UNIVERSIDAD DEL BÍO-BÍO
FACULTAD DE CIENCIAS EMPRESARIALES
DEPARTAMENTO DE CIENCIAS DE LA COMPUTACIÓN Y
TECNOLOGÍAS DE LA INFORMACIÓN
CAMPUS CHILLÁN

"Desarrollo de plug-in Eclipse

para facilitar el desarrollo de aplicaciones android que usen SQLite"

KEVIN STEVE SANDOVAL SOLIS

DANIEL EDUARDO JELDRES FUENTES

MEMORIA PARA OPTAR AL TÍTULO DE

INGENIERO CIVIL EN INFORMÁTICA

Chillán, julio 2013

Resumen

Este proyecto se presenta para dar conformidad a los requisitos exigidos por la Universidad de Bío-Bío en el proceso de titulación para a la carrera de Ingeniería Civil en Informática. El proyecto titulado "Desarrollo de plug-in eclipse para facilitar el desarrollo de aplicaciones android que usen SQLite"

Este proyecto se centra en el área de desarrollo de aplicaciones android que utilizan BD SQLite. Apunta a resolver la problemática que significa trabajar con BD en el desarrollo de aplicaciones android en las cuales se debe trabajar directamente con código SQL dentro del código java en el desarrollo, para poder realizar operaciones básicas de BD como son la inserción, eliminación, actualización y consulta, además resuelve los problemas de integridad de tipos de datos y de claves foráneas dentro de la BD. Este proyecto implementa las funciones de inserción, eliminación, actualización y consulta de datos mediante métodos de clases java generadas desde la BD, como la clase Fachada, y no mediante código SQL directamente. Además valida los tipos de datos que se insertan y las posibles violaciones de referencia en la inserción. Este proyecto se desarrolla en lenguaje java orientado a objetos, utilizando arquitectura de modelo vista controlador (MVC).

Con este proyecto se logra facilitar la creación e implementación de BD en las aplicaciones android tornando más fácil, rápido y sencillo el desarrollo para los programadores, permitiendo ahorrar tiempo y código, sin mencionar que es una aplicación integrada en el IDE Eclipse.

Abstract

The following project is presented to fulfill all the compulsory requirements asked for the University of Bío-bío in the certification process for the Civil Computer Engineering Bachelor's degree. The project entitled "Development of plug-ins for eclipse to facilitate the development of applications using SQLite android"

This project is proposed as a self-developed theme by the authors-students of this work, which focuses on the area of application development android using SQLite DB. It focuses on the problem of working with DB in the development of android applications in which they must work directly with SQL code within java code in development to perform basic DB such as insert, delete, update and consultation , it solves the integrity of data types problems and foreign keys in the DB as well. This project implements the functions insert, delete, update and consultation of data using methods java classes generated from the database, like the Fachada class, and not through SQL code directly. It also validates the data types that are inserted and the possible reference violations in the insert. This project is developed in object-oriented Java using architecture model view controller (MVC).

This project facilitates the creation and implementation of DB in android applications turning the development for programmers easier, faster and simpler and therefore saving time and code, not to mention that it is an integrated application in the Eclipse IDE.

Índice General

Índice Tablas

Índice Figuras

1 INTRODUCCIÓN

Actualmente las aplicaciones android son una de las más desarrolladas, ya sea a nivel empresarial como personal. Todo esto debido a la gran masificación de dispositivos móviles existentes en el mercado con dicho sistema operativo. La mayoría de las aplicaciones android utilizan Base de Datos para manejar la persistencia de la información. Hoy en día android utiliza el motor SQLite para manejar la persistencia de la información, en este sentido, a nivel de desarrollo es difícil de implementar ya que se debe introducir código SQL entre las líneas de Java, lo cual hace que la programación sea difícil y desordenada.

El presente informe explica el proceso de desarrollo de un plug-in para el IDE Eclipse con la finalidad de facilitar el desarrollo de aplicaciones para android que necesiten implementar bases de datos SQLite. Con la cual se podrá realizar una programación más fácil y más ordenada en lo que respecta al código desarrollado, siempre manteniendo la orientación a objetos. Con esto se pretende dejar detallados todos los pasos del desarrollo para que cualquiera pueda entenderlo claramente y así facilitar futuras modificaciones.

A continuación se detalla el contenido de cada capítulo:

El Capítulo 2, presenta la descripción del problema actual, que motiva la necesidad de una solución adecuada.

El Capítulo 3 detalla los objetivos generales y específicos del proyecto, el ambiente de la ingeniería del software, como metodología de desarrollo utilizada, técnicas, estándares y herramientas de apoyo. También presenta una sección dedicada a definiciones de siglas, abreviaciones y conceptos del negocio en sí, que facilitan el entendimiento del presente documento.

El Capítulo 4 especifica los requerimientos del software, sus alcances y objetivos. Además, describe las características que tienen las interfaces de usuario, la relación entre éstas últimas y los periféricos hardware y las interfaces de comunicación. Detalla una lista

de requerimientos funcionales específicos del sistema, interfaces externas de entrada, de salida y los atributos del producto.

En el Capítulo 5 se analiza la problemática, se definen los actores quienes interactúan con el sistema y los casos de uso que representan los requerimientos funcionales del proyecto, estos casos de uso son especificados uno por uno. Además se incluye el diagrama de casos de uso de la especificación. Finalmente se describe el modelo de datos, explicando las clases utilizadas para este proyecto y se incluye el diagrama de clases que lo representa.

En el Capítulo 6, correspondiente al diseño, se especifican el diseño físico de la BD en la cual se explican el tipo de persistencia que se utiliza en la implementación, que en este caso es un XML, para el cual se aclaran todos los tag que lo componen. Además se presenta el diagrama de arquitectura funcional, donde se detalla desde el nivel más alto, el plug-in, hasta llegar a los módulos que componen el proyecto. Se detallan las interfaces de usuario que se utilizan indicando su contenido para cada una de ellas. Además incluye el diagrama de navegación y de menú. Finalmente se especifican cada uno de los módulos que componen el proyecto.

El Capítulo 7 corresponde a las pruebas, donde se especifican que características o funcionalidades se probarán, se incluye un plan de pruebas a ejecutar en el cual se indica el tipo la característica a probar y el criterio con el cual se dará por concluida la prueba y si esta fue un éxito o un fracaso. Además, se indica el responsable de ejecutar cada prueba y el calendario de pruebas que se representa con una carta Gantt. Finalmente se muestra el resultado de la ejecución de las pruebas.

En el Capítulo 8 se incluyen las tablas con los esfuerzos requeridos por cada alumno para la realización del proyecto.

El Capítulo 9 corresponde a la conclusión del proyecto, donde se dan a conocer las cosas relevantes en el desarrollo de la aplicación, sus impresiones personales y las dificultades que se encontraron a lo largo del desarrollo.

En el Capítulo 10 se detalla la bibliografía utilizada a lo largo del proyecto y las páginas de internet visitadas, que fueron apoyo para el desarrollo.

En el Capítulo 11 se especifica la planificación inicial del proyecto además de un estudio y cálculo de Puntos de Casos de Uso, en el cual se detalla desde los casos de uso la estimación de horas necesarias para el desarrollo.

En los anexos se detallan el diccionario de datos, donde se especifican los datos del modelo. Además, se incluye el manual de usuario de la aplicación.

2 DEFINICION DEL PROBLEMA

2.1 Descripción de la problemática

Android es en la actualidad el Sistema Operativo para dispositivos móviles más usado a nivel mundial. El desarrollo de aplicaciones para este es muy variado, desde aplicaciones empresariales hasta juegos de recreación.

Gran parte de estas aplicaciones, necesitan de una Base de Datos para manejar su persistencia, por lo que android nos ofrece SQLite para esta función.

SQLite es sin duda ideal para android, dado que es una Base de Datos "liviana" y los aparatos móviles no son tan potentes como computadores de escritorio o servidores donde se pueden usar Base de Datos mucho más potentes, sin embargo el trabajo con SQLite en android resulta bastante engorroso y tedioso para el programador, ya que se debe crear toda la estructura de la Base de Datos a través de código, trabajando con muchos comandos SQL directamente, lo cual hace que este trabajo esté propenso a errores de digitación, muchas veces difíciles de encontrar y que hacen que se pierda bastante tiempo en el desarrollo. Además, SQLite no permite manejar de manera directa la integridad de la Base de Datos y si se requiere de ésta, se debe implementar manualmente mediante procedimientos o triggers.

.

El mismo problema se genera al crear manualmente el código necesario para la inserción, búsqueda, modificación y eliminación de datos.

Teniendo en cuenta estos problemas, se hace necesario implementar alguna forma de crear la estructura de la Base de Datos, manejar sus restricciones de integridad (chequeo de validez e integridad referencial) y manejar la inserción, búsqueda, modificación y eliminación de una forma más sencilla y eficiente.

2.2 Diagrama de relación de elementos

A continuación se presenta un esquema, en el cual se visualiza la interacción de los elementos para el desarrollo android y donde tiene lugar el plugin desarrollado.

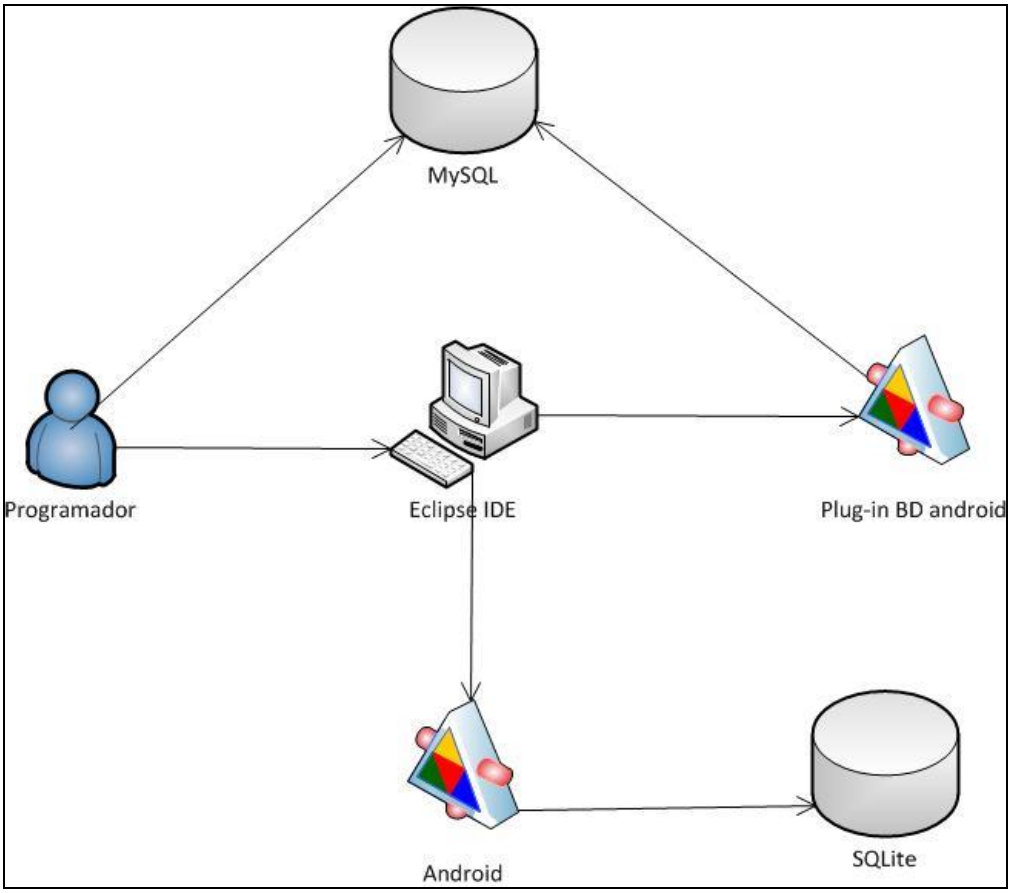


Figura Nº 1: Esquema de Relación.

3 DEFINICIÓN PROYECTO

3.1 Objetivos del proyecto

- Diseñar e implementar una herramienta de software para evitar errores de digitación y facilita la generación de código java para la creación de la estructura de una Base de Datos SQLite en android usando el IDE Eclipse.
- Diseñar e implementar una solución para suplir la carencia de manejo de restricciones de integridad (integridad referencial y chequeo de validez) de SQLite en android.
- Diseñar e implementar una solución para apoyar la generación manual de código para crear clases POJO e implementar patrones de diseño como DAO y fachada para una Base de Datos SQLite en android usando el IDE Eclipse.

3.2 Ambiente de Ingeniería de Software

Metodología: Modelo de desarrollo iterativo incremental que nos permite en cada iteración generar un módulo con nuevas funcionalidades para la aplicación, y gracias a esto poder tempranamente ver resultados tangibles y corregir posibles errores.

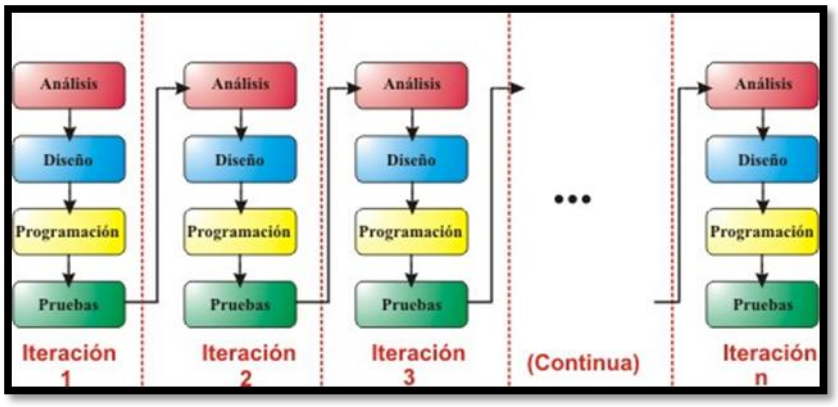


Figura Nº 2: Metodología de desarrollo.

Técnicas: diagramas de casos de uso, MER.

Usaremos estas técnicas ya que son de gran utilidad en el diseño, dejando claro las funcionalidades de la aplicación y el modelo de datos respectivamente. Para detallar los procesos de cada funcionalidad de nuestra solución, utilizaremos diagramas de caso de uso, donde se refleje cómo se llevará a cabo cada proceso y así ver al final de cada

iteración si se ha cumplido con estos. El MER será de mucha utilidad para crear nuestro modelo de datos y definir así la estructura para la persistencia.

Herramientas:
- Diagramas de Casos de Uso: ArgoUML v0.28
- Modelo Entidad-Relación : DIA v0.97.2
- Programación: Eclipse Helios (IDE)
- Diseño de Interfaz: Windows Builder v1.5.0
- Base de Datos: SQLite (solo para hacer pruebas)
  MySQL Server 5.5 (Para pruebas)
  MySQL Front 5.2 (Creación de BD externa de pruebas)

3.3 Glosario de Términos

IDE: sigla en inglés de *Integrated Development Environment* (Entorno de Desarrollo Integrado)

CU: *Diagrama de Casos de Uso*, representa la forma en cómo un Cliente (Actor) opera con el sistema en desarrollo.

POJO: *Plain Old Java Objec, sigla* utilizada por programadores Java para enfatizar el uso de clases simples y que no dependen de un framework en especial.

DAO: *Data Access Object* (Objeto de Acceso a Datos),

MER: *Modelo Entidad Relación.*

BD: *Base de Datos.*

4 ESPECIFICACIÓN DE REQUERIMIENTOS DE SOFTWARE

4.1 Alcances

La herramienta permitirá ahorrar tiempo en el desarrollo de aplicaciones para android que utilicen SQLite.

Las aplicaciones existentes actualmente para trabajar con SQLite en android, están más orientadas a la visualización de los datos de las bases de datos ya creadas, y las que permiten la generación del código para la creación de la estructura de la Base de Datos carecen de facilidad de uso, ya que requieren trabajar manualmente con archivos XML.

La aplicación no creará la Base de Datos SQLite en el dispositivo android, sino que generará el código para crear la Base de Datos, que el desarrollador incluirá en su aplicación. Por esto, tampoco permitirá agregar, modificar, consultar o eliminar datos de la Base de Datos, pues no se pueden hacer dichas operaciones sobre una Base de Datos que aún no ha sido creada, sino que generará el código que el desarrollador incluirá en su aplicación para realizar dichas operaciones.

La aplicación permitirá importar la estructura de una Base de Datos externa (MySQL). Es importante dejar claro que no importará los datos de dicha Base de Datos, por los mismos motivos expuestos en el párrafo anterior, sino sólo importará su estructura y permitirá generar el código que el desarrollador puede incluir en su aplicación android para crear una Base de Datos SQLite con la misma estructura de la Base de Datos MySQL importada.

El software permitirá manejar restricciones de integridad, pero no en su totalidad. Permitirá tener un control de validez de tipo de datos, rango de valores para los datos, asignar valores por defecto e integridad referencial sólo en la inserción, dejando la opción de implementar en un futuro la integridad referencial para todas las operaciones.

El software permitirá, además, al usuario generar clases POJO que representen las tablas de su Base de Datos, y el código generado para realizar las operaciones de

inserción, modificación, eliminación y búsqueda será implementado con los patrones DAO y fachada.

4.2 Objetivo del software

General:

- Desarrollar un software el cual manejará información sobre la estructura de la Base de Datos que el usuario desea crear, apoyando el proceso de generación de código fuente para la creación y manejo de esta, logrando una disminución en tiempo que toma dicho proceso.

Específicos:

- El software manejará información sobre la estructura de las tablas de la Base de Datos que el usuario desea crear, apoyando el proceso de generación de código fuente para la creación y manejo de estas (implementación de clases POJO, y patrones DAO y fachada para las operaciones de inserción, modificación, eliminación y búsqueda), logrando una disminución en tiempo que toma dicho proceso.
- El software manejará información sobre las restricciones de integridad que tendrá la Base de Datos que el usuario desea crear, apoyando el proceso de generación de código fuente para la implementación de chequeo de validez de tipo de datos, manejo de rango de valores, asignación de valores por defecto y restricciones de integridad referencial para la inserción de datos, logrando una disminución en tiempo que toma dicho proceso.

4.3 Descripción Global del Producto

4.3.1 Interfaz de usuario

Según lo requerido por el usuario, hemos creado una interfaz que cumple con las características básicas de cualquier administrador de BD y que resulta familiar para cualquier programador con conocimientos en productos de Base de Datos, en donde la iconografía es estandarizada, los paneles, accesos y navegabilidad es simple, además, de sencilla. Para esto se crearon los siguientes paneles:

- **Ventanas o Paneles:**
  - Vista Principal:

    Esta es la ventana principal la cual muestra siempre al usuario el esquema de la Base de Datos y los paneles de ingreso de datos. Sus características son:
    - Color: Gris del mismo tono que el entorno Eclipse.
    - Contenido: una barra de menú tipo JMenuItem en la parte superior, un Jpanel en el lado izquierdo y un JPanel en el lado derecho.
    - Dimensiones: 800x600.
    - Iconos: icono principal de la ventana igual al del entorno eclipse.
  - Panel Árbol de BD:

    Este panel es el encargado de mostrar siempre al programador el árbol que se va creando a medida que se implementa la Base de Datos, siempre se encuentra en el lado izquierdo de la ventana principal. Sus características son:
    - Color: Posee un fondo Blanco.
    - Contenido: un JTree que representa la Base de Datos implementada.
    - Dimensiones: 226x550.
    - Iconos: contiene iconos estandarizados de Base de Datos para atributos, claves primarias, Base de Datos, tablas.
  - Nueva BD:

    Esta es la interfaz en la cual el programador crea una nueva Base de Datos proporcionando un nombre a esta, además sirve como panel de edición para poder cambiar el nombre de una BD, siempre se ubica en el lado derecho de la vista principal. Su contenido es:
    - Color: morado que se degrada de izquierda a derecha.
    - Contenido: un JLabel que indica lo que se escribirá en el campo de texto, un JTextFiel donde se escribe el nombre de la BD y un JButton que Actualiza la BD.
    - Dimensiones: 557x550.

- Cargar BD:

  Esta es la interfaz encargada de proporcionar al usuario las bases de datos guardadas en el proyecto, para que este seleccione una y la cargue en la ventana principal. Sus características son:

  - Color: morado que se degrada de izquierda a derecha.
  - Contenido: un JPanel que contiene la lista de bases de datos guardadas, un JButton para cargar y un JButton para cancelar.
  - Dimensiones: 350x250.
  - Iconos: icono principal de la ventana igual al del entorno eclipse.

- Importar BD:

  Esta es la interfaz encargada de conectarse a una Base de Datos externa MySQL y de importar la Base de Datos requerida por el programador. Sus características son:

  - Color: morado que se degrada de izquierda a derecha.
  - Contenido: 6 etiquetas tipo JLabel para indicar la función de los campos, 6 campos tipo JTextField donde se ingresan los datos de conexión como el nombre de la Base de Datos, el puerto, el servidor, nombre de usuario y contraseña, un botón tipo JButton para testear la conexión, un botón tipo JButton para conectarse y un JTextArea para mostrar posible errores de conexión.
  - Dimensiones: 558x597.
  - Iconos: Iconos: icono principal de la ventana igual al del entorno eclipse.

- Nueva Tabla:

  Este panel es el que se encarga de agregar una nueva tabla a la Base de Datos, solicitando al usuario un nombre para esta. Además sirve para la edición de la tabla, mostrar los atributos de ella y cambiarle el nombre a una tabla. Sus características son:

  - Color: morado que se degrada de izquierda a derecha.
  - Contenido: un JLabel para indicar la función del campo, un JTextField para escribir el nombre de la tabla, un JButton

para crear la tabla y un JTable para mostrar los atributos de la tabla.

- Dimensiones: 557x550.
- Iconos: icono que representa una tabla en todos los sistemas de gestión de base de datos.

- Nuevo Atributo

  Este panel se encarga de recopilar los datos para crear un nuevo atributo en una tabla determinada. Además, sirve para la edición de dichos atributos. Sus características son:

  - Color: morado que se degrada de izquierda a derecha.
  - Contenido: 6 JLabel para etiquetar los campos de ingreso de datos, 5 JTextField para el ingreso de datos por pate del programador, 1 JComboBox para seleccionar el tipo de dato, 3 JCheckBox para seleccionar datos boléanos como clave primaria, acepta nulos o si es auto incrementable y 1 JButton para guardar o crear según corresponda.
  - Dimensiones: 557x550.
  - Iconos: icono que representa un atributo en todos los sistemas de gestión de base de datos.

- Claves Foráneas:

  Este panel es el encargado de la administración de claves foráneas de una tabla determinada, en él se pueden ver las clave de una tabla, editarlas, borrarlas o agregar una nueva. Sus características son:

  - Color: morado que se degrada de izquierda a derecha.
  - Contenido: un JTextField donde se muestra el nombre de la tabla, una JTable donde se muestran todas las claves foráneas de la tabla, 1 JButton para agregar una nueva clave, un JButton para borrar una clave, un JButton para editar una clave.
  - Dimensiones: 557x550.

- Nueva Clave Foránea:

  Este panel es el encargado de solicitar los datos al programador para agregar una nueva clave foránea a una tabla, además sirve para poder editar los datos de una clave ya creada. Sus características son:

  - Color: morado que se degrada de izquierda a derecha.
  - Contenido: 6 JLabel para etiquetar los campos, 3 JTextField para ingresar datos, 3 JComboBox para seleccionas datos y un JButton para guardar o agregar según corresponda.
  - Dimensiones: 557x550.

❖ **Especificación de iconografía:**

Icono	Descripción
	**Icono de Eclipse IDE**
	**Icono de Base de Datos**
	**Icono de un atributo**
	**Icono de una clave primaria**
	**Icono de una tabla**

Tabla Nº 1: Iconografía.

Las pantallas se cargarán siempre en el lado derecho de la ventana principal. El lado izquierdo está reservado solo para el árbol expandible que representa la Base de Datos. Las únicas ventanas emergentes serán la de cargar BD, importar BD y las ventanas de mensajes de errores.

Cuando se ingresen datos erróneos en los paneles de formularios la aplicación mostrará un mensaje de error tornando de un color rojo el fondo del campo donde se ingresó un dato erróneo y mostrando un mensaje al programador, cuando el error es corregido el mensaje desaparece y el color vuelve a la normalidad. Todo esto se ejecuta en tiempo real, mientras se escribe en el campo y no es necesario presionar ningún botón para hacer la validación de estos.

Algunas opciones como el editar BD, editar tabla o editar atributo están disponibles sólo con hacer doble click sobre el árbol de la estructura, teniendo además la opción desde los menús.

Cuando se edite una tabla o se haga doble click en ella aparecerá el panel de nueva tabla cargando los datos de la tabla que se clicó donde se verá el nombre de la tabla. Este nombre será editable y además se mostrará una lista de atributos que posee la tabla, esta última solo de información ya que no se puede realizar ninguna operación sobre ella.

Para editar un atributo basta con hacer doble click sobre el atributo y modificar los datos de este.

En lo que respecta a claves foráneas estas estarán disponible en el menú contextual del botón derecho sobre una tabla, y se cargará un panel donde se muestra el nombre de la tabla, el cual no es editable, y una lista de atributos foráneos donde si se quiere editar alguno se seleccionará y se presionará el botón editar o borrar si esto es lo que se desea. Si se quiere agregar una nueva basta con presionar agregar nueva y se mostrará el formulario para esto.

4.3.2 Interfaz Software

- Nombre: Eclipse SDK
- Abreviación: Eclipse
- Número especificación o Versión: Helios versión 3.2.6
- Fuente: http://www.apache.org/

- Nombre: MySQL Server
- Abreviación: MySQL
- Número especificación o Versión: 5.5

4.4 Requerimientos Específicos

4.4.1 Requerimientos Funcionales del sistema

ID	Nombre	Descripción
RF01	Crear BD	Permite crear una Base de Datos vacía, en la cual se podrá posteriormente agregar tablas y atributos.
RF02	Editar BD	Permite cambiarle el nombre a la Base de Datos, ya sea creada, cargada o importada.

RF03	Ver tablas de una BD	Permite al programador ver las tablas que tiene la base de datos, para poder acceder a ellas.
RF04	Cargar BD	Permite cargar una Base de Datos desde un archivo XML guardado previamente en el proyecto.
RF05	Guardar BD	Permite guardar la estructura de la Base de Datos en el proyecto como un XML, que posteriormente se puede cargar nuevamente en el editor.
RF06	Importar BD	Permite importar una Base de Datos desde un motor externo a la aplicación que es MySQL. Se debe contar con los datos de acceso a dicho motor, ya sea un nombre de usuario y contraseña, puerto, servidor y una Base de Datos.
RF07	Generar código java	Permite generar código java de la Base de Datos creada, para poder utilizarlo en la aplicación que se construya. Se generan los patrones de Fachada, DAO y POJOs de la estructura.
RF08	Crear tabla	Permite crear una nueva tabla en la Base de Datos. No se permitirá la creación

		de tablas con nombre iguales. O que en el nombre contengan caracteres inválidos.
RF09	Cambiar nombre de tabla	Permite cambiarle el nombre a una tabla de la Base de Datos. No admite caracteres inválidos o especiales.
RF10	Ver atributos de una tabla	Permite al programador ver todos los atributos de una tabla. Solo es una opción de lectura ya que desde aquí no se permite la edición de dichos atributos
RF11	Ver claves foráneas de una tabla	Permite ver las claves foráneas de una tabla. Desde esta interfaz se permite la edición, la agregación y la eliminación de las claves foráneas.
RF12	Agregar una clave foránea	Permite la agregación de una clave foránea nueva. Se debe tener en cuenta que los tipos de datos deben ser iguales para poder referenciar dos atributos.
RF13	Editar una clave foránea	Permite la edición de una clave foránea ya creada. En este requerimiento se debe tener cuidado con los tipos de datos que se referencias para que no sean de

		distintos tipos.
RF14	Borrar una clave foránea	Permite borrar una clave foránea de una tabla.
RF15	Borrar una tabla	Permite la eliminación de una tabla de la Base de Datos. No se podrá eliminar tablas que tengan atributos que son foráneos para otras tablas.
RF16	Agregar un atributo	Permite agregar un nuevo atributo a una tabla. Se validan los datos del atributo como son que no se ingresen caracteres donde solo deben ir números o caracteres inválidos en los nombres.
RF17	Editar un atributo	Permite editar un atributo de una tabla. Se debe tener cuidado del tipo de dato cuando se modifica ya que si es clave foránea y se cambia el tipo de datos puede crear errores posteriores. No debería permitirse el cambio.
RF18	Borrar un atributo	Permite eliminar un atributo que no sea referenciado por otro en otra tabla. Si es referenciado este no se debe borrar.

Tabla Nº 2: Requerimientos funcionales.

4.4.2 Interfaces externas de entrada

Identificador	Nombre del ítem.	Detalle de Datos contenidos en ítem
DE_01	Importar BD	Motor BD, Nombre BD, puerto, servidor, nombre usuario, contraseña.
DE_02	Crear BD	Nombre de la Base de Datos
DE_03	Editar BD	Nombre de la Base de Datos
DE_04	Crear Tabla	Nombre tabla.
DE_05	Editar Tabla	Nombre tabla.
DE_06	Crear Atributo	Nombre, tipo, valor por defecto, val mínimo, valor máximo, longitud máxima, auto-increment, clave primaria, admitir null
DE_07	Editar Atributo	Nombre, tipo, valor por defecto, val mínimo, valor máximo, longitud máxima, auto-increment, clave primaria, admitir null
DE_08	Agregar Clave Foránea	Nombre, Base de Datos, tabla, atributo, tabla de referencia, atributo de referencia
DE_09	Editar Cave Foránea	Nombre, Base de Datos, tabla, atributo, tabla de referencia, atributo de referencia

Tabla Nº 3: Interfaces externas de entrada.

4.4.3 Interfaces externas de Salida

Identificador	Nombre del ítem.	Detalle de Datos contenidos en ítem	Medio Salida
IS_01	Tablas de la Base de Datos	Nombre de la tabla	pantalla
IS_02	Atributos de una tabla	Nombre, tipo, PK, null, defecto, auto-incrementable	pantalla

IS_03	Claves foráneas de una tabla	Nombre atributo, nombre FK, tabla de referencia, atributo referenciado	pantalla
IS_04	Generar código java	AdminSQLiteOpenHelper, Fachada, DAOs, POJOs	Archivos Java
IS_05	Guardar BD	Base de Datos, tablas, atributos	Archivo XML

Tabla Nº 4: Interfaces externas de salida.

4.4.4 Atributos del producto

- USABILIDAD- OPERABILIDAD: El sistema cuenta con mensajes de error para el programador que cuentan con especificación de en qué lugar se cometió un error, además muestra cuál es el error cometido ya sea de caracteres inválidos como de conexiones erróneas.

5 ANÁLISIS

5.1 Diagrama de casos de uso

5.1.1 Actores

- Programador : El programador es el encargado de desarrollar aplicaciones en android y de utilizar la aplicación en su gran mayoría. Sus conocimientos deben ser altos en la programación en andoid y debe tener conocimientos de bases de datos y conocer algunos motores de bases de datos. El programador tendrá acceso al 100% de las funciones del software.

- Base de Datos Externa : La BD externa es la encargada de crear una Base de Datos en el motor MySQL, su función es proporcionar una Base de Datos a la cual se conecte la aplicación para poder importar datos. Este solo interactúa con los sistemas en el módulo de importar una Base de Datos externa.

5.1.2 Casos de Uso y descripción

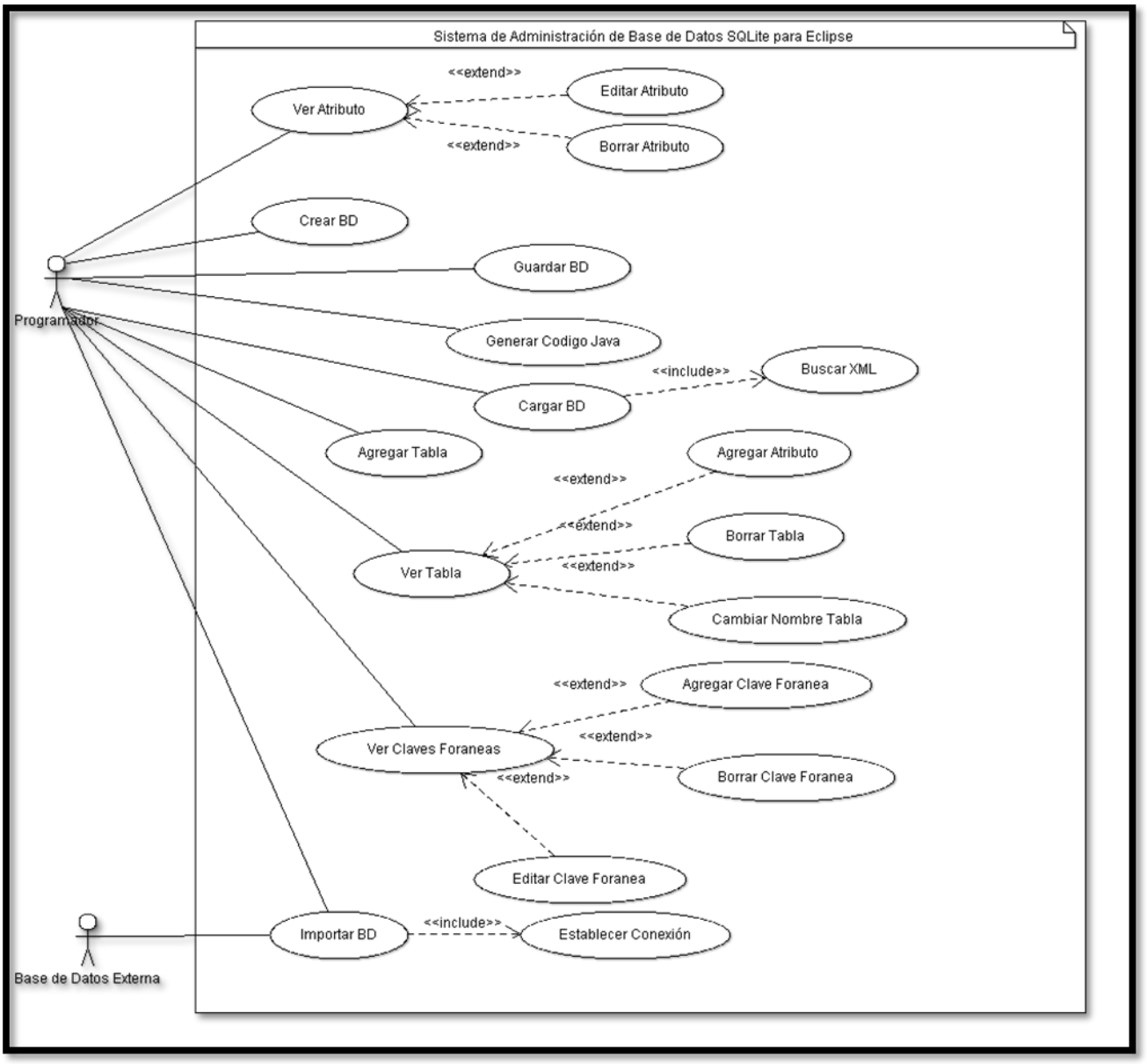


Figura Nº 3: Modelo de Casos de Uso.

El caso de uso tiene dos actores principales el programador y la Base de Datos externa, esta última solo tiene participación en el caso de uso importar BD ya que es donde interactúa el programador con ella para tener acceso a sus Base de Datos ajenas a la aplicación y que están escritas en otro motor.

El programador tiene acceso a todos los casos de uso ya que es el principal usuario del sistema este puede hacer cosas como crear una nueva BD, Cargar una BD, Guardar una BD e importar BD, además opciones de edición de BD como agregar o modificar

tabla, agregar o modificar atributo, agregar o modificar claves foráneas y todas las opciones de borrar correspondientes. Por último tiene acceso a la opción de generar código java el cual le permite trabajar con la Base de Datos creada en su aplicación.

5.1.3 Especificación de los Casos de Uso

5.1.3.1 Caso de Uso: <Crear BD>

- ID: PE01
- Descripción: Crear una nueva estructura de Base de Datos para una aplicación.
- Pre-Condiciones: Crear el proyecto para android donde se creará la estructura.
- Flujo de Eventos Básicos:

    1. El caso de uso comienza cuando el programador necesita crear una nueva Base de Datos para una aplicación android.
    2. El programador accede a la interfaz gráfica del plug-in.
    3. El sistema muestra la interfaz al usuario con las opciones disponibles.
    4. El programador selecciona crear nueva Base de Datos.
    5. El sistema solicita un nombre para la Base de Datos.
    6. El programador proporciona un nombre.
    7. El sistema crea la nueva Base de Datos en proyecto.

- Post-Condiciones: el sistema crea la Base de Datos además de dejar la interfaz en la edición de tal Base de Datos.
- Flujo alternativo:
    9. El nombre de la Base de Datos posee caracteres inválidos.
    10. El sistema muestra el error al programador.
    11. El programador cambia el nombre de la Base de Datos.

5.1.3.2 Caso de Uso: <Guardar BD>

- ID: PE02
- Descripción: guardar la Base de Datos en formato XML dentro del proyecto eclipse.
- Pre-Condiciones: se debe haber cargado previamente una Base de Datos y haber realizado modificaciones a la estructura de esta o haber creado una nueva Base de Datos.
- Flujo de Eventos Básicos:

    1. El caso de uso comienza cuando el programador desea guardar los cambios realizados en una estructura de la Base de Datos cargada o nueva.
    2. El programador selecciona la opción guardar.
    3. El sistema recopila y guarda los cambios en la estructura.

- Post-Condiciones: El sistema debe quedar con la Base de Datos guardada o actualizada según corresponda.

5.1.3.3 Caso de Uso: <Cargar BD>

- ID: PE03
- Descripción: Cargar una Base de Datos previamente guardada desde un archivo XML en el proyecto eclipse.
- Pre-Condiciones: Se debe haber guardado previamente al menos una Base de Datos para poder cargarla.
- Flujo de Eventos Básicos:

    1. El caso de uso comienza cuando el programador necesita cargar una Base de Datos en el administrador.
    2. El programador selecciona la opción cargar BD
    3. <<include Buscar XML>>
    4. El sistema proporciona los XML o bases de datos disponibles.
    5. El programador selecciona la Base de Datos.
    6. El sistema carga la Base de Datos en el administrador.
- Post-Condiciones: El sistema debe quedar con la Base de Datos cargada y lista para posibles ediciones.

5.1.3.4 Caso de Uso: <Buscar XML>

- ID: PE04
- Descripción: Buscar los XML disponibles y guardados en el proyecto para mostrarlos al usuario.
- Pre-Condiciones: El sistema debe haber requerido la acción cargar Base de Datos.
- Flujo de Eventos Básicos:

    4. El caso de uso comienza cuando el programador requiere ver las bases de datos disponibles en el proyecto.
    5. El sistema busca los XML en el proyecto.
    6. El sistema carga los XML del proyecto.
    7. El usuario ve los XML del proyecto.
- Post-Condiciones: El usuario debe poder ver todos los archivos XML del proyecto.

5.1.3.5 Caso de Uso: <Generar código Java>

- ID: PE05
- Descripción: Generar código de patrones (DAO, POJOs, Fachada) automáticamente dentro del proyecto.
- Pre-Condiciones: El sistema debe estar con el proyecto cargado y además con la Base de Datos de la cual se desea obtener cualquiera de los códigos.
- Flujo de Eventos Básicos:

    1. El caso de uso comienza cuando el programador desea generar patrones.
    2. El sistema muestra en la interfaz al usuario la opción de generar código y patrones.
    3. El programador escoge la opción.
    4. El sistema genera el código en el proyecto.

Post-Condiciones: El sistema debe quedar actualizado, esto es, que el proyecto debe quedar con todos los patrones generados en un paquete separado.

5.1.3.6 Caso de Uso: <Importar estructura>

- ID: PE06
- Descripción: importar una estructura de una Base de Datos externa configurada en el motor MySQL.
- Pre-Condiciones:
- Flujo de Eventos Básicos:

    1. El caso de uso comienza cuando el programador necesita cargar una Base de Datos escrita en el motor MySQL.
    2. El programador selecciona la opción de importar Base de Datos MySQL.
    3. El sistema solicita los datos de conexión y acceso.
    4. El programador proporciona los datos de acceso y conexión.
    5. <<include Establecer Conexión>>
    6. El sistema importa la estructura de la Base de Datos.

- Post-Condiciones: El sistema debe actualizarse y quedar con una nueva Base de Datos creada correspondiente a la estructura importada.

- Flujo alternativo:
    5. La conexión no se realiza.
    6. El sistema muestra el error al usuario.
    7. El programador proporciona los datos nuevamente.

5.1.3.7 Caso de Uso: <Establecer Conexión>

- ID: PE07
- Descripción: Establecer la conexión con el motor de MySQL.
- Pre-Condiciones: Se debe haber solicitado la opción importar BD.
- Flujo de Eventos Básicos:

    7. El caso de uso comienza cuando al programador solicita conexión al motor de MySQL.
    8. El sistema estable la conexión con los datos dados.
    9. El sistema muestra al usuario el resultado de la conexión.

- Post-Condiciones: El sistema debe quedar conectado al motor MySQL.

5.1.3.8 Caso de Uso: <Ver tablas>

- ID: PE08
- Descripción: Mostrar al usuario los datos de la tabla y los atributos que esta posee.
- Pre-Condiciones: Se debe haber creado con anterioridad la tabla que se desee ver.
- Flujo de Eventos Básicos:

    1. El caso de uso comienza cuando el programador desea ver el contenido de una tabla.
    2. El programador selecciona la opción de ver tabla.
    3. El sistema proporciona los datos de esta al usuario. Post-Condiciones: El sistema debe quedar con una nueva tabla en la Base de Datos y la interfaz debe quedar lista para la inserción de datos en dicha tabla.

5.1.3.9 Caso de Uso: <Agregar tabla>

- ID: PE09
- Descripción: crear una nueva tabla en la Base de Datos.
- Pre-Condiciones: Se debe haber creado o cargado previamente una Base de Datos.
- Flujo de Eventos Básicos:

    1. El caso de uso comienza cuando el programador desea crear una nueva tabla en la Base de Datos.
    2. El programador selecciona crear tabla.
    3. El sistema solicita un nombre de tabla.
    4. El programador proporciona un nombre para la nueva tabla.
    5. El sistema crea la nueva tabla.

- Post-Condiciones: El sistema debe quedar con una nueva tabla en la Base de Datos.

- Flujo alternativo:
    4. El nombre de la tabla ya existe.
    5. El sistema solicita un nuevo nombre de tabla.
    6. El programador proporciona un nuevo nombre de tabla

5.1.3.10 Caso de Uso: <Cambiar Nombre Tabla>

- ID: PE10
- Descripción: modificar el nombre de una tabla ya creada en la Base de Datos.
- Pre-Condiciones: Se debe haber creado o cargado previamente una Base de Datos y además haber accedido a la estructura de una tabla.
- Flujo de Eventos Básicos:

    1. El caso de uso comienza cuando el programado desea cambiar el nombre de una tabla en la Base de Datos.
    2. El programador escoge la tabla a modificar.

3. El sistema muestra el nombre de una tabla.
4. El programador modifica el nombre de la tabla.
5. El programador guarda los cambios.
6. El sistema guarda los cambios en la base de datos.

- Post-Condiciones: El sistema debe quedar con la tabla actualizada.

- Flujo alternativo:
    4. El nombre de la tabla posee caracteres inválidos
    5. El sistema muestra mensajes de error.
    6. El programador modifica el nombre

5.1.3.11 Caso de Uso: <Borrar tabla>

- ID: PE11
- Descripción: Borrar una tabla en la Base de Datos.
- Pre-Condiciones: Se debe haber creado o carado la Base de Datos en el administrador.
- Flujo de Eventos Básicos:

    1. El caso de uso comienza cuando el programador desea borrar una tabla de la Base de Datos.
    2. El sistema muestra las tablas de la Base de Datos.
    3. El programador escoge la tabla.
    4. El programador escoge eliminar tabla.
    5. El sistema verifica las dependencias.
    6. El sistema borra la tabla.

- Post-Condiciones: El sistema debe quedar actualizado sin la tabla que se eliminó.

- Flujo alternativo:
    5. La tabla posee referencias o dependencias.
    6. El sistema muestra mensajes de error.

5.1.3.12 Caso de Uso: <Agregar Atributo>

- ID: PE12
- Descripción: Permite agregar un atributo a una tabla recién creada.
- Pre-Condiciones: Se debe haber creado la tabla previamente.
- Flujo de Eventos Básicos:

    1. El caso de uso comienza cuando el programador desea agregar un atributos a una tabla
    2. El sistema da la opción de agregar atributos
    3. El programador ingresa los datos.
    4. El programador guarda el atributo
    5. El sistema guarda el atributo en la estructura de la Base de Datos.

- Post-Condiciones: El sistema debe quedar actualizado con un nuevo atributo en la tabla.

- Flujo alternativo:
    3. Los datos son inválidos.
    4. El sistema informa al programador de los errores.
    5. El programador cambia los datos.

5.1.3.13 Caso de Uso: <Ver Atributos>

- ID: PE13
- Descripción: permite ver todos los atributos de una tabla.
- Pre-Condiciones: Se debe haber creado la tabla previamente con sus respectivos atributos.
- Flujo de Eventos Básicos:

    1. El caso de uso comienza cuando el programador desea ver los atributos de una tabla.
    2. El sistema da la opción de ver atributos
    3. El programador accede la opción.
    4. El sistema muestra los atributos al programador.

- Post-Condiciones: El sistema debe quedar mostrando los atributos al programador.

5.1.3.14 Caso de Uso: <Borrar Atributo>

- ID: PE14
- Descripción: Permite borrar un atributo de una tabla.
- Pre-Condiciones: Se debe haber creado la tabla y el atributo previamente además de acceder a la opción de ver atributos.
- Flujo de Eventos Básicos:

    1. El caso de uso comienza cuando el programador desea borrar el atributo de una tabla
    2. El sistema da la opción de borrar atributos
    3. El programador accede a la opción.
    4. El sistema borra el atributo de la tabla.

- Post-Condiciones: El sistema debe quedar actualizado y sin el atributo borrado.

- Flujo alternativo:
    4. El atributo es referenciado por otro atributo o es clave foránea.
    5. El sistema informa al programador de la condición con un mensaje.

5.1.3.15 Caso de Uso: <Editar Atributo>

- ID: PE15
- Descripción: Permite editar un atributo para modificar sus datos.
- Pre-Condiciones: Se debe haber accedido a la opción de ver atributos.
- Flujo de Eventos Básicos:

    1. El caso de uso comienza cuando el programador desea editar un atributo de una tabla.
    2. El sistema proporciona al programador la opción de editar el atributo.
    3. El programador accede a la opción.
    4. El sistema muestra los datos del atributo.
    5. El programador cambia los datos del atributo y guarda.
    6. El sistema guarda los cambios.

  Post-Condiciones: El sistema debe quedar con el atributo editado actualizado.
- Flujo alternativo:

5. Los datos son inválidos.
6. El sistema informa al programador de los errores.
7. El programador cambia los datos.

- Flujo alternativo:
    5. El atributo editado es clave foránea de otra tabla.
    6. El sistema informa al programador de los errores.
    7. El programador declina la acción.

5.1.3.16 Caso de Uso: <Ver Claves Foráneas>

- ID: PE16
- Descripción: permite ver todas las claves foráneas de una tabla.
- Pre-Condiciones: Se debe haber creado la tabla previamente.
- Flujo de Eventos Básicos:

    1. El caso de uso comienza cuando el programador desea ver las claves foráneas de una tabla.
    2. El sistema da la opción de ver claves foráneas.
    3. El programador accede la opción.
    4. El sistema muestra las claves foráneas al programador.

- Post-Condiciones: El sistema debe quedar mostrando las claves foráneas al programador.

5.1.3.17 Caso de Uso: <Editar Clave Foránea>

- ID: PE17
- Descripción: permite editar los datos de una clave foránea.
- Pre-Condiciones: Se debe haber accedido a la opción de ver claves foráneas.
- Flujo de Eventos Básicos:

    1. El caso de uso comienza cuando el programador desea editar los datos de una clave foránea.
    2. El sistema da la opción de editar clave foránea.
    3. El programador accede la opción.
    4. El sistema muestra los datos de la clave foránea.
    5. El programador modifica los datos de la clave y guarda.
    6. El sistema guarda los cambios.

- Post-Condiciones: El sistema debe quedar con la clave editada actualizada y en la interfaz de ver claves foráneas.
- Flujo alternativo:
    6. Los datos modificados violan restricciones de tipos de datos.
    7. El sistema informa al programador de los errores.
    8. El programador cambia las referencias.

5.1.3.18 Caso de Uso: <Agregar Clave Foránea>

- ID: PE18
- Descripción: permite agregar una nueva clave foránea.
- Pre-Condiciones: Se debe haber accedido a la opción de ver claves foráneas.
- Flujo de Eventos Básicos:

    1. El caso de uso comienza cuando el programador desea agregar una nueva clave foránea.
    2. El sistema da la opción de agregar clave foránea.
    3. El programador accede la opción.
    4. El sistema muestra el formulario de nueva clave.
    5. El programador ingresa los datos y guarda.
    6. El sistema guarda los cambios.

- Post-Condiciones: El sistema debe quedar actualizado con una nueva clave foránea y en la interfaz de ver claves foráneas de la tabla.
- Flujo alternativo:
    6. Los datos modificados violan restricciones de tipos de datos.
    7. El sistema informa al programador de los errores.
    8. El programador cambia las referencias.

5.1.3.19 Caso de Uso: <Borrar Clave Foránea>

- ID: PE19
- Descripción: permite borrar una clave foránea de una tabla.
- Pre-Condiciones: Se debe haber accedido a la opción de ver claves foráneas.
- Flujo de Eventos Básicos:

    1. El caso de uso comienza cuando el programador desea borrar una clave foránea.
    2. El sistema muestra las claves foráneas de la tabla.
    3. El programador selecciona la clave foránea a eliminar y la borra.
    4. El sistema borra la clave foránea.

- Post-Condiciones: El sistema debe quedar actualizado con una nueva clave foránea y en la interfaz de ver claves foráneas de la tabla.
- Flujo alternativo:
    3. La clave foránea a borrar es referenciada por otra entidad.
    4. El sistema muestra el mensaje de error.
    5. El programador declina la acción.

5.2 Diagrama de clases de implementación

Se describe a continuación un diagrama de clases representativo de la implementación de la aplicación. En el cual se distinguen varias clases. En primer lugar definimos las clases que representan el diseño básico de una estructura de Base de Datos como son la clase BaseDeDatos, Entidad (que representa una tabla de una Base de Datos) y Atributo (que representa los campos de las tablas). Para representar la importación o conexión con una Base de Datos externa se creó una clase llamada ConexionBDExterna encargada de conectarse a otro motor de Base de Datos y obtener la estructura de esta. Para el caso de la persistencia de la Base de Datos ya sea creada o importada la clase Persistencia es la encargada de guardar la estructura de manera permanente mediante archivos XML dentro del proyecto donde se esté utilizando el plug-in. Para poder generar los códigos java de patrones como Fachada, DAO y POJO se creó la clase GeneradorDeCodigo, la cual contiene objetos Clase, Métodos, Variables, LineaCodigo, que son los encargados de generar los archivos .java dentro del proyecto.

Como se utilizó una arquitectura de implementación tipo MVC (modelo-vista-controlador), las clases anteriores pasan a ser parte del modelo. Además de ellas se implementó una clase o súper clase que interactúa con ellas y con la interfaz de usuario llamada Control, la cual es la encargada de coordinar y llamar a las clases del modelo y hacer que los resultados de las operaciones se traspasen a la vista.

Finalmente se creó una clase representativa de vista o interfaz de usuario la cual es la encargada de interactuar con el usuario de modo gráfico, de esta manera esta clase solo se comunica con la clase controladora y las clases del modelo son transparentes para ella, así si se modifica la interfaz el modelo y el controlador no sufrirán cambios mayores, y no se requerirá una restructuración total del modelo de implementación.

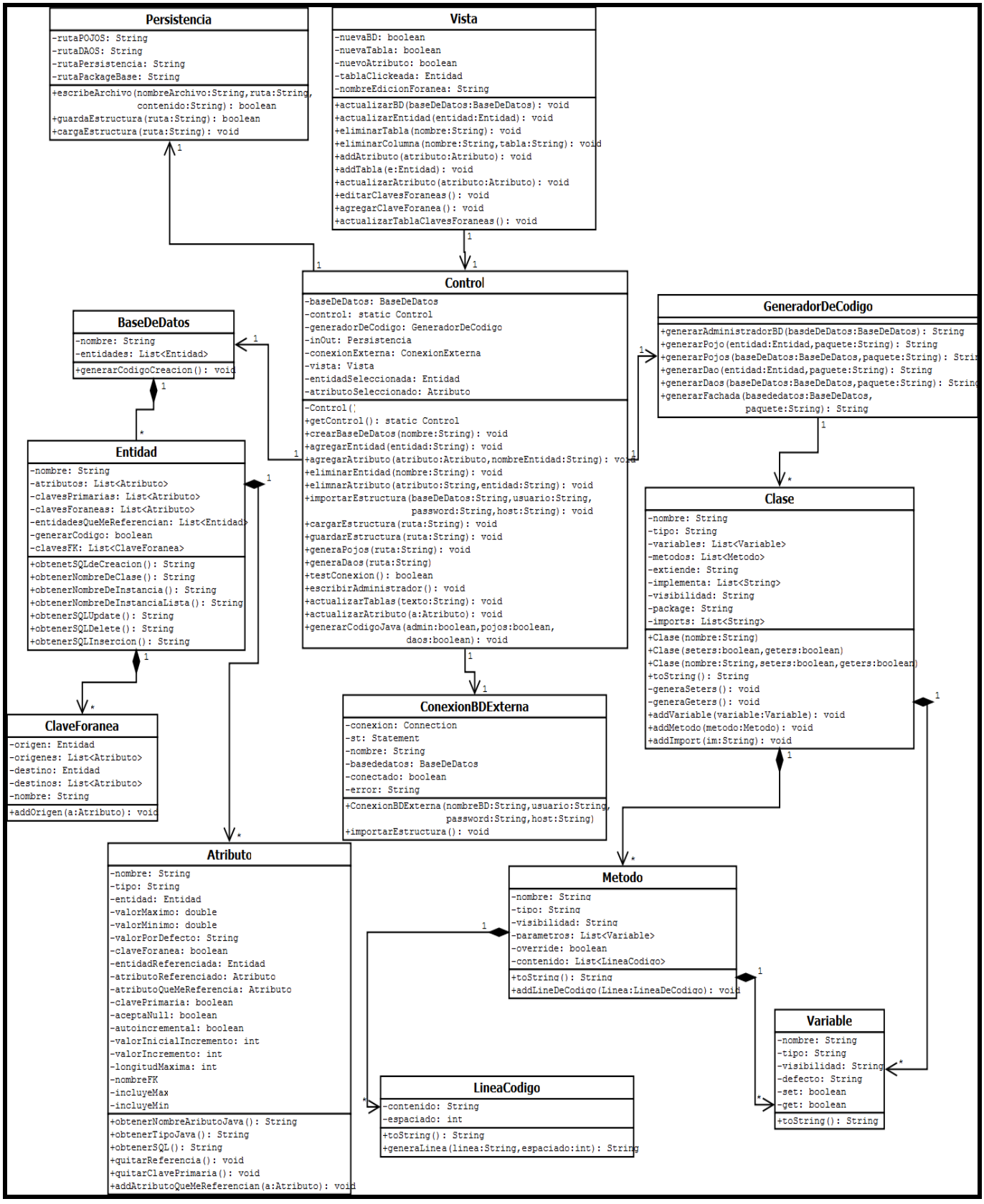


Figura Nº 4: Diagrama de Clases.

5.3 Diagramas de Secuencia

Se presentan los diagramas de secuencia para mostrar la relación entre las distintas clases u objetos. Cabe señalar que solo se describen los diagramas de los procesos más significativos de la aplicación.

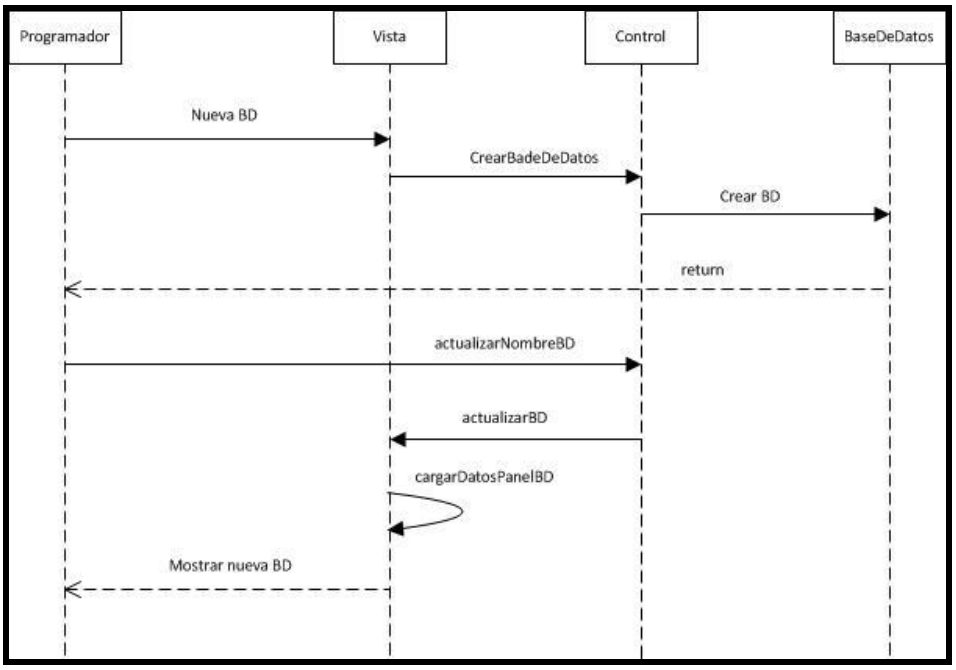


Figura Nº 5: Diagrama de Secuencia, Creación de BD.

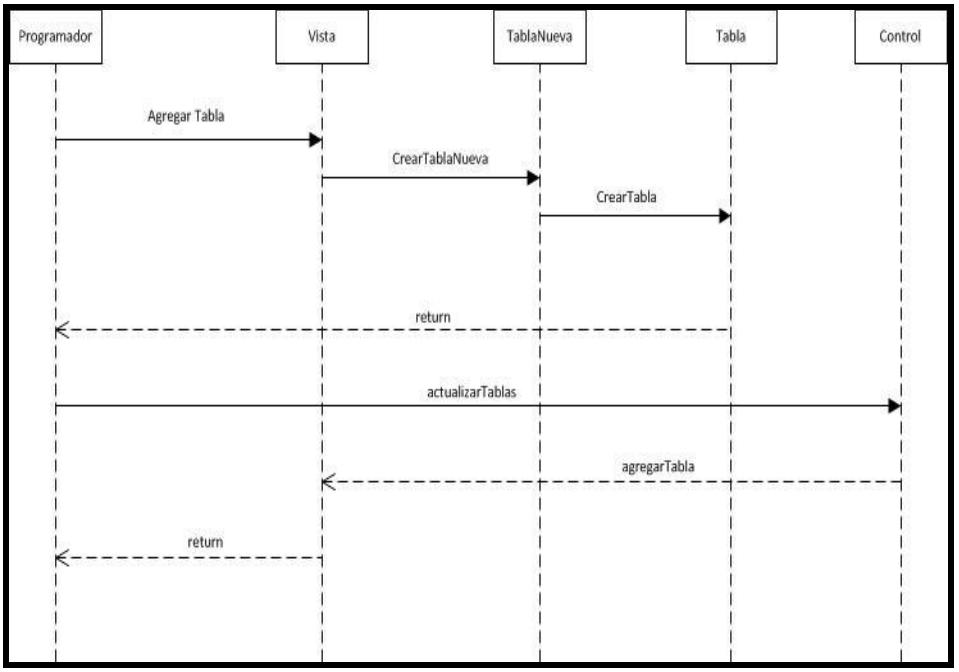


Figura Nº 6: Diagrama de Secuencia, Creación de Tabla.

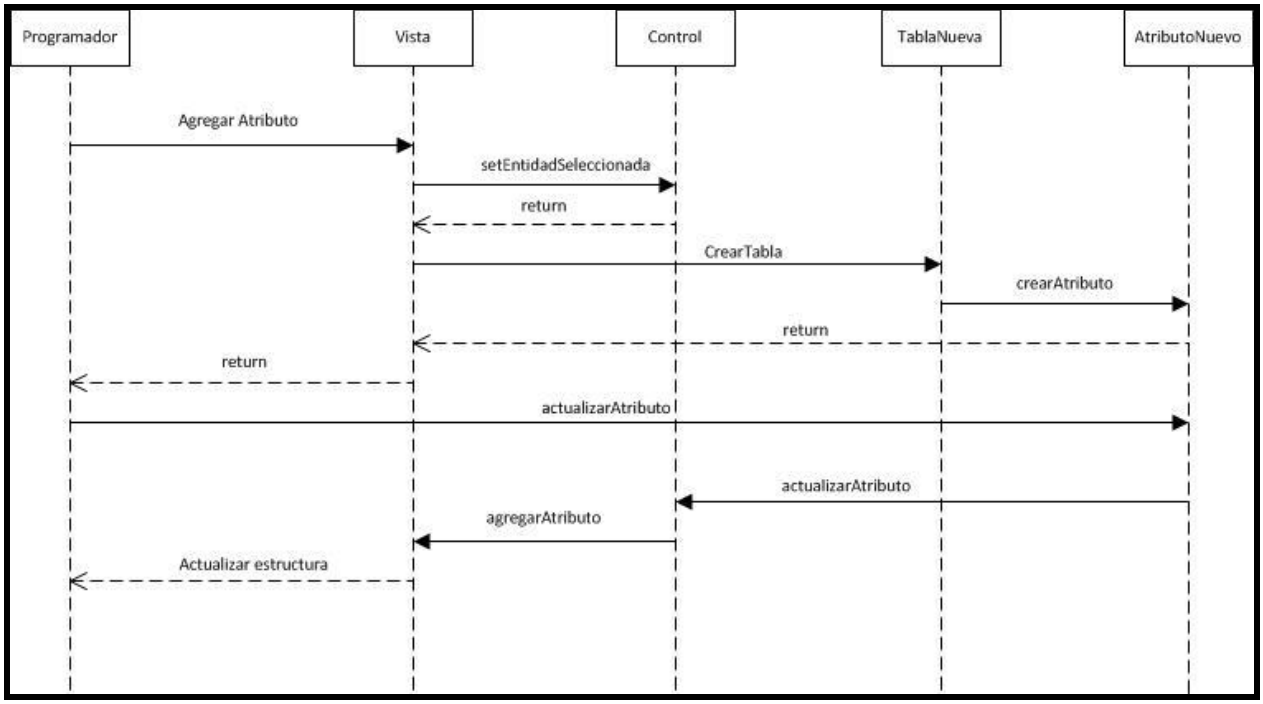


Figura Nº 7: Diagrama de Secuencia, Creación de Atributo.

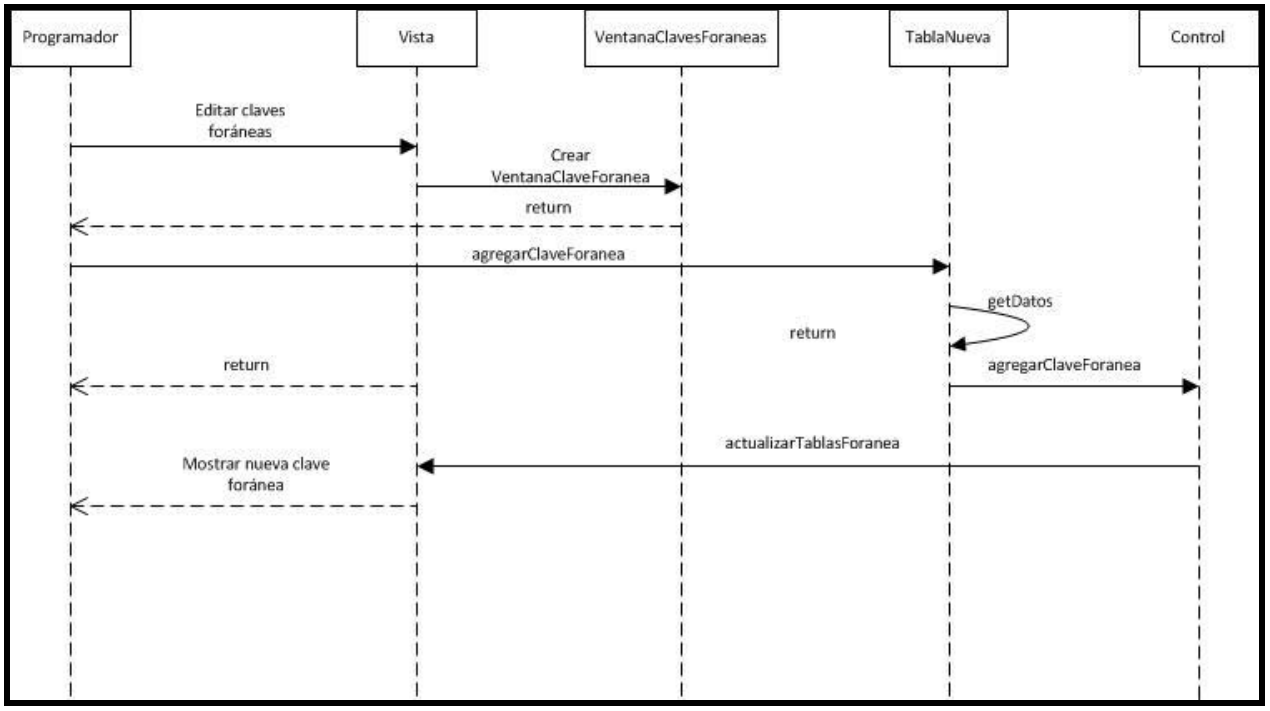
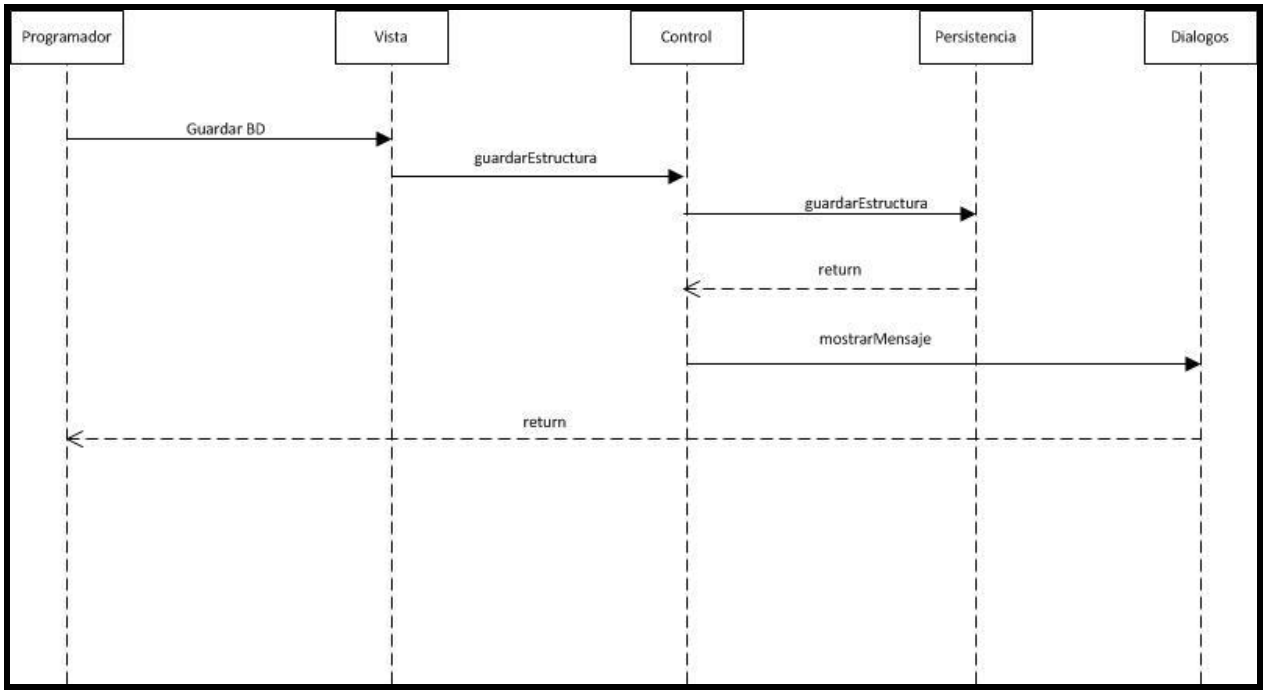


Figura Nº 8: Diagrama de Secuencia, Creación de Clave Foránea



. Figura Nº 9: Diagrama de Secuencia, Guardar BD

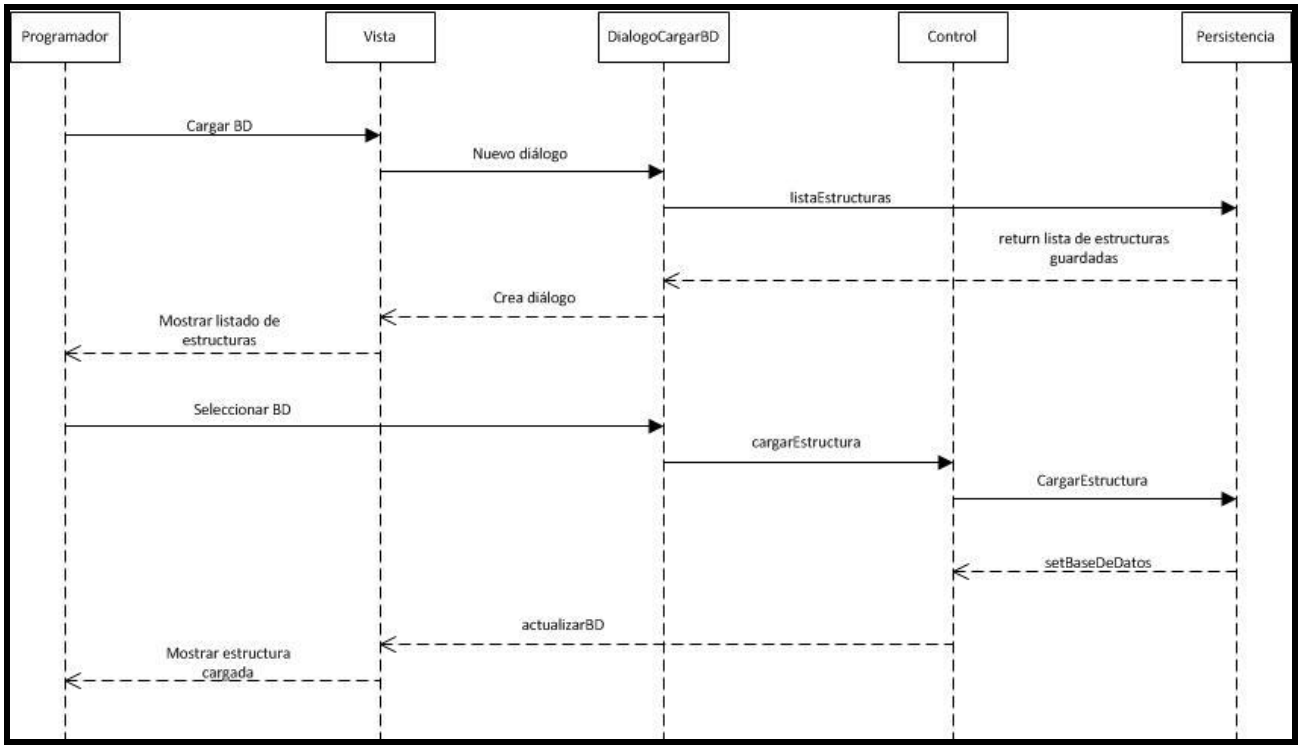


Figura Nº 10: Diagrama de Secuencia, Cargar BD

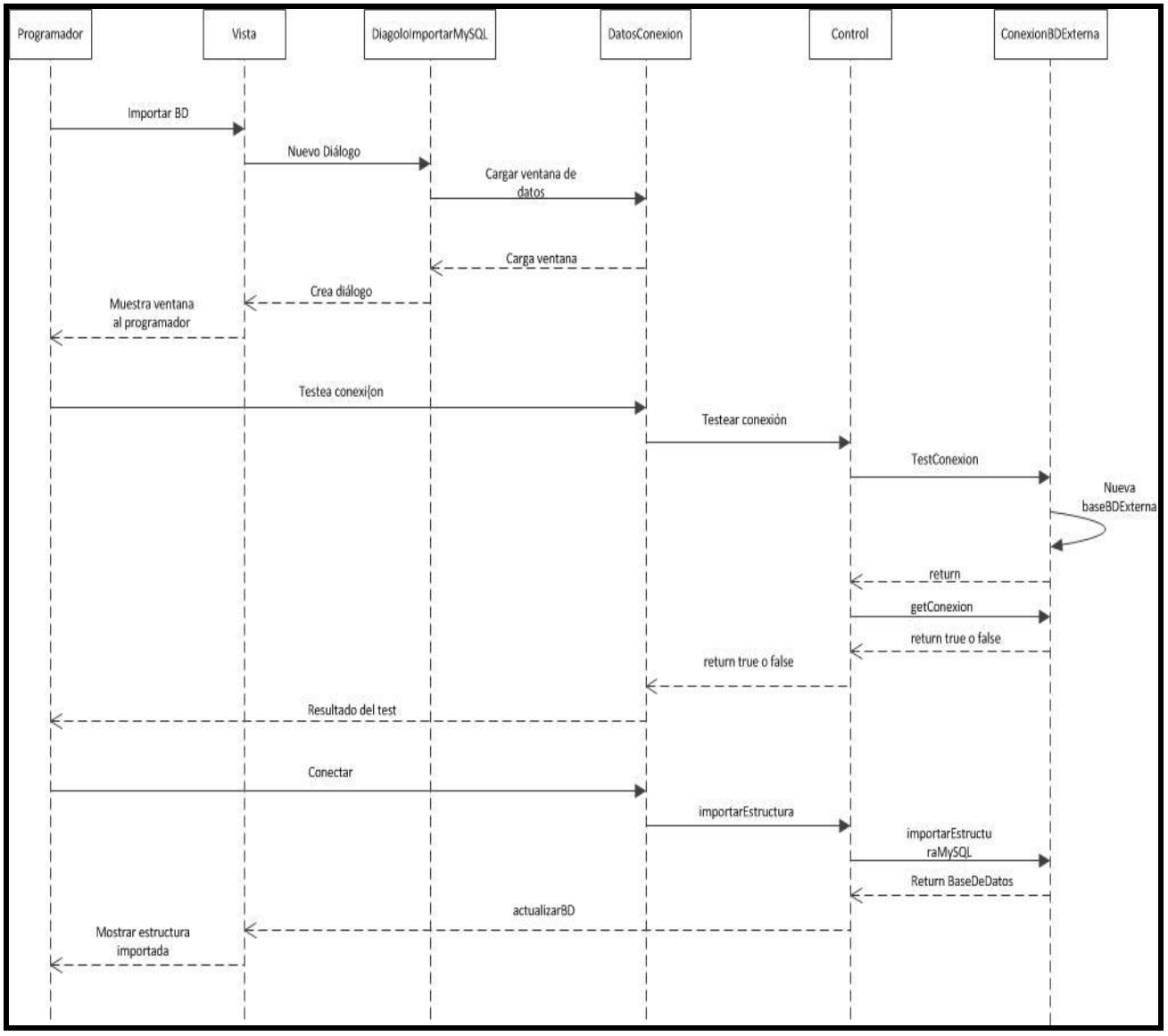


Figura Nº 11: Diagrama de Secuencia, Importar BD

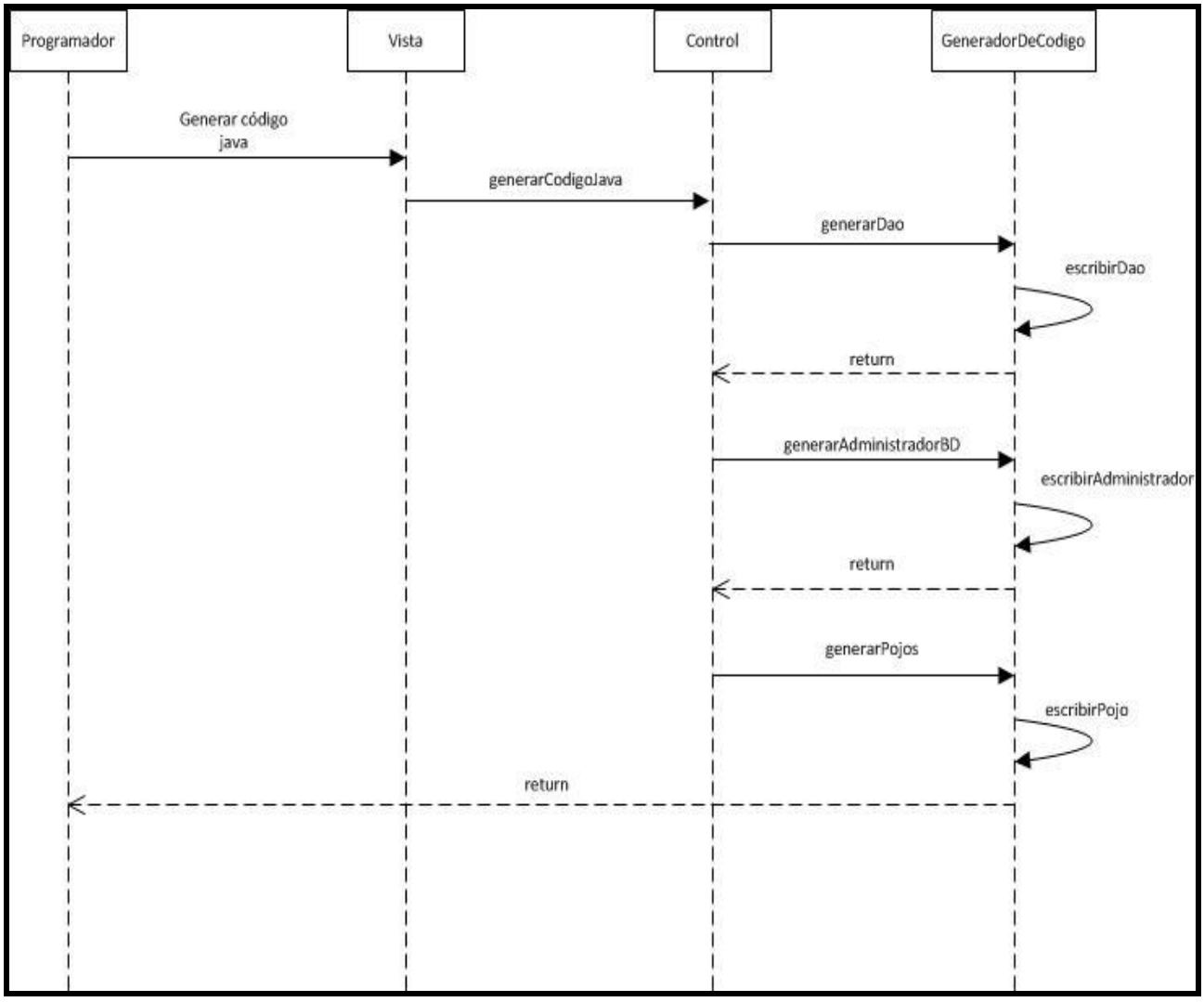


Figura Nº 12: Diagrama de Secuencia, Generar Código Java

6 DISEÑO

6.1 Diseño de Físico de la Base de Datos

En la aplicación se utilizó una estructura de almacenamiento tipo XML ya que no se trabaja con muchos datos en la persistencia, además las entradas para conformar dicho archivo siempre serán las mismas por lo que se optó por este tipo de estructura. Además es más cómodo su procesamiento al momento de recuperar la estructura y cargarla para el programador.

Para crear este archivo se utilizan los siguientes tags.

Tag	Utilización
<baseDeDatos></baseDeDatos>	Este tag encierra todo el archivo xml. Y especifica que dentro de él se contienen los datos de la BD (tablas, atributo, claves, etc).
<nombreBD></nombreBD>	Este tag se utiliza para especificar el nombre de la BD. Dentro de él solo hay un string que representa a este.
<tablas></tablas>	Este tag se utiliza para especificar que lo que está dentro son tablas pueden haber una o varias tablas dentro de este tag.
<tabla></tabla>	Este tag se utiliza para encerrar los datos de una sola tabla. Este se utiliza siempre dentro del tag tablas.
<nombreTabla></nombreTabla>	En este tag se especifica el nombre de la tabla.
<columnas></columnas>	Este tag sirve para especificar los atributos de una tabla. Este puede contener uno o varios atributos en su interior.
<nombreColumna></nombreColumna>	Este tag sirve para especificar el nombre de una columna.
<tipo></tipo>	Este tag se utiliza para especificar el tipo de dato del atributo (integer, text, real).
<valorMaximo></valorMaximo>	Este tag se utiliza para especificar el valor máximo que puede tomar el atributo.

	Es representado con un número.
<valorMinimo></valorMinimo>	Este tag se utiliza para especificar el valor mínimo que puede tomar un atributo. Es representado con un número.
<valorPorDefecto></valorPorDefecto>	En este tag se especifica el valor por defecto. Este debe ser un texto si el atributo es text o numérico si el atributo es integer o real. Es decir, debe existir una correspondencia entre estos datos.
<clavePrimaria></clavePrimaria>	En este tag se especifica si el atributo es o no clave primaria, esto se hace con true o false dentro de este tag.
<aceptaNull></aceptaNull>	En este tag se especifica si el atributo acepta valores nulos. Esto se hace con los valores true o false dentro de este tag.
<autoIncremental></autoIncremental>	En este tag se especifica si el atributo es autoincremental. Esto se hace con los valores true o false dentro del tag.
<valorInicialIncremento></valorInicialIncremento>	En este tag se especifica el valor inicial del incremento, es decir el valor numérico con el cual se comenzara a incrementar el atributo.
<longitudMaxima></longitudMaxima>	En este tag se especifica la longitud máxima del atributo, esto quiere decir los caracteres o cantidad de números que puede poseer el valor de este atributo.
<incluyeMaximo></incluyeMaximo>	En este tag se especifica el valor máximo del atributo. Este tag encierra un valor numérico.
<incluyeMinimo></incluyeMinimo>	En este tag se especifica el valor mínimo del atributo. Este tag encierra un valor numérico.
<clavesForaneas></clavesForaneas>	En este tag es el que encierra la especificación de las claves

	foráneas de la BD. Puede contener una o varias claves foráneas.
<claveForanea></claveForanea>	Este tag se utiliza para especificar una clave foránea de la BD.
<nombreFK></nombreFK>	Este tag especifica el nombre de la clave foránea.
<tablaOrigen></tablaOrigen>	Este tag especifica el nombre de la tabla de origen de la clave foránea.
<columnaOrigen></columnaOrigen>	Este tag especifica el nombre del atributo de origen de la clave foránea.
<tablaDestino></tablaDestino>	Este tag especifica el nombre de la tabla de destino a la cual apunta la clave foránea.
<columnaDestino></columnaDestino>	Este tag especifica el nombre del atributo al cual se está referenciando.

Tabla Nº 5: Tags del archivo xml de persistencia.

Se especifica a continuación como sería la anidación de los tag en el archivo xml.

```
<baseDeDatos>
        <nombreBD></nombreBD>
        <tablas>
          <tabla>
                <nombreTabla>…</nombreTabla>
                    <columnas>
                    <nombreColumna>…</nombreColumna>
                    <tipo>…</tipo>
                    <valorMaximo>…</valorMaximo>
                    <valorMinimo>…</valorMinimo>
                    <valorPorDefecto>…</valorPorDefecto>
                    <clavePrimaria>…</clavePrimaria>
                    <aceptaNull>…</aceptaNull>
                    <autoIncremental>…</autoIncremental>
                    <valorInicialIncremento>…</valorInicialIncremento>
                    <longitudMaxima>…</longitudMaxima>
                    <incluyeMaximo>…</incluyeMaximo>
                    <incluyeMinimo>…</incluyeMinimo>
                    </columnas>

                    <columnas>
                    .
                    .
                    .
                    </columnas>
```

```
        </tabla>
        <tabla>
                .
                .
                .
        </tabla>
      </tablas>
      <clavesForaneas>
        <claveForanea>
                <nombreFK>…</nombreFK>
                <tablaOrigen>…</tablaOrigen>
                <columnaOrigen>…</columnaOrigen>
                <tablaDestino>…</tablaDestino>
                <columnaDestino>…</columnaDestino>
        </claveForanea>

        <claveForanea>
                .
                .
                .
        </claveForanea>
  </clavesForaneas>
</baseDeDatos>
```

Los archivos generados por el plug-in son almacenados en una carpeta dentro de la ruta donde se encuentra eclipse. Por lo que se pueden encontrar fácilmente y revisar su contenido.

6.2 Diseño de arquitectura funcional

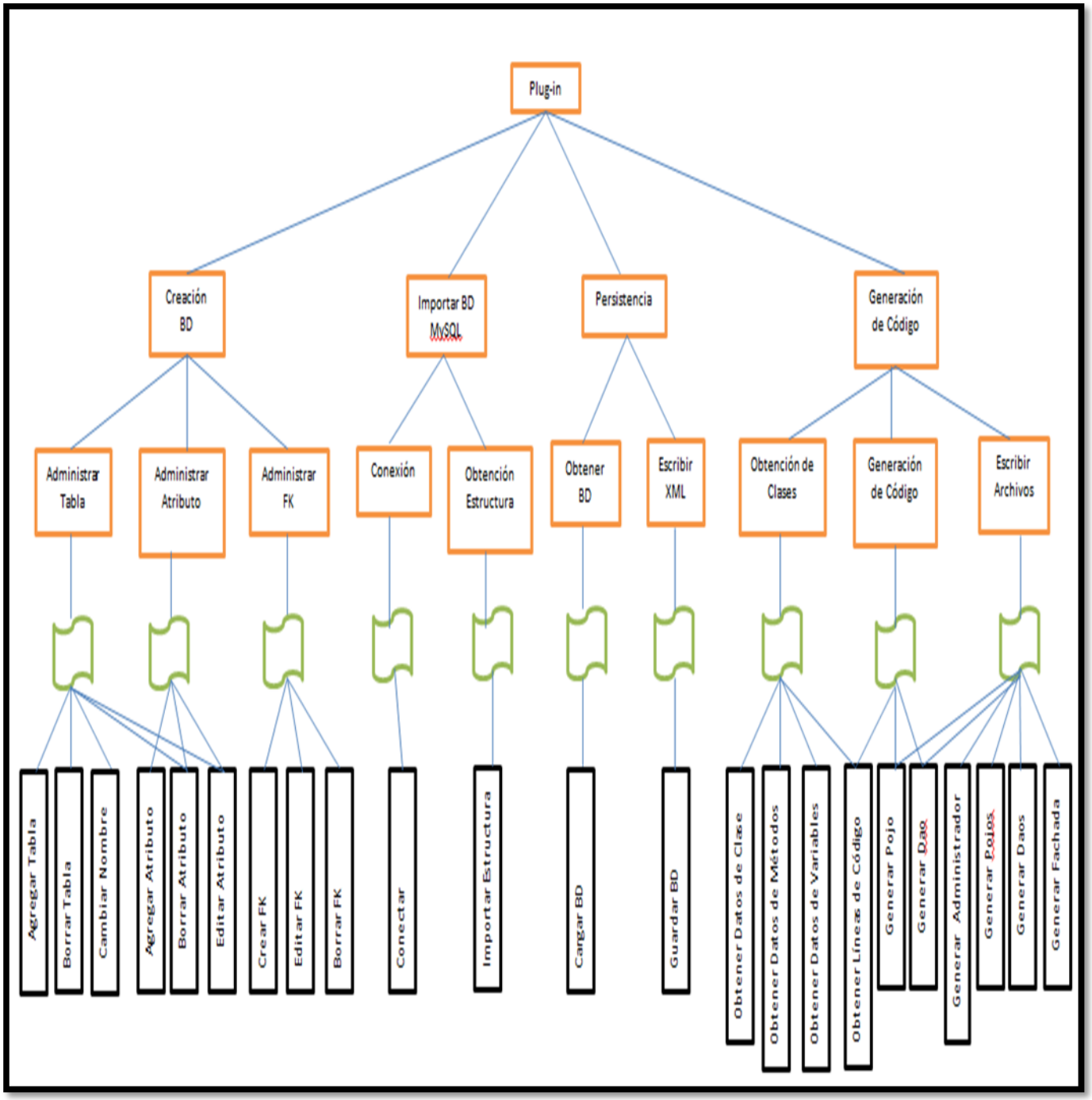


Figura Nº 13: Diseño de arquitectura funcional.

6.3 Diseño interfaz y navegación

6.3.1 Esquema especificación de interfaz

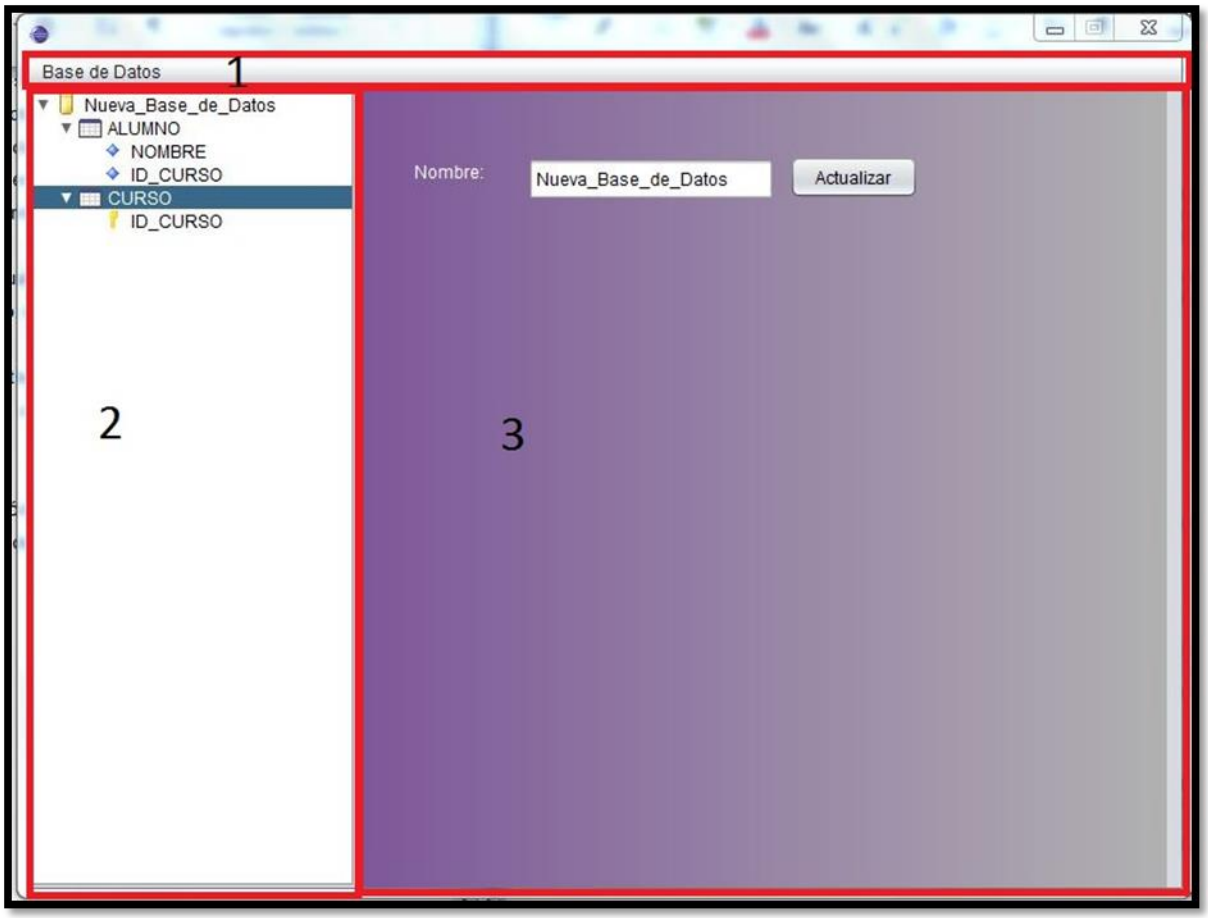


Figura Nº 14: Vista Principal.

Área 1 de la figura N° 14: Desplegar Menú. Incluye ítems como:

- *"Nueva BD" donde el usuario podrá crear una nueva Base de Datos.*
- *"Cargar BD"* donde el programador podrá cargar en la ventana un XML de una Base de Datos que se haya creado con anterioridad.
- *"Importar MySQL"* donde el programador podrá acceder a un panel que le pida los datos de conexión a una Base de Datos MySQL y posteriormente importarla a la ventana para su edición o utilización en el proyecto.
- *"Guardar"* donde el programador podrá guardar la estructura ya sea creada o importada como un XML persistente dentro del proyecto para su posterior uso en la aplicación.

- *"Agregar Tabla"* Donde el programador podrá agregar nuevas tablas a la Base de Datos.
- *"Borrar Todas las Tablas"* donde el usuario podrá acceder a borrar todas las tablas de la Base de Datos y dejarla vacía.
- *"Generar código java"* Donde el programador podrá generar los patrones en archivos con extensión java dentro del proyecto, de la Base de Datos creada.
- "Acerca de…" donde el usuario podrá ver información de la aplicación, versión, autores, etc.

Área 2 de la figura N° 14: área de árbol de estructura de Base de Datos, donde se puede tener acceso a diferentes opciones en el menú del botón derecho del mouse dependiendo de dónde se escoja:

Sobre una Base de Datos: se tendrá acceso a las mismas opciones que en la barra de menú, con la excepción de "generar código java" y "acerca de…".

Sobre una Tabla:

- "Agregar Atributo" que permite agregar un atributo a la tabla seleccionada.
- "Borrar Tabla" que permite borrar la tabla seleccionada de la Base de Datos.
- "Borrar todos los Atributo" que permite borrar todos los atributos de la tabla seleccionada.
- "Editar Claves Foráneas" permite ver las claves foráneas de la tabla, además de editarlas, borrarlas o agregar una nueva.

Sobre un atributo:

- "Borrar Columna" permite borrar el atributo seleccionado.

Además existen accesos de doble click sobre los distintos elementos del árbol:

- Sobre una Base de Datos: permite editar el nombre de la Base de Datos y actualizarlo.
- Sobre una tabla: permite editar el nombre de la tabla y actualizarlo. Además permite ver los atributos que posee dicha tabla.
- Sobre un atributo: permite editar los datos de un atributo y guardar los cambios.

Área 3 de la figura N° 14: Área de paneles, en él se cargarán todos los paneles u opciones de la aplicación.

- El color de las ventanas será gris manteniendo el todo de eclipse, y el área de paneles será morado en degradación de izquierda a derecha, con una combinación de dos colores que en RGB son:

Color 1: (181, 126, 220).

Color 2: (255, 255, 255).

- La Letra será de color blanco RGB (240,240,240), tipo Tahoma.
- Paneles que se cargara:

  - Figura N° 15, Panel de nueva BD: contiene un campo donde ingresar el nombre de la BD y un botón para guardar el nombre.

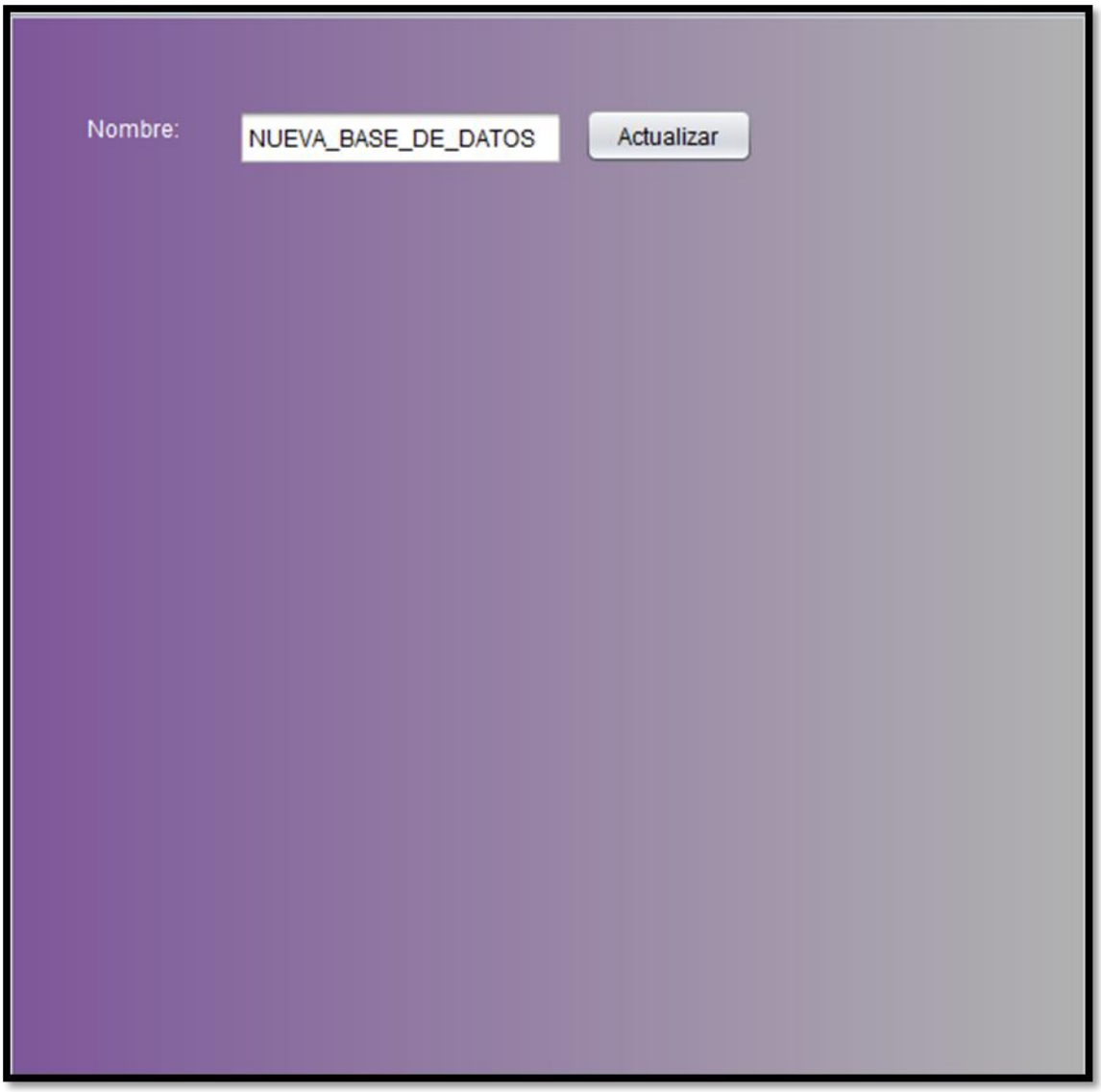


Figura Nº 15: Panel nueva BD.

  - Figura N° 16, Panel nueva tabla: contiene un campo para el nombre de la tabla, un botón guardar y una lista para mostrar los atributos de la tabla.

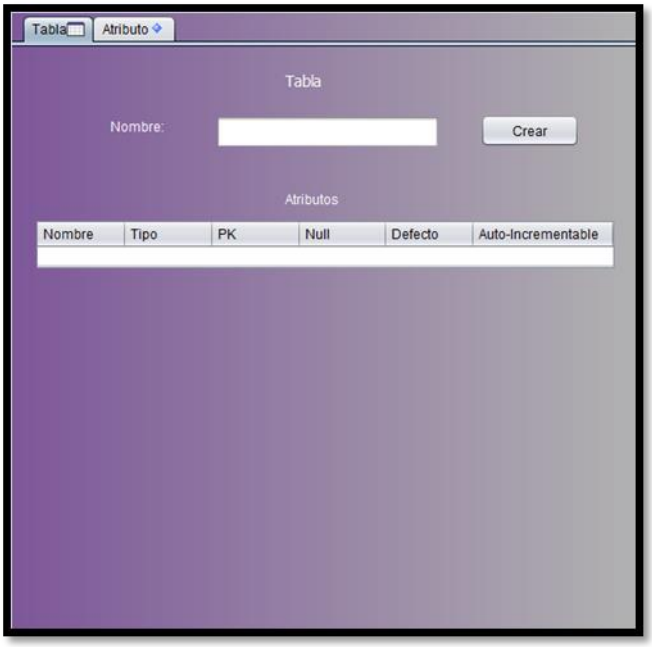


Figura Nº 16: Panel nueva tabla.

- Figura N° 17, Panel nuevo atributo: contiene campos para ingresar los datos del atributo como el nombre, el tipo, si es clave primaria, etc. y un botón para guardar el atributo.

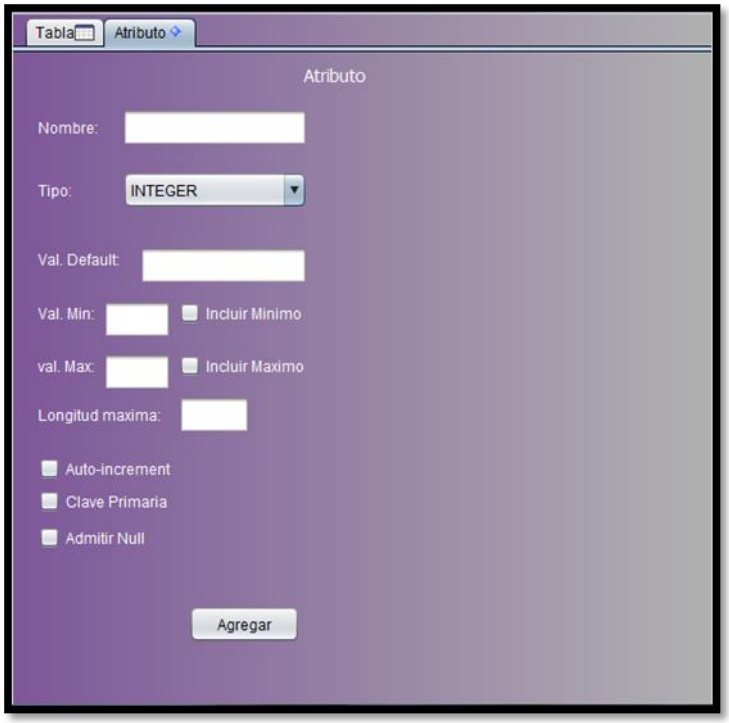


Figura Nº 17: Panel nuevo atributo.

- Figura N° 18, Panel de edición de claves foráneas: contiene un campo no editable que muestra el nombre de la tabla, una lista seleccionable que muestra los atributos foráneos de la tabla y botones para las acciones de agregar una nueva FK, borrar una FK y editar una FK.

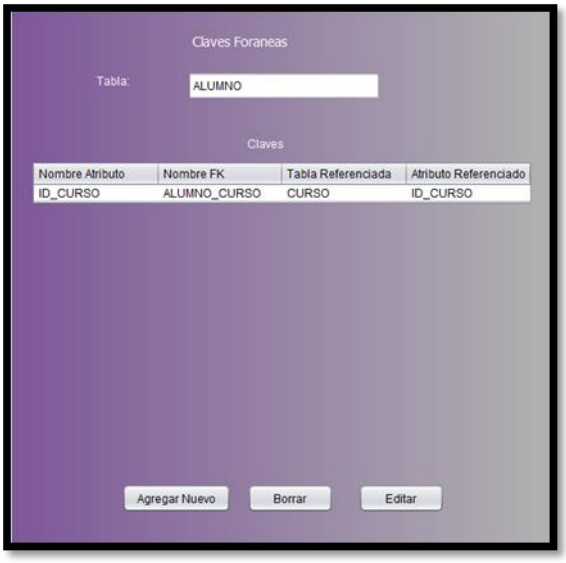


Figura Nº 18: Panel edición de claves foráneas.

- Figura N° 19, Panel nueva clave foránea: contiene los campos necesarios para la creación de una nueva clave foránea, esto es campo para el nombre y campos de selección para escoger las referencias y un botón para guardar el atributo foráneo.

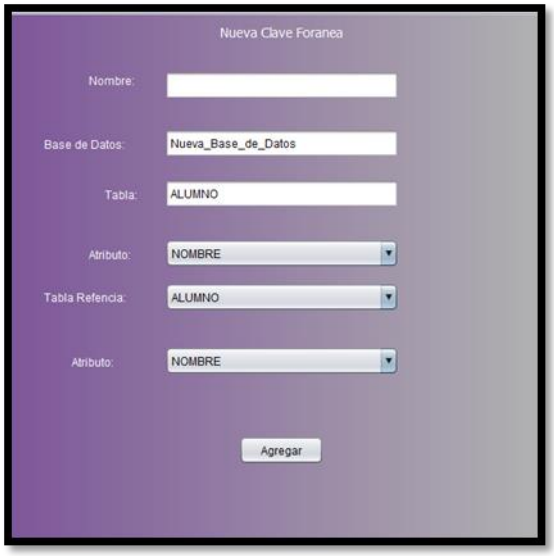


Figura Nº 19: Panel edición de claves foráneas.

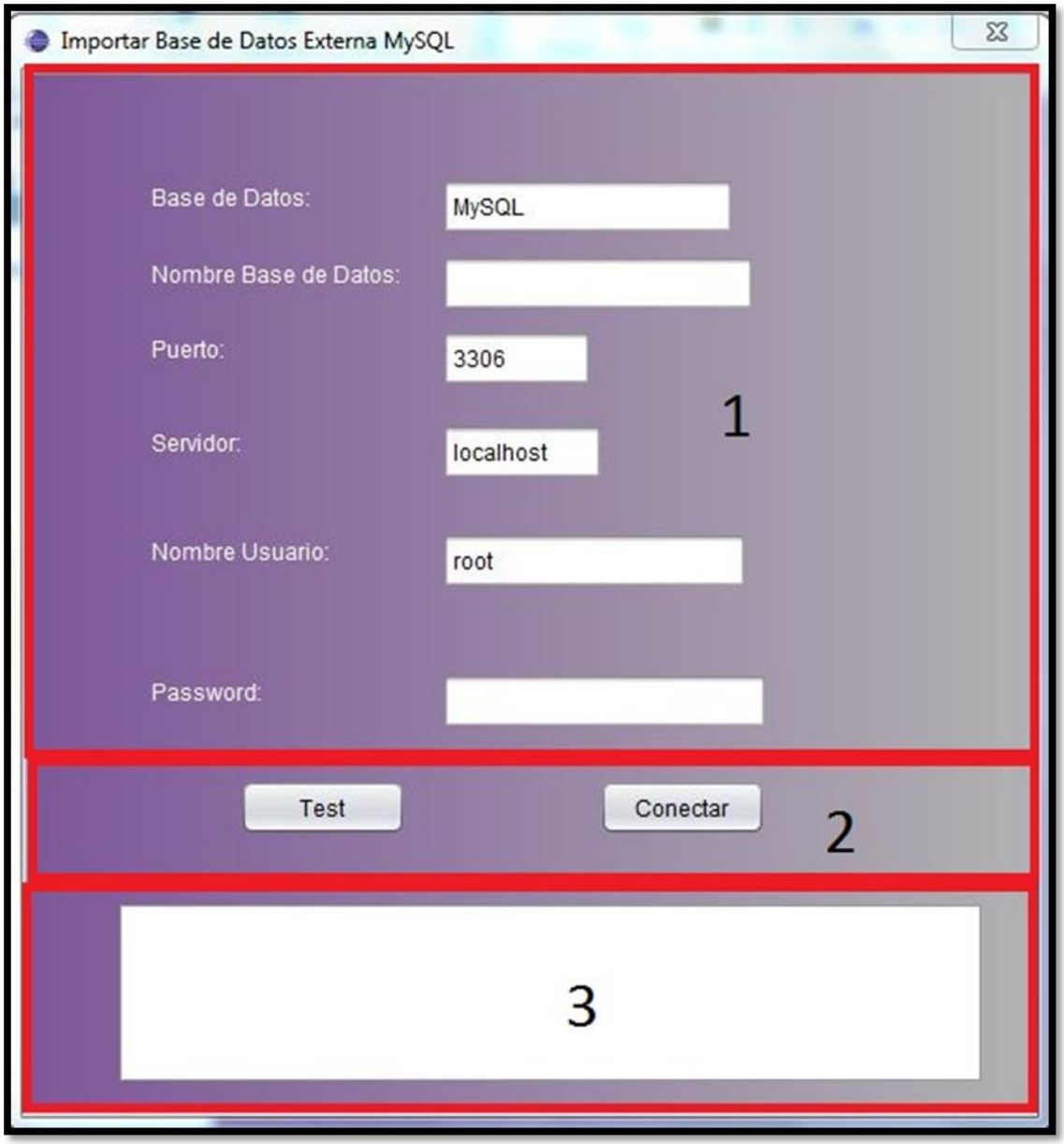


Figura Nº 20: Ventana de conexión a BD externa.

Área 1 de la figura N° 20: formulario de datos de conexión.

Área 2 de la figura N° 20: botones de testeo de conexión y conectar.

Área 3 de la figura N° 20: área de notificación.

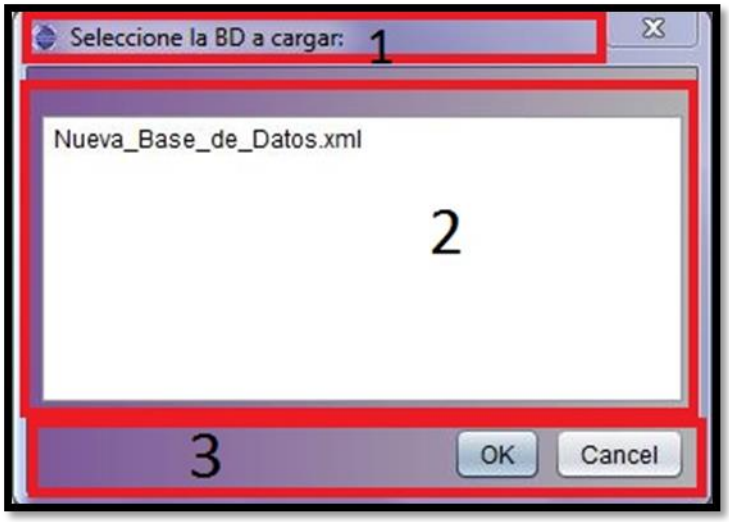


Figura Nº 21: Ventana para cargar un BD.

Área 1 de la figura N° 21: título de la ventana e instrucción.

Área 2 de la figura N° 21: lista de Base de Datos guardados.

Área 3 de la figura N° 21: botones de carga y cancelación.

6.3.2 Esquema de Navegación

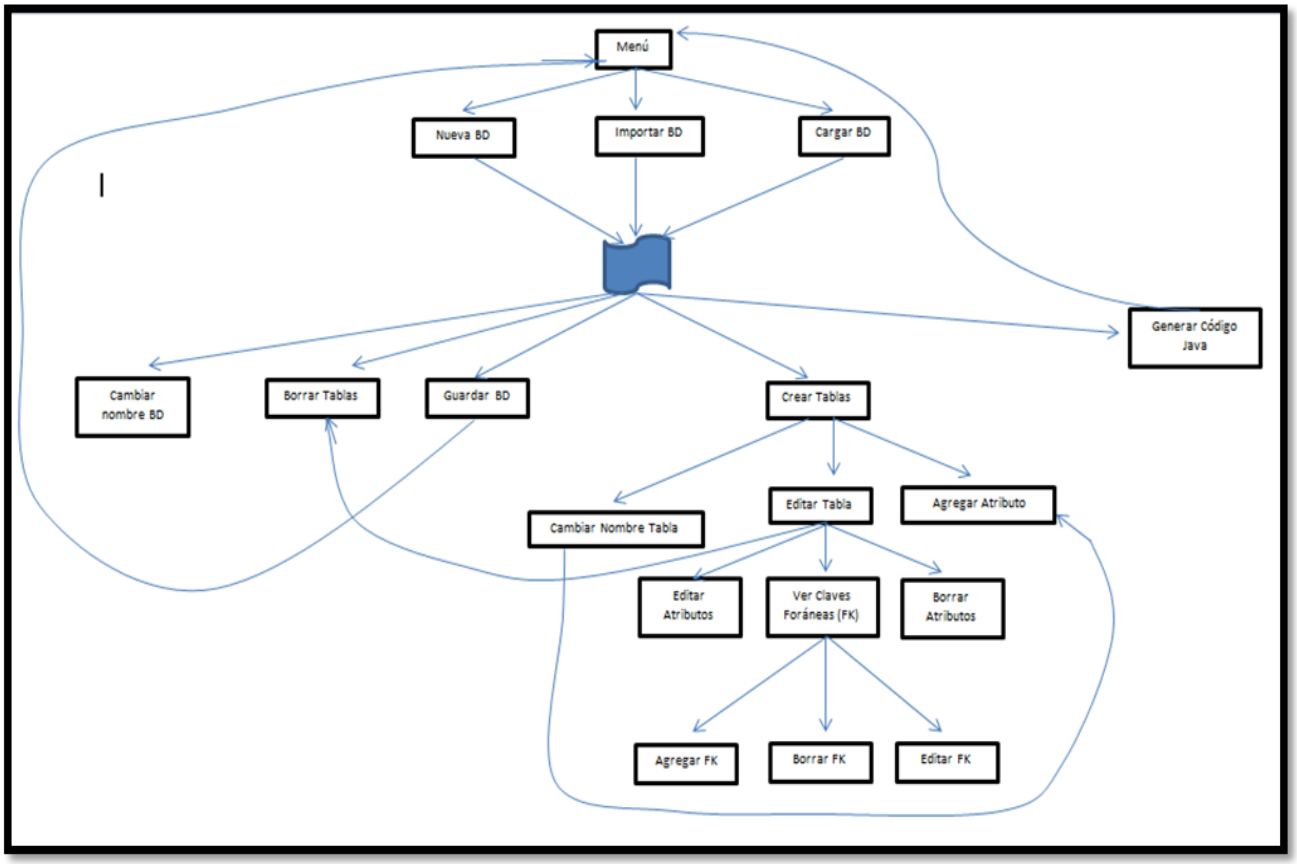


Figura Nº 22: Esquema de navegación.

6.3.3 Especificación de menú

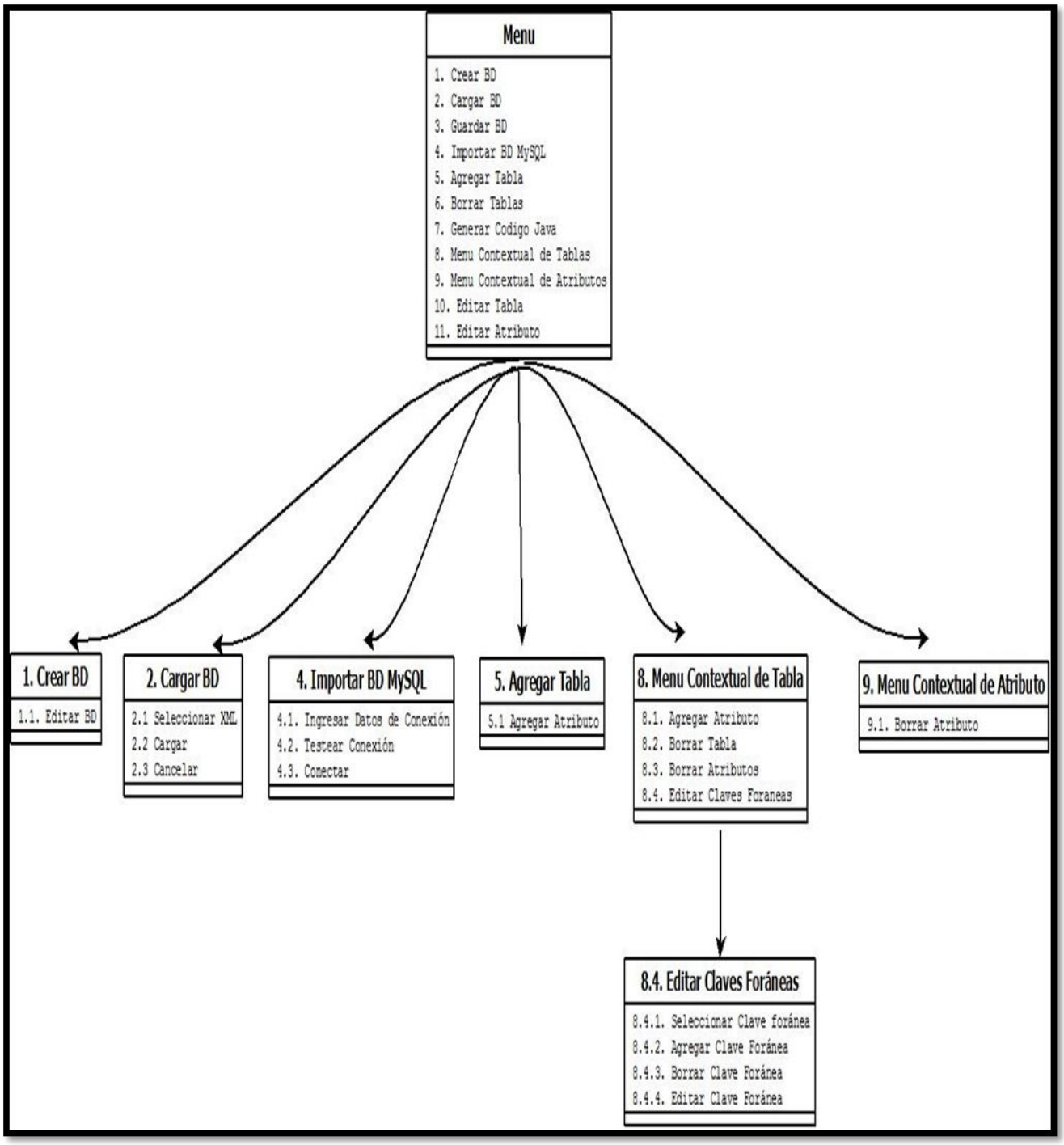


Figura Nº 23: Esquema de menú.

6.4 Especificación de módulos

Nombre Módulo: Agregar Tabla			
Parámetros de entrada		**Parámetros de Salida**	
Nombre:	**Tipo de dato:**	**Nombre:**	**Tipo de dato:**
nombreTabla	String	Tabla	Entidad

Tabla Nº 6: Especificación de módulo agregar tabla.

Nombre Módulo: Borrar Tabla			
Parámetros de entrada		**Parámetros de Salida**	
Nombre:	**Tipo de dato:**	**Nombre:**	**Tipo de dato:**
nombreTabla	String		

Tabla Nº 7: Especificación de módulo borrar tabla.

Nombre Módulo: Cambiar Nombre Tabla			
Parámetros de entrada		**Parámetros de Salida**	
Nombre:	**Tipo de dato:**	**Nombre:**	**Tipo de dato:**
nuevoNombre	String	Tabla	Entidad

Tabla Nº 8: Especificación de módulo cambiar nombre tabla.

Nombre Módulo: Agregar Atributo			
Parámetros de entrada		**Parámetros de Salida**	
Nombre:	**Tipo de dato:**	**Nombre:**	**Tipo de dato:**
nombre	String	atributo	Atributo
tabla	Entidad		
Tipo	String		
valorDefecto	Int		
valorMinimo	Int		
valorMaximo	Int		
longitudMaxima	Int		
Auto-increment	boolean		
Clave primaria	boolean		
Admitir null	boolean		

Tabla Nº 9: Especificación de módulo agregar atributo.

Nombre Módulo: Editar Atributo			
Parámetros de entrada		**Parámetros de Salida**	
Nombre:	**Tipo de dato:**	**Nombre:**	**Tipo de dato:**
nombre	String	atributo	Atributo
tabla	Entidad		
Tipo	String		
valorDefecto	Int		
valorMinimo	Int		
valorMaximo	Int		
longitudMaxima	Int		
Auto-increment	boolean		
Clave primaria	boolean		
Admitir null	boolean		

Tabla Nº 10: Especificación de módulo editar atributo.

Nombre Módulo: Borrar Atributo			
Parámetros de entrada		**Parámetros de Salida**	
Nombre:	**Tipo de dato:**	**Nombre:**	**Tipo de dato:**
nombreAtributo	String	tabla	Entidad
nombreTabla	String		

Tabla Nº 11: Especificación de módulo borrar atributo.

Nombre Módulo: Agregar FK			
Parámetros de entrada		**Parámetros de Salida**	
Nombre:	**Tipo de dato:**	**Nombre:**	**Tipo de dato:**
nombre	String	atributo	Atributo
tabla	String		
atributo	String		
tablaReferenciada	String		
atributoReferenciado	String		

Tabla Nº 12: Especificación de módulo agregar FK.

Nombre Módulo:Editar FK			
Parámetros de entrada		**Parámetros de Salida**	
Nombre:	**Tipo de dato:**	**Nombre:**	**Tipo de dato:**
nombre	String	atributo	Atributo
tabla	String		
atributo	String		
tablaReferenciada	String		
atributoReferenciado	String		

Tabla Nº 13: Especificación de módulo editar FK.

Nombre Módulo: Borrar FK			
Parámetros de entrada		**Parámetros de Salida**	
Nombre:	**Tipo de dato:**	**Nombre:**	**Tipo de dato:**
tabla	String	atributo	Atributo
atributo	String		

Tabla Nº 14: Especificación de módulo borrar FK.

Nombre Módulo: Conectar			
Parámetros de entrada		**Parámetros de Salida**	
Nombre:	**Tipo de dato:**	**Nombre:**	**Tipo de dato:**
Nombre BD	String	conexión	Connection
puerto	int		
servidor	localhost		
nombreUsuario	String		
password	String		

Tabla Nº 15: Especificación de módulo conectar.

Nombre Módulo: Importar Estructura			
Parámetros de entrada		**Parámetros de Salida**	
Nombre:	**Tipo de dato:**	**Nombre:**	**Tipo de dato:**
Nombre BD	String	basededatos	BaseDeDatos

Tabla Nº 16: Especificación de módulo importar estructura.

Nombre Módulo: Cargar BD			
Parámetros de entrada		**Parámetros de Salida**	
Nombre:	**Tipo de dato:**	**Nombre:**	**Tipo de dato:**
ruta	String	basededatos	BaseDeDatos
nombreFichero	String		

Tabla Nº 17: Especificación de módulo cargar BD.

Nombre Módulo: Guardar BD			
Parámetros de entrada		**Parámetros de Salida**	
Nombre:	**Tipo de dato:**	**Nombre:**	**Tipo de dato:**
Ruta	String	Archivo que representa la BD	XML
basededatos			

Tabla Nº 18: Especificación de módulo guardar BD.

Nombre Módulo: Obtener Datos de Clase			
Parámetros de entrada		**Parámetros de Salida**	
Nombre:	**Tipo de dato:**	**Nombre:**	**Tipo de dato:**
nombreClase	String	toString	String

Tabla Nº 19: Especificación de módulo obtener datos de clase.

Nombre Módulo: Obtener Datos de Métodos			
Parámetros de entrada		**Parámetros de Salida**	
Nombre:	**Tipo de dato:**	**Nombre:**	**Tipo de dato:**
nombreMetodos	String	toString	String

Tabla Nº 20: Especificación de módulo obtener datos de métodos.

Nombre Módulo: Obtener Datos de Variables			
Parámetros de entrada		**Parámetros de Salida**	
Nombre:	**Tipo de dato:**	**Nombre:**	**Tipo de dato:**
nombreVariablea	String	toString	String

Tabla Nº 21: Especificación de módulo obtener datos de variables.

Nombre Módulo: Obtener Líneas de Código			
Parámetros de entrada		**Parámetros de Salida**	
Nombre:	**Tipo de dato:**	**Nombre:**	**Tipo de dato:**
contenido	String	lineaDeCodigo	String
espaciado	Int		

Tabla Nº 22: Especificación de módulo obtener líneas de código.

Nombre Módulo: Generar Pojo			
Parámetros de entrada		**Parámetros de Salida**	
Nombre:	**Tipo de dato:**	**Nombre:**	**Tipo de dato:**
nombreClase	String	pojo	String
paquete	String		
clavesFK	List<ClaveForanea>		
atributos	List<Atributo>		
entidadesQueMeReferencian	List<Entidad>		

Tabla Nº 23: Especificación de módulo generar pojo.

Nombre Módulo: Generar DAO			
Parámetros de entrada		**Parámetros de Salida**	
Nombre:	**Tipo de dato:**	**Nombre:**	**Tipo de dato:**
nombreDeClase	String	DAO	String
nombre	String		
nombreDeInstancia	String		
SQLInsercion	String		
SQLUpdate	String		
SQLDelete	String		
atributos	List<Atributo>		
entidadesQueMeReferencian	List<Entidad>		

Tabla Nº 24: Especificación de módulo generar DAO.

Nombre Módulo: Generar Administrador			
Parámetros de entrada		**Parámetros de Salida**	
Nombre:	**Tipo de dato:**	**Nombre:**	**Tipo de dato:**
tablas	List<Entidad>	AdminSQLOpenHelper	Archivo Java
nombreTabla	String		
SQLDeCreacion	String		
clavesFK	List<ClaveForanea>		
atributos	List<Atributo>		
nombreFK	String		
origenFK	String		
destinoFK	String		
origenesFK	List<Atributo>		
destinosFK	List<Atributo>		

Tabla Nº 25: Especificación de módulo generar administrador.

Nombre Módulo: Generar Pojos			
Parámetros de entrada		**Parámetros de Salida**	
Nombre:	**Tipo de dato:**	**Nombre:**	**Tipo de dato:**
paquete	String	Pojos	Archivos Java
tablas	List<Entidad>		
nombreDeClase	String		
nombreTabla	String		

Tabla Nº 26: Especificación de módulo generar Pojos.

Nombre Módulo: Generar Daos			
Parámetros de entrada		**Parámetros de Salida**	
Nombre:	**Tipo de dato:**	**Nombre:**	**Tipo de dato:**
paquete	String	Daos	Archivos Java
tablas	List<Entidad>		
nombreDeClase	String		

Tabla Nº 27: Especificación de módulo generar Daos.

Nombre Módulo: Generar Fachada			
Parámetros de entrada		**Parámetros de Salida**	
Nombre:	**Tipo de dato:**	**Nombre:**	**Tipo de dato:**
paquete	String	Fachada	Archivo Java
tablas	List<Entidad>		
nombreDeClase	String		
nombreTabla	String		
nombreDeInstancia	String		
nombreDeInstanciaLista	String		

Tabla Nº 28: Especificación de módulo generar fachada.

7 PRUEBAS

7.1 Elementos de prueba

A continuación se detallan los elementos de pruebas, en los cuales representan tanto pruebas de unidad, como de integración y sistema, evaluando con cada una los requisitos funcionales del software.

Verificar la creación de una nueva BD: se debe probar la creación de una nueva Base de Datos, la cual solo acepte como nombre caracteres del alfabeto.

Verificar la creación de una tabla: se debe probar la creación de una tabla la cual debe ser creada dentro de una BD y la cual acepte por nombre caracteres válidos del alfabeto.

Verificar la creación de un atributo: se debe probar la creación de un atributo, el cual debe crearse dentro de la tabla que se seleccione, con validaciones de datos y además que se cargue correctamente en la vista.

Verificación de eliminación de un atributo: se debe probar la correcta eliminación de un atributo, lo cual debe eliminarlo de la estructura de datos y además eliminarlo de la vista según corresponda. Además debe validar que si esta es una clave foránea no se elimine.

Verificar la edición de un atributo: se debe probar que al editar un atributo, los datos sean cargados correctamente en el set de edición, siendo estos los del atributo seleccionado para la edición, los nuevos datos ingresados sean validados, al igual que no permitir modificaciones que alteren la integridad de la BD, actualizar correctamente los datos del atributo y actualizarlos en la tabla que corresponda y en el atributo seleccionado.

Verificar la creación de una FK: se debe probar la correcta creación de una clave foránea, respetando la integridad de los datos relacionados, agregándola a la tabla seleccionada y actualizando la vista donde aparecen.

Verificar la edición de una FK: se debe probar que la clave foránea al momento de ser editada valida los datos, la integridad entre atributos relacionados y se actualiza correctamente tanto en la estructura de datos como en la vista.

Verificar la eliminación de una FK: se debe probar que la eliminación de una clave foránea libera a los atributos que relacionaba, se elimina de la estructura y se elimina de la vista.

Verificar la importación de una BD externa en MySQL: se debe probar que la aplicación es capaz de conectarse a un motor MySQL mediante una Base de Datos, usuario y contraseña los que deben ser validados y testear la conexión. Además debe extraer la estructura de la Base de Datos a la cual se conecta (tablas, atributos, claves, tipos de atributos y referencias) y crear una estructura en memoria principal que represente a esta, posteriormente debe cargarla correctamente en la vista donde se verá en forma de árbol anidado.

Verificar la creación de una estructura de BD nueva y completa: se debe probar que la aplicación es capaz de crear una BD completa (tablas, atributos, claves, tipos de atributos y referencias) y esta debe reflejarse en la vista en forma de árbol anidado a medida que se vaya creando, además debe quedar en memoria principal.

Verificar el guardado de una BD: la aplicación debe probar que es capaz de guardar una estructura de BD que se encuentre en memoria principal y que esté disponible en la vista, de forma tal que, se guarde de manera persistente en la aplicación y pueda posteriormente cargarse aun cuando se cierre esta.

Verificar la carga de una BD: se debe probar que la aplicación es capaz de cargar una BD anteriormente guardada en la aplicación y que puede cargarla en la vista tal cual como era originalmente.

Verificar la generación de código java: se debe probar que la aplicación es capaz de generar el código java que representa la estructura de la BD, el cual debe generarse físicamente en un paquete dentro de la aplicación android en construcción (específicamente donde se activó el plug-in).

Verificar las clases generadas: se debe probar que las clases java generadas con el plug-in son funcionales esto es confirmar que son aptas para la inserción, edición, consulta y eliminación de datos dentro de una aplicación android,

Verificar los tipos de datos y claves foráneas: se debe probar que la BD creada con el plug-in y mediante las clases java generadas es posible controlar la integridad de tipos de datos y la integridad de las claves de referencia (FK)

Verificar la creación del archivo físico de BD: se debe probar que el archivo físico que representara a la estructura de BD en la aplicación android es creado dentro de esta.

.

7.2 Especificación de las pruebas

Características a probar	Nivel de prueba	Objetivo de la Prueba	Enfoque para la definición de casos de prueba	Técnicas para la definición de casos de prueba	Actividades de prueba	Criterios de cumplimiento
Crear una nueva BD	Prueba de unidad	Se pretende detectar si se está creando la estructura principal que corresponde a un TDA, el cual poseerá toda la estructura de la BD (tablas, atributos, PK, FK). Y que se guarda en memoria principal.	Caja blanca	Valores limites	Obtener el código fuente, establecer los valores limites, probar el código con dichos valores.	La prueba se dará por concluida cuando cree un TDA que representara la BD, con un nombre valido y no permitiendo la creación de esta con nombre que contengan caracteres especiales.
Creación de una tabla	Prueba de unidad	Se tiene por objetivo comprobar la correcta creación dentro del TDA principal del objeto que represent	Caja negra	Valores limites	Ejecutar la aplicación, establecer los valores limites, probar la aplicación con estos valores, comprobar lo que ocurre en la	La prueba se dará por terminada cuando la aplicación agregue la tabla a la BD con un nombre valido y esta se muestre en la vista.

		a una tabla de la BD.			vista.	
Crear un atributo	Prueba de unidad	Tiene por objetivo primordial comprobar que se cree correctamente dentro del objeto que representa una tabla, un atributo, el atributo con todas sus características asociadas (PK, null, etc.).	Caja negra	Partición equivalente	Ejecutar la aplicación, tener previamente una BD y una tabla creadas, definir las particiones, ejecutar la prueba, verificar que se agreguen a la tabla correspondiente, verificar que se agreguen en la vista.	La prueba se dará por concluida cuando el atributo se agregue a la tabla correspondiente, con valores válidos y se muestre en la vista.
Borrar un atributo	Prueba de unidad	Tiene por objetivo comprobar que al momento de eliminar un atributo, este se borre de la lista de atributos que posee la tabla.	Caja blanca	complejidad dicromática	Obtener el código de la operación, realizar el grafo correspondiente, obtener los caminos, probar los caminos.	La prueba se dará por concluida cuando se compruebe que se elimina el atributo y que se validan los casos especiales (el atributo es FK por lo tanto no se debe eliminar)
Editar un atributo	Prueba de unidad	Tiene por objetivo probar que cuando se edite un atributo, los cambios hechos se guarden correctamente en el atributo y tabla al cual pertenece.	Caja negra	Partición equivalente	Tener creada la BD, la tabla y el atributo previamente, seleccionas las particiones equivalentes, ejecutar la prueba, verificar que los cambios se realicen en el atributo y tabla correspondiente	La prueba se dará por concluida cuando al editar un atributo se guarden correctamente sus cambios y se validen estos previamente (que no se violen integridad referencial cuando corresponda) y se muestren los cambios en la vista.
Agregar FK	Prueba de unidad	Tiene como objetivo principal comprobar que si un atributo se	Caja negra	Valores limites	Se debe ejecutar la aplicación, se debe tener creada una BD con tablas y	La prueba concluye cuando se agrega correctamente una clave foránea y se

		seleccion a como clave foránea, se le sete su propiedad de clave foránea.			atributos para relacionar, se ejecuta la prueba, se observan y verifican los cambios en la vista.	validan la integridad de datos referenciados en el proceso, además se mostrar cambios en la vista.
Editar FK	Prue ba de unidad	Se evalúa que cuando se edite una clave foránea no se viole la integridad de los atributos y sus tipos de datos, además que se guarden los cambios correctam ente en el atributo y tabla que correspon da.	Caja negra	Partición equivalente	Se debe ejecutar la aplicación, se debe tener creada una BD con tablas y atributos y asignada la clave foránea para editarla, se ejecuta la prueba, se verifican los cambios en la vista.	La prueba concluye cuando se guarda la edición de la FK y se validan la integridad de datos en el proceso.
Eliminar FK	Prue ba de unidad	Se comprueb a que al momento de eliminar una FK se quite de las propiedad es de dicho atributo esa cualidad y que los cambios queden guardado s para el atributo y tabla correspon diente.	Caja blanca	Complejidad dicromática	Se debe ejecutar la aplicación, se debe tener creada una BD con tablas y atributos y asignada la clave foránea para eliminarla, ejecutar el plan de pruebas, verificar su eliminación de la vista.	La prueba concluirá cuando se elimine la clave foránea de la BD y se elimine la FK de la vista.
Importar una BD	Prue ba de unidad	Se pretende evaluar que se cargue en memoria principal la Base de Datos exportada desde otro	Caja negra	Partición equivalente	Se crea una clase main para probar el método, se ejecuta la prueba, se verifica que la información este en la memoria	La prueba se dará por concluida cuando en memoria principal se tengan los datos de la BD externa.

		servidor, esto es que cree correctamente el TDA con todas las tablas y atributos y características de estos y los deje en memoria principal listo para la vista.			principal.	
Crear una estructura de BD	Prueba de integración	Se pretende probar que se cree correctamente una Base de Datos desde cero con la interfaz de la aplicación esto es, agregar tablas, atributos, PK, FK. Y comprobar que se genere el TDA con la estructura creada.	Caja negra	Valores limites	Se debe ejecutar la aplicación, ejecutar la prueba y verificar que todos al final la estructura concuerde con la que se pretendía.	La prueba se dará por concluida cuando se logre crear una BD completa con tablas, atributos y claves, donde la estructura ingresada se refleje en la vista.
Guardar una BD	Prueba de integración	Se pretende probar que una vez creada una estructura esta se guarde correctamente en memoria secundaria, esto es comprobar la creación del archivo XML que representara la BD en memoria	Caja negra	Prueba basada en requisitos o casos de uso	Se debe haber iniciado la aplicación previamente, se debe tener una estructura cargada en la aplicación, se ejecuta la prueba, se verifica que se guarde el archivo (XML) de manera persistente.	La prueba concluirá cuando se guarde correctamente la estructura de BD de manera persistente en la aplicación.

		secundaria.				
Cargar una BD	Prueba de integración	Se pretende evaluar que se cargue correctamente una BD desde un archivo guardado previamente. Esto es que desde el XML se genere el TDA en memoria principal y que este se cargue en la vista correctamente y sin pérdida de información.	Caja negra	Prueba basada en requisitos o casos de uso	Tener archivos guardados previamente en la aplicación, ejecutar la prueba, verificar que se cargue la estructura y se muestre en la vista.	La prueba se dará por concluida cuando se cargue una estructura correctamente desde un archivo XML y se muestre en la vista.
Generar Código java	Prueba de integración	Se pretende evaluar que se genere el código en memoria principal y que este sea escrito en los archivos .java correspondientes.	Caja negra	Prueba basada en requisitos o casos de uso	Tener cargada una BD en la aplicación, ejecutar la aplicación, verificar que se generen los archivos java en el proyecto android.	La prueba concluirá cuando se generen los archivos java dentro del proyecto que usa el plug-in.
Generación de código java	Prueba de sistema	Se pretende evaluar que la generación de código se haga en el paquete que corresponde y que genere todas las clases predispuestas y que estas queden listas para su utilización.	Caja negra	Prueba basada en requisitos o casos de uso	La aplicación debe tener una BD cargada en ella, se ejecuta la prueba, se comprueba la generación de los archivos en donde corresponde y que su contenido sea completo, listos para ser usados.	La prueba concluye cuando los archivos generados se pueden utilizar en la programación inmediatamente sin realizarles ningún tipo de cambio. Y que además se generen en el paquete correcto dentro del proyecto.
Inserción de datos con clase fachada	Prueba de sistema	Se evaluara la	Caja negra	Prueba basada en requisitos o casos de uso	Se debe crear un proyecto	La prueba concluirá cuando la

		inserción de datos con la clase fachada generada desde la aplicación, esto es comprobar que la clase fachada este insertando los datos dentro de la BD con du método correspondiente.			android y generar el código java desde el plug in, se ejecuta la prueba, se verifica su funcionamiento con la aplicación android.	inserción con la clase fachada se correcta para todos los casos, incluyendo las validaciones de integridad de datos y referencias.
Eliminación de datos con clase fachada	Prueba de sistema	Se pretende evaluar que con la clase fachada generada desde la aplicación se pueda eliminar los datos ingresados, esto es comprobar que en la BD se borren los registros deseados con el método correspondiente de la clase.	Caja negra	Prueba basada en requisitos o casos de uso	Se debe crear un proyecto android y generar el código java desde el plug in, se ejecuta la prueba, se verifica su funcionamiento con la aplicación android.	La prueba concluirá cuando la eliminación con la clase fachada se correcta para todos los casos.
Consulta de datos con clase fachada	Prueba de sistema	Se comprueba que con la clase fachada generada con la aplicación se puede obtener desde la BD los registros de las tablas con los métodos correspondientes.	Caja negra	Valores limites	Se debe crear un proyecto android y generar el código java desde el plug in, se ejecuta la prueba, se verifica su funcionamiento con la aplicación android.	La prueba concluirá cuando la consulta con la clase fachada retorne los datos requeridos por el programador.
Edición de datos con clase fachada	Prueba de sistema	Se pretende evaluar	Caja negra	Valores limites	Se debe crear un proyecto	La prueba concluirá cuando la

		que la fachada edite correctamente los datos de una tabla. Esto es que guarde los cambios físicamente con el método correspondiente.			android y generar el código java desde el plug in, se ejecuta la prueba, se verifica su funcionamiento con la aplicación android.	edición con la clase fachada se correcta para todos los casos y los cambios sean guardados.
Validación de integridad de tipos metadatos	Prueba de sistema	Se pretende evaluar que cuando se desea realizar una inserción los datos cumplan con las restricciones de tipo, los máximos, y los datos null.	Caja negra	Valores limites	Se debe crear un proyecto android y generar el código java desde el plug in, se ejecuta la prueba, se verifica su funcionamiento con la aplicación android.	La prueba concluirá cuando al momento de insertar datos con la clase fachada esta valide los tipos de datos y su integridad.
Validaciones de integridad entre claves foráneas	Prueba de sistema	Se pretende evaluar que al momento de ingresar datos o eliminar o editar datos se respeten las restricciones de claves foráneas de la BD. Impidiendo en algunos casos la eliminación o edición de estos.	Caja negra	Valores limites	Se debe crear un proyecto android y generar el código java desde el plug in, se ejecuta la prueba, se verifica su funcionamiento con la aplicación android.	La prueba concluirá cuando al momento de insertar datos con la clase fachada esta valide la integridad referencial de los datos.
Generación de archivo de Base de Datos en la aplicación android	Prueba de sistema	Se pretende comprobar la correcta generación del archivo de BD en la aplicación	Caja negra	Prueba basada en requisitos o casos de uso	Generar código java desde el plug in, utilizarlo en algún desarrollo android, ejecutar la prueba,	La prueba concluirá cuando al ejecutar la aplicación android que utilizo el código generado cree el fichero de

		android el cual se genera físicamente en una carpeta predeterminada.			**verificar fichero.**	**BD en su interior.**

Tabla Nº 29: Especificación de pruebas.

7.3 Responsables de las pruebas

Nombre	Responsabilidad
Kevin Sandoval	Crear una nueva BD
Kevin Sandoval	Creación de una tabla
Kevin Sandoval	Crear un atributo
Kevin Sandoval	Borrar un atributo
Kevin Sandoval	Editar un atributo
Kevin Sandoval	Agregar FK
Kevin Sandoval	Editar FK
Kevin Sandoval	Eliminar FK
Kevin Sandoval	Importar una BD
Kevin Sandoval	Crear una estructura de BD
Daniel Jeldres	Guardar una BD
Daniel Jeldres	Cargar una BD
Daniel Jeldres	Generar Código java
Daniel Jeldres	Inserción de datos con clase fachada
Daniel Jeldres	Eliminación de datos con clase fachada
Daniel Jeldres	Consulta de datos con clase fachada
Daniel Jeldres	Edición de datos con clase fachada
Daniel Jeldres	Validación de integridad de tipos metadatos
Daniel Jeldres	Validaciones de integridad entre claves foráneas
Kevin Sandoval	Generación de archivo de Base de Datos en la aplicación android

Tabla Nº 30: Responsable de pruebas.

7.4 Calendario de pruebas

7.4.1

Nombre de tarea	Duración	Comienzo	Fin	Predecesoras	Nombres de los recursos
Elaboracion plan de pruebas	5 días?	lun 04-03-13	vie 08-03-13		Kevin Sandoval
pruebas de unidad	**0,88 días**	**lun 11-03-13**	**lun 11-03-13**	**1**	
Crear una nueva BD	1 hora	lun 11-03-13	lun 11-03-13		Kevin Sandoval
Creación de una tabla	1 hora	lun 11-03-13	lun 11-03-13	3	Kevin Sandoval
Crear un atributo	1 hora	lun 11-03-13	lun 11-03-13	3,4	Kevin Sandoval
Borrar un atributo	1 hora	lun 11-03-13	lun 11-03-13	5	Kevin Sandoval
Editar un atributo	2 horas	lun 11-03-13	lun 11-03-13	5	Kevin Sandoval
Agregar FK	2 horas	lun 11-03-13	lun 11-03-13	5	Kevin Sandoval
Editar FK	2 horas	lun 11-03-13	lun 11-03-13	8	Kevin Sandoval
Eliminar FK	1 hora	lun 11-03-13	lun 11-03-13	8	Kevin Sandoval
Importar una BD	3 horas	lun 11-03-13	lun 11-03-13		Kevin Sandoval
pruebas de integridad	**2 días**	**mar 12-03-13**	**mié 13-03-13**	**2**	
Crear una estructura de BD	5 horas	mar 12-03-13	mar 12-03-13		Kevin Sandoval
Guardar una BD	3 horas	mar 12-03-13	mar 12-03-13	13,11	Daniel Jeldres
Cargar una BD	5 horas	mié 13-03-13	mié 13-03-13	14	Daniel Jeldres
Generar Código java	3 horas	mié 13-03-13	mié 13-03-13	11,15	Daniel Jeldres
pruebas de sistema	**4 días**	**jue 14-03-13**	**mar 19-03-13**	**12**	
Generación de código java	1 día	jue 14-03-13	jue 14-03-13		Daniel Jeldres
Inserción de datos con clase fachada	1 día	vie 15-03-13	vie 15-03-13	18	Daniel Jeldres
Eliminación de datos con clase fachada	1 día	vie 15-03-13	vie 15-03-13	18	Daniel Jeldres
Consulta de datos con clase fachada	1 día	vie 15-03-13	vie 15-03-13	18	Daniel Jeldres
Edición de datos con clase fachada	1 día	vie 15-03-13	vie 15-03-13	18	Daniel Jeldres
Validación de integridad de tipos metadatos	1 día	lun 18-03-13	lun 18-03-13	19,20,21,22	Daniel Jeldres
Validaciones de integridad entre claves foráneas	1 día	lun 18-03-13	lun 18-03-13	19,20,21,22	Daniel Jeldres
Generación de archivo de base de datos en la aplicación android	1 día	mar 19-03-13	mar 19-03-13	19,20,21,22,18	Kevin Sandoval

Figura Nº 24: Calendario de pruebas.

7.4.2 Carta Gantt

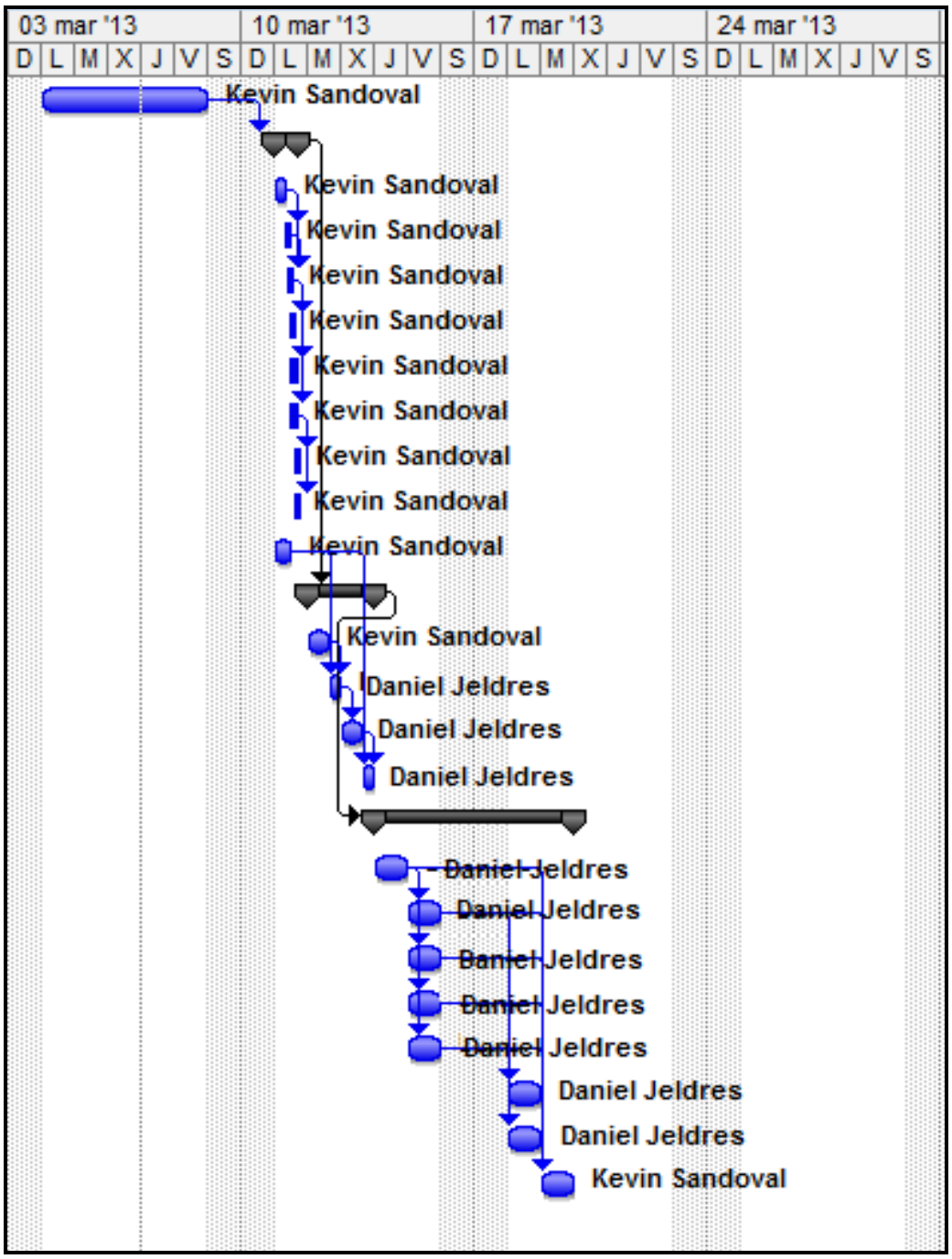


Figura Nº 25: Carta Gantt de pruebas.

7.5 Detalle de las pruebas

7.5.1 <Crear nueva Estructura BD>

- Configuración: Se debe tener instalado el plug-in en el eclipse, y se debe ejecutar para que muestre la pantalla principal.
- Precondición: Que la BD de datos que se cree sea con la opción “Nueva BD”

ID Caso De Prueba	Características a Probar	Datos de Entrada	Salida esperada	Salida Obtenida	Éxito / Fracaso	Observaciones
		nombre				
001	Se crea la BD con el nombre por defecto correctamente	Nueva_Base_de_Datos	Se crea correctamente la nueva BD	Se crea la BD	Éxito	Sin observación
002	Se prueba un nombre solo con caracteres (a-z A-Z)	BaseDeDatos	Se crea correctamente la nueva BD	Se crea la BD	Éxito	La acción solo actualiza el nombre, ya que el nombre por defecto queda guardado como primera instancia al crear una nueva BD.
003	Se prueba un nombre solo con números (0-9)	123456789	Se crea correctamente la nueva BD	Se crea la BD	Éxito	La acción solo actualiza el nombre, ya que el nombre por defecto queda guardado como primera instancia al crear una nueva BD.
004	Se prueba con un nombre combinando números y caracteres (0-9,a-z,A-Z)	baseDatos001	Se crea correctamente la nueva BD	Se crea la BD	Éxito	La acción solo actualiza el nombre, ya que el nombre por defecto queda guardado como primera instancia al crear una nueva BD.
005	Se prueba un nombre con caracteres especiales (*,+,}, (), etc.)	+***`[]{}	La BD no se crea y se muestra un mensaje de error al usuario	No se puede crear la BD y se emite una alerta al usuario	Éxito	La alerta es en tiempo real no cuando el usuario presiona el botón para actualizar el nombre.

006	Se prueba un nombre combinado de letras y caracteres especiales (a-z, A-Z,+,*, {}, etc.)	{baseDatos+.*}	La BD no se crea y se muestra un mensaje de error al usuario	No se puede crear la BD y se emite una alerta al usuario	Éxito	La alerta es en tiempo real no cuando el usuario presiona el botón para actualizar el nombre.

Tabla Nº 31: Detalle de prueba crear nueva BD

7.5.2 <Crear nueva Tabla>

- Configuración: Se debe tener instalado el plug-in en el eclipse, y se debe ejecutar.
- Precondición: Se debe haber creado una BD.

ID Caso De Prueba	Características a Probar	Datos de Entrada	Salida esperada	Salida Obtenida	Éxito / Fracaso	Observaciones
		nombre				
007	Se probara crear una tabla con un nombre vacío.		No debe crear la tabla y debe mostrar un mensaje al usuario indicando error.	No crea la tabla, además alerta al usuario del error.	Éxito	Alerta al usuario en tiempo real y no necesariamente cuando se presiona el botón.
008	Se probara crear una tabla con nombre solo letras (a-z ,A-Z)	tabla	Se crea la tabla con el nombre especificado.	Crea la tabla con el nombre ingresado.	Éxito	Crea la tabla, sin embargo si se cambia el nombre una vez creada la tabla al presionar el botón actualizar, se actualiza el nombre de la tabla recién creada, de esta forma para crear una tabla nueva se debe acceder a la opción en el menú o en el menú contextual de la BD.
009	Se probara crear tabla con nombre solo números (0-9)	01	Se crea la tabla con el nombre especificado.	Crea la tabla con el nombre ingresado	Éxito	Crea la tabla, sin embargo si se cambia el nombre una vez creada la tabla al presionar el botón actualizar, se actualiza el nombre de la tabla recién creada, de esta forma para crear una tabla nueva se debe acceder a la opción en el menú o en el menú contextual
010	Se probara crear tabla con nombre combinando letras y números (0-9, a-z, A-Z)	Tabla 01	Se crea la tabla con el nombre especificado.	Crea la tabla con el nombre ingresado	Éxito	Crea la tabla, sin embargo si se cambia el nombre una vez creada la tabla al presionar el botón actualizar, se actualiza el nombre de la tabla recién creada, de esta forma para crear una tabla nueva se debe acceder a la opción en el menú o en el menú contextual
011	Se probara crear una tabla cuyo nombre contenga caracteres especiales(*, +, {}, etc)	[****]	No se crea la tabla y se da una alerta al usuario con el error de entrada.	No crea la tabla, además alerta al usuario del error.	Éxito	Alerta al usuario en tiempo real y no necesariamente cuando se presiona el botón.
012	Se probara crear una tabla que contenga caracteres	{tabla.[]}	No se crea la tabla y se da una alerta al usuario con el error de	No crea la tabla, además alerta al	Éxito	Alerta al usuario en tiempo real y no necesariamente cuando se presiona el botón.

	especiales y letras combinados (a-z, A-Z, 0-9)		entrada.	usuario del error.		

Tabla Nº 32: Detalle de prueba crear nueva tabla

7.5.3 <Crear nuevo Atributo>

- Configuración: Se debe tener instalado el plug-in en el eclipse, y se debe ejecutar.
- Precondición: se debe haber creado una BD y una tabla en la cual agregar el atributo.

ID Caso De Prueba	Características a Probar	Datos de Entrada							Salida esperada	Salida Obtenida	Éxito / Fracaso	Observaciones
		nombre	defecto	Máx.	min.	longitud	tipo	Primaria/null				
013	Probar que si se presiona el botón agregar sin ingresar los datos, no crea el atributo ya alerta al usuario								No crea el atributo y alerta al usuario del error.	No crea el atributo y alerta al usuario del error.	Éxito	El aviso se produce cuando se presiona el botón agregar.
014	Probar que los campos no aceptan datos con caracteres especiales.	Atributo[]	*****	+++	+*	{}			No debe crear el atributo y debe alertar al usuario de los errores de datos.	Crea el atributo.	Fracaso	No se validan los campos defecto, Max, min pudiendo ingresar en ellos caracteres especiales y letras.
015	Probar que si el tipo de dato es real o integer no se puede poner un valor por defecto que sea carácter		inicio				Real, integer		No se debe agregar el atributo y se debe alertar al usuario del error.	No se agrega el atributo y se alerta al usuario del error	Éxito	El aviso es en tiempo real al usuario y no necesariamente cuando se presiona el botón.
016	Probar que si el atributo							Primaria=true y	Cuando se seleccione	Cuando se seleccion	Éxito	Sin observaciones.

	es clave primaria este no puede admitir null y viceversa							null=true	clave primaria se debe dejar como no seleccionable el null y viceversa.	a null clave primaria no es seleccionable y cuando se selecciona primaria campo null no es seleccionable		
017	Probar que los únicos datos obligatorios para agregar el atributo son el nombre y el tipo de dato.	atributo					integer		Se crea el atributo correctamente.	Se crea el nuevo atributo.	Éxito	Sin observaciones.

Tabla Nº 33: Detalle de prueba crear nuevo atributo

7.5.4 <Borrar Atributo>

- Configuración: Se debe tener instalado el plug-in en el eclipse, y se debe ejecutar.
- Precondición: se debe haber creado una BD, una tabla y un atributo, el cual se desea borrar.

ID Caso De Prueba	Características a Probar	Datos de Entrada	Salida esperada	Salida Obtenida	Éxito / Fracaso	Observaciones
		D1				
018	Se probara que cuando se selecciona borrar un atributo, este se borra correctamente, es decir, se elimina de la tabla donde se encontraba y se borra de la vista		Se borra el atributo que corresponde en la tabla que corresponde y se actualiza la vista.	El atributo se borra correctamente y se actualiza la vista.	Éxito	La interfaz muestra al usuario un mensaje de confirmación de la eliminación para prevenir errores.
019	Probar que si el atributo es clave foránea este no se elimina y alerta al usuario de esta situación.		Se alerta al usuario de la condición y no se borra el atributo.	Se informa al usuario de que es una clave foránea y el atributo no es eliminado.	Éxito	Para borrar el atributo se debe eliminar la clave foránea que la involucra.

Tabla Nº 34: Detalle de prueba borrar atributo

7.5.5 <Editar Atributo>

- Configuración: Se debe tener instalado el plug-in en el eclipse, y se debe ejecutar.
- Precondición: se debe haber creado una BD, una tabla y un atributo, el cual se desea editar. Además se probara la edición de un atributo “at01” el cual es clave foránea con “at02” y un atributo “at00” el cual no es clave foránea, por lo cual se debe tener estos atributos creados.

ID Caso De Prueba	Características a Probar	Datos de Entrada							Salida esperada	Salida Obtenida	Éxito / Fracaso	Observaciones
		nombre	defecto	Máx.	min.	longitud	tipo	Primaria/null				
020	Se prueba que los datos que se cargan en el panel de edición son los datos del atributo seleccionado								Se cargan los datos del atributo seleccionado en el panel de edición.	Se cargan los datos del atributo seleccionado en el panel de edición.	Éxito	Sin observaciones.
021	Se prueba que no se aceptan nombres y valores con caracteres especiales.	At[01]	***	treinta	cero	mucho	integer		No se aceptan nombres con caracteres especiales en la edición. Se alerta al usuario del error.	Se guardan los cambios.	Fracaso	No se validan los campos default, Max, min y se puede en estos ingresar datos inválidos en la edición.
022	Se prueba que no se puede cambiar el tipo de dato si el atributo es clave foránea.						Tipo de la relación=integer. Tipo de dato en la edición=text		No se puede guardar un cambio de tipo de datos si el atributo es clave foránea o está siendo referenciado, alerta al	No permite editar el tipo de datos si el atributo es clave foránea o si es referenciado.	Éxito	Muestra un aviso al usuario indicando que no se puede realizar la edición del tipo de dato.

									usuario del error.			
023	Se prueba que los cambios se guardan correctamente en el atributo editado y se actualiza en la vista.								Se guardan los datos correctamente y se actualiza el atributo en la vista.	Los cambios se guardan correctamente y se actualiza la vista	Éxito.	Sin observaciones.

Tabla Nº 35: Detalle de prueba editar atributo

7.5.6 <Crear nueva FK>

- Configuración: Se debe tener instalado el plug-in en el eclipse, y se debe ejecutar.
- Precondición: se debe haber creado una BD, dos tablas y dos atributos, los cuales se relacionaran.

ID Caso De Prueba	Características a Probar	Datos de Entrada					Salida esperada	Salida Obtenida	Éxito / Fracaso	Observaciones
		nombre	Tipo atributo	Tipo Atributo referenciado	tabla	Tabla referenciada				
024	Se probara la creación de una FK con un nombre con caracteres especiales.	[tabla]					No se crea la clave foránea y se emite un mensaje al usuario indicando el error.	No se puede crear la clave y además se alerta al usuario del error.	Éxito	El alerta al usuario es de forma inmediata y no necesariamente cuando se presiona el botón crear.
025	Se probara si se puede relacionar atributos de distintos tipos	FK	integer	text	Ta01	Ta02	No se puede crear la clave y se informa al usuario del error.	No permite crear la clave ya que son tipos de datos distintos y se alerta al usuario.	Éxito	Sin observaciones.
026	Se probara si se puede relacionar el atributo	FK	integer	integer	Ta01	Ta01	No se puede crear la clave y	Permite crear una clave relaciona	fracaso	corregir

	consigo mismo.						se informa al usuario del error.	da consigo mismo		
027	Se probara relacionar atributos del mismo tipo y tablas diferentes	FK	integer	integer	Ta01	Ta02	Se crea la clave y se carga en la vista.	Se crea la FK y se carga en la vista.	Éxito	Al momento de crearla se muestra la pantalla donde se puede visualizar la clave foránea creada.

Tabla Nº 36: Detalle de prueba crear nueva FK

7.5.7 <Editar FK>

- Configuración: Se debe tener instalado el plug-in en el eclipse, y se debe ejecutar.

  Precondición: se debe haber creado una BD, dos tablas, dos atributos de tipo integer, un atributo tipo text y crear una clave foránea (la cual relaciona los dos atributos tipo integer que se encuentran en distintas tablas) que los relacione la cual será editada.

ID Caso De Prueba	Características a Probar	Datos de Entrada					Salida esperada	Salida Obtenida	Éxito / Fracaso	Observaciones
		nombre	Tipo atributo	Tipo Atributo referenciado	tabla	Tabla referenciada				
028	Se probara cambiar a un nombre inválido.	[FK]					No se pueden guardar los cambio y se alerta al usuario del error	No se guardan los cambios y se alerta al usuario del error.	Éxito	La validación es en tiempo real y no cuando se presiona el botón guardar.
029	Se probara cambiar la relación a un atributo de distinto tipo.	FK	integer	text	Ta01	Ta02	No se puede realizar el cambio ya que los tipos de datos son distinto	Se realizar el cambio y se guardan.	Fracaso	Se debe validar en la edición

								s.			
030	Se probara cambiar la relación al mismo atributo.	FK	integer	integer	Ta01	Ta01		No se guardan los cambios y se alerta al usuario.	Se guardan los cambios.	fra caso	Se debe validar el caso.

Tabla Nº 37: Detalle de prueba editar FK

7.5.8 <Eliminar FK>

- Configuración: Se debe tener instalado el plug-in en el eclipse, y se debe ejecutar.

  Precondición: se debe haber creado una BD, dos tablas, dos atributos y crear una clave foránea, la cual se eliminara.

ID Caso De Prueba	Características a Probar	Datos de Entrada	Salida esperada	Salida Obtenida	Éxito / Fracaso	Observaciones
		D1				
031	Se probara la eliminación cuando no se ha seleccionado ninguna FK		Se alerta al usuario que debe seleccionar una clave foránea para eliminar.	Se muestra un mensaje al usuario para informarle que debe seleccionar una FK para eliminar	Éxito	Sin observaciones
032	Se probara la eliminación cuando se selecciona una FK		Se elimina la clave foránea seleccionada y no otra. Se actualiza la tabla en la vista.	Se elimina la FK seleccionada y se actualiza la vista correctamente.	Éxito	Tal vez sea necesario confirmar la eliminación.

Tabla Nº 38: Detalle de prueba eliminar FK

7.5.9 <Importar BD>

- Configuración: Se debe tener instalado el plug-in en el eclipse, y se debe ejecutar. Además se debe tener un motor MySQL instalado y ejecutándose.
- Precondición: se debe haber creado previamente la BD a importar en el motor MySQL, se debe tener el nombre de la Base de Datos, el usuario y la contraseña para la conexión. Para esta prueba se configura una pequeña BD en MySQL donde su nombre es memoria, pass: memoria, user:root, puerto:3306, servidor:localhost y su contenido es una tabla alumno (id,idramo,nombre,rut,edad), ramo(id,nombre,cupos) , además un FK que relaciona alumno.idramo con ramo.id.

ID Caso De Prueba	Características a Probar	Datos de Entrada					Salida esperada	Salida Obtenida	Éxito / Fracaso	Observaciones
		Nombre BD	puerto	servidor	pass	user				
033	Se probara la conexión con usuario y password inválidos.	memoria	3306	localhost	Memoria01	memo	No se realiza la conexión y se informa al usuario del error.	No se realiza la conexión falla el test y se muestra el error al usuario.	Éxito	Sin observaciones.
034	Se probara la conexión a una BD inexistente.	Memoria1020	3306	localhost	memoria	root	No se realiza la conexión y se alerta al usuario del error.	No se realiza la conexión falla el test y se muestra el error al usuario.	Éxito	Sin observaciones.
035	Se probara el test de una conexión valida.	memoria	3306	localhost	memoria	root	Se realiza la conexión y se muestra al usuario por pantalla que la conexión es exitosa.	Se conecta correctamente, el test es exitoso e importa la estructura.	Éxito	Sin observaciones.
036	Se probara el test de una conexión inválida.	memoria	8080	172.15.244.23	memoria	root	No se realiza la conexión y se alerta al usuario del error.	No se realiza la conexión falla el test y se muestra el error al usuario.	Éxito	El mensaje de error es demasiado extenso para el log de error implementado.
037	Se probara la validación de los campos de entrada de datos de conexión.	Memoria[]	3r4	L[]cal	memoria	root	Se muestran al usuario alertas de errores en los datos de entrada.	Se valida solo caracteres en el puerto, pero no caracteres especiales en los otros campos	fracaso	Verificar validaciones.

038	Se probara que la BD generada sea exactamente igual a la que se importó desde MySQL	memoria	3306	localhost	memoria	root	Se crea la estructura y es la misma que en MySQL.	Se importa la estructura y las tablas con sus atributos correctamente. Se ve un error en las claves foráneas.	fracaso	Corregir error de FK

Tabla Nº 39: Detalle de prueba importar BD

7.5.10 <Crear Estructura Completa de BD>

- La condición es tener un esquema el cual realizar y el que se utilizara posteriormente en la aplicación. Este esquema se creara desde cero con ayuda del plug-in hasta obtener la estructura deseada.
- Se debe tener instalado el plug-in en el eclipse, haber creado un proyecto android en el que se desee trabajar con dicho esquema.

Id	Descripción Requerimiento Funcional	Entrada			Salida esperada	Salida Obtenida	Evaluación	
		D1	D2	D3			Éxito / Fracaso	Criticidad en caso Fracaso
039	Se prueba que es posible crear una estructura que contenga todo lo necesario para su utilización en una aplicación android.	tablas	atributos	claves	Crear una estructura de BD real agregando tablas y atributos mediante la interfaz de plug-in sin contratiempos.	Se logra crear una estructura de BD desde cero agregando tablas y atributos, además de claves ya sean primarias y foráneas. Se validan los datos y las restricciones de integridad referencial, y todos los accesos a las opciones funcionan correctamente.	Éxito	

Tabla Nº 40: Detalle de prueba crear estructura de BD

7.5.11 < Guardar una BD>

En este ítem se especifican:

- La condición es tener un esquema creado en el plug-in y el que se necesita guardar de manera persistente como respaldo por posibles problemas, porque no se ha terminado aún y se debe pausar la construcción o para futuras modificaciones.

  Se debe tener instalado el plug-in en el eclipse, haber creado un proyecto android en el que se desee trabajar con dicho esquema y se debe haber creado la estructura con anterioridad.

Id	Descripción Requerimiento Funcional	Entrada	Salida esperada	Salida Obtenida	Evaluación	
		D1			Éxito / Fracaso	Criticidad en caso Fracaso
39	Se debe probar el requerimiento de guardar la BD creada en el plug-in de forma persistente, de manera que se pueda utilizar una vez que se reinicie el plug-in.	Estructura de BD	Se guarda la estructura de forma permanente en el proyecto y esta queda accesible para el programador en cualquier momento que la requiera.	Se guarda la estructura correctamente en un archivo de tipo XML en el proyecto en el cual se está utilizando el plug-in de esta manera se puede acceder a él en cualquier momento que se requiera.	Éxito	

Tabla Nº 41: Detalle de prueba guardar BD

7.5.12 < Cargar una BD>

- La condición es tener un esquema guardado en el proyecto eclipse con anterioridad, el cual el usuario necesita cargarlo y editarlo o simplemente generar su código nuevamente.
- Se debe tener instalado el plug-in en el eclipse, haber creado un proyecto android en el que se desee trabajar con dicho esquema y se debe haber guardado la estructura con anterioridad.

Id	Descripción Requerimiento Funcional	Entrada	Salida esperada	Salida Obtenida	Evaluación	
		D1			Éxito / Fracaso	Criticidad en caso Fracaso
040	Se prueba el requerimiento de cargar una BD desde memoria secundaria.	Archivo XML	Se carga una BD guardada anteriormente en la aplicación y se muestra su contenido en el plug-in.	Se carga la BD que se había guardado anteriormente de forma correcta, lee el archivo desde memoria secundaria y es capaz de cargar toda la estructura como era en un principio sin errores en ningún caso.	Éxito	

Tabla Nº 42: Detalle de prueba cargar BD

7.5.13 < Generar Código Java>

- La condición es tener un esquema en el plug-in con el cual ya se desea comenzar a trabajar, de esta manera solo generar el código para comenzar a utilizarlo en la construcción de la aplicación.
- Se debe tener instalado el plug-in en el eclipse, haber creado un proyecto android en el que se desee trabajar con dicho esquema y se debe tener una estructura cargada en el plug-in, además se debe haber abierto el plug-in en el paquete que se desea generar el código.

Id	Descripción Requerimiento Funcional	Entrada	Salida esperada	Salida Obtenida	Evaluación	
		D1			Éxito / Fracaso	Criticidad en caso Fracaso
041	Se prueba que el plug-in es capaz de generar las clases en java que representan la BD (DAO,POJOs) y las clases con las cuales se administrara dicha BD (Fachada y SQL y adminSQLiteOpenHelper)	Estructura de BD	Archivos con código java que representan las BD (DAO, POJO) y archivos con los cuales se podrá administrar esta BD (Fachada y SQL y adminSQLiteOpenHelper) todos archivos con extensión .java y en paquetes separados.	Se generan correctamente los archivos java en los paquetes que corresponden y representan en forma correcta a la BD.	Éxito	

Tabla Nº 43: Detalle de prueba generar código java

7.5.14 < Pruebas de clase Fachada>

- La condición es tener que realizar un proyecto android en el cual es necesario realizar operaciones a la BD ya sea, inserción, eliminación, actualización y consultas.
- Se debe haber generado el código con anterioridad de la BD con la cual se trabajara. Además se debe construir una aplicación android en la cual se utilicen los métodos generados con el plug-in.

Id	Descripción Requerimiento Funcional	Entrada	Salida esperada	Salida Obtenida	Evaluación	
		D1			Éxito / Fracaso	Criticidad en caso Fracaso
042	Se prueba la inserción en la BD con la clase Fachada.	Objeto (POJO) a insertar	Se espera que la clase fachada realice con éxito la inserción de datos en la BD y que esta se haga solo a través del método que corresponde en la clase.	Inserción exitosa	Éxito	
043	Se prueba la actualización en la BD con la clase Fachada	Objeto(POJO) a actualizar	Se espera que la clase Fachada realice correctamente la actualización de datos en	Actualización correcta	Éxito	

			la BD y que esto se realice solo con el método especificado en la clase.			
044	**Se prueba la consulta de datos en la BD con la clase Fachada**	**String con la condición de la búsqueda**	**Se espera que la clase Fachada realice correctamente consultas en la BD y devuelva lo solicitado correctamente y solo a través del método creado para esto.**	**Lista con los Objetos (POJOs) correctamente según la condición de búsqueda**	**Éxito**	
045	**Se prueba la Eliminación de la BD con la clase Fachada**	**String con la condición de la eliminación**	**Se espera que la clase Fachada realice la eliminación correctamente de la BD y solo con el método destinado para esto.**	**Eliminación correcta**	**Éxito**	

Tabla Nº 44: Detalle de prueba clase fachada

7.5.15 < Validaciones de integridad>

- La condición es tener que realizar un proyecto android en el cual es necesario realizar operaciones a la BD ya sea, inserción, eliminación, actualización y consultas.
- Se debe haber generado el código con anterioridad de la BD con la cual se trabajara tener claro los tipos de datos en la BD con lo que comprobara las validaciones de integridad.

Id	**Descripción Requerimiento Funcional**	**Entrada**	**Salida esperada**	**Salida Obtenida**	**Evaluación**	
		D1			**Éxito / Fracaso**	**Criticidad en caso Fracaso**
046	**Se prueba si la clase Fachada es capaz de validar al momento de la inserción que los tipos de datos insertados sean íntegros con los tipos de datos de la BD.**	**String con los valores a insertar**	**Se espera que la Fachada controle la integridad de los tipos de datos al insertar y que si no cumplen con esto no realice cambios en la BD además de informar al programador de la violación de integridad.**	**Se captura el error al tratar de transformar los String al tipo de dato correspondiente**	**Éxito**	
047	**Se probara si la Base de Datos es capaz de validar al momento de la inserción que no se violen las referencias entre las tablas con los datos insertados.**	**Objeto(POJO) haciendo referencia a elementos inexistentes**	**Se espera que si existe alguna violación de referencia entre tablas con los datos que se pretenden insertar, la Base de Datos devuelva un error indicando el problema.**	**La Base de Datos devuelve un error indicando el problema**	**Éxito**	

Tabla Nº 45: Detalle de prueba Validaciones de Integridad

7.5.16 < Generar Archivo BD Físico >

- La condición es tener que ejecutar la aplicación creada con el código generado y ver qué es lo que hay en el interior del archivo generado que representa la BD físicamente en el dispositivo.
- Se debe haber terminado la programación del código fuente y solo se debe ejecutar el apk en un dispositivo o emulador.

Id	Descripción Requerimiento Funcional	Entrada	Salida esperada	Salida Obtenida	Evaluación	
		D1			Éxito / Fracaso	Criticidad en caso Fracaso
048	Se probara si con el código generado se crea correctamente el archivo de BD que representa a esta de manera física en el teléfono.	String con nombre del archivo a crear	Se espera que se cree un archivo en el cual al ser abierto con el administrador de SQLite contenga la estructura de la BD tal cual se diseñó en el plug-in.	Archivo creado correctamente	Éxito	

Tabla Nº 46: Detalle de prueba creación de archivo físico

7.6 Conclusiones de Prueba

Todas las pruebas realizadas se centran en posibles errores de validaciones que pudieren existir en la aplicación desde caracteres inválidos hasta números ingresados en vez de palabras o viceversa, se validan además que los cambios o modificaciones siempre se realicen en donde corresponde y no en otros objetos, las validaciones más importantes son las de referencia e integridad de tipos de datos las cuales son fundamentales en las BD. Además se prueba las clases generadas con el plug-in y su real utilidad en un ambiente de desarrollo.

Las pruebas arrojan como conclusión que todos los datos que se ingresan en la construcción de un esquema de BD son validados, con algunas excepciones mínimas de errores encontrados en algunos casos. Además se comprobó mediante las pruebas que la aplicación es capaz de validar que no se puedan violar las restricciones de integridad en los tipos de datos y en la referencias. Esto es que no se pueda borrar un atributo que es referenciado y que no se pueda cambiar el tipo de datos de un atributo que es referenciado. Esto es una de los principales objetivos que se esperaba de la aplicación.

Se prueba que la aplicación es capaz indudablemente de importar una BD externa, para este caso, del motor MySQL, y obtener su estructura completa para dejarla disponible, ya sea, para edición o utilización inmediata en el desarrollo.

Finalmente se prueba la aplicación en ejecución y se comprueba que esta es capaz de realizar las acciones básicas de BD (inserción, eliminación, consulta y actualización) mediante la clase Fachada si problemas, además se comprueba que la aplicación es capaz de validar en tiempo real las restricciones de tipo de dato y clave foráneas al momento de la inserción, siendo esto el principal objetivo en tipo de ejecución real para la aplicación.

8 RESUMEN ESFUERZO REQUERIDO

A continuación se detallan las horas utilizadas para el desarrollo del proyecto.

Actividades/fases	N° Horas
Planificación	**96**
Definición del problema	**24**
Planificación del proyecto	**48**
Definición del proyecto	**24**
Análisis	**58**
Diagrama de casos de uso	**20**
Especificación de casos de uso	**25**
Diagrama de clases	**13**
Diseño	**115**
Diseño de interfaces	**70**
Diseño de BD	**20**
Diseño de menú	**25**
Desarrollo	**570**
Implementación de interfaces (frame)	**100**
Implementación de navegación	**50**
Implementación de modelo de clases	**10**
Implementación de importar BD	**30**
Implementación guardar BD	**15**
Implementación de cargar BD	**20**
Implementación de Generación de clases	**20**
Implementación de crear estructura de BD	**255**
Integración con eclipse	**20**
Pruebas	**95**
Plan de pruebas	**45**
Ejecución de pruebas	**50**
Documentación	**34**
Manual de usuario	**20**
Calendarios y cartas Gantt	**10**
Diccionario de datos	**4**
TOTAL	**968**

Tabla Nº 47: Detalle esfuerzo alumno Kevin Sandoval

Actividades/fases	N° Horas
Planificación	**100**
Definición del problema	**26**
Planificación del proyecto	**46**
Definición del proyecto	**28**
Análisis	**50**
Diagrama de casos de uso	**15**
Especificación de casos de uso	**10**
Diagrama de clases	**25**
Diseño	**45**
Diseño de interfaces	**15**
Diseño de BD	**20**
Diseño de menú	**10**
Desarrollo	**780**
Implementación de interfaces (frame)	**130**
Implementación de navegación	**90**
Implementación de modelo de clases	**20**
Implementación de importar BD	**10**
Implementación guardar BD	**85**
Implementación de cargar BD	**75**
Implementación de Generación de clases	**300**
Implementación de crear estructura de BD	**50**
Integración con eclipse	**20**
Pruebas	**40**
Plan de pruebas	**5**
Ejecución de pruebas	**35**
Documentación	**10**
Manual de usuario	**4**
Calendarios y cartas Gantt	**3**
Diccionario de datos	**3**
TOTAL	**925**

Tabla Nº 48: Detalle esfuerzo alumno Daniel Jeldres

9 CONCLUSIONES

El proyecto que se planteó al inicio del desarrollo debía ser un plug-in eclipse el cual se pueda incluir en el IDE, para poder lograr facilitar la programación e implementación de BD para el desarrollo de aplicación android. Este plug-in debe permitir la creación de una BD considerando tablas, atributos, y claves. Debe tener especial cuidado con las validaciones tanto de tipos de datos como de integridad referencial entre tablas y atributos. El plug-in debe permitir generar a partir de una estructura de BD creada, las clases java de tipo POJO, DAO, Fachada y AdminSQLopenHelper. Con las cuales el programador pueda realizar operaciones básicas de inserción, eliminación, actualización y consulta sin necesidad de escribir código SQL dentro de sus clases, sólo con llamadas a los métodos de la clase Fachada. Además, la BD debe ser capaz de validar que no se puedan ingresar datos si el tipo es incorrecto y que no se puedan ingresar datos si las claves foráneas son violadas. Finalmente debe permitir exportar una estructura desde MySQL para poder utilizarla de forma móvil.

Con estas metas se logra construir una aplicación que es capaz de realizar esto de manera correcta. Se logró crear un plug-in que se incorpora a eclipse y se puede utilizar en el desarrollo de aplicación android. Posee una interfaz estándar y muy similar a la de los actuales motores de BD por lo cual su uso es muy similar a estos. Permite crear nuevas BD, tablas y atributos. Además de permitir crear claves foráneas a las cuales se valida que en la relación no se produzcan violaciones de tipo de datos y cuando se inserta no se produzcan violaciones de referencias. Se logró un plug-in que permite la importación de BD de un motor externo como es MySQL, además queda abierta la posibilidad de ampliarlo a otros motores de BD. La aplicación logra Guardar de manera persistente las BD creadas para no correr riesgos de perderlas en el tiempo. Genera de manera correcta clases java como son POJOs, DAOs, Fachada y AdminSQLOpenHelper, los cuales se cumplen con su función especificada simplificando de manera notable en líneas de código y tiempo en la programación.

Con respecto a las herramientas cabe destacar que no se utilizaron grandes configuraciones, solo bastó con tener instalado el IDE eclipse y crear un nuevo proyecto del tipo plug-in. En lo que respecta al lenguaje de programación, se utilizó java para toda

la aplicación, salvo en las partes de importar y generación de código donde se debió incorporar leguaje SQL para las consultas y operaciones a BD. Para la parte gráfica se utilizó el plug-in de eclipse para desarrollo de interfaz gráficas (Windows Builder), la cual facilitó de manera notable el trabajo en esta fase del desarrollo.

Se tomó como ejemplo para el desarrollo gráfico del plug-in las interfaces de los administradores de BD actuales y se siguió la misma línea de interfaz, para que de esta manera el usuario no se enfrentara a algo desconocido.

Se utilizó una metodología iterativa en la cual se fueron agregando funcionalidades en cada iteración, de esta manera se pudo ver desde un inicio la cáscara del plug-in a la cual a lo largo del desarrollo se le agregaron sus funcionalidades poco a poco.

Con lo que respecta a la planificación inicial del proyecto, esta se extendió debido a los problemas encontrados a lo largo del desarrollo, ya sean problemas de conocimientos de programación como de conocimientos técnicos a nivel de configuración y método de desarrollo para un plug-in con lo cual fue necesario realizar investigación y estudio personal.

En lo académico este plug-in logra ser un gran aporte para aquellos programadores que utilizan las BD en sus aplicaciones android, ya que con él se pueden ahorrar bastantes líneas de código, horas de programación y orden en el código, todo esto según la complejidad o tamaño de la BD. Además se logra una aplicación que controle la integridad de los datos y de la BD ya que actualmente SQLite no proporciona esto a los programadores.

En el ámbito personal se puede decir que a lo largo del desarrollo de este proyecto se lograron conocimientos tanto de BD como de programación en java, mencionando además el cómo programar un plug-in para eclipse y como integrarlo al IDE.

Se logró conocimientos avanzados en SQL a nivel de obtener información del esquema de la BD para conocer sus atributos, tipos, claves primarias y foráneas, saber con qué tablas y atributos se relacionaban, etc.

Se logró generar clases de java con las cuales se ofrecen servicios al programador, el cual puede insertar datos y realizar otras operaciones básicas en BD solo con la llamada a un método y no utilizando el engorroso código SQL dentro de java.

10 BIBLIOGRAFÍA

SOMMERVILLE Ian, Ingeniería de Software. 7° ed. Madrid, España, Ediciones Pearson Education 2005. 687p

Roger S. Pressman. Ingeniería de Software un enfoque práctico. 5° ed. Madrid, España, Ediciones Mc Geaw Hill, 2002. 601p.

Grant Allen y Mike Owens. The Definitive Guide to SQLite. 2° ed. E.E.U.U, Ediciones Apress, 2010. 347p.

Ian Gilfilland. La biblia MySQL. Ediciones Anaya. 880p.
Eric Clayberg y Dan Rubel. Eclipse building commercial-quality Plug-ins. 2° ed. Ediciones instantiations, 2008. 184p.

Ed Burnette, Hello. Introducing Google's Mobile Development Platform, Android. 2° ed. UU.EE, 2009. 291p.

The Eclipse Foundation. Eclipse documentation - Archived Release. [En línea] <http://help.eclipse.org/helios/index.jsp> [consulta: 8 octubre 2012]

11 ANEXO 1: PLANIFICACION INICIAL DEL PROYECTO

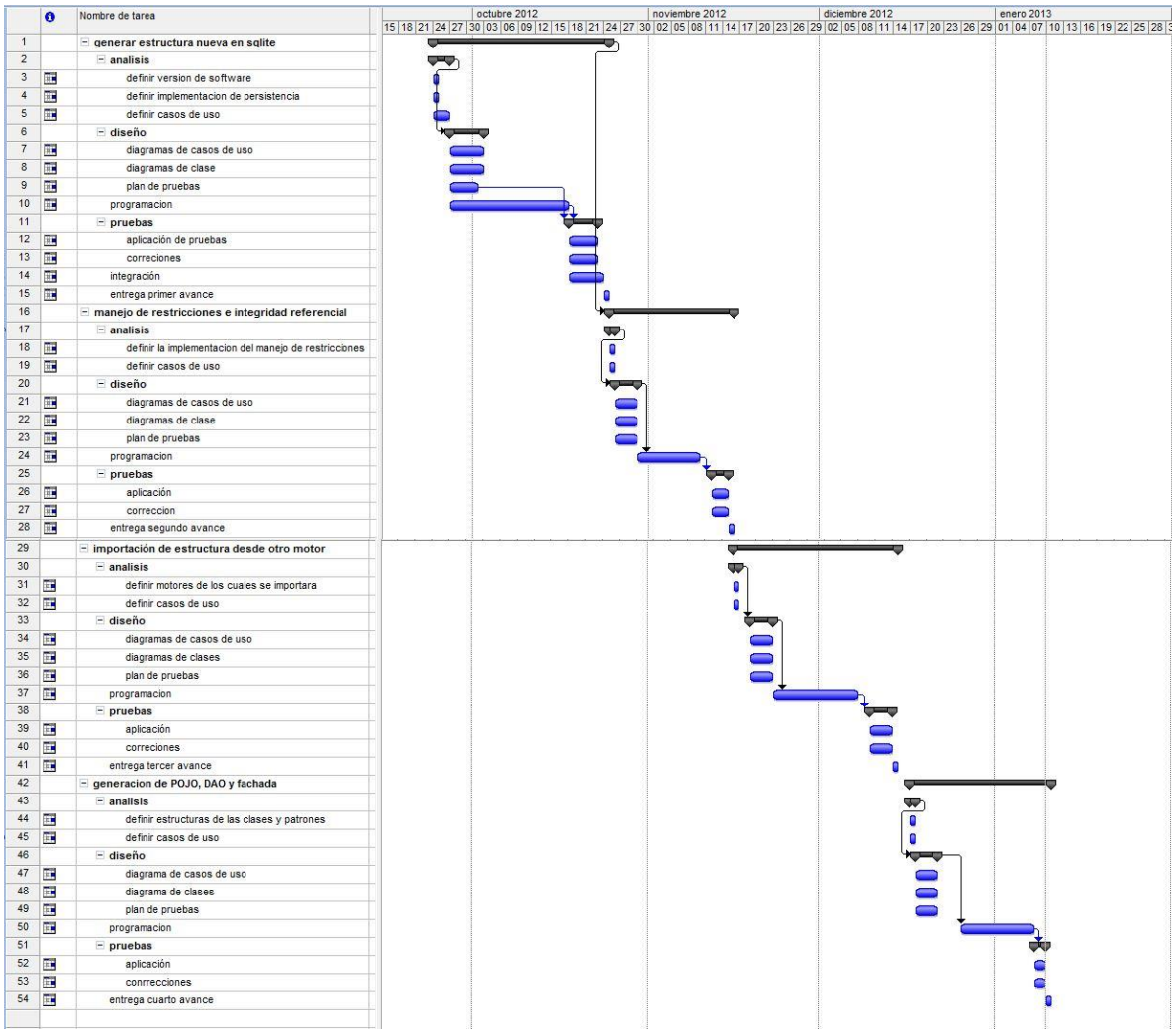


Figura Nº 26: Carta Gantt del Proyecto.

Se presenta el plan de trabajo seguido a lo largo del desarrollo del tema, este plan es representado en la Carta Gantt de la imagen superior. Se utiliza una metodología iterativa, con la cual se pretende, en cuatro iteraciones, terminar el desarrollo de la aplicación. En cada iteración de implementan las siguientes funcionalidades.

Primera iteración: Se implementan las funcionalidades básicas del sistema que son crear una estructura de Base de Datos desde la nada, esto es, creando desde tablas, atributos y claves foráneas una Base de Datos cualquiera. Además se entrega la persistencia de esta en el tiempo, esto implementado como archivos XML, de los cuales se leerá la

estructura para cargarla y en los cuales se guardara la estructura para hacerla persistente. También en esta iteración se debe tener todas las interfaces que se utilizaran en la aplicación aunque no todas tengan funcionalidad.

Segunda iteración: Se implementan las funciones de integridad de las bases de datos, en este caso, tipos de datos, claves foráneas, validaciones y las opciones de validaciones en la inserción de datos a través de triggers en SQL. Todo esto pensado para bases de datos SQLite que son las utilizadas por los dispositivos móviles.

Tercera iteración: Se implementa la opción de importar una Base de Datos externa a SQLite, en este caso se trabaja con MySQL. Esta funcionalidad permite al programador poder conectarse a al motor MySQL, en específico a una Base de Datos que ya exista en dicho motor y que se desee importar al motor compatible con android, copiando la estructura base de la BD importada, esto es, tablas, atributos y claves foráneas, para posteriormente utilizarla en la programación en android.

Cuarta iteración: se implementa la generación de código java, con la cual se podrá generar un paquete con archivos java en la aplicación que se utilice el plug-in. Se generan los archivos DAO, POJO, Fachada y un adminSQLiteOpenHelper, con la finalidad de facilitar la programación al usuario de Base de Datos en sus aplicaciones.

11.1.1 Estimación inicial de tamaño

Puntos de Casos de Uso

Clasificación de actores

Actor	Tipo
Programador	Complejo
BD externa	Medio

Tabla Nº 49: clasificación de actores

Calculo de factor de peso de los actores sin ajustar (UAW)

Tipo	Peso	Cantidad de Actores	resultado
Simple	1	0	0
Medio	2	1	2
complejo	3	1	3
Valor UAW			6

Tabla Nº 50: cálculo de UAW

Clasificación de casos de uso

Caso de Uso	Clases	Nº Transacciones	Tipo
Crear BD	2	2	Simple
Cargar BD	6	4	Medio
Guardar BD	6	4	Medio
Generar Código Java	7	6	Medio
Agregar Tabla	3	2	Simple
Ver Tabla	3	1	Simple
Importar BD	5	4	Medio
Establecer conexión	1	1	Simple
Ver Claves FK	2	2	Simple
Agregar FK	3	3	Simple
Borrar FK	3	3	Simple
Editar FK	4	5	Medio
Agregar Atributo	3	2	Simple
Borrar Tabla	2	2	Simple
Cambiar Nombre Tabla	3	2	Simple
Buscar XML	2	2	Simple
Ver Atributos	1	1	Simple
Editar Atributo	6	6	Medio
Borrar Atributo	2	2	Simple

Tabla Nº 51: Clasificación de casos de uso

Calculo de factor de peso de los casos de uso sin ajustar (UUCW)

Tipo	Peso	Cantidad de Casos de Uso	Resultado en UUCW
Simple	5	13	65
Medio	10	6	60
complejo	15	0	0
Valor UUCW			125

Tabla Nº 52: cálculo de UUCW

Calculo de puntos de caso de uso sin ajustar (UUCP)

UCW	UUCW	Formula	UUCP
6	125	UCW+UUCW=UUCP	131

Tabla Nº 53: cálculo de UUCP

Calculo de factores técnicos

Factor técnico	Descripción	Multiplicador	Influencia	Resultado
1	Sistema distribuido	2	0	0
2	Rendimiento o tiempo de respuesta	1	3	3
3	Eficiencia del usuario final	1	5	5
4	Procesamiento interno complejo	1	4	4
5	El código debe ser reutilizable	1	3	3
6	Facilidad de instalación	0.5	2	1
7	Facilidad de uso	0.5	3	1.5
8	Portabilidad	2	5	10
9	Facilidad de cambio	1	3	3
10	Concurrencia	1	2	2
11	Características especiales de seguridad	1	0	0
12	Provee acceso directo a terceras partes	1	0	0
13	Se requiere facilidades especiales de entrenamiento al usuario	1	1	1
Factor total técnico				33.5

Tabla Nº 54: cálculo de factores técnicos

Rangos de Influencia

Descripción	Valor
Irrelevante	0 a 2
Medio	3 a 4
Esencial	5

Tabla Nº 55: rangos de influencia

Calculo del factor de complejidad técnica (TCF)

Formula	Calculo	Resultado
0.6+(0.01*factor total técnico)	0.6+(0.01*33.5)	0.935

Tabla Nº 56: cálculo de TCF

Calculo de factores de entorno

Factor ambiental	Descripción	multiplicador	Influencia	Resultado
1	Familiaridad con el modelo de proyecto utilizado, familiaridad con UML	1.5	3	4.5
2	Estabilidad de los requerimientos	2	1	2
3	Capacidad del analista líder	0.5	5	2
4	Experiencia en la aplicación	0.5	4	2
5	Experiencia en OO	1	3	3
6	Motivación	1	4	4
7	Dificultad del lenguaje	-1	0	0
8	Personal tiempo parcial	-1	0	0
Factor ambiental total				17.5

Tabla Nº 57: calculo factor ambiental

Calculo del factor de entorno (EF)

Formula	Calculo	Resultado
1.4+(-0.03*factor ambiental total)	1.4+(-0.03*17.5)	0.875

Tabla Nº 58: calculo EF

Calculo de puntos de casos de uso ajustados (UCP)

Formula	Calculo	Resultado
UCP=UUCP*TCF*EF	125*0.935*0.875	102.26

Tabla Nº 59: calculo UCP

Calculo de las horas-hombre

Para este cálculo se utilizara un LOE de 20.

Formula	Calculo	Resultado
UCP*LOE	102.26*20	2045.2

Tabla Nº 60: calculo horas hombre

11.1.2 Contabilización final del tamaño del Sw

11.1.2.1 Cantidad total de líneas de código

Clase	Líneas de código
FocusTraversalOnArray	56
Activator	24
Conversor	71
Control	376
GeneradorDeCodigo	681
Atributo	267
BaseDeDatos	57
ClaveForanea	73
Entidad	238
Clase	177
LineaCodigo	32
Metodo	80
Variable	50
Persistencia	315
ConexionBDExterna	118
CreaeEstructuraBDAction	40
NodoBD	26
JPColor	39
JRTextField	69
MiRender	52
Imagenes	95
Dialogos	132
DialogoImportarMySQL	44
DialogoCargarBD	107
DatosConexion	133
Constantes	12
TablaNueva	70
Tabla	151
ClavesForaneas	187
VentanaClaveForanea	208
AtributoNuevo	355
Verificador	360
Vista	681
TOTAL	5376

Tabla Nº 61: Líneas de código

12 ANEXO 2: COMPARACIÓN

Se realizó un estudio de las ventajas que se pueden obtener al utilizar el plug-in en el desarrollo de aplicaciones. Estos estudios dan a conocer las diferencias en el tiempo de programación y en ahorro de líneas de código que se pueden producir si se utiliza el plug-in para desarrollar.

Para esto se desarrollara una pequeña aplicación android la cual utiliza una pequeña BD que se especificara a continuación:

Tabla Alumno		
columna	**tipo**	**Otras propiedades**
Rut	Text	Valor minimo:0 longitud máxima:8 clave primaria: true
Nombre	Text	
Apellido	Text	
Anio_nacimiento	Integer	Valor mínimo: 1980 Valor máximo: 2010
Curso	Text	
Nombre_colegio	Text	Clave foránea

Tabla Colegio		
columna	**tipo**	**Otras propiedades**
Nombre	Text	clave primaria: true
Dirección	Text	
Anio_fundacion	Integer	Valor mínimo: 1500 Valor máximo: 2013
Director	Text	

Tabla Nº 62: Definición BD prueba.

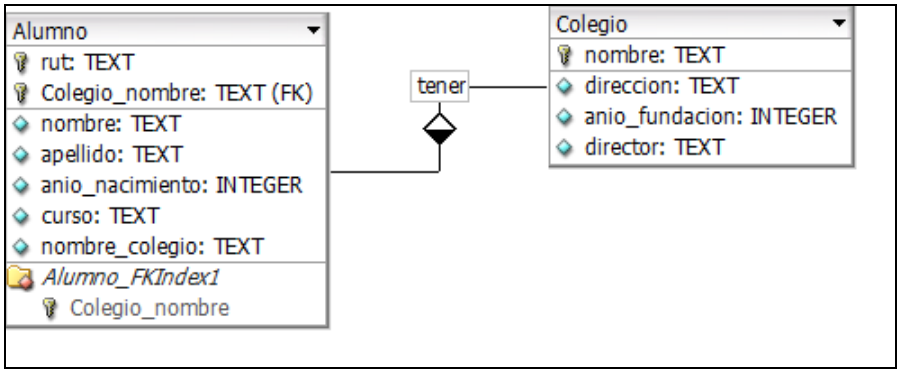


Figura Nº 27: Diagrama BD de prueba

Observaciones:

1. Ningún valor debe ser null.
2. No se debe permitir romper las restricciones.
3. No se debe permitir crear un alumno en un colegio que no existe.
4. No debe permitir modificar al alumno para ponerle un alumno que no existe.
5. Si se borra el colegio se deben borrar sus alumnos.

La aplicación android es una pequeña app que permite realizar operaciones básicas en BD, inserción, consulta, eliminación y actualización. Además genera pequeños informes. Para esta prueba se tiene las mismas clases para ambos desarrollo con la única diferencia que una aplicación se desarrollara con ayuda del plug-in y otra se desarrollara solo con programación manual.

Se tienen las clases siguientes clases en el proyecto pestanaalumno, pestanacolegio, pestanalistaalumnos y en estas se debe agregar el código sql en el caso de programación manual y las llamadas a las clases generadas (Fachada) en el caso de la programación con plug-in.

12.1 Tiempos

12.1.1 Implementación y tiempos manuales

Para la implementación de la aplicación manual se obtuvieron los siguientes resultados, en la generación de la clase adminSQLite, se obtuvo un tiempo de 27 minutos para la implementación de la clase y 46 minutos para poder terminar la aplicación utilizando este código.

```
package tesis.prueba;
import android.content.ContentValues;
import android.content.Context;
import android.database.Cursor;
import android.database.sqlite.SQLiteDatabase;
import android.database.sqlite.SQLiteDatabase.CursorFactory;
import android.database.sqlite.SQLiteOpenHelper;
import android.util.Log;
import android.widget.Toast;
```

```
public class AdminSQLiteOpenHelper extends SQLiteOpenHelper {
  public AdminSQLiteOpenHelper(
          Context context,
          String nombre,
          CursorFactory factory,
          int version){
          super(context, nombre, factory, version);
  }
@Override
public void onCreate(SQLiteDatabase db){

  db.execSQL("create table ALUMNO(RUT INTEGER NOT NULL, NOMBRE TEXT NOT NULL, APELLIDO
  TEXT NOT NULL, ANIO_NACIMIENTO INTEGER NOT NULL, CURSO TEXT NOT NULL, NOMBRE_COLEGIO
  TEXT NOT NULL, PRIMARY KEY(RUT))");

  db.execSQL("CREATE TRIGGER ALUMNO_COLEGIO BEFORE INSERT ON ALUMNO FOR EACH ROW BEGIN
  SELECT CASE WHEN(( NEW.NOMBRE_COLEGIO IS NOT NULL) AND ((SELECT NOMBRE FROM COLEGIO
  WHERE NOMBRE= NEW.NOMBRE_COLEGIO) IS NULL))THEN RAISE(ABORT, 'insert on table
  \"ALUMNO\" violates foreign key \"ALUMNO_COLEGIO\"') END; END;");

  db.execSQL("CREATE TRIGGER ALUMNO_COLEGIOUpdate BEFORE UPDATE ON ALUMNO FOR EACH ROW
  BEGIN SELECT CASE WHEN(( NEW.NOMBRE_COLEGIO IS NOT NULL) AND ((SELECT NOMBRE FROM
  COLEGIO WHERE NOMBRE= NEW.NOMBRE_COLEGIO) IS NULL))THEN RAISE(ABORT, 'update on table
  \"ALUMNO\" violates foreign key \"ALUMNO_COLEGIO\"') END; END;");

  db.execSQL("CREATE TRIGGER ALUMNO_RUTLongitudInsert BEFORE INSERT ON ALUMNO FOR EACH
  ROW BEGIN SELECT CASE WHEN((NEW.RUT IS NOT NULL) AND ( length(NEW.RUT)>8))THEN
  RAISE(ABORT, 'insert on table \"ALUMNO\" violates length') END; END;");

  db.execSQL("CREATE TRIGGER ALUMNO_RUTLongitudUpdate BEFORE UPDATE ON ALUMNO FOR EACH
  ROW BEGIN SELECT CASE WHEN((NEW.RUT IS NOT NULL) AND ( length(NEW.RUT)>8))THEN
  RAISE(ABORT, 'update on table \"ALUMNO\" violates length') END; END;");

  db.execSQL("CREATE TRIGGER ALUMNO_ANIO_NACIMIENTOInsertMax BEFORE INSERT ON ALUMNO FOR
  EACH ROW BEGIN SELECT CASE WHEN((NEW.ANIO_NACIMIENTO IS NOT NULL) AND (
  NEW.ANIO_NACIMIENTO > 2010))THEN RAISE(ABORT, 'insert on table \"ALUMNO\" violates max
  value') END; END;");

  db.execSQL("CREATE TRIGGER ALUMNO_ANIO_NACIMIENTOUpdateMax BEFORE UPDATE ON ALUMNO FOR
  EACH ROW BEGIN SELECT CASE WHEN((NEW.ANIO_NACIMIENTO IS NOT NULL) AND (
  NEW.ANIO_NACIMIENTO > 2010))THEN RAISE(ABORT, 'update on table \"ALUMNO\" violates max
  value') END; END;");

  db.execSQL("CREATE TRIGGER ALUMNO_ANIO_NACIMIENTOInsertMin BEFORE INSERT ON ALUMNO FOR
  EACH ROW BEGIN SELECT CASE WHEN((NEW.ANIO_NACIMIENTO IS NOT NULL) AND (
  NEW.ANIO_NACIMIENTO < 1980))THEN RAISE(ABORT, 'insert on table \"ALUMNO\" violates min
  value') END; END;");

  db.execSQL("CREATE TRIGGER ALUMNO_ANIO_NACIMIENTOUpdateMin BEFORE UPDATE ON ALUMNO FOR
  EACH ROW BEGIN SELECT CASE WHEN((NEW.ANIO_NACIMIENTO IS NOT NULL) AND (
  NEW.ANIO_NACIMIENTO < 1980))THEN RAISE(ABORT, 'update on table \"ALUMNO\" violates min
  value') END; END;");

  db.execSQL("CREATE TRIGGER ALUMNO_ANIO_NACIMIENTOLongitudInsert BEFORE INSERT ON
  ALUMNO FOR EACH ROW BEGIN SELECT CASE WHEN((NEW.ANIO_NACIMIENTO IS NOT NULL) AND (
  length(NEW.ANIO_NACIMIENTO)>4))THEN RAISE(ABORT, 'insert on table \"ALUMNO\" violates
  length') END; END;");

  db.execSQL("CREATE TRIGGER ALUMNO_ANIO_NACIMIENTOLongitudUpdate BEFORE UPDATE ON
  ALUMNO FOR EACH ROW BEGIN SELECT CASE WHEN((NEW.ANIO_NACIMIENTO IS NOT NULL) AND (
```

```
 length(NEW.ANIO_NACIMIENTO)>4))THEN RAISE(ABORT, 'update on table \"ALUMNO\" violates
 length') END; END;");

 db.execSQL("create table COLEGIO(NOMBRE TEXT NOT NULL, DIRECCION TEXT NOT NULL,
 ANIO_FUNDACION INTEGER NOT NULL, DIRECTOR TEXT NOT NULL, PRIMARY KEY(NOMBRE))");

 db.execSQL("CREATE TRIGGER COLEGIO_ANIO_FUNDACIONInsertMax BEFORE INSERT ON COLEGIO
 FOR EACH ROW BEGIN SELECT CASE WHEN((NEW.ANIO_FUNDACION IS NOT NULL) AND (
 NEW.ANIO_FUNDACION > 2013))THEN RAISE(ABORT, 'insert on table \"COLEGIO\" violates max
 value') END; END;");

 db.execSQL("CREATE TRIGGER COLEGIO_ANIO_FUNDACIONUpdateMax BEFORE UPDATE ON COLEGIO
 FOR EACH ROW BEGIN SELECT CASE WHEN((NEW.ANIO_FUNDACION IS NOT NULL) AND (
 NEW.ANIO_FUNDACION > 2013))THEN RAISE(ABORT, 'update on table \"COLEGIO\" violates max
 value') END; END;");

 db.execSQL("CREATE TRIGGER COLEGIO_ANIO_FUNDACIONInsertMin BEFORE INSERT ON COLEGIO
 FOR EACH ROW BEGIN SELECT CASE WHEN((NEW.ANIO_FUNDACION IS NOT NULL) AND (
 NEW.ANIO_FUNDACION < 1500))THEN RAISE(ABORT, 'insert on table \"COLEGIO\" violates min
 value') END; END;");

 db.execSQL("CREATE TRIGGER COLEGIO_ANIO_FUNDACIONUpdateMin BEFORE UPDATE ON COLEGIO
 FOR EACH ROW BEGIN SELECT CASE WHEN((NEW.ANIO_FUNDACION IS NOT NULL) AND (
 NEW.ANIO_FUNDACION < 1500))THEN RAISE(ABORT, 'update on table \"COLEGIO\" violates min
 value') END; END;");

 db.execSQL("CREATE TRIGGER COLEGIO_ANIO_FUNDACIONLongitudInsert BEFORE INSERT ON
 COLEGIO FOR EACH ROW BEGIN SELECT CASE WHEN((NEW.ANIO_FUNDACION IS NOT NULL) AND (
 length(NEW.ANIO_FUNDACION)>4))THEN RAISE(ABORT, 'insert on table \"COLEGIO\" violates
 length') END; END;");

 db.execSQL("CREATE TRIGGER COLEGIO_ANIO_FUNDACIONLongitudUpdate BEFORE UPDATE ON
 COLEGIO FOR EACH ROW BEGIN SELECT CASE WHEN((NEW.ANIO_FUNDACION IS NOT NULL) AND (
 length(NEW.ANIO_FUNDACION)>4))THEN RAISE(ABORT, 'update on table \"COLEGIO\" violates
 length') END; END;");

}
@Override
public void onUpgrade(SQLiteDatabase db,       int versionAnte,       int versionNue){
 db.execSQL("drop table if exists ALUMNO");

 db.execSQL("create table ALUMNO(RUT INTEGER NOT NULL, NOMBRE TEXT NOT NULL, APELLIDO
 TEXT NOT NULL, ANIO_NACIMIENTO INTEGER NOT NULL, CURSO TEXT NOT NULL, NOMBRE_COLEGIO
 TEXT NOT NULL, PRIMARY KEY(RUT))");

 db.execSQL("drop table if exists COLEGIO");
        db.execSQL("create table COLEGIO(NOMBRE TEXT NOT NULL, DIRECCION TEXT NOT NULL,
 ANIO_FUNDACION INTEGER NOT NULL, DIRECTOR TEXT NOT NULL, PRIMARY KEY(NOMBRE))");

}

}
```

12.1.2 Implementación y tiempos con plug-in

En el caso de la implementación con el plug-in los tiempos fueron de 5 minutos para la implementación del adminSQLite y de 40 minutos para terminar la aplicación utilizando las clases generadas.

En este caso se genera una clase que manualmente no se tiene a menos que se programe. Esta clase 'Fachada' es la que se encarga de proporcionar los métodos con los cuales se puede hacer las operaciones básicas en BD (insertar, eliminar, actualizar, consultar), sin necesidad de implementar código SQL.

Esta clase se detalla a continuación:

```
package tesis.prueba.daos;
import java.util.List;
import java.util.ArrayList;
import java.util.Iterator;
import tesis.prueba.pojos.*;
import tesis.prueba.*;

public class Fachada {

 public AdminSQLiteOpenHelper helper;
 public Fachada(AdminSQLiteOpenHelper helper){
        this.helper=helper;
 }

 public void insert(Object pojo){

        if(pojo instanceof Persona){
               PersonaDAO dao=new PersonaDAO(helper);
               dao.insert((Persona)pojo);
        }
        if(pojo instanceof Vehiculo){
               VehiculoDAO dao=new VehiculoDAO(helper);
               dao.insert((Vehiculo)pojo);
        }
        if(pojo instanceof Casa){
               CasaDAO dao=new CasaDAO(helper);
               dao.insert((Casa)pojo);
        }
 }

 public void update(Object pojo){
        if(pojo instanceof Persona){
               PersonaDAO dao=new PersonaDAO(helper);
               dao.update((Persona)pojo);
        }
        if(pojo instanceof Vehiculo){
               VehiculoDAO dao=new VehiculoDAO(helper);
               dao.update((Vehiculo)pojo);
        }
        if(pojo instanceof Casa){
               CasaDAO dao=new CasaDAO(helper);
               dao.update((Casa)pojo);
        }
 }
```

```
public void delete(Object pojo){

	if(pojo instanceof Persona){
		PersonaDAO dao=new PersonaDAO(helper);
		dao.delete((Persona)pojo);
	}
	if(pojo instanceof Vehiculo){
		VehiculoDAO dao=new VehiculoDAO(helper);
		dao.delete((Vehiculo)pojo);
	}
	if(pojo instanceof Casa){
		CasaDAO dao=new CasaDAO(helper);
		dao.delete((Casa)pojo);
	}
}

public void insertPersona(String columnas,String values){
	PersonaDAO dao=new PersonaDAO(helper);
	dao.insert(columnas, values);
}

public void updatePersona(String set,String condiciones){
	PersonaDAO dao=new PersonaDAO(helper);
	dao.update(set, condiciones);
}

public void deletePersona(String condiciones){
	PersonaDAO dao=new PersonaDAO(helper);
	dao.delete(condiciones);
}

public List<Persona> selectAllPersona(){
	PersonaDAO dao=new PersonaDAO(helper);
	return dao.selectAll();
}

public List<Persona> selectPersona(String condiciones){
	PersonaDAO dao=new PersonaDAO(helper);
	return dao.select(condiciones);
}

public void insertVehiculo(String columnas,String values){
	VehiculoDAO dao=new VehiculoDAO(helper);
	dao.insert(columnas, values);
}

public void updateVehiculo(String set,String condiciones){
	VehiculoDAO dao=new VehiculoDAO(helper);
	dao.update(set, condiciones);
}

public void deleteVehiculo(String condiciones){
	VehiculoDAO dao=new VehiculoDAO(helper);
	dao.delete(condiciones);
}

public List<Vehiculo> selectAllVehiculo(){
	VehiculoDAO dao=new VehiculoDAO(helper);
	return dao.selectAll();
}
```

```
public List<Vehiculo> selectVehiculo(String condiciones){
        VehiculoDAO dao=new VehiculoDAO(helper);
        return dao.select(condiciones);
}

public void insertCasa(       String columnas,       String values){
        CasaDAO dao=new CasaDAO(helper);
        dao.insert(columnas, values);
}

public void updateCasa(       String set,String condiciones){
        CasaDAO dao=new CasaDAO(helper);
        dao.update(set, condiciones);
}

public void deleteCasa(       String condiciones){
        CasaDAO dao=new CasaDAO(helper);
        dao.delete(condiciones);
}

public List<Casa> selectAllCasa(){
        CasaDAO dao=new CasaDAO(helper);
        return dao.selectAll();
}

public List<Casa> selectCasa(String condiciones){
        CasaDAO dao=new CasaDAO(helper);
        return dao.select(condiciones);
}

}
```

12.1.3 Comparación final de tiempo

Manual		**Con plug-in**	
Clase	**Tiempo (min)**	**Clase**	**Tiempo (min)**
adminSQLite	27	adminSQLite	5
Aplicación	46	Aplicación	30
Total desarrollo	***73***	***Total desarrollo***	***35***

Tabla Nº 63: Tiempo final de desarrollo.

12.2 Código

En esta subsección estudiaremos el ahorro de código que se produce al utilizar el plug-in en el desarrollo de aplicaciones con BD. Para esto primeramente explicaremos la diferencia en el código a la hora de realizar las operaciones básicas en BD.

Operación	Manual	Plug-in
Inserción	bd.execSQL("INSERT INTO ALUMNO (RUT, NOMBRE, APELLIDO, ANIO_NACIMIENTO, CURSO, NOMBRE_COLEGIO) VALUES ("+alumno.getRut()+", '"+alumno.getNombre()+"', '"+alumno.getApellido()+"', "+alumno.getAnioNacimiento()+", '"+alumno.getCurso()+"', '"+alumno.getNombreColegio()+"');");	fachada.insertAlumno((Alumno.RUT+" , "+Alumno.NOMBRE+", "+Alumno.APELLIDO+", "+Alumno.ANIO_NACIMIENTO+", "+Alumno.CURSO+", "+Alumno.NOMBRE_COLEGIO), (rut+"', '"+nombre+"', "+"', '"+apellido+"', "+"', "+anioNacimiento+", "+"', '"+curso+"', "+"', '"+nombreColegio));
Eliminación	bd.execSQL("DELETE FROM ALUMNO WHERE RUT="+alumno.getRut()+" ");	fachada.deleteAlumno(Alumno.RUT+"= "+rut);
Consulta	bd.rawQuery("SELECT * FROM ALUMNO WHERE "+condiciones+" ", null);	fachada.selectAlumno(Alumno.RUT+"= "+rut);
Actualización	bd.execSQL("UPDATE ALUMNO SET RUT="+alumno.getRut()+", NOMBRE='"+alumno.getNombre()+"', APELLIDO='"+alumno.getApellido()+"', ANIO_NACIMIENTO="+alumno.getAnioNacimiento()+", CURSO='"+alumno.getCurso()+"', NOMBRE_COLEGIO='"+alumno.getNombreColegio()+"' WHERE RUT="+alumno.getRut()+" ");	fachada.updateAlumno(Alumno.APELLIDO+"='Araya', "+Alumno.RUT+"=17343755", "+Alumno.RUT+"=17090851 AND "+Alumno.NOMBRE+"='Gabriel'");

Tabla Nº 64: Comparación de código

De esta manera podemos observar las diferencias a la hora de realizar las operaciones básicas. Manualmente tendremos que escribir bastante código SQL, y con ayuda del plug-in solo tenemos que llamar a los métodos de la Fachas con los parámetros que necesitemos, ahorrándonos incrustar código SQL.

12.3 Pruebas

El objetivo de esta prueba es principalmente ver el rendimiento con grandes cantidades de tablas y columnas, usando cantidades que no se verán en una app real 100 tablas con 50 columnas c/u, 99 tablas con 1 FK, cada tabla con una PK simple.

Para este proceso se tomó en tiempo para la generación de código 10747 milisegundos en generar todas las clases.

Clase Generada	Numero de Clases	Promedio de Líneas de código por DAO	Código Total de las Clases
DAOs	100	496	49590
POJOs	100	479	47874
Fachada	1	3822	3822
AdminSQLHelper	1	526	526
		Total	101812

Tabla Nº 65: Especificación de código para 100 tablas.

Las siguientes pruebas tienen por objetivo medir la cantidad de código generado, el cual se le evita al programador escribir manualmente, usando cantidades de tablas y columnas razonables para una aplicación móvil.

1) 10 tablas con 5 columnas c/u, 9 tablas con 1 FK, cada tabla con una PK simple. Para este proceso le tomó en tiempo para la generación de código, 501 milisegundos. en generar todas las clases.

Clase Generada	Numero de Clases	Promedio de Líneas de código por DAO	Código Total de las Clases
DAOs	10	135	1350
POJOs	10	71	714
Fachada	1	408	408
AdminSQLHelper	1	76	76
		Total	2542

Tabla Nº 66: Especificación de código para 10 tablas.

2) 5 tablas con 5 columnas c/u, 4 tablas con 1 FK, cada tabla con una PK simple. Para este proceso le tomo en tiempo para la generación de código, 269 ms. en generar todas las clases.

Clase Generada	Numero de Clases	Promedio de Líneas de código por DAO	Código Total de las Clases
DAOs	5	134	670
POJOs	5	69	344
Fachada	1	212	408
AdminSQLHelper	1	51	76
		Total	1277

Tabla Nº 67: Especificación de código para 5 tablas.

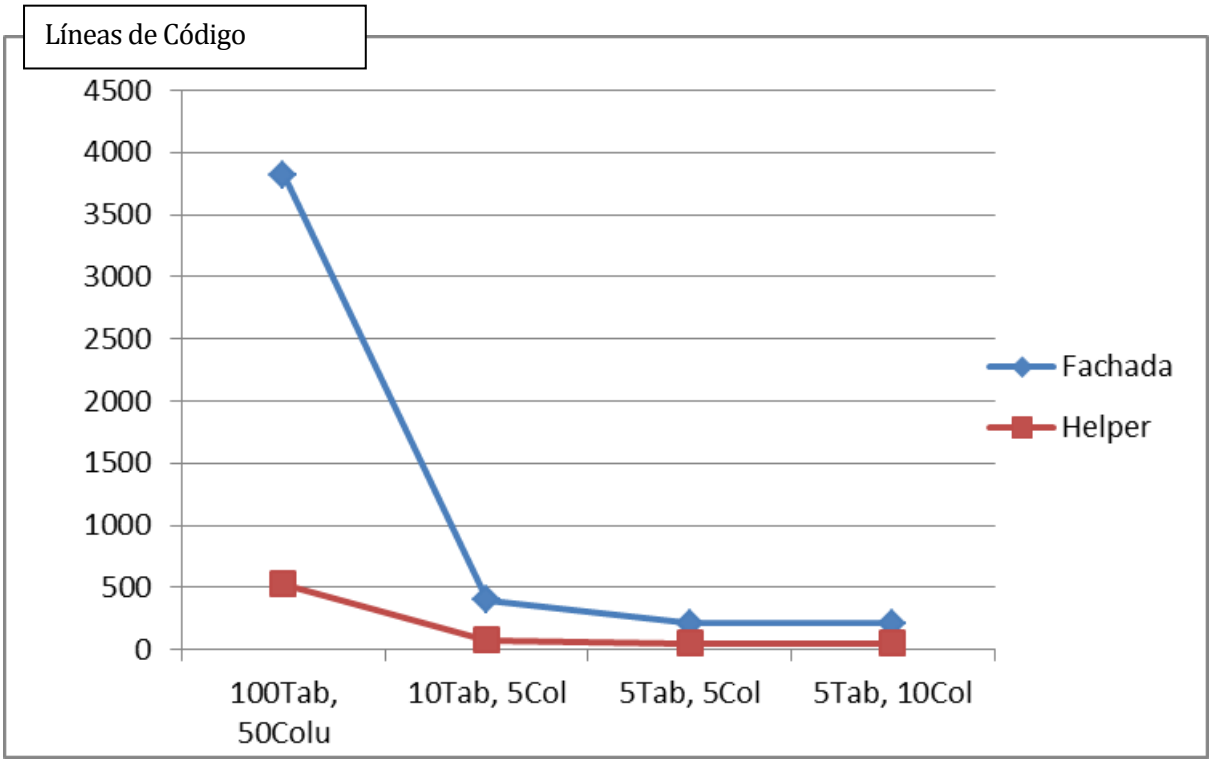


Figura Nº 28: Grafico de Líneas de código generado en las clases Fachada y Helper en cada prueba.

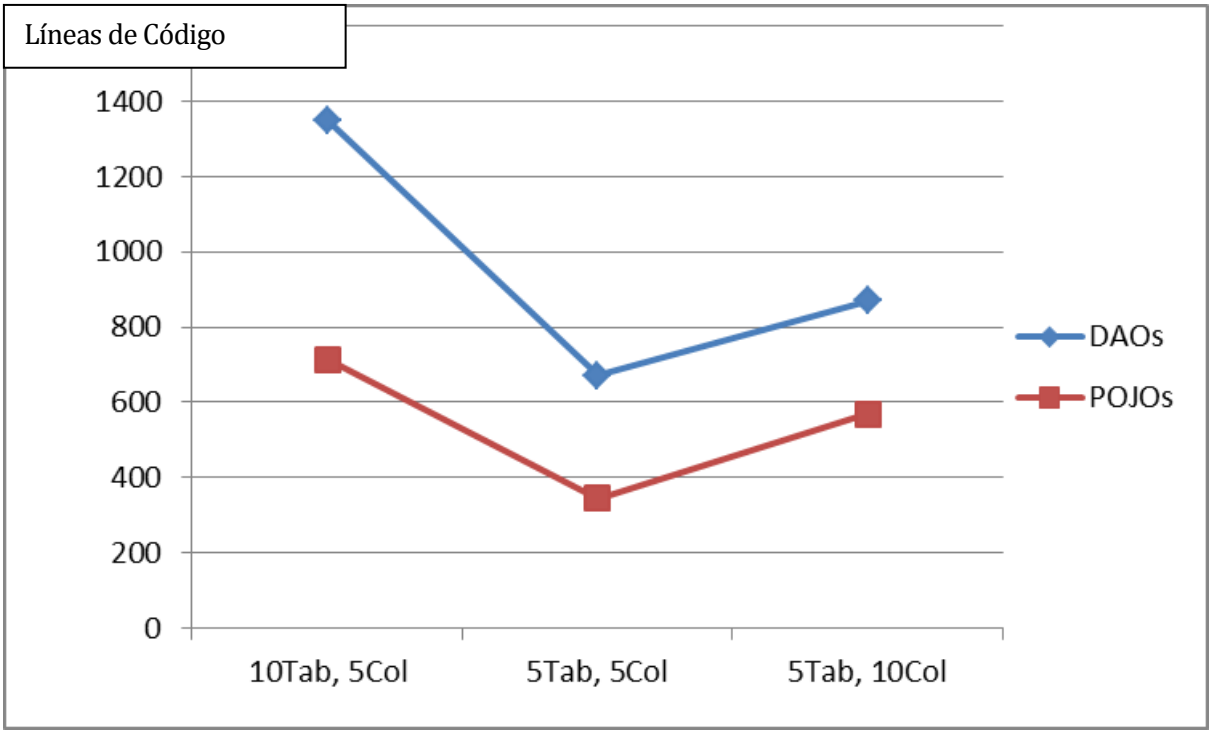


Figura Nº 29: Grafico de Líneas de código total generadas entre todas las clases DAOs y POJOs en cada prueba.

13 ANEXO 3: DICCIONARIO DE DATOS DEL MODELO DE DATOS

TAG	SUBTAGS	TIPO	DESCRIPCIÓN
baseDe Datos	nombreBD	Texto	Atributo de tipo texto el cual tiene como finalidad guardar el nombre de la BD. Solo se especifica uno por archivo XML.
Tabla	nombreTabla	Texto	Atributo de tipo texto que tiene como finalidad guardar el nombre de la tabla. Existen tantos como tablas poséala BD en el archivo XML.
Atributo	nombreColumna	Texto	Atributo de tipo texto que tiene como finalidad albergar el nombre del atributo. Existen tantos como atributos tenga la tabla en el archivo XML.
	tipo	Texto	Atributo que almacena el tipo de dato del atributo el cual puede ser del tipo 'integer', 'real' o 'text'.
	valorMaximo	Numérico	Atributo numérico el cual almacena el valor máximo que puede llegar a alcanzar el atributo.
	valorMinimo	Numérico	Atributo numérico el cual almacena el valor mínimo que puede llegar a alcanzar el atributo.
	valorPorDefecto	Texto	Atributo de tipo texto que almacena el valor que puede tomar el atributo cuando no se especifica un dato de entrada. Este puede ser numérico o cadena.
	clavePrimaria	Binario	Atributo de tipo binario o booleano en el cual se especifica si el atributo es clave primaria o no. Como se guarda en un archivo XML este se especifica de la manera 'true' o 'false'.
	aceptaNull	Binario	Atributo de tipo binario o booleano en el cual se especifica si el atributo puede aceptar valores null o no. Como se guarda en un archivo XML este se especifica de la manera 'true' o 'false'.
	valorInicialIncremento	Numérico	Atributo de tipo numérico en el cual se almacena el valor inicial con el cual comenzara el autoincremento el cual va de uno en uno.
	longitudMaxima	Numérico	Valor de tipo numérico en el cual se especifica cual es la longitud máxima que puede alcanzar el atributo en

			capacidad de caracteres.
	incluyeMinimo	Binario	Atributo de tipo binario con el cual se define si el atributo incluye o no el valor mínimo especificado anteriormente.
	incluyeMaximo	Binario	Atributo de tipo binario con el cual se define si el atributo incluye o no el valor máximo especificado anteriormente.
ClaveForanea	nombreFK	Texto	Atributo de tipo texto en el cual se almacena el nombre de la clave foránea.
	tablaOrigen	Texto	Atributo de tipo texto en el cual se almacena el nombre de la tabla en la cual se creó la clave foránea.
	columnaOrigen	Texto	Atributo de tipo texto en el cual se especifica el nombre de la columna que se declara como clave foránea.
	tablaDestino	Texto	Atributo de tipo texto en el cual se almacena el nombre de la tabla a la cual se encuentra el atributo que se referenciara.
	columnaDestino	Texto	Atributo de tipo texto en el cual se especifica el nombre de la columna a la que se hace referencia desde el atributo de origen.

Tabla Nº 68: Diccionario de datos.

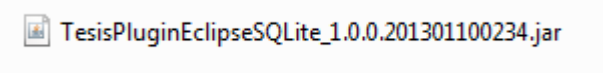
14 ANEXO 4: MANUAL DE USUARIO

Manual de Usuario

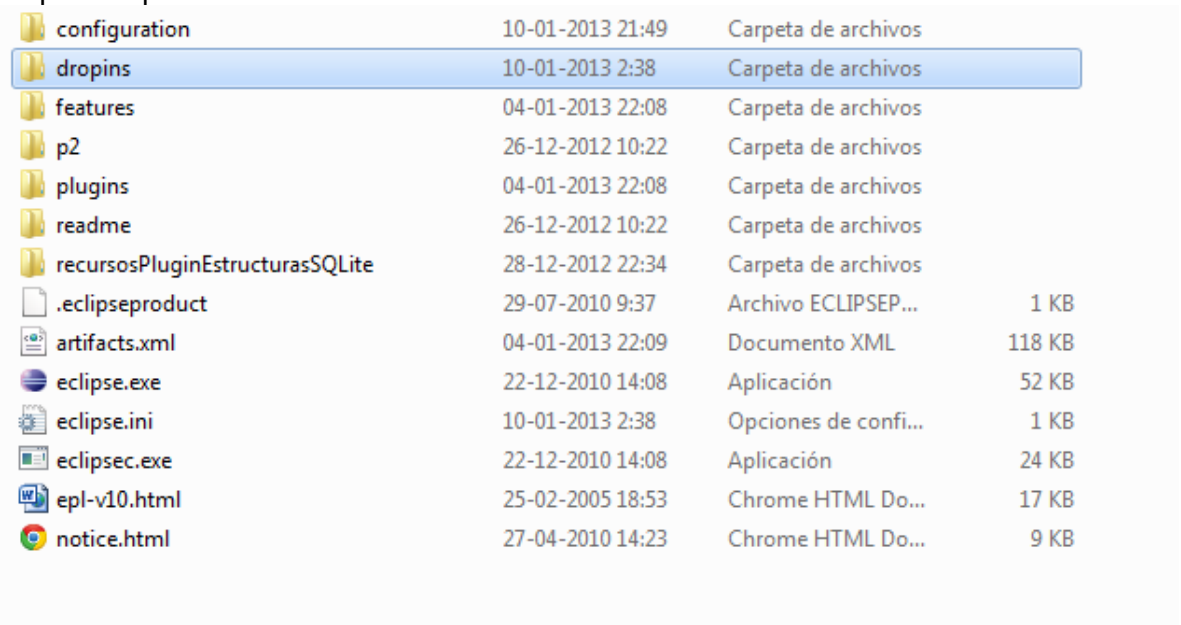
Índice

1 INSTALACIÓN DE PLUG-IN

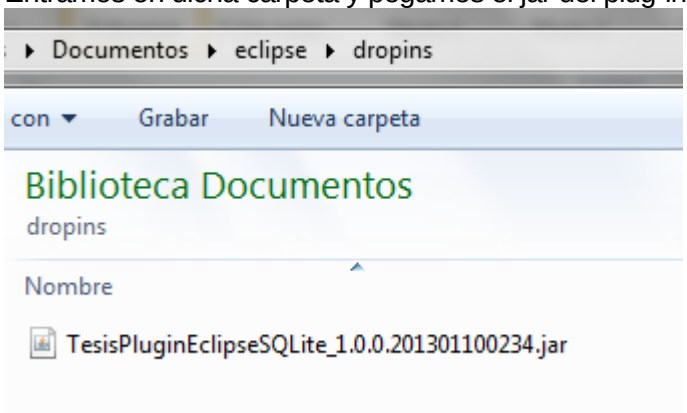
El archivo del plug-in para eclipse que tenemos que instalar es:



Para instalarlo simplemente vamos a la carpeta donde tenemos eclipse y buscamos la carpeta dropins



Entramos en dicha carpeta y pegamos el jar del plug-in:



Luego ejecutamos eclipse normalmente y el plug-in estará instalado.

2 ACCESO AL PLUG-IN

Para tener acceso al plug-in de administración de SQLite para eclipse, en primer lugar se debe crear un proyecto android, el cual debe contener algún paquete de desarrollo, sobre este paquete se pulsa botón derecho del mouse y en el menú contextual que aparece se busca el ítem “SQLite” y luego “Crear Estructura BD” como se muestra en la Figura 1.

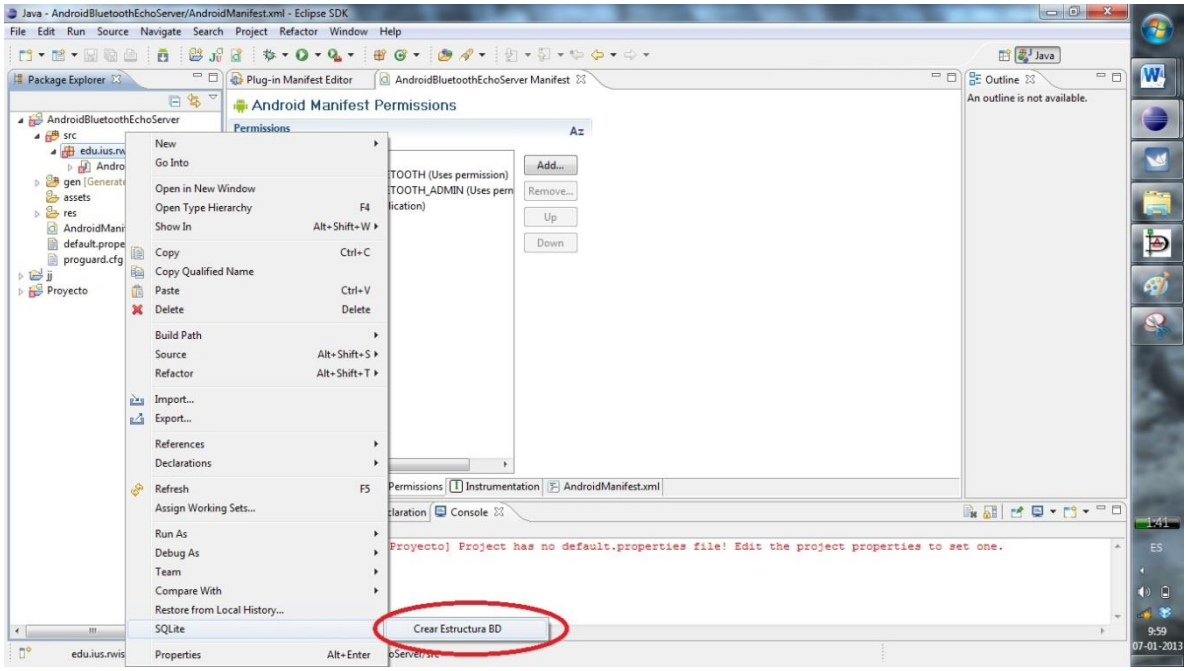


Figura 1: Acceso al plug-in

Cuando se abra la aplicación se mostrará una pantalla como la de la Figura 2, donde la aplicación aparece con una nueva BD creada la cual se puede renombrar si se desea.

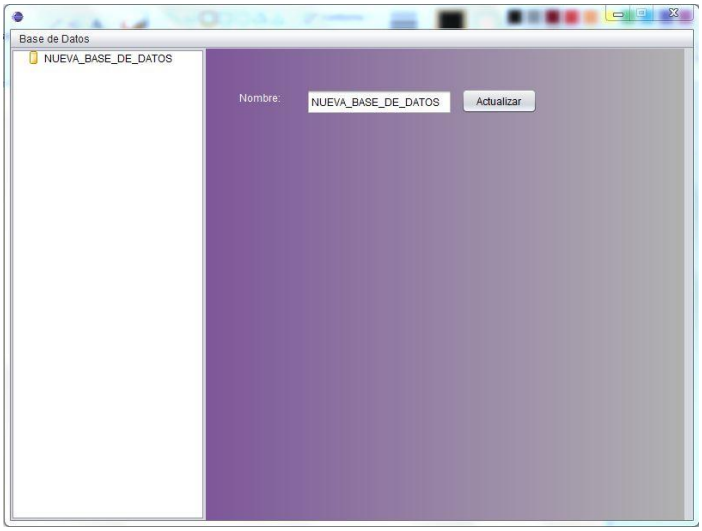


Figura 2: pantalla inicial del plug-in

3 NUEVA BD

Para crear una nueva BD se debe acceder al menú de la ventana (Figura 3) y presionar la opción “Nueva BD” también posee un acceso rápido presionando las teclas Crtl+N, al presionar esta opción, sino no ha guardado la Base de Datos en la que está trabajando emitirá un mensaje de advertencia (Figura 4), de lo contrario se creará automáticamente una nueva BD y se dará la opción para cambiarle nombre ya que el nombre se le asigna automáticamente. Si cambia el nombre de la BD debe presionar el botón “Actualizar” para guardar el nuevo nombre de la BD. En la Figura 4 se muestra la interfaz de una nueva BD, donde con un círculo negro se muestra la Base de Datos creada en el árbol de estructura, en color rojo se muestra el campo para cambiar el nombre y en verde se muestra el botón para guardar los cambios.

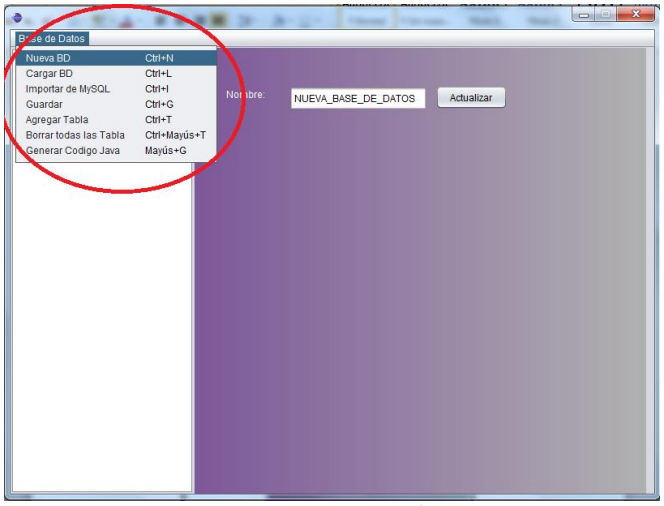


Figura 3: Menú

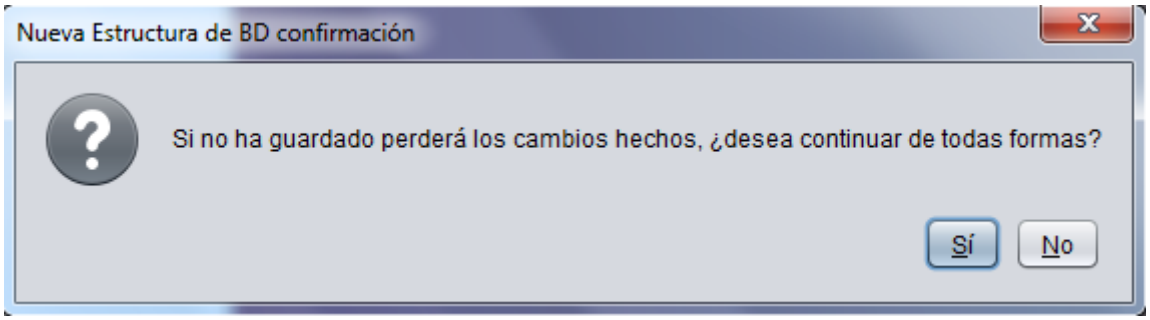


Figura 4: Advertencia de al crear una Nueva BD

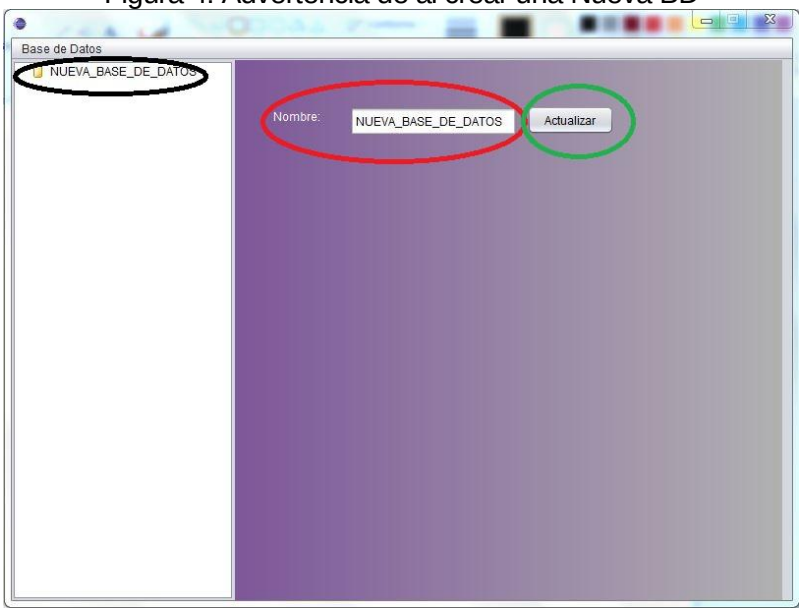


Figura 5: Nueva BD

4 CARGAR BD

Para poder cargar una BD que se encuentre guardada en el proyecto, se debe acceder al menú de la ventana principal y presionar la opción “Cargar BD”, al hacer esto aparecerá una ventana emergente (Figura 5) en la cual se debe seleccionar de una lista (destacado en rojo en la Figura 5) el archivo que se desea cargar, una vez seleccionado se debe presionar el botón “OK” (destacado en negro en la Figura 5) para poder cargar el archivo. Si se desea declinar la operación se debe presionar el botón “Cancelar” (destacado en verde en la Figura 5).

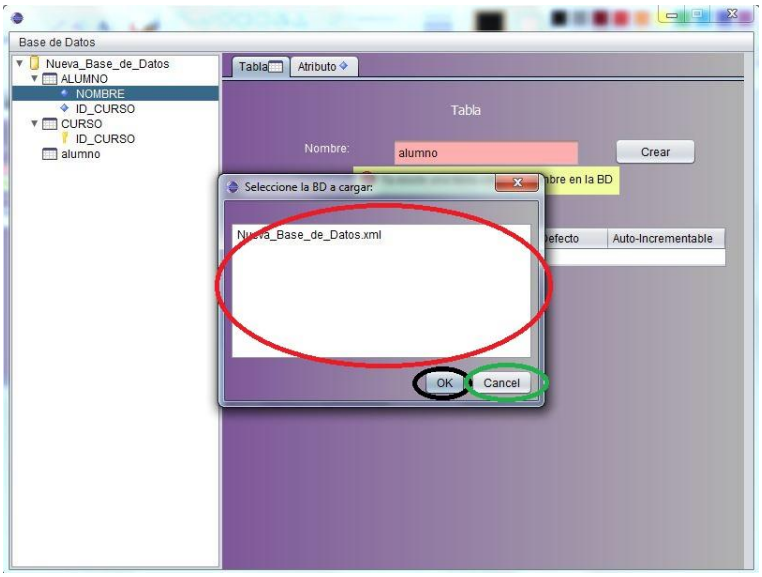


Figura 6: Cargar BD

5 IMPORTAR DE MYSQL

Para poder importar una BD desde el motor de MySQL, se escoge la opción de "Importar de MySQL" desde la barra de menú (Figura 3) o con el acceso rápido Ctrl+I, una vez se presiona la opción aparece una ventana emergente donde se solicitan los datos de conexión como son nombre de usuario, contraseña, puerto, etc. Como se ve en la Figura 7, una vez ingresados los datos de conexión se puede hacer dos acciones, testear la conexión y conectarse, si se procede a testear la conexión se mostrara el resultado en el área de notificación de la ventana (resaltado en rojo), mostrando si la conexión es exitosa o errónea, posteriormente se debe proceder a conectar al hacer esto automáticamente se obtiene la Base de Datos desde MyQSL y se carga en la aplicación.

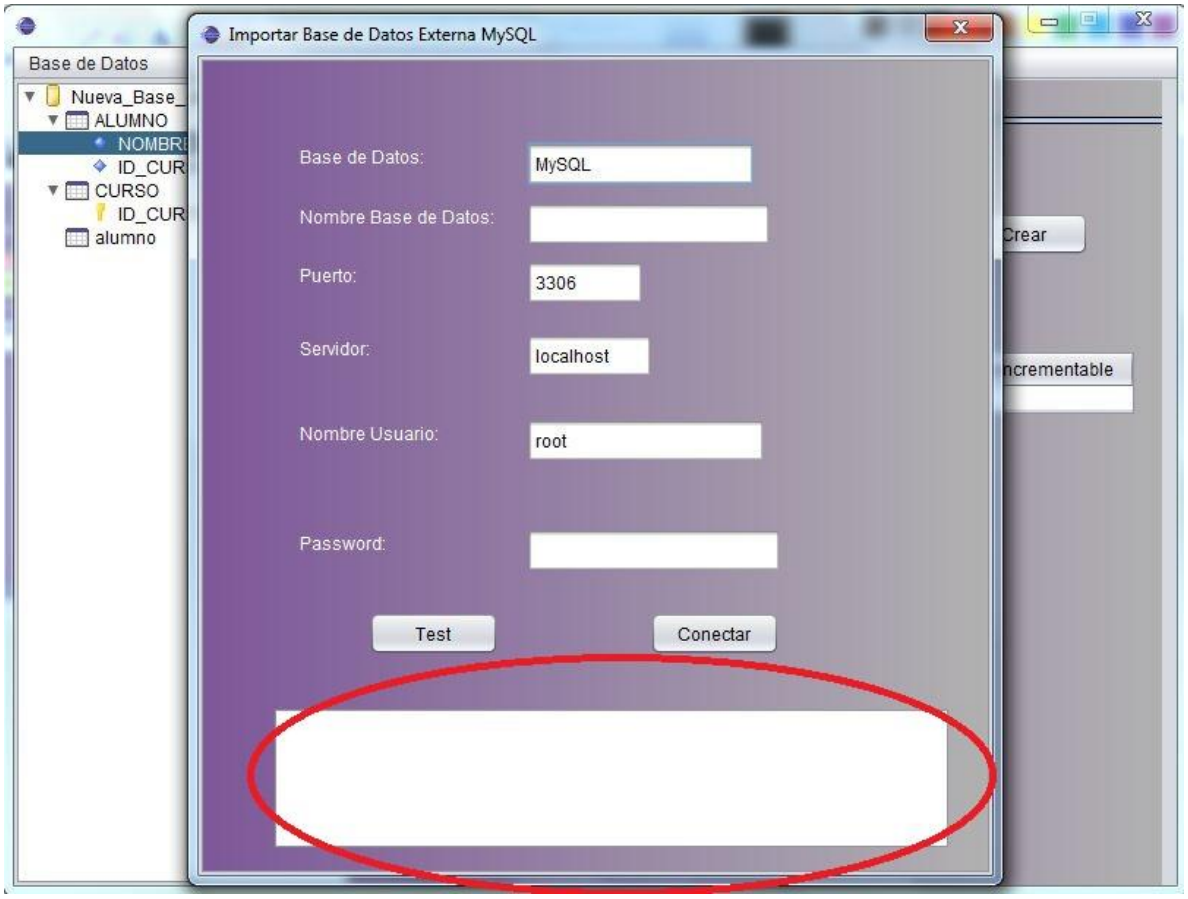


Figura 7: conectar a BD externa MySQL

6 GUARDAR BD

Para poder guardar la estructura de la Base de Datos con la que se está trabajando y mantenerla en forma persistente dentro del proyecto, se debe acceder a la opción de "Guardar" en el menú o presionar las teclas Ctrl+G y de esta forma la aplicación guardara la estructura en formato XML dentro del proyecto.

7 TABLAS

7.1 Agregar Tabla

Para poder agregar tabla debe acceder a la opción “Agregar tabla” del menú o presionar las teclas de acceso rápido Ctrl+T, al hacer esta operación se abrirá la ventana nueva tabla (Figura 8) en donde se pedirá un nombre de tabla. Además en esta interfaz se muestran todos los atributos que posee dicha tabla (resaltado en azul). Cabe decir que solo es información de lectura y no se puede realizar acciones en ellas.

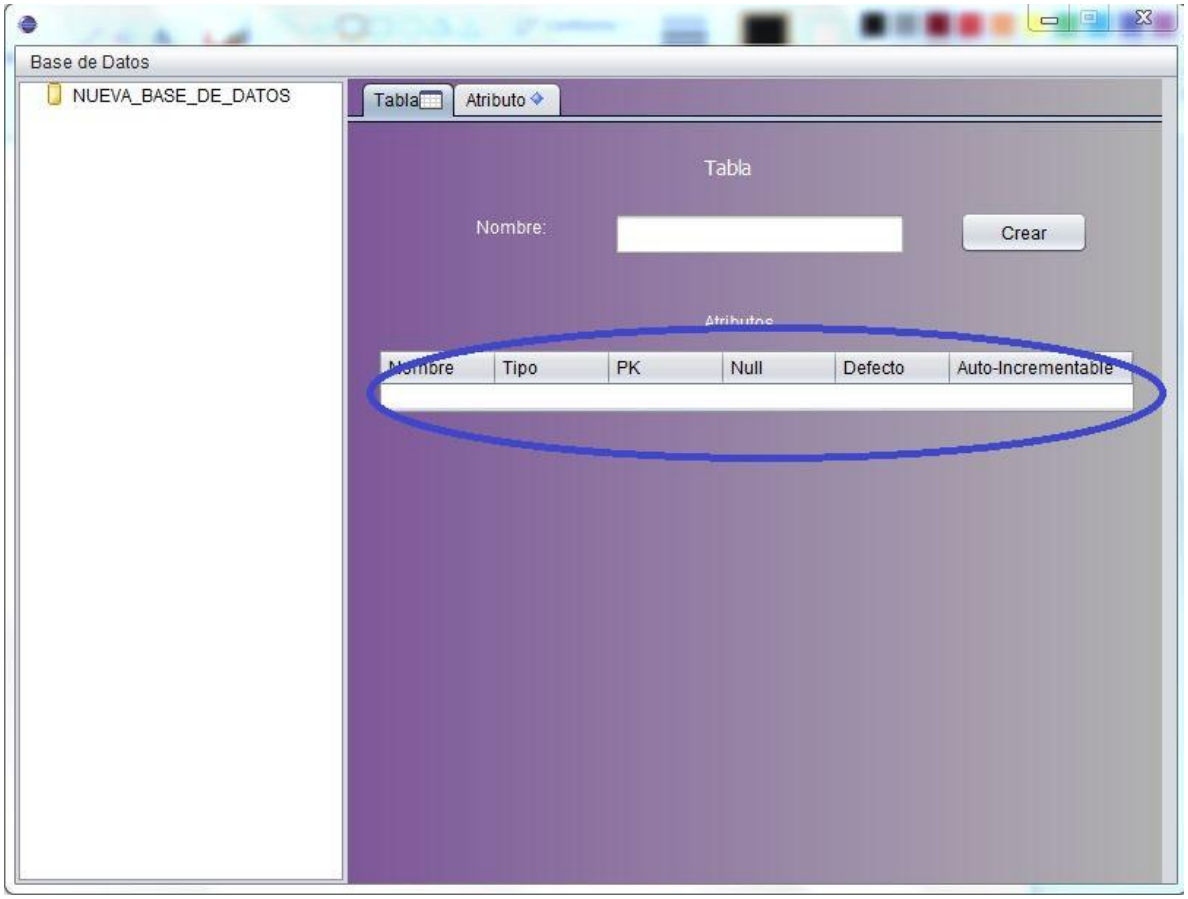


Figura 8: Agregar Tabla

7.2 Editar Tabla

Para editar los datos de una tabla, esto es el nombre, se debe hacer doble click sobre la tabla que se quiera renombrar, al hacer esto se verá el mismo panel de una nueva tabla, solo que esta vez el nombre estará cargado y listo para editar y se podrá ver la lista de

atributo de la tabla (ver figura 9), acá se debe cambiar el nombre y presionar el botón actualizar.

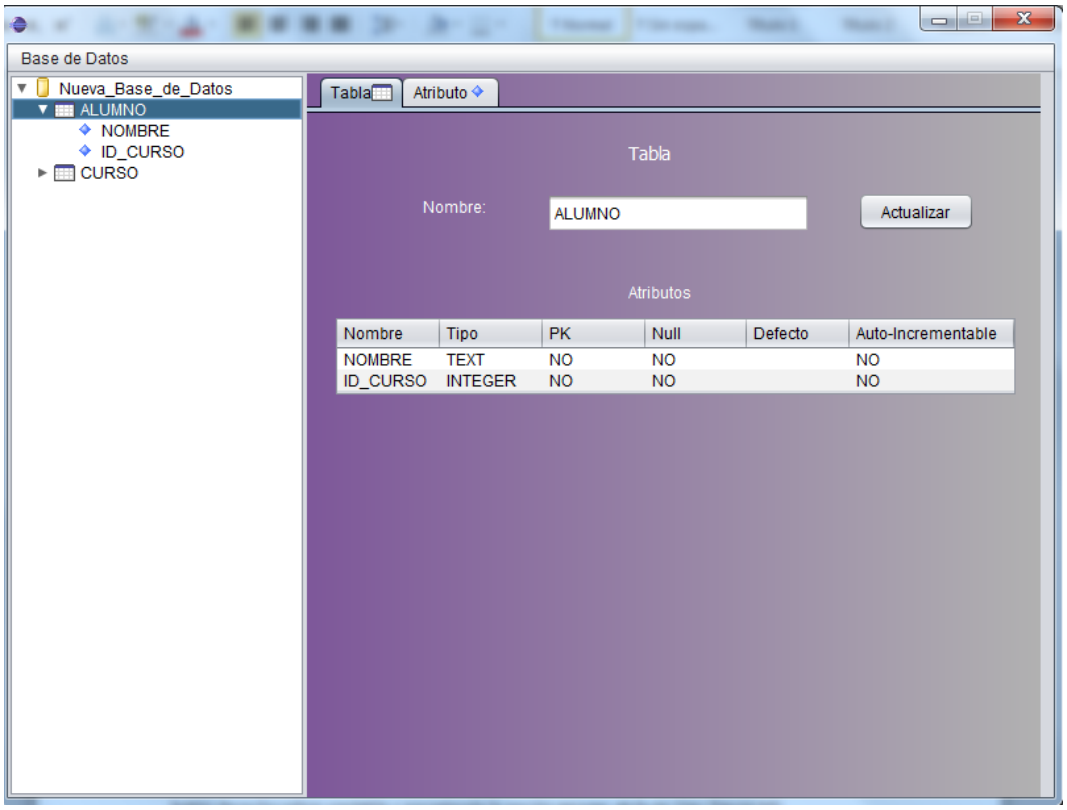


Figura 9: Editar tabla

7.3 Agregar Atributo

Para poder agregar un atributo a una tabla existen varias manera, la primera es al momento de crear la tabla se le pueden asignar todos los atributos necesario a dicha tabla, esto se hace en la pestaña atributo en el panel de nueva tabla como se ve en la Figura 10, otra forma es presionando botón derecho sobre una tabla y escogiendo la opción agregar atributo (Ver Figura 11), una última forma de agregar atributos es en la opción de editar tabla escogiendo la pestaña Atributo de la misma forma que cuando se crea una nueva tabla (figura 12).

Desde cualquiera de las tres formas en las que se puede acceder a agregar un atributo aparecerá el panel del atributo que se muestra en la figura 13 en el cual se pide como mínimo el nombre del atributo.

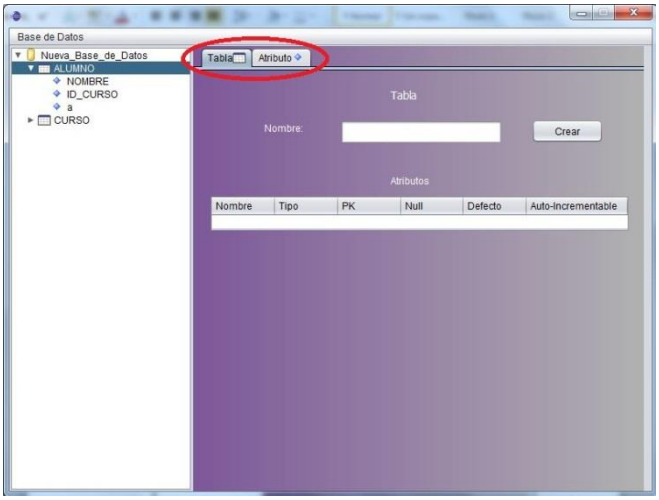


Figura 10: Agregar atributo desde nueva Tabla

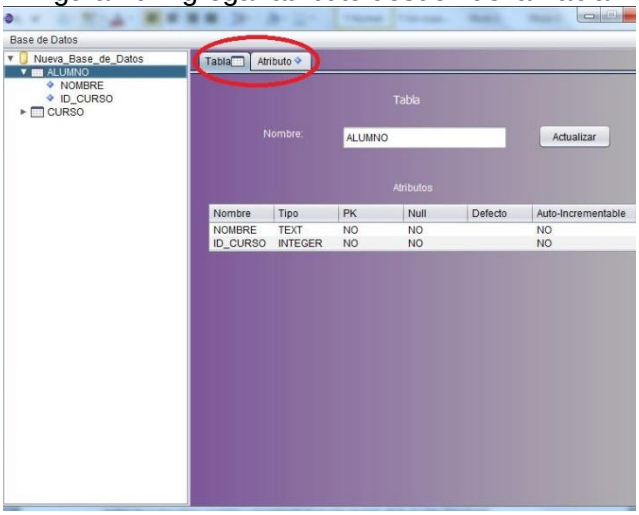


Figura 11: Agregar atributo desde editar tabla

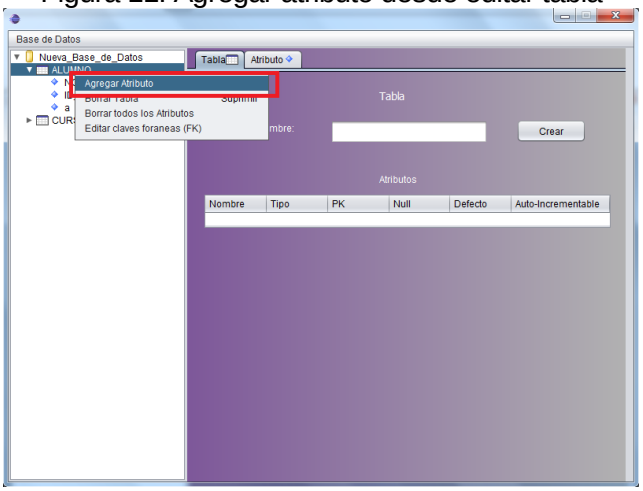


Figura 12: Agregar atributo desde menú contextual de una tabla

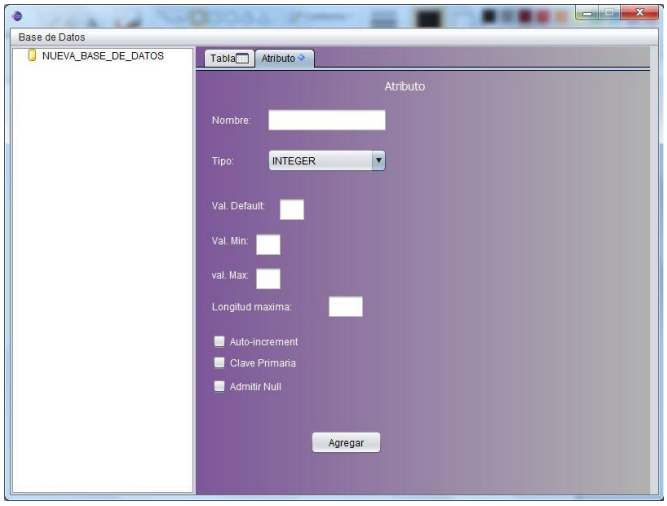


Figura 13: formulario agregar atributo

7.4 Editar Atributo

Para poder editar los datos de un tributo se debe expandir desde el árbol de la estructura de la Base de Datos hasta el nivel de los atributos y hacer doble click sobre el atributo que se desea editar (Figura 14). De esta forma se mostrara la pantalla con los datos del atributo listos para ser editados. Una vez editados se debe guardar los cambios presionando el botón guardar de la misma ventana.

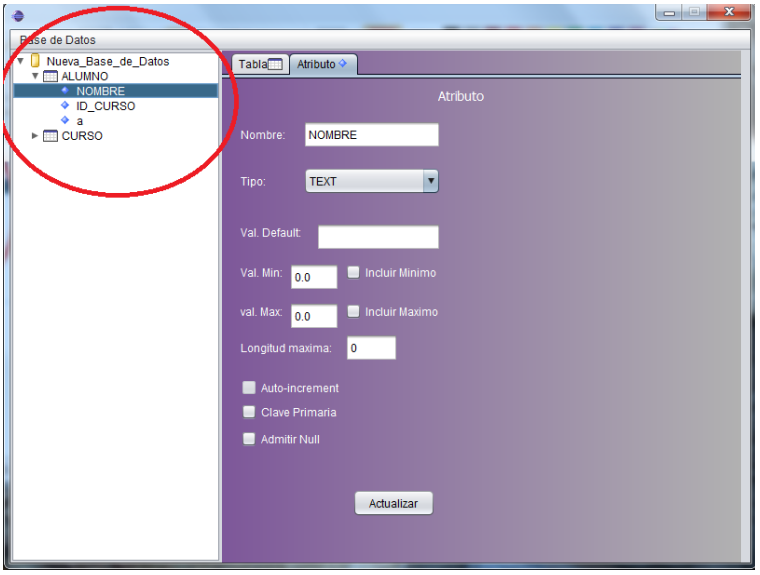


Figura 14: Editar un atributo

7.5 Borrar atributo

Para borrar un atributo se puede hacer de dos maneras una es presionando botón derecho sobre un atributo en el árbol de estructura y escogiendo la opción “Borrar atributo” del menú contextual (Figura 15). Otra forma es en el menú contextual de la tabla en la opción “Borrar todos los atributos” (Figura 16), con la diferencia que esta opción borra todos los atributos de la tabla y no solo alguno en particular.

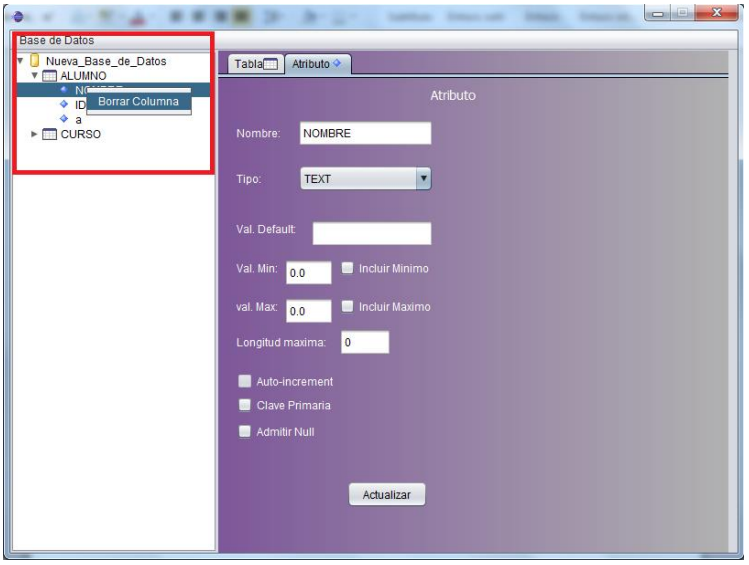


Figura 15: Borrar atributo

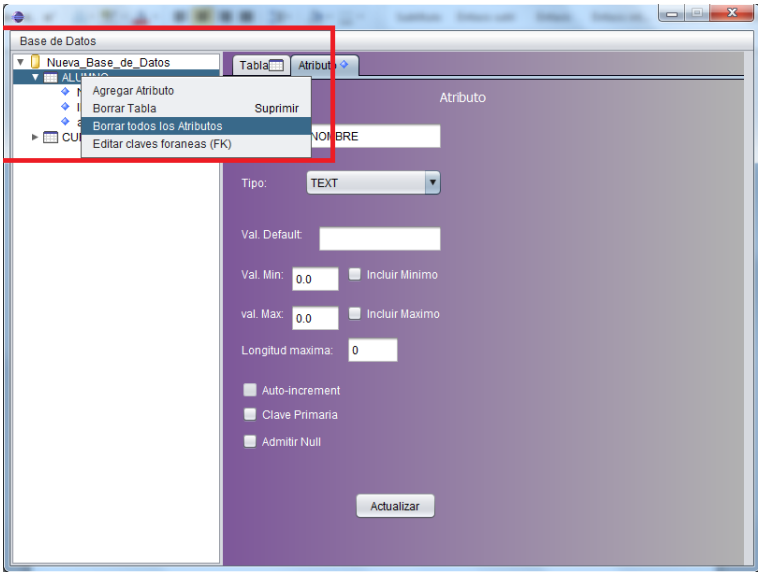


Figura 16: borrar todos los atributos de una tabla

7.6 Borrar Tabla

Para poder borrar una tabla de la Base de Datos existen tres formas de hacerlo, una es en el menú contextual de la tabla en la opción “Borrar tabla” (Figura 17), otra se encuentra en el menú contextual de la Base de Datos y en el menú de la barra de menú de la ventana en la opción “Borrar todas las tablas” (Figura 18) solo que esta última opción borra todas las tablas de la Base de Datos.

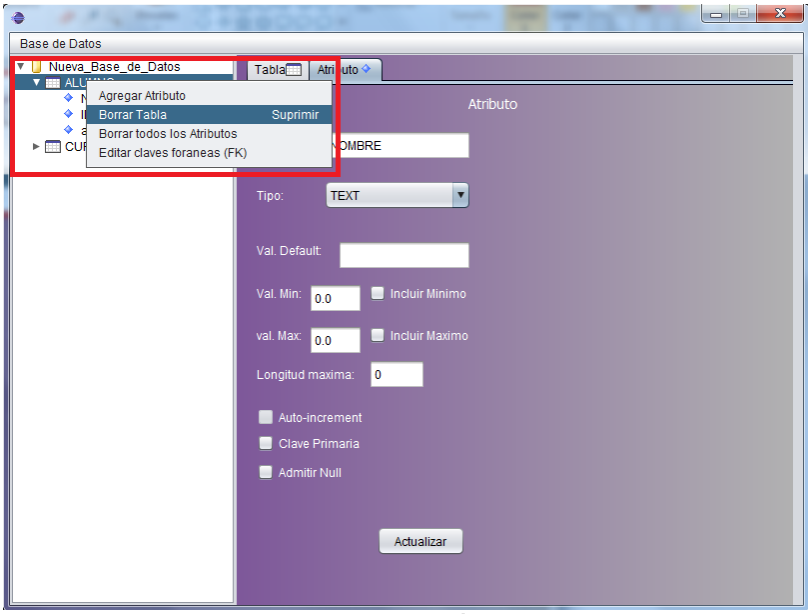


Figura 17: Borrar tabla de menú contextual de la tabla

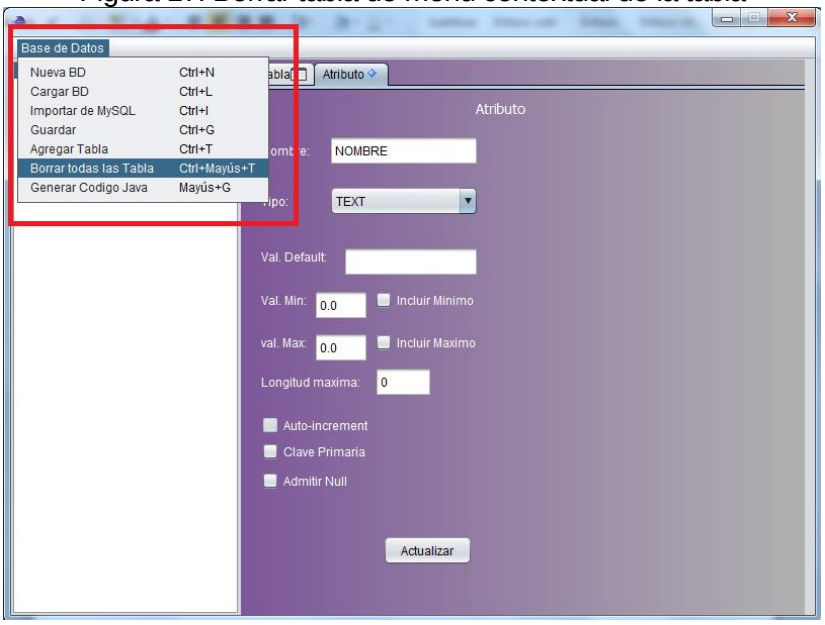


Figura 18: Borrar todas las tablas

7.7 Ver claves Foráneas

Para poder ver las claves foráneas de una tabla se puede hacer desde el menú contextual de esta en la opción de “Editar Claves Foráneas (FK)” (Figura 19), se cargara un panel donde se muestra el nombre de la tabla y una lista con las claves foráneas de esta (Figura 20), en esta pantalla se pueden hacer tres acciones con las claves foráneas, las que son:

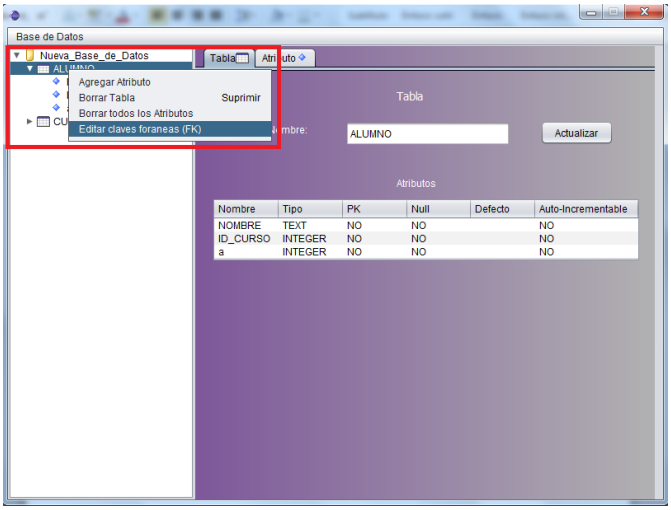


Figura 19: menú contextual de una tabla opción de editar claves foráneas

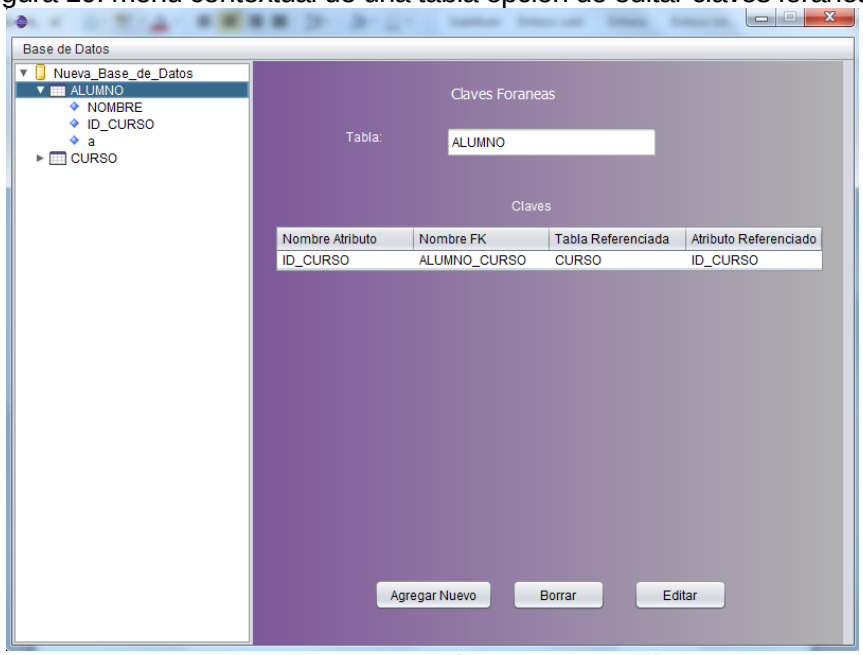


Figura 20: panel de edición de claves foráneas

Agregar Clave Foránea: en esta opción se puede agregar una nueva clave foránea, se accede a ella presionando el botón “Agregar clave” de la pantalla y se muestra el panel de creación de una clave foránea (Figura 21), una vez ingresado los datos se guarda la clave foránea y la aplicación vuelve a la pantalla anterior.

Editar Calve Foránea: en esta opción se puede editar una clave foránea, se accede a ella con el botón “Editar clave” de la pantalla, presionado el botón se muestra el mismo panel de la figura 21, salvo que este viene precargado con los datos de la clave foránea. Cabe mencionar que antes de presionar el botón se debe seleccionar un atributo foráneo de la lista que se muestra en la pantalla (Figura 22).

Borrar Clave Foránea: permite borrar una clave foránea de la tabla, esto se hace seleccionando la clave de la lista y presionando el botón “Borrar” de la pantalla.

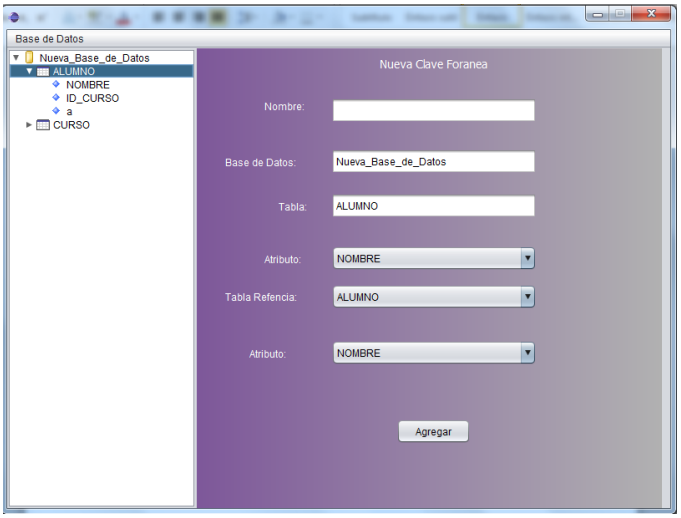


Figura 21: formulario de nueva clave foránea

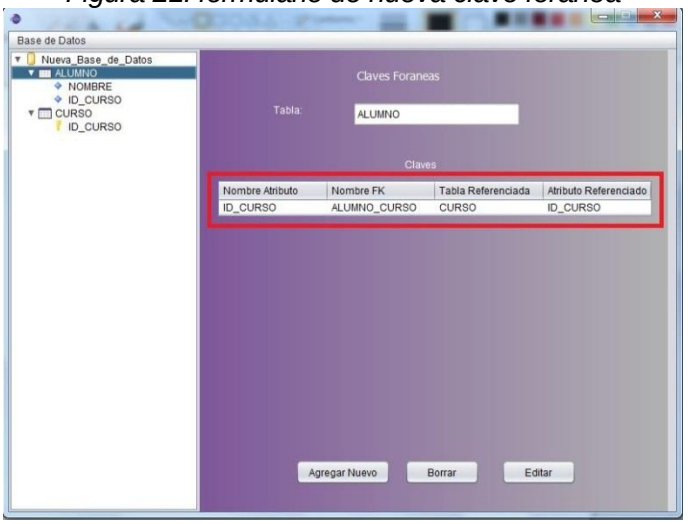


Figura 22: lista de atributos foráneos

8 GENERACIÓN DE CÓDIGO JAVA

Esta opción está disponible en el menú de la barra de la aplicación con el nombre de "Generar Código Java" esta opción permite generar patrones de diseño como son Fachada, POJO y DAO esto lo genera en un paquete aparte en el proyecto (Figura 23), son archivos java los cuales ayudaran al programador en la programación de una Base de Datos en andoid.

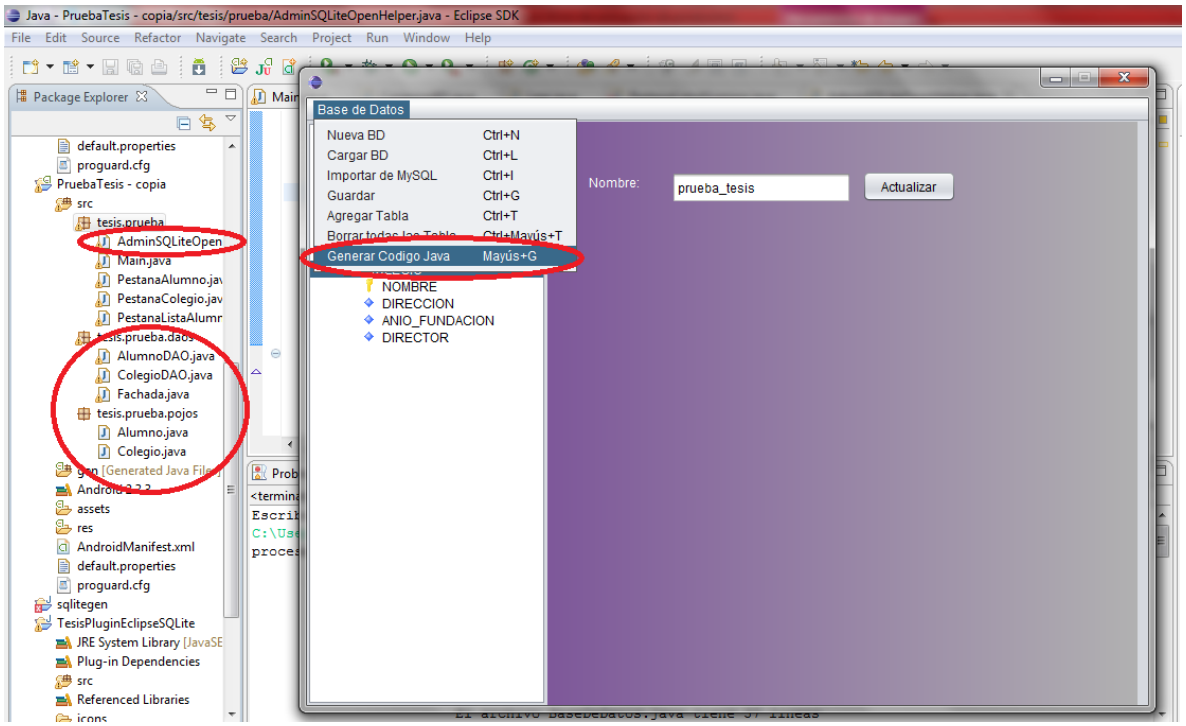
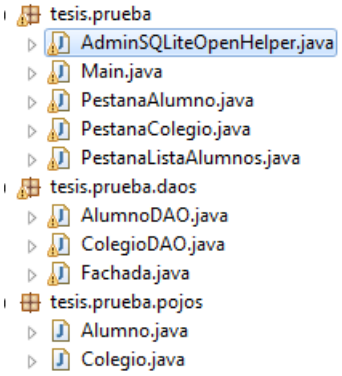


Figura 22: Generación de código java.

8.1 UTILIZACION DEL CODIGO GENERADO

Tras dar clic en generar código podemos cerrar la interfaz del plug-in si queremos, y veremos que en el package en el cual dimos clic derecho al principio, se nos ha generado una clase "AdminSQLiteOpenHelper" y dos subpackages "pojos" y "daos" que contendrán clases de tipo POJO y DAO respectivamente, además de una clase "Fachada" para no tener que interactuar directamente con cada DAO generado.



Este código generado podemos usarlo para interactuar con la BD desde alguna clase de tipo Activity

```
public class PestanaAlumno extends Activity
```

Desde donde podemos crear la BD, insertar, actualizar, consultar o eliminar, para ello siempre usaremos primeramente las líneas:

```
AdminSQLiteOpenHelper admin=new AdminSQLiteOpenHelper(this,"pruebatesis",null,1);
Fachada fachada=new Fachada(admin);
```

Con ellas creamos un administrador de la BD, que creará la BD "pruebatesis" (lógicamente podemos usar otro nombre si queremos) en caso de que no exista (normalmente la primera vez que se ejecuta), o accederá a ella si ya se creó. Y creamos una Fachada, que utilizara al administrador y a los DAOs por nosotros.

Inserción:

Con esto, si por ejemplo queremos insertar un alumno, hacemos

```
Alumno alumno=new Alumno();
alumno.setRut(rut);
alumno.setNombre(nombre);
alumno.setApellido(apellido);
alumno.setAnioNacimiento(anioNacimiento);
alumno.setCurso(curso);
alumno.setNombreColegio(nombreColegio);
fachada.insert(alumno);
```

O bien:

```
//(columnas, valores)
fachada.insertAlumno((Alumno.RUT+", "+Alumno.NOMBRE+", "+Alumno.APELLIDO
        +", "+Alumno.ANIO_NACIMIENTO+", "+Alumno.CURSO+", "+Alumno.NOMBRE_COLEGIO),
        (rut+"', '"+nombre+"', "+"', '"+apellido+"', "+"', "+anioNacimiento+", "
        +"', '"+curso+"', "+"', '"+nombreColegio));
```

Ambos casos nos darán el mismo resultado, pero este último nos da más libertades, como el omitir alguna columna para que tome el valor por defecto (aunque en el caso anterior bastaría con no setear un valor al objeto para que tome el valor por defecto), o darle el valor NULL a alguna columna (eso no lo permite el caso anterior). Aunque en

este caso debemos tener cuidado de usar la comilla simple si la columna es de tipo "text". Se recomienda usar el primer caso a menos que se necesite pasar el valor NULL a una columna.

Si la inserción falla por incumplir alguna restricción de clave foránea, de máximo, mínimo, longitud, o no null, lanzará una Exception, por lo que deberíamos usa un try catch

Selección:

Si queremos seleccionar alumnos, podemos usar la fachada para seleccionar una lista con todos, o una lista con los que cumplan una condición:

```
List<Alumno> alumnos=fachada.selectAlumno(Alumno.RUT+"="+rut);
```

Para una condición múltiple al String que se pasa de parámetro como condición, se le concatenan condiciones unidas con "AND" o "OR" según corresponda.

Si sabemos que el resultado será solo 1 o 0 alumno, y no queremos la lista, podemos hacer

```
Alumno alumno=null;
List<Alumno> alumnos=fachada.selectAlumno(Alumno.RUT+"="+rut);
if(!alumnos.isEmpty())
    alumno=alumnos.get(0);
else{
    mostrarMensajeEmergente("Alumno no encontrado");
    return;
}
```

En ese caso, si la lista contiene elementos el alumno que queremos será el primero, sino, pues no se encontró el elemento buscado.

Update:

Para modificar un alumno podemos obtener un objeto tipo Alumno usando selección con la condición adecuada, luego setearle nuevos valores ha dicho objeto (sin modificar la clave primaria, sino no podrá encontrar al objeto que se está modificando) y luego usar:

```
fachada.update(alumno);
```

También se puede hacer:

```
//(set, condiciones)//en este caso rut es numerico
fachada.updateAlumno(Alumno.APELLIDO+"='Araya', "+Alumno.RUT+"=17343755",
        Alumno.RUT+"=17090851 AND "+Alumno.NOMBRE+"='Gabriel'");
```

En este caso si podemos cambiar la clave primaria si queremos, en este caso específico le asignamos el apellido 'Araya' y el rut 17343755 a todos los alumnos que cumplan la condición de tener el rut 17090851 Y su nombre sea 'Gabriel'. Aunque en este caso,

como rut es clave primaria, a lo más se modificará 1 alumno, pero dependiendo la condición podríamos modificar varios a la vez.

Si el update falla por incumplir alguna restricción de clave foránea, de máximo, mínimo, longitud, o no null, lanzará una Exception, por lo que deberíamos usa un try catch

Delete

Si queremos borrar un alumno, podemos usar la selección para obtener un Alumno según la condición dada, y luego usar:

```
fachada.delete(alumno);
```

O bien podemos usar:

```
fachada.deleteAlumno(Alumno.RUT+"="+rut);
```

En este caso se borraran todos los alumnos que cumplan la condición, aunque en éste caso específico, por ser rut clave primaria, solo se borrara a lo más 1. Podría usarse por ejemplo para el caso en que se elimina un colegio, eliminar todos los alumnos cuyo colegio sea el eliminado.

Claves foráneas

Al declarar claves foráneas en la estructura, por ejemplo si se declara que la columna "nombre_colegio" de "ALUMNO" hace referencia a la columna "nombre" de "COLEGIO", la clase "Colegio" contendrá una lista de Alumnos. De paso mostraremos también como recorrer una lista:

```
List<Colegio> colegios= fachada.selectAllColegio();
for (Iterator iterator = colegios.iterator(); iterator.hasNext();) {
    Colegio colegio = (Colegio) iterator.next();
    List <Alumno> alumnos= colegio.getAlumnos();
```

En este caso obtenemos una lista de todos los colegios, recorremos esa lista de colegios y de cada colegio obtenemos la lista de sus alumnos.

(En el ejemplo falta cerrar el ciclo for, que se debería cerrar después de implementado lo que queramos hacer con el colegio y su lista de alumnos).

9 ANEXO

Mensajes y avisos al programador y usuario

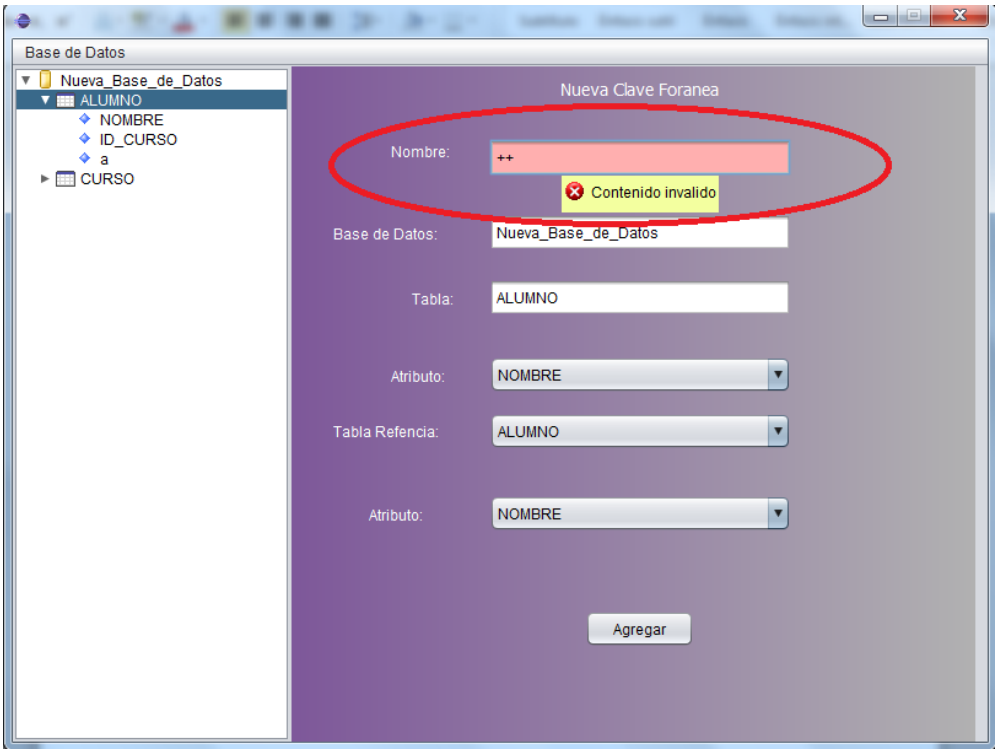


Figura 23: mensaje de error de caracteres. Se muestra cuando el nombre o en algún se insertan caracteres inválidos.

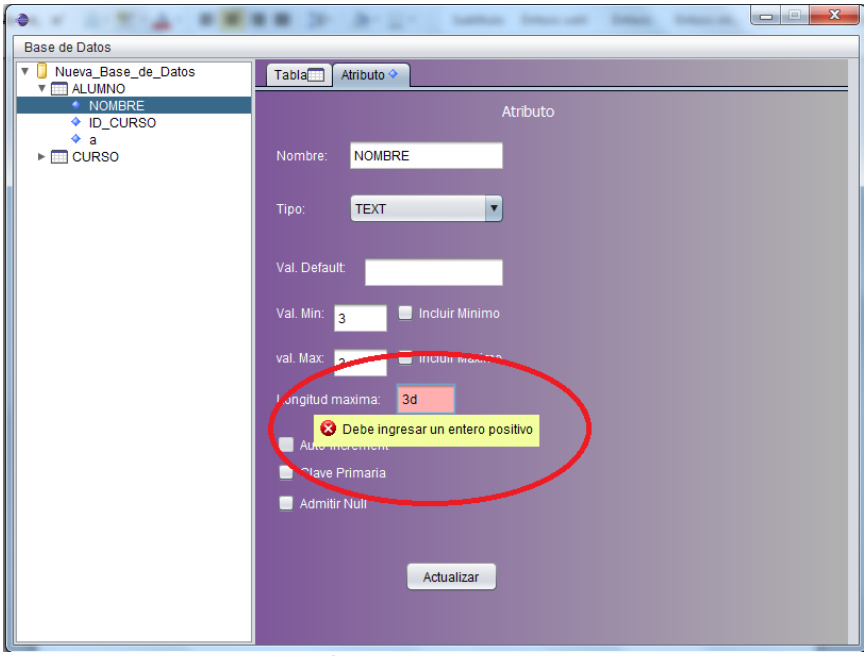


Figura 24: mensaje de error de número. Se muestra cuando se ingresan letras a un campo que solo acepta números.

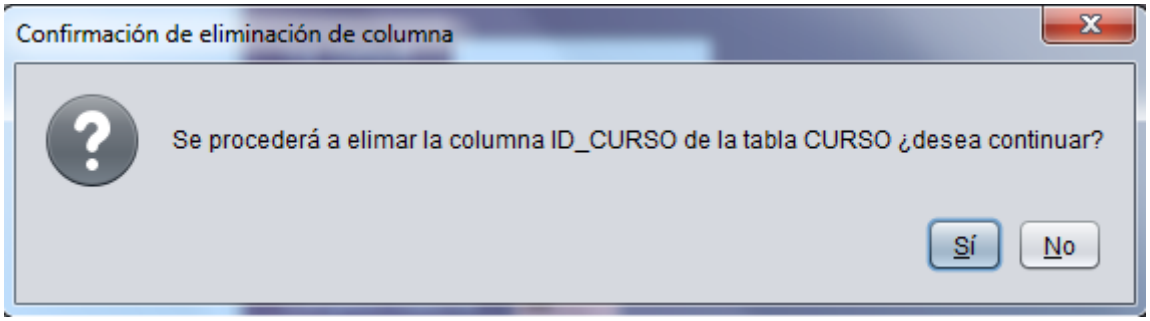


Figura 25: mensaje de advertencia cuando se eliminara un atributo

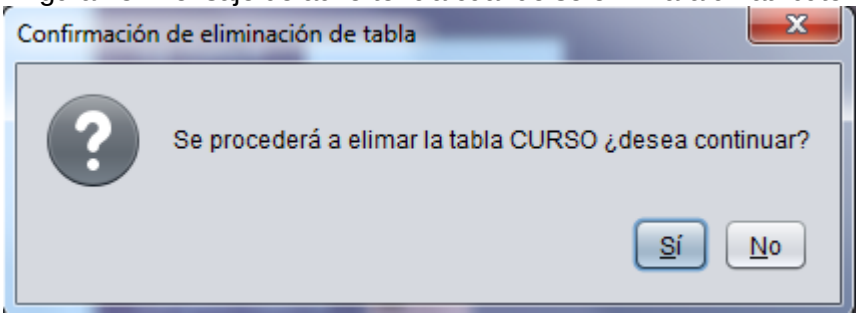


Figura 26: mensaje de advertencia cuando se eliminara una tabla.

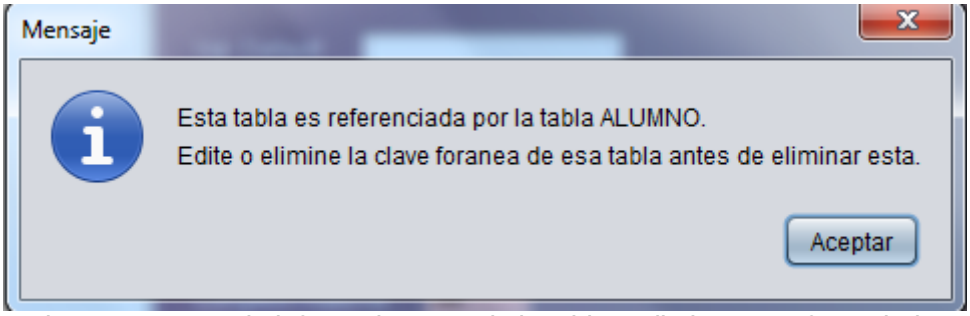


Figura 27: mensaje informativo cuando la tabla a eliminar es referenciada.

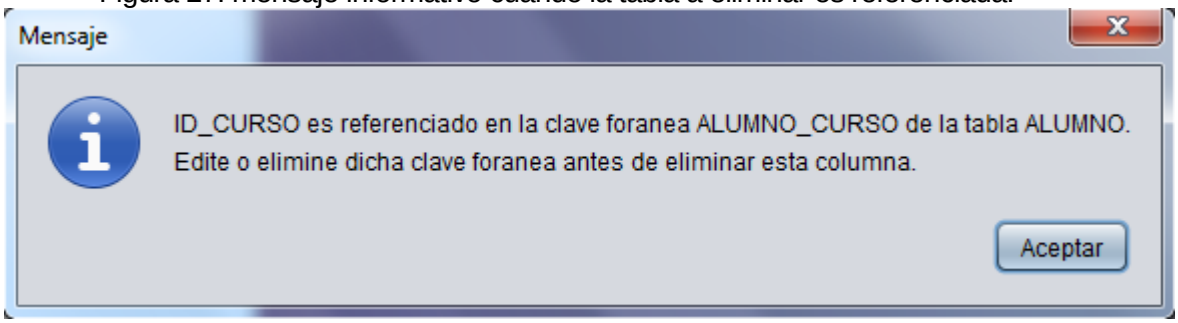


Figura 28: mensaje informativo cuando el atributo a eliminar es referenciado.

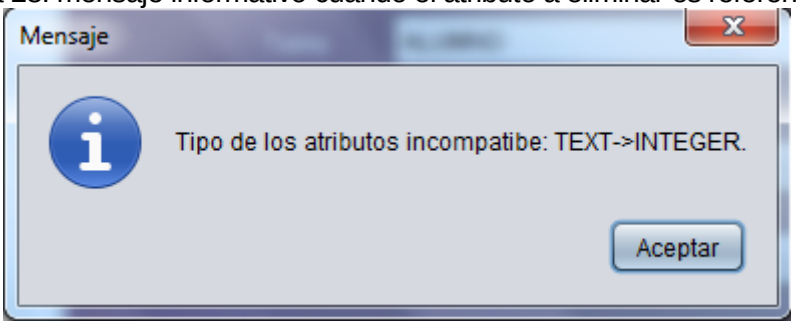


Figura 29: mensaje de error de tipos de datos incompatibles. Se produce cuando se referencian datos con diferentes tipos.