Universidad Cooperativa de Colombia

CAMPUS: Montería

Proyecto Final

CURSO: Estructura de Datos

PROGRAMA: Ingeniería de Sistema

NOMBRES DE LOS ESTUDIANTES:

Omar Osorio

Juan Diego Salgado

Juan Camilo Vega

DOCENTE: Rubén Baena

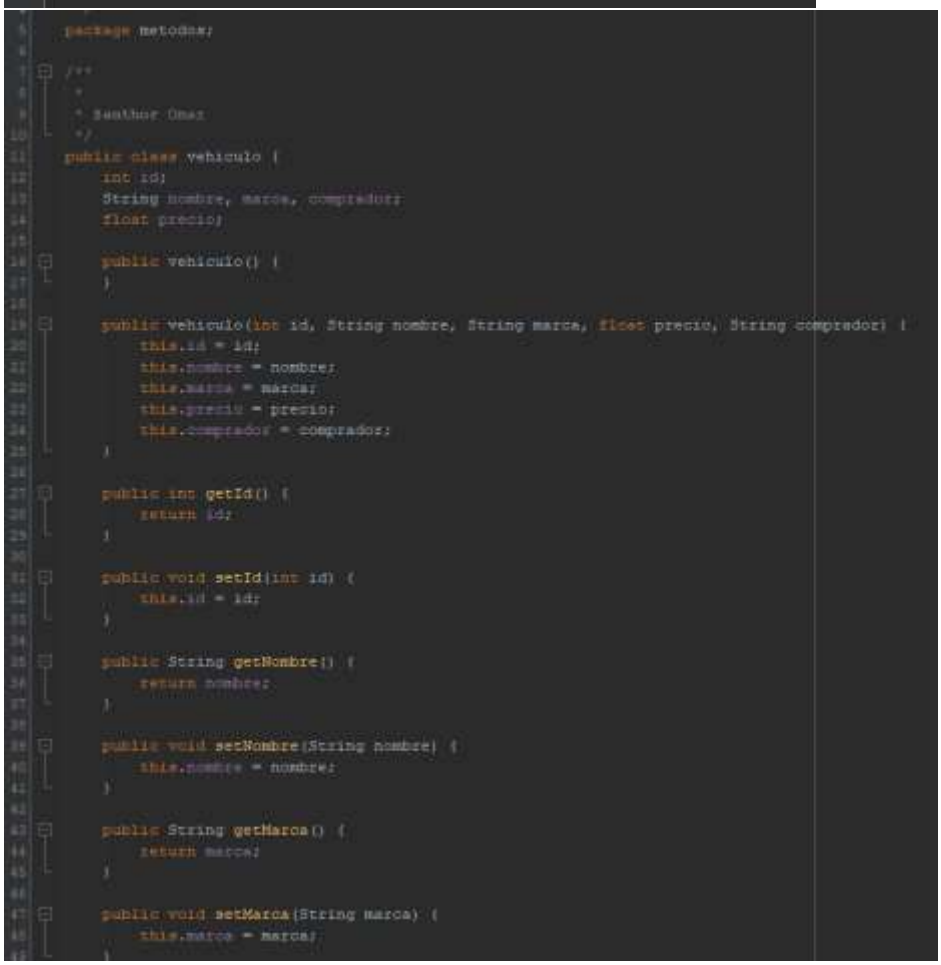
2024

Desarrollo de proyecto final en JavaFX:

Este es un manual paso a paso que explica el desarrollo de una aplicación en JavaFX.

1. Primero se crean los objetos que utilizaremos, esta será una tienda de vehículos entonces usaremos el objeto de usuarios y de vehículos.

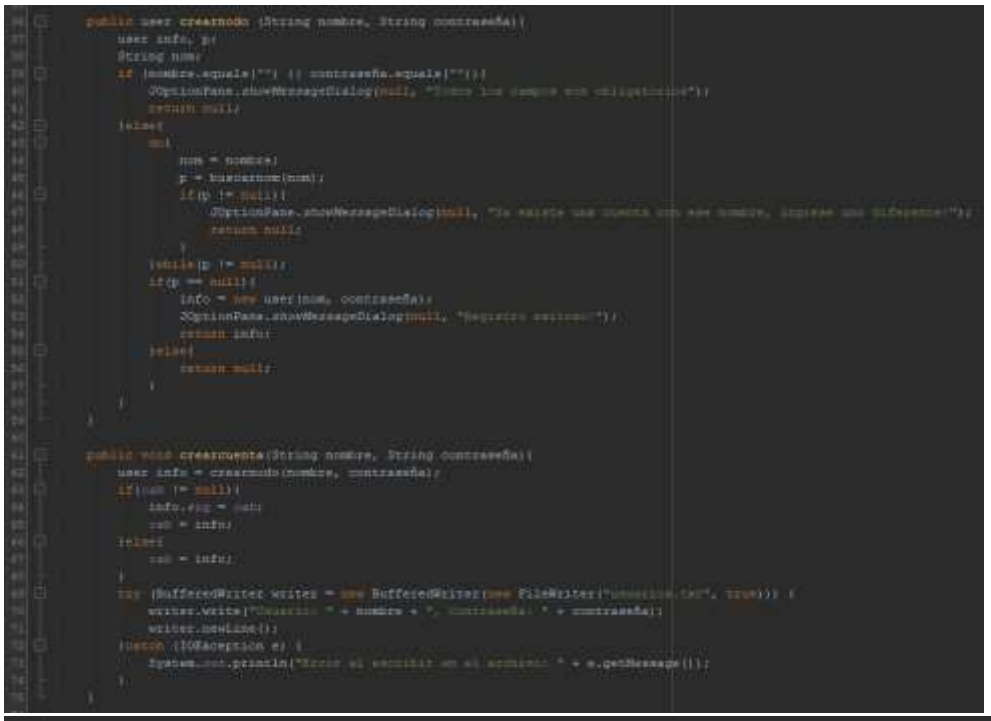
```java
package metodos;

/**
 *
 * @author Omar
 */
public class user {
    String nombre, contraseña;
    user sig;

    public user(String nombre, String contraseña) {
        this.nombre = nombre;
        this.contraseña = contraseña;
        sig = null;
    }
}
```



```
    public float getPrecio() {
        return precio;
    }

    public void setPrecio(float precio) {
        this.precio = precio;
    }

    public String getComprador() {
        return comprador;
    }

    public void setComprador(String comprador) {
        this.comprador = comprador;
    }


}
```

2. Ahora creamos las listas, pilas y colas que utilizaremos
-Empezando por la lista de usuarios

```
package metodos;

import java.io.BufferedWriter;
import java.io.FileWriter;
import java.io.IOException;
import javax.swing.JOptionPane;

/**
 *
 * @author Omar
 */
public class listacuentas {
    public static user cab;

    public listacuentas () { cab=null; }

    public user buscarnom(String user){
        user p = cab;
        if(cab == null)
            return null;
        else{
            while (p != null){
                if ((p.nombre).equals(user))
                    return p;
                else
                    p=p.sig;
            }
            return null;
        }
    }

```



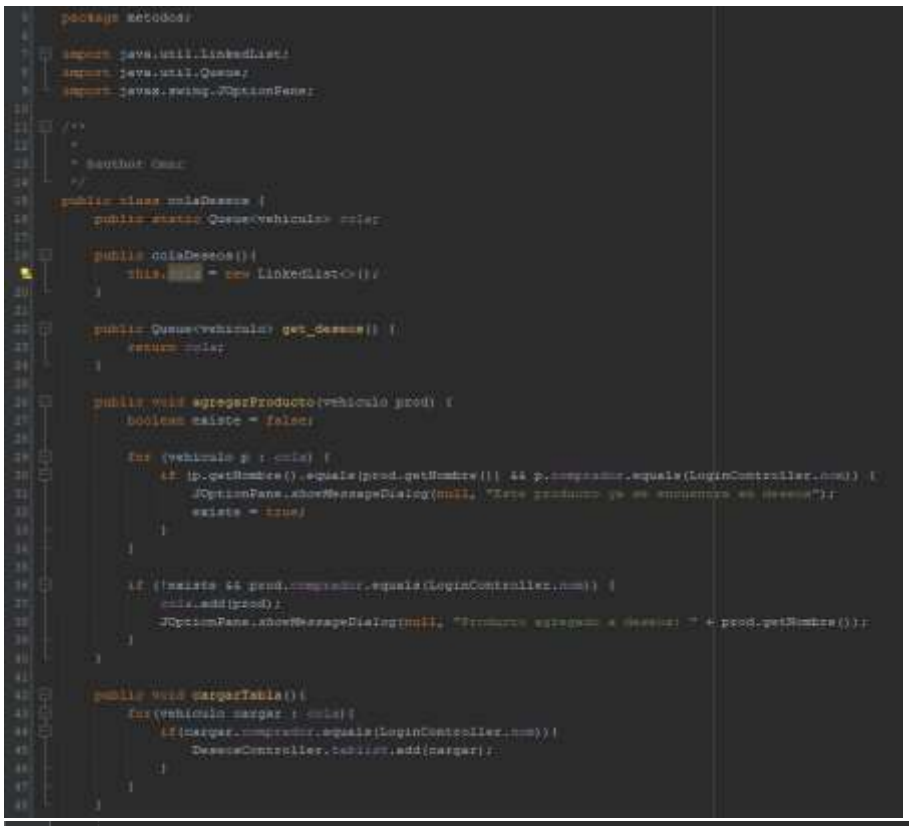
```
    public user logcuenta(String nombre, String contraseña){
        user p = cab;
        if(cab == null)
            return null;
        else{
            while(p != null){
                if((p.nombre).equals(nombre) && (p.contraseña).equals(contraseña))
                    return p;
                else
                    p=p.sig;
            }
            return null;
        }
    }

}
```

-Ahora la pila del carrito de compras.

```java
package metodos;

import java.util.Stack;
import javax.swing.JOptionPane;

/**
 *
 * @author Omar
 */
public class pilaCarrito {

    public static Stack<vehiculo> pila;

    public pilaCarrito(){
        this.pila = new Stack();
    }

    public void setPushProductos (vehiculo p){
        pila.push(p);
        JOptionPane.showMessageDialog(null,"Agregado al carrito exitosamente!");
    }

    public void cargarTabla(){
        for(vehiculo cargar : pila){
            if(cargar.comprador.equals(LoginController.nom)){
                CarritoController.tablist.add(cargar);
            }
        }
    }

    public float valorTotal(){
        float total = 0;
        for(vehiculo v : pila){
            if(v.comprador.equals(LoginController.nom)){
                total += v.precio;
            }
        }
        return total;
    }
```

```java
    public void eliminarProducto (int id){
        for(vehiculo elim : pila){
            if(elim.id == id && elim.comprador.equals(LoginController.nom)){
                pila.remove(elim);
                CarritoController.tablist.remove(elim);
                JOptionPane.showMessageDialog(null,"Producto eliminado");
                break;
            }
        }
    }

    public void comprarProducto (int id){
        for(vehiculo comprar : pila){
            if(comprar.id == id && comprar.comprador.equals(LoginController.nom)){
                PrincipalController.cola.añadirProducto(comprar);
                pila.remove(comprar);
                CarritoController.tablist.remove(comprar);
                JOptionPane.showMessageDialog(null,"Producto comprado!");
                break;
            }
        }
    }

    public void comprarTodo(){
        Stack<vehiculo> nuevaPila = new Stack<>();
        while(!pila.isEmpty()){
            vehiculo elem = pila.pop();
            if (!elem.comprador.equals(LoginController.nom)){
                nuevaPila.push(elem);
            }else{
                PrincipalController.cola.añadirProducto(elem);
            }
        }
        while(!nuevaPila.isEmpty()){
            pila.push(nuevaPila.pop());
        }
        JOptionPane.showMessageDialog(null, "Se compraron todos los productos");
    }
}
```

-Ahora la cola de la lista de deseos.

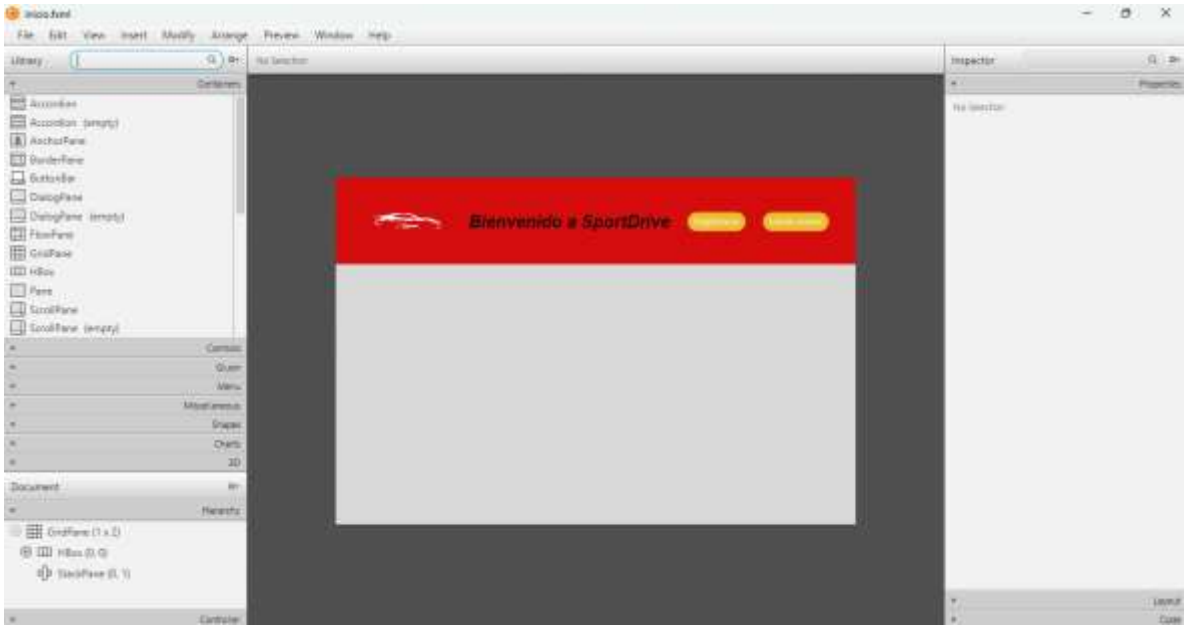


```
50      public void eliminarProducto (int id){
51          for(vehiculo elim : cola){
52              if(elim.id == id && elim.comprador.equals(LoginController.nom)){
53                  cola.remove(elim);
54                  DeseosController.tablist.remove(elim);
55                  JOptionPane.showMessageDialog(null,"Producto eliminado");
56                  break;
57              }
58          }
59      }
60
61      public void comprarProducto (int id){
62          for(vehiculo comprar : cola){
63              if(comprar.id == id && comprar.comprador.equals(LoginController.nom)){
64                  PrincipalController.cola.añadirProducto(comprar);
65                  cola.remove(comprar);
66                  DeseosController.tablist.remove(comprar);
67                  JOptionPane.showMessageDialog(null,"Producto comprado!");
68                  break;
69              }
70          }
71      }
72  }
```

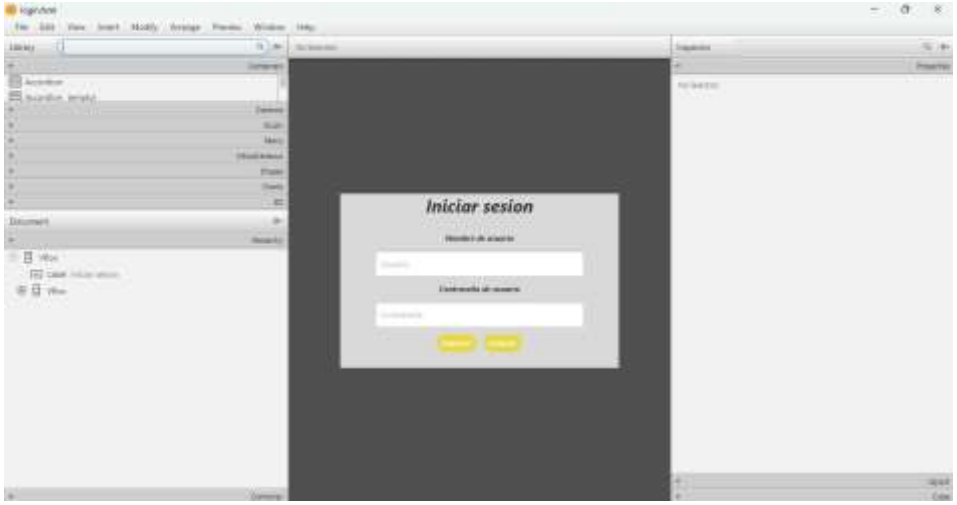
-Finalmente la cola del historial de compras.

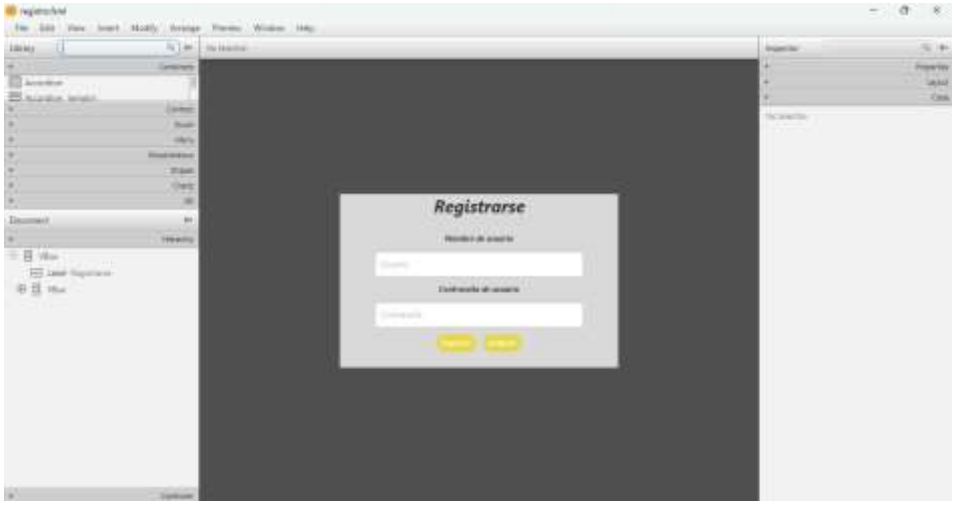
```java
package metodos;

import java.util.LinkedList;
import java.util.Queue;

/**
 *
 * @author Omar
 */
public class colaHistorial {

    public static Queue<vehiculo> cola;

    public colaHistorial(){
        this.cola = new LinkedList<>();
    }

    public void añadirProducto(vehiculo p){
        cola.add(p);
        System.out.println(cola.peek());
    }
    public void cargarCola(){
        for(vehiculo cargar : cola){
            if(cargar.comprador.equals(LoginController.nom)){
                HistorialController.tablist.add(cargar);
            }
        }
    }
}
```

3. Ahora se crean las interfaces del proyecto usando archivos fxml

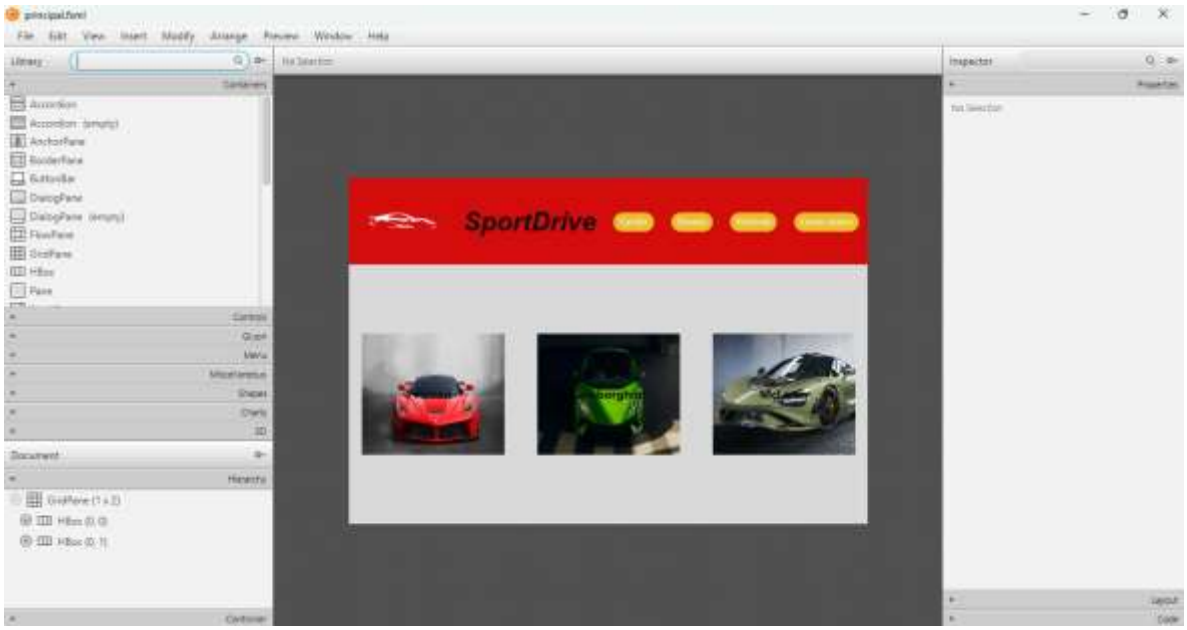


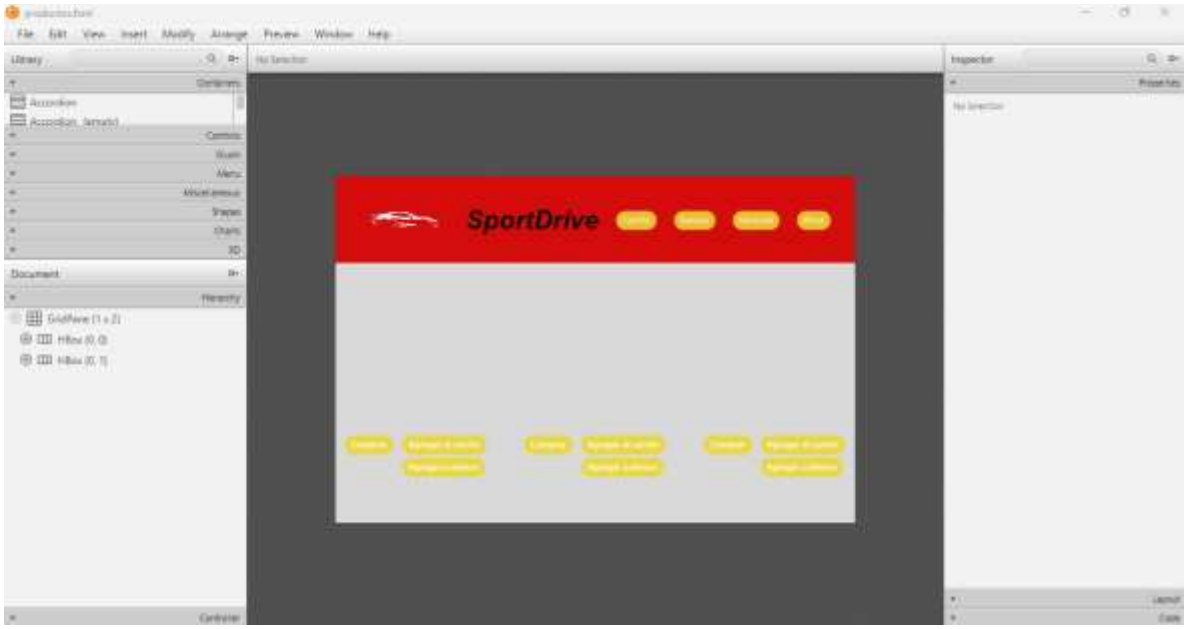
-En el inicio usamos un stackpanel para sobreponer el formulario de registro o de login



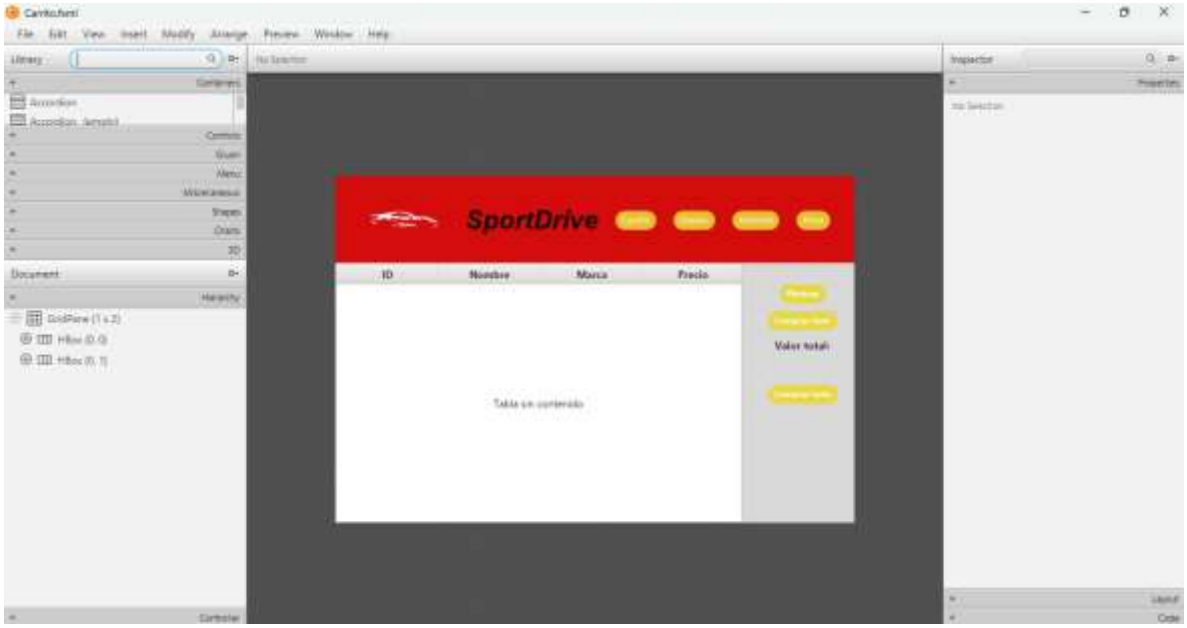


-Ahora se crean las demás interfaces

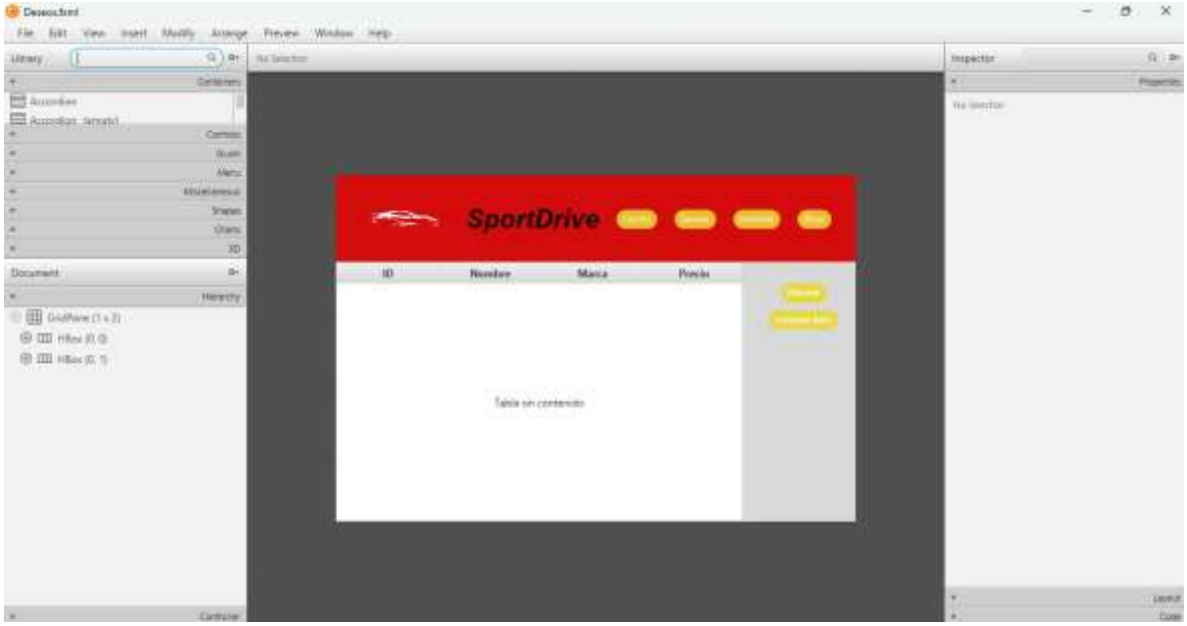




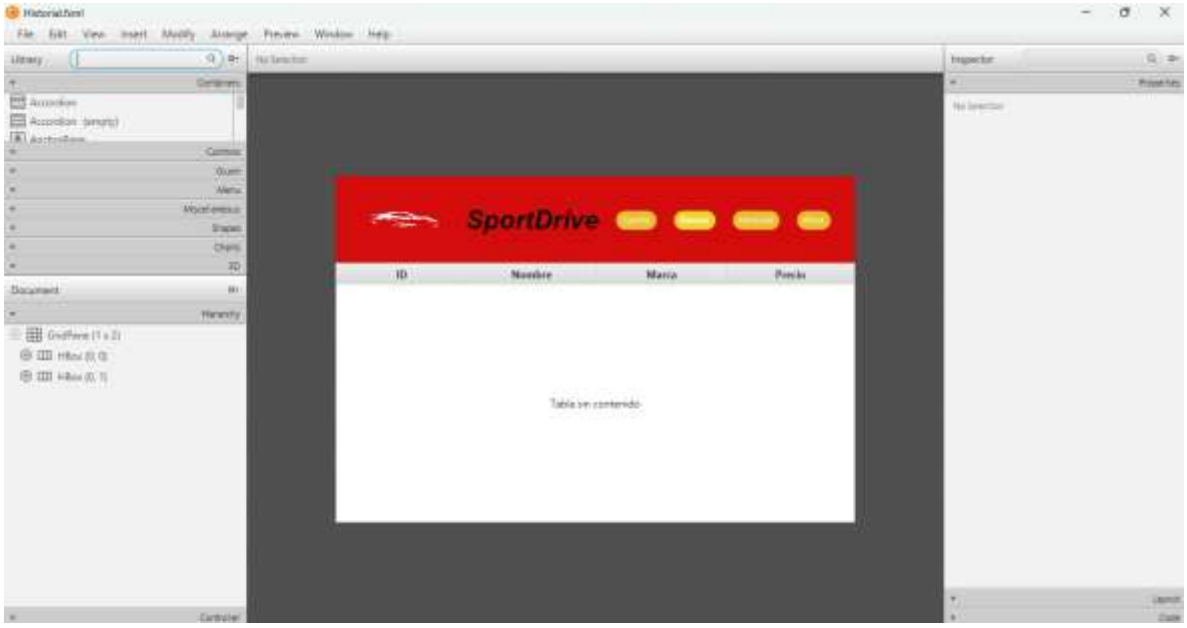
-Aquí los espacios vacíos se llenan dependiendo de la marca que seleccione el usuario.



-Carrito de compras



-Lista de deseos



-Historial de compras

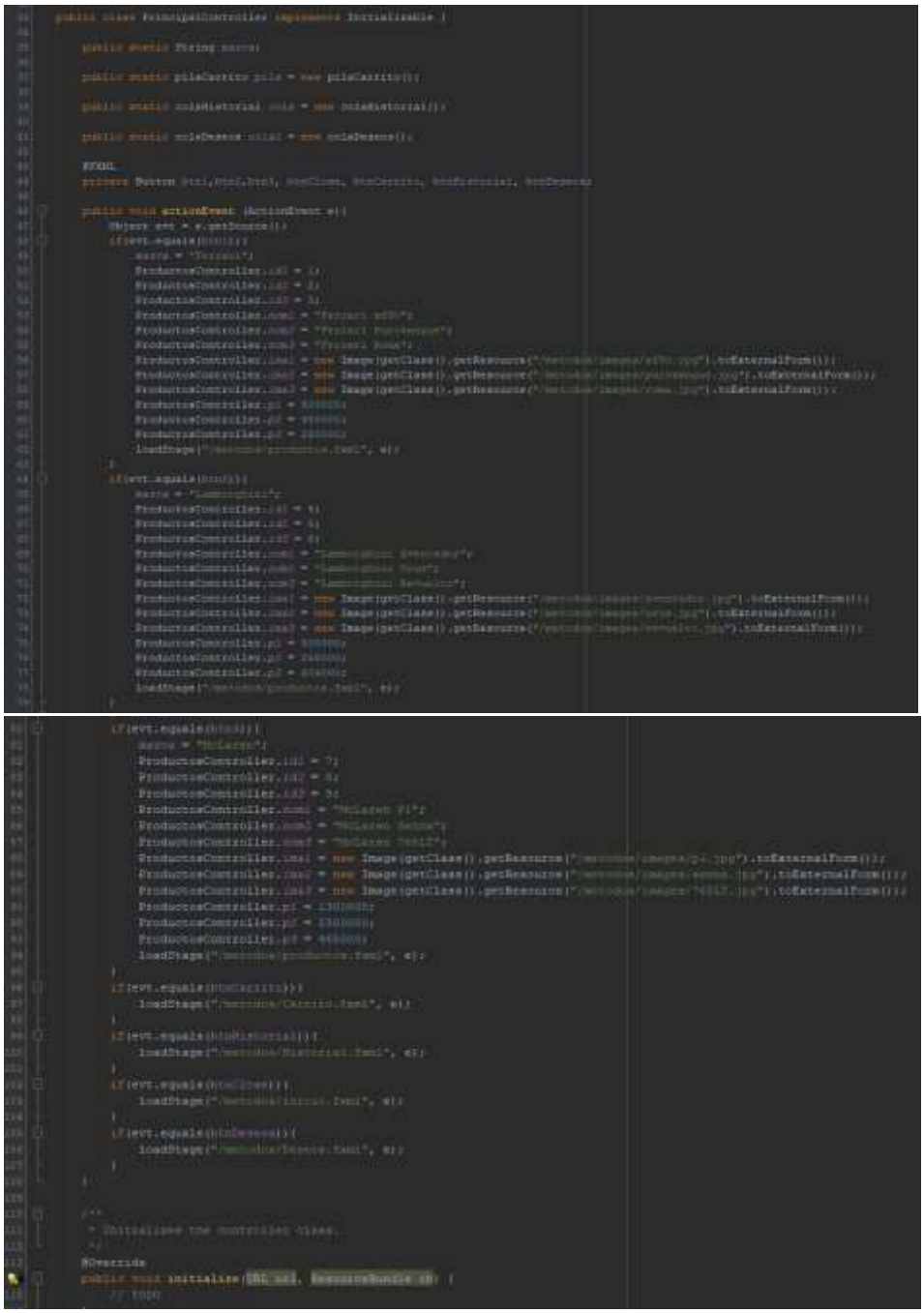
4. Ahora se crean los controladores para los archivos fxml

```
public class InicioController implements Initializable {

    @FXML
    private Button registrarse, iniciar;

    @FXML
    private StackPane stack;

    private VBox log, reg;

    @FXML
    private void actionEvent(ActionEvent e){

        Object evt = e.getSource();

        if(evt.equals(iniciar)){
            log.setVisible(true);
            reg.setVisible(false);
        }else if(evt.equals(registrarse)){
            log.setVisible(false);
            reg.setVisible(true);
        }

    }

    /**
     * Initializes the controller class.
     */
    @Override
    public void initialize(URL url, ResourceBundle rb) {
        try {
            log = LoadForm("/metodos/login.fxml");
            reg = LoadForm("/metodos/registro.fxml");
            stack.getChildren().addAll(log, reg);

            log.setVisible(true);
            reg.setVisible(false);

        } catch (IOException ex) {
            Logger.getLogger(InicioController.class.getName()).log(Level.SEVERE, null, ex);
        }
        // TODO
    }
    private VBox LoadForm(String url) throws IOException{
        return (VBox)FXMLLoader.load(getClass().getResource(url));
    }

}
```

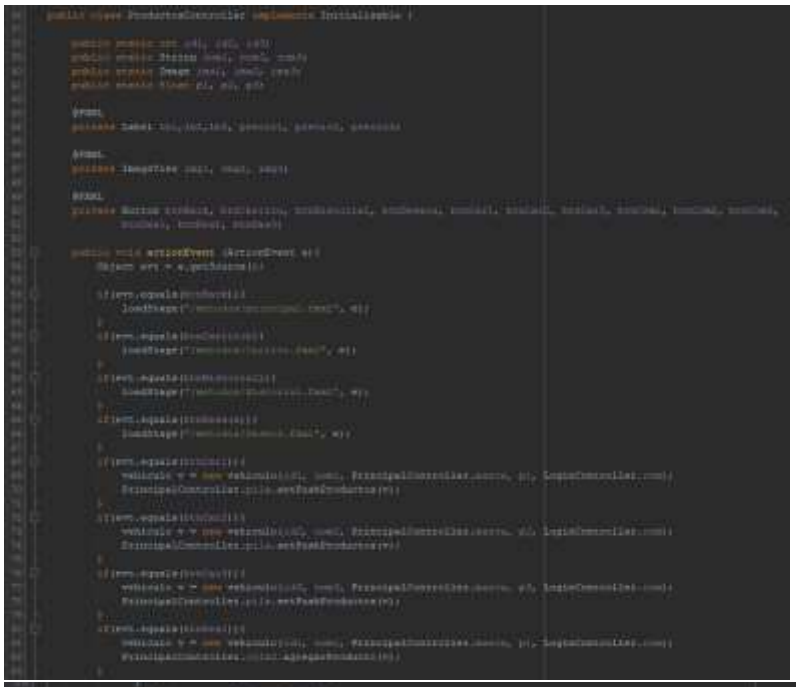
-Este es el controlador de la pagina de inicio, permite configurar la pagina que se esta cargando, ya sea la de login o la de registro

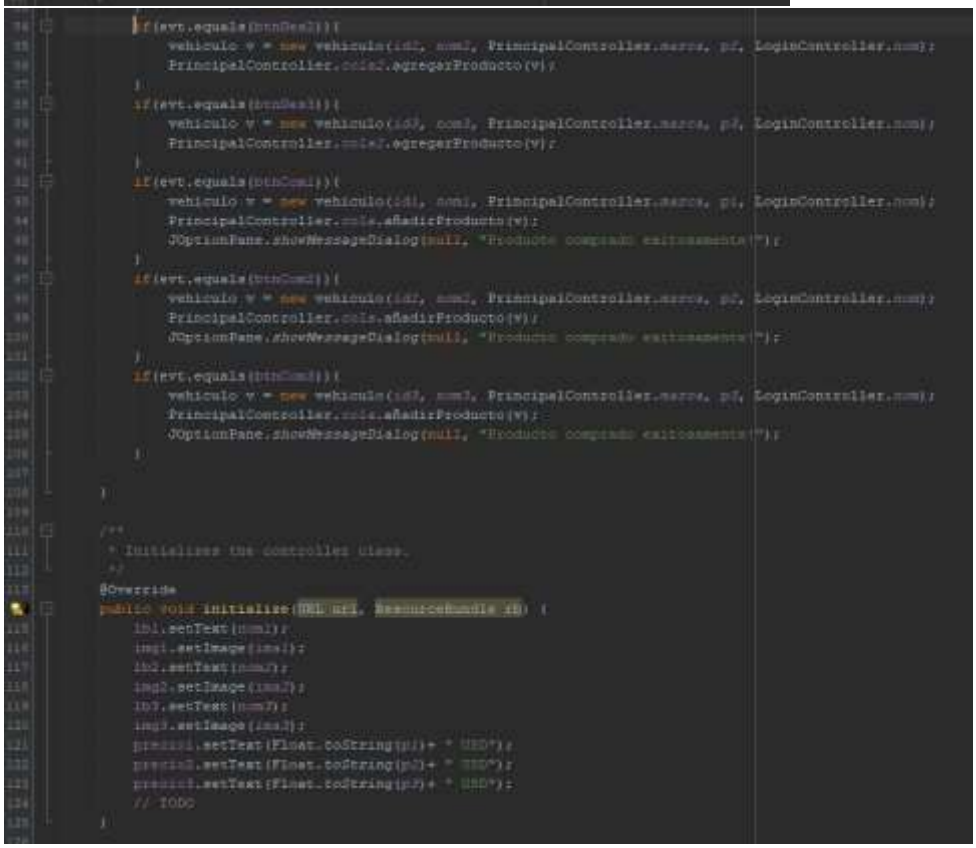
-Ese es el controlador del formulario del login

-Este es el controlador del formulario de registro.



-Ese es el controlador de la pagina principal, permite navegar a cualquier página del programa, además de que envía la información necesaria para cargar los productos en la página de productos.





-Este es el controlador de los productos, aquí se recibe la información de la pagina principal y se cargan los vehículos para vender según la marca

-Ese es el controlador del carrito, ahí se inicializa una tabla de javafx la cual se llena con los vehículos almacenados en la pila del carrito, además da las opciones de eliminar del carrito, comprar unidad, comprar todo y ver el valor total del carrito.

```
public class DeseosController implements Initializable {

    @FXML
    private TableView<vehiculo> tabla;
    @FXML
    private TableColumn<vehiculo, String> id;
    @FXML
    private TableColumn<vehiculo, String> nom;
    @FXML
    private TableColumn<vehiculo, String> mar;
    @FXML
    private TableColumn<vehiculo, String> pre;

    public static ObservableList<vehiculo> tablist = FXCollections.observableArrayList();

    @FXML
    private Button btnBack, btnCarrito, btnHistorial, btnEliminar, btnComprar1, btnDeseos;

    public void actionEvent (ActionEvent e){
        Object evt = e.getSource();
        if(evt.equals(btnEliminar)){
            int id = tabla.getSelectionModel().getSelectedItem().getId();
            PrincipalController.cola2.eliminarProducto(id);
            tablist.clear();
            PrincipalController.cola2.cargarTabla();
        }
        if(evt.equals(btnComprar1)){
            int id = tabla.getSelectionModel().getSelectedItem().getId();
            PrincipalController.cola2.comprarProducto(id);
            tablist.clear();
            PrincipalController.cola2.cargarTabla();
        }
        if(evt.equals(btnHistorial)){
            tablist.clear();
            loadStage("/metodos/Historial.fxml", e);
        }
        if(evt.equals(btnCarrito)){
            loadStage("/metodos/Carrito.fxml", e);
        }
        if(evt.equals(btnBack)){
            tablist.clear();
            loadStage("/metodos/principal.fxml", e);
        }
        if(evt.equals(btnDeseos)){
            JOptionPane.showMessageDialog(null, "Ya se encuentra en la pagina de deseos!");
        }
    }
```

```
    @Override
    public void initialize(URL url, ResourceBundle rb) {
        PrincipalController.cola2.cargarTabla();
        id.setCellValueFactory(new PropertyValueFactory<vehiculo, String>("id"));
        nom.setCellValueFactory(new PropertyValueFactory<vehiculo, String>("nombre"));
        mar.setCellValueFactory(new PropertyValueFactory<vehiculo, String>("marca"));
        pre.setCellValueFactory(new PropertyValueFactory<vehiculo, String>("precio"));
        tabla.setItems(tablist);
        // TODO
    }
```

-Ese es el controlador del carrito, en ese controlador se carga otra tabla de javafx con la diferencia de que se llena con los productos que se encuentren en la cola de la lista de deseos.

```
public class HistorialController implements Initializable {

    @FXML
    private Button btnBack, btnCarrito, btnHistorial, btnDeseos;

    @FXML
    private TableView<vehiculo> tablo;
    @FXML
    private TableColumn<vehiculo, String> id;
    @FXML
    private TableColumn<vehiculo, String> nom;
    @FXML
    private TableColumn<vehiculo, String> mar;
    @FXML
    private TableColumn<vehiculo, String> pre;

    public static ObservableList<vehiculo> tablist = FXCollections.observableArrayList();


    public void actionEvent (ActionEvent e){
        Object evt = e.getSource();

        if(evt.equals(btnHistorial)){
            JOptionPane.showMessageDialog(null, "Ya se encuentra en la pagina del historial!");
        }
        if(evt.equals(btnCarrito)){
            tablist.clear();
            loadStage("/metodos/Carrito.fxml", e);
        }
        if(evt.equals(btnBack)){
            tablist.clear();
            loadStage("/metodos/principal.fxml", e);
        }
        if(evt.equals(btnDeseos)){
            loadStage("/metodos/Deseos.fxml", e);
        }
    }
```

```
    @Override
    public void initialize(URL url, ResourceBundle rb) {
        // TODO
        PrincipalController.cola.cargarCola();
        id.setCellValueFactory(new PropertyValueFactory<vehiculo, String>("id"));
        nom.setCellValueFactory(new PropertyValueFactory<vehiculo, String>("nombre"));
        mar.setCellValueFactory(new PropertyValueFactory<vehiculo, String>("marca"));
        pre.setCellValueFactory(new PropertyValueFactory<vehiculo, String>("precio"));
        tabla.setItems(tablist);
    }
```

-Finalmente el controlador del historial que también carga una tabla la cual se llena con los vehículos almacenados en la cola del historial.

5. Link del proyecto subido a GitHub:
   https://github.com/OmarYessid/ProyectoFinal.git