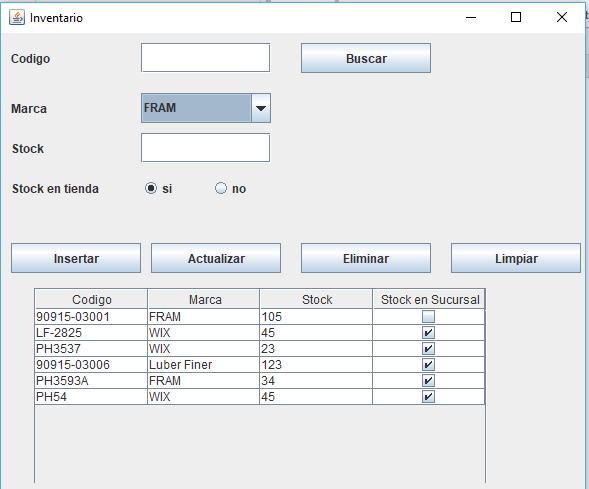
Conexión a Bases de Datos.

JDBC significa J ava D ata B ase C onnectivity, que es una API Java estándar para la conectividad independiente de la base de datos entre el lenguaje de programación Java y una amplia gama de bases de datos.

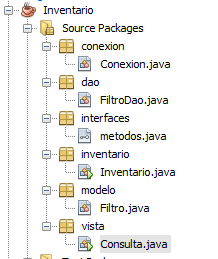
La biblioteca JDBC incluye API para cada una de las tareas comúnmente asociadas con el uso de la base de datos:

* Haciendo una conexión a una base de datos
* Crear instrucciones SQL.
* Ejecutando consultas SQL.
* Visualización y modificación de los registros resultantes

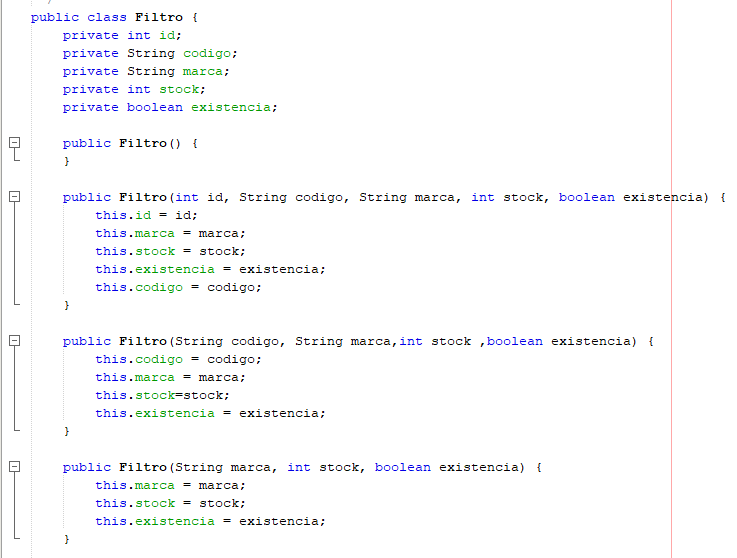
Practica Guiada

El objetivo de la practica será realizar el siguiente formulario que nos permitirá administrar un pequeño inventario de filtros de aceite para vehículos.

1. Crearemos la siguiente estructura de paquetes.



1. Clase Filtro.java

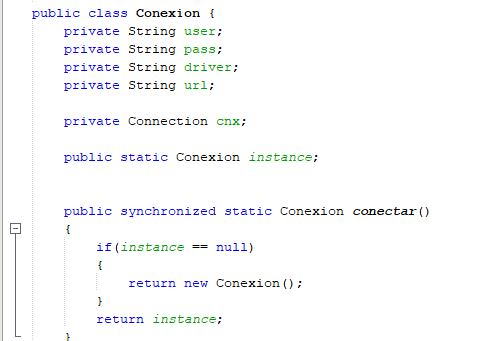


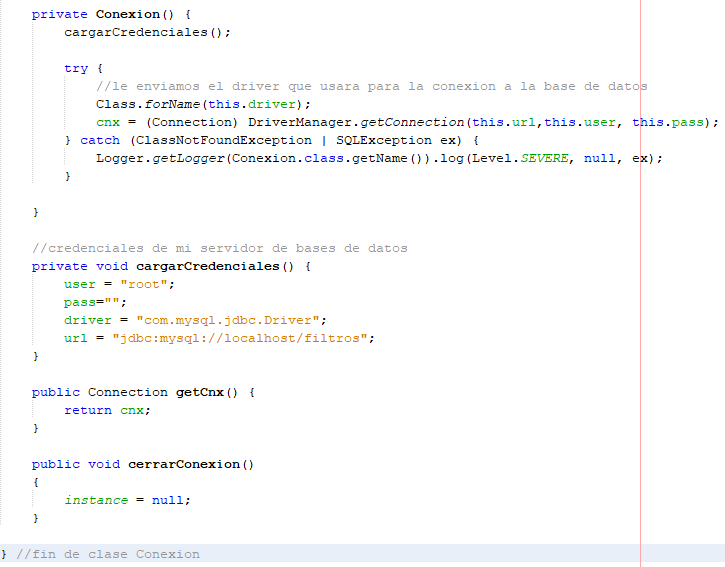
1. Clase Conexión.java.

**Class.forName(“Driver”):** Nos permite abrir un canal para poder establecer la comunicación con la base de datos.

**DriverManager.getConnection ():** Devuelve un objeto Connection, el cual representa la conexión física a la base de datos. Recibe como parámetros:

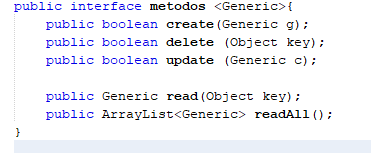
* Url: Es el lugar donde está alojado nuestra base de datos, esta se escribe de la siguiente forma: jdbc:mysql://server/nombre\_bd, en donde server es la dirección ip o hostname del servidor.
* User: Usuario de la bd.
* Pass: Es la contraseña del usuario de User.





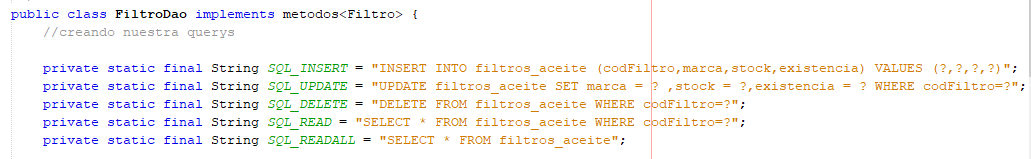
1. Clase Interfaz metodos.java.

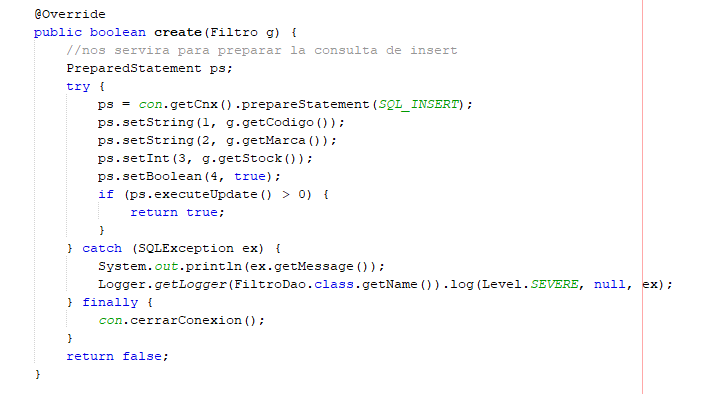
El objetivo de la interfaz será para establecer los métodos para insertar, eliminar, buscar y actualizar.

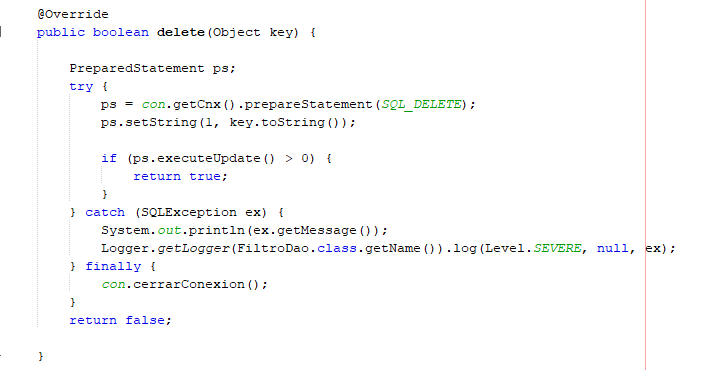


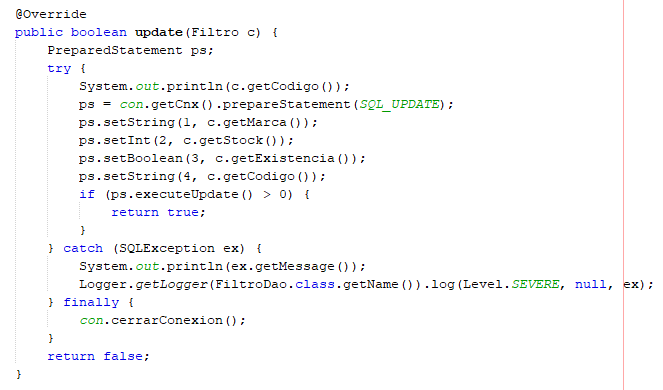
1. Clase FiltroDao.java.

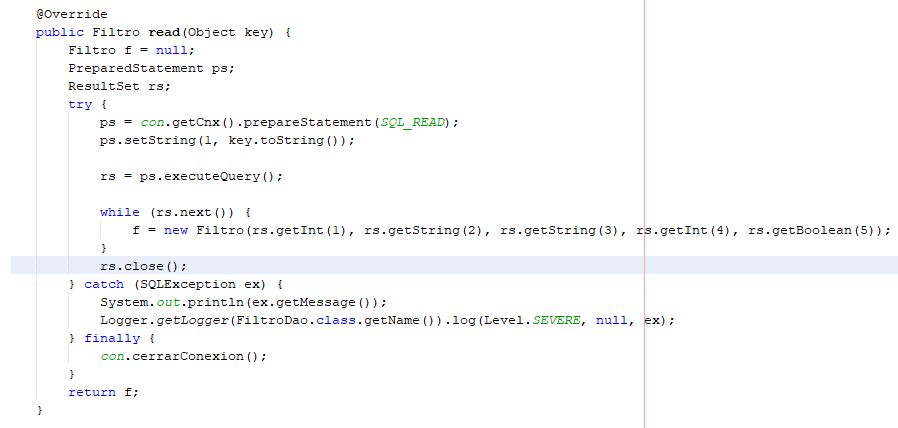
|  |  |
| --- | --- |
| Objeto | Descripción |
| PreparedStatement | Se usa para ejecutar consultas parametrizadas. Con lo cual podemos evitar ataques de tipo SQLInjection. Retorna un objeto de tipo Resulset.  Métodos  **.prepareStatement(“SQL”)** -> Enviamos una consulta como la siguiente:  DELETE FROM filtros\_aceite WHERE codFiltro=?, donde “?” es el valor que parametrizaremos o en otra palabras preparamos la consulta.  **.setTipoDato(posición,dato)** -> Nos permite cambiar el “?” por el dato que enviaremos a la base datos. La posición comienza a partir del número 1. Para la consulta del ejemplo anterior nos quedaría:  .setString(1, “dato”).  **.executeUpdate()** -> Ejecutamos la consulta, si fue exitosa nos devolverá un número mayor a 0. |
| Satement | proporciona métodos para ejecutar consultas con la base de datos. Retorna un objeto de tipo Resulset.  **Métodos**  .executeQuery(“SQL”) -> Ejecuta una consulya sin parámetros decir no lleva “?”. |
| Resulset | Mantiene un cursor apuntando a una fila de una tabla. Inicialmente, el cursor apunta a antes de la primera fila.  Métodos.  **.Next()** -> Mueve el cursor a la siguiente fila de la tabla.  **Rs.getTipoDato()** -> Extrae el dato de una columna especifica. Puede recibir como parámetro el número de la columna (ojo aca inicia apartir del numero 1) o el nombre de la columna.  **.close()** -> Cierra el cursor. |

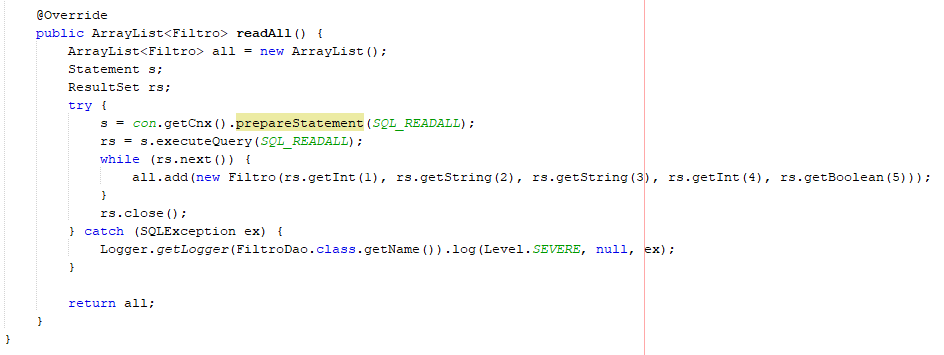




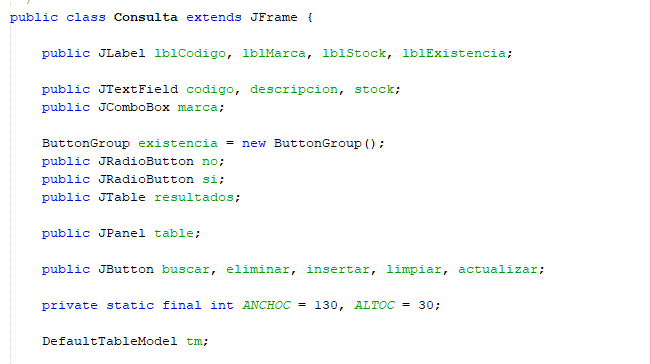




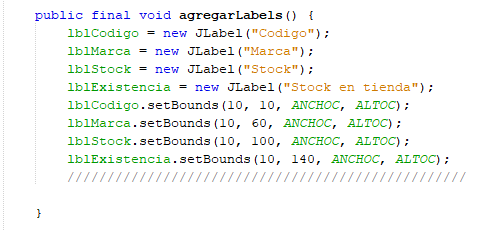


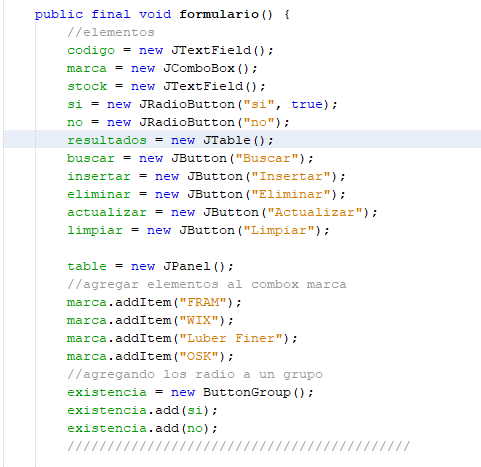


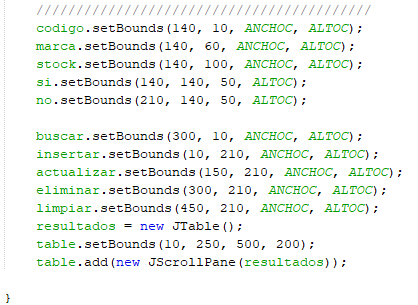
1. Clase Consulta.java



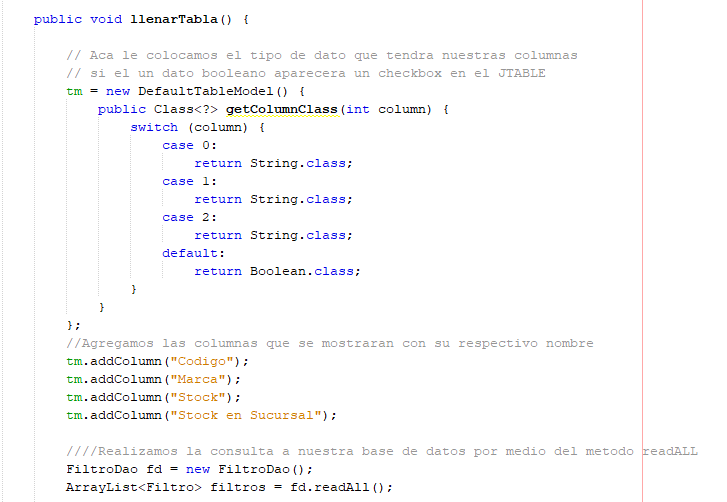






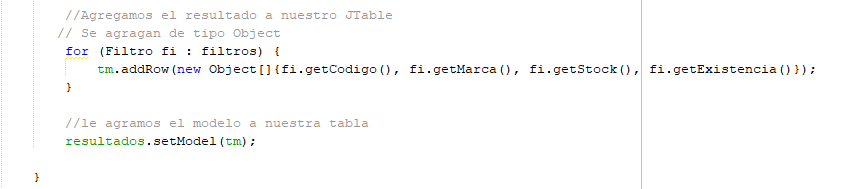


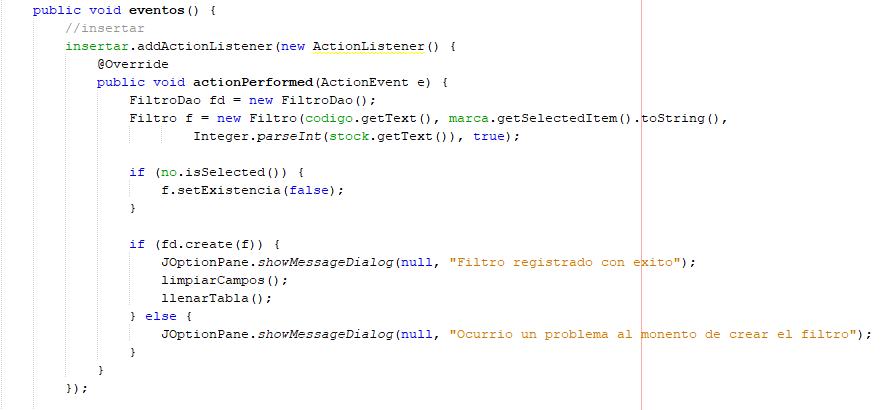
Agregamos datos al combo box

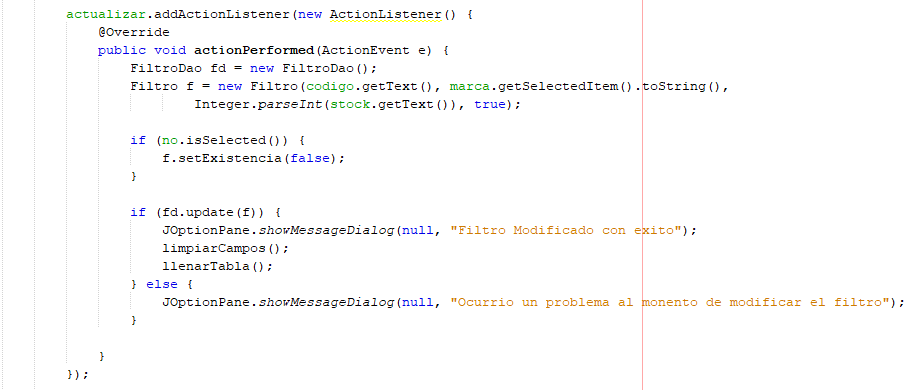


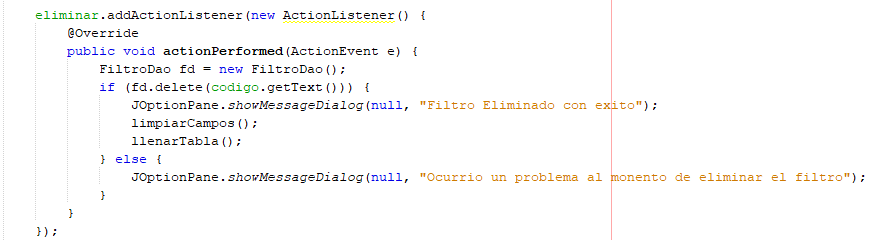
Indicamos el tipo de datos de las columnas

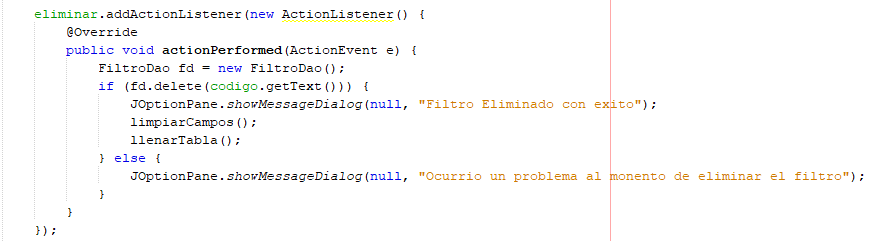
Agregamos las columnas a nuestro modelo

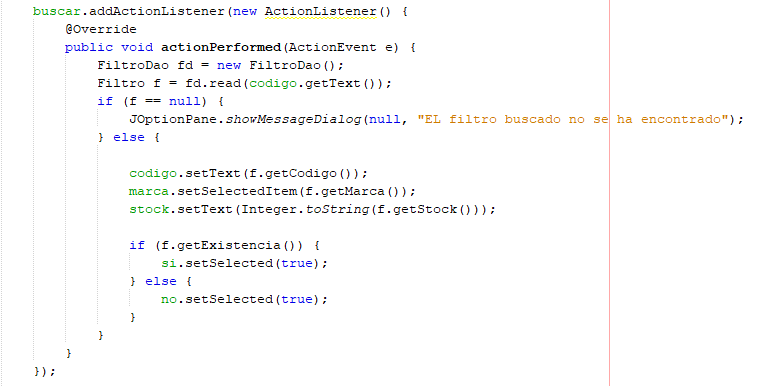


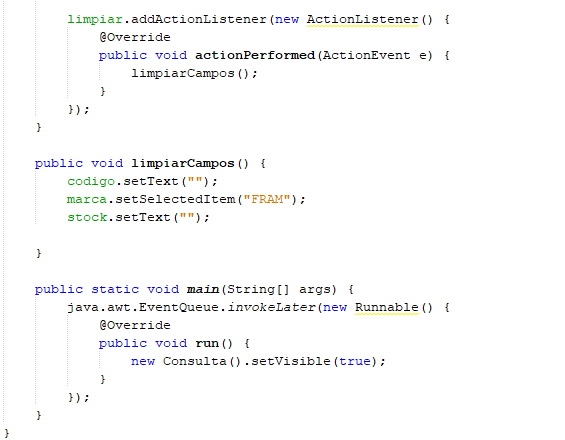












**Base de datos.**

Nombre de la base: filtros.

Gestor de Bases de Datos: MariaDB.

