



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.

```
5 package metodos;  
6  
7 /**  
8  *  
9  * @author Omar  
10 */  
11 public class user {  
12     String nombre, contraseña;  
13     user sig;  
14  
15     public user(String nombre, String contraseña) {  
16         this.nombre = nombre;  
17         this.contraseña = contraseña;  
18         sig = null;  
19     }  
20 }  
21
```

```
5 package metodos;  
6  
7 /**  
8  *  
9  * @author Omar  
10 */  
11 public class vehiculo {  
12     int id;  
13     String nombre, marca, comprador;  
14     float precio;  
15  
16     public vehiculo() {  
17     }  
18  
19     public vehiculo(int id, String nombre, String marca, float precio, String comprador) {  
20         this.id = id;  
21         this.nombre = nombre;  
22         this.marca = marca;  
23         this.precio = precio;  
24         this.comprador = comprador;  
25     }  
26  
27     public int getId() {  
28         return id;  
29     }  
30  
31     public void setId(int id) {  
32         this.id = id;  
33     }  
34  
35     public String getNombre() {  
36         return nombre;  
37     }  
38  
39     public void setNombre(String nombre) {  
40         this.nombre = nombre;  
41     }  
42  
43     public String getMarca() {  
44         return marca;  
45     }  
46  
47     public void setMarca(String marca) {  
48         this.marca = marca;  
49     }  
50 }  
51
```

```

    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

```

5   package metodos;
6
7   import java.io.BufferedWriter;
8   import java.io.FileWriter;
9   import java.io.IOException;
10  import javax.swing.JOptionPane;
11
12  /**
13   *
14   * @author Omar
15   */
16  public class listacuentas {
17      public static user cab;
18
19      public listacuentas () { cab=null; }
20
21      public user buscardom(String user){
22          user p = cab;
23          if(cab == null)
24              return null;
25          else{
26              while (p != null){
27                  if ((p.nombre).equals(user))
28                      return p;
29                  else
30                      p=p.sig;
31              }
32              return null;
33          }
34      }
35  }

```

```

36 public user crearnodo (String nombre, String contraseña){
37     user info = p;
38     String nom;
39     if (nombre.equals("") || contraseña.equals("")){
40         JOptionPane.showMessageDialog(null, "Faltan los campos con obligatorios");
41         return null;
42     }
43     while (
44         nom = nombre;
45         p = buscardatos(nom);
46         if (p != null){
47             JOptionPane.showMessageDialog(null, "Ya existe una cuenta con ese nombre, ingrese uno diferente");
48             return null;
49         }
50         while (p != null){
51             if (p == null){
52                 info = new user (nom, contraseña);
53                 JOptionPane.showMessageDialog(null, "Registro exitoso");
54                 return info;
55             }
56             return null;
57         }
58     }
59 }
60
61 public void crearcuenta (String nombre, String contraseña){
62     user info = crearnodo (nombre, contraseña);
63     if (info != null){
64         info.sig = info;
65         cab = info;
66     }
67     cab = info;
68 }
69 try {BufferedWriter writer = new BufferedWriter(new FileWriter("usuarios.txt", true)); {
70     writer.write("Usuario " + nombre + ", Contraseña " + contraseña);
71     writer.newLine();
72 } catch (IOException e) {
73     System.out.println("Error al escribir en el archivo " + e.getMessage());
74 }
75 }

```

```

36 public user logcuenta (String nombre, String contraseña){
37     user p = cab;
38     if (cab == null)
39         return null;
40     else{
41         while (p != null){
42             if ((p.nombre).equals(nombre) && (p.contraseña).equals(contraseña))
43                 return p;
44             else
45                 p=p.sig;
46         }
47         return null;
48     }
49 }
50
51 }

```

-Ahora la pila del carrito de compras.

```
package metodos;

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

/**
 *
 * @author Oscar
 */
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 cargarFila() {
        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;
    }

    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.agregarProducto(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.agregarProducto(elem);
            }
        }
        while(!nuevaPila.isEmpty()) {
            pila.push(nuevaPila.pop());
        }
        JOptionPane.showMessageDialog(null, "Se compraron todos los productos");
    }
}
```

-Ahora la cola de la lista de deseos.

```
1 package metodos;
2
3 import java.util.LinkedList;
4 import java.util.Queue;
5 import javax.swing.JOptionPane;
6
7 /**
8  *
9  * @author Oscar
10  */
11 public class colaDeseos {
12     public static Queue<vehiculo> cola;
13
14     public colaDeseos() {
15         this.cola = new LinkedList<>();
16     }
17
18     public Queue<vehiculo> get_deseos() {
19         return cola;
20     }
21
22     public void agregarProducto(vehiculo prod) {
23         boolean existe = false;
24
25         for (vehiculo p : cola) {
26             if (p.getIdentific() .equals(prod.getIdentific()) && p.comprador.equals(LoginController.nom)) {
27                 JOptionPane.showMessageDialog(null, "Este producto ya se encuentra en deseos");
28                 existe = true;
29             }
30         }
31
32         if (!existe && prod.comprador.equals(LoginController.nom)) {
33             cola.add(prod);
34             JOptionPane.showMessageDialog(null, "Producto agregado a deseos: " + prod.getIdentific());
35         }
36     }
37
38     public void cargarTabla() {
39         for (vehiculo cargar : cola) {
40             if (cargar.comprador.equals(LoginController.nom)) {
41                 DeseosController.tablist.add(cargar);
42             }
43         }
44     }
45 }
```

```
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.agregarProducto(comprar);
65             cola.remove(comprar);
66             DeseosController.tablist.remove(comprar);
67             JOptionPane.showMessageDialog(null,"Producto comprado!");
68             break;
69         }
70     }
71 }
72 }
```

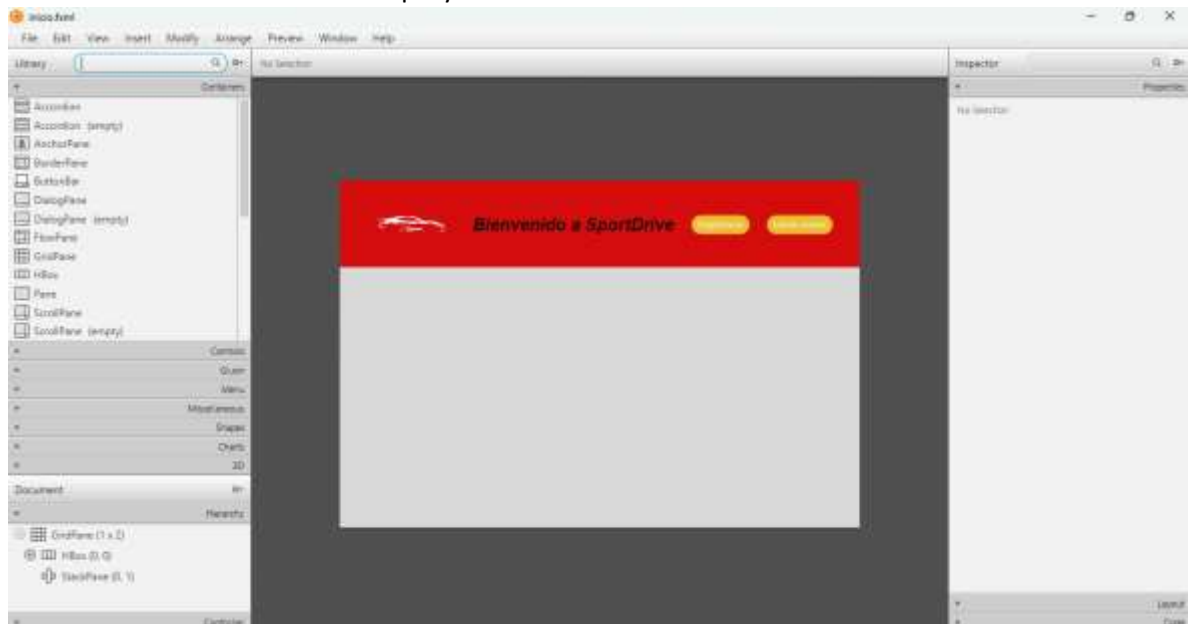
-Finalmente la cola del historial de compras.

```

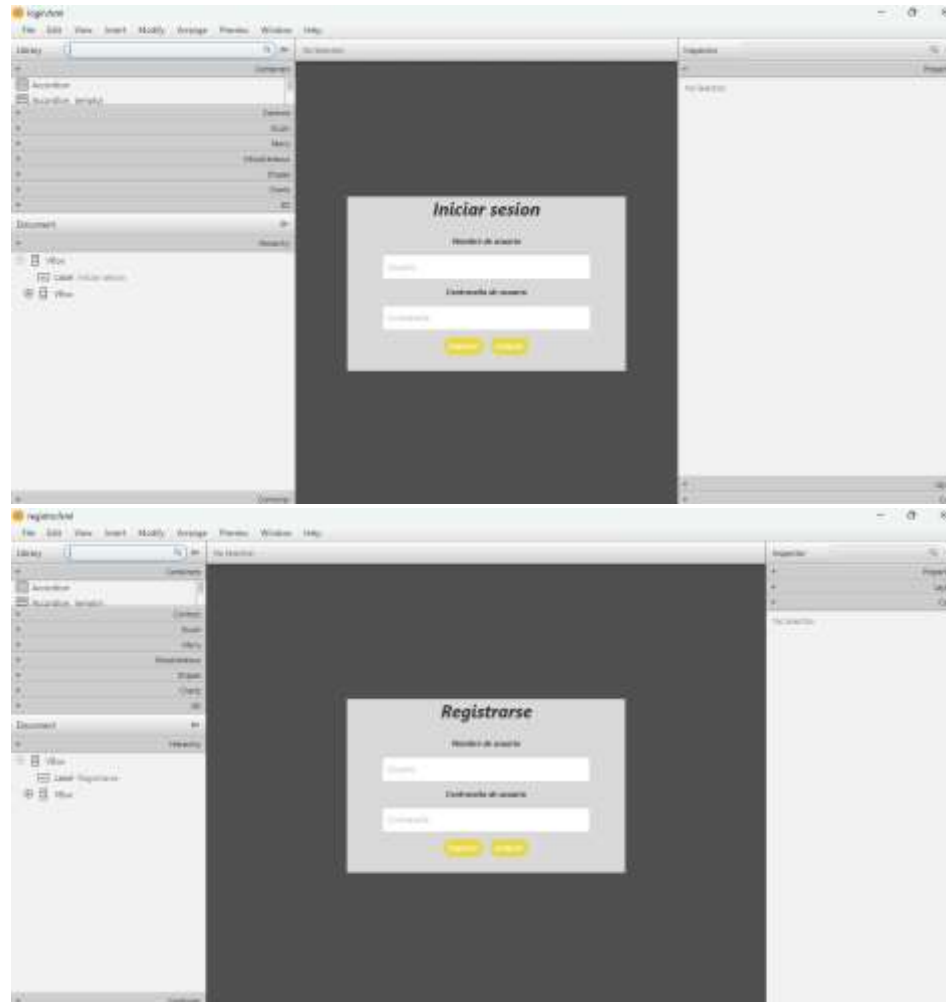
5 package metodos;
6
7 import java.util.LinkedList;
8 import java.util.Queue;
9
10 /**
11  *
12  * @author Omar
13  */
14 public class colaHistorial {
15
16     public static Queue<vehiculo> cola;
17
18     public colaHistorial() {
19         this.cola = new LinkedList<>();
20     }
21
22     public void añadirProducto(vehiculo p) {
23         cola.add(p);
24         System.out.println(cola.peek());
25     }
26
27     public void cargarCola() {
28         for(vehiculo cargar : cola) {
29             if(cargar.comprador.equals(LoginController.nom)) {
30                 HistorialController.tablist.add(cargar);
31             }
32         }
33     }
34 }

```

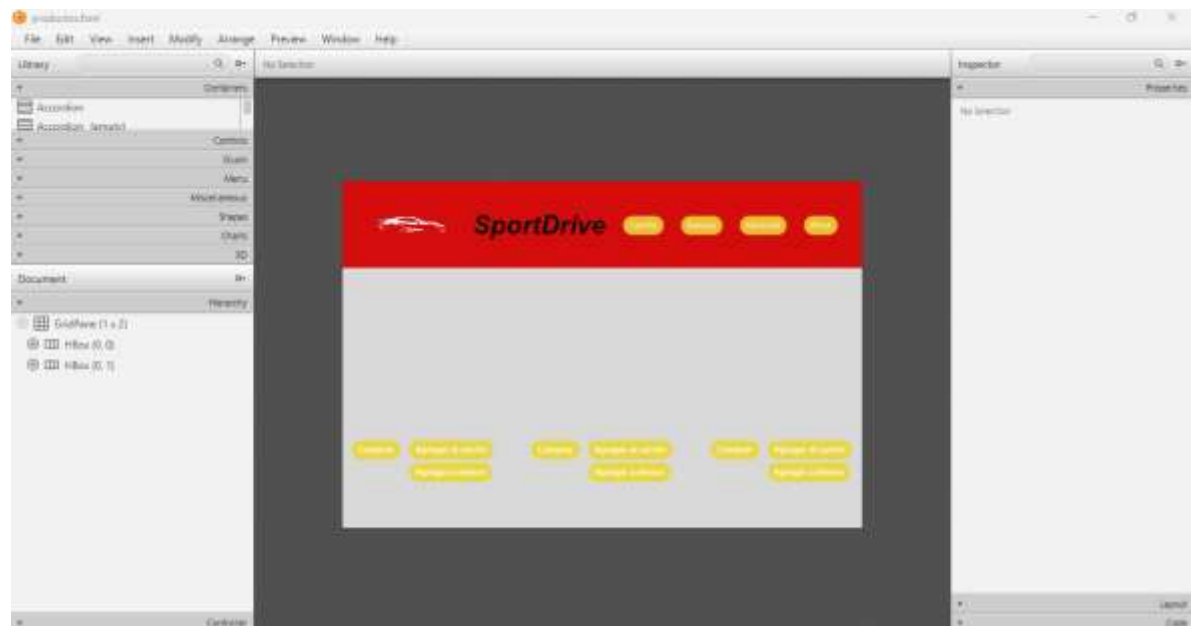
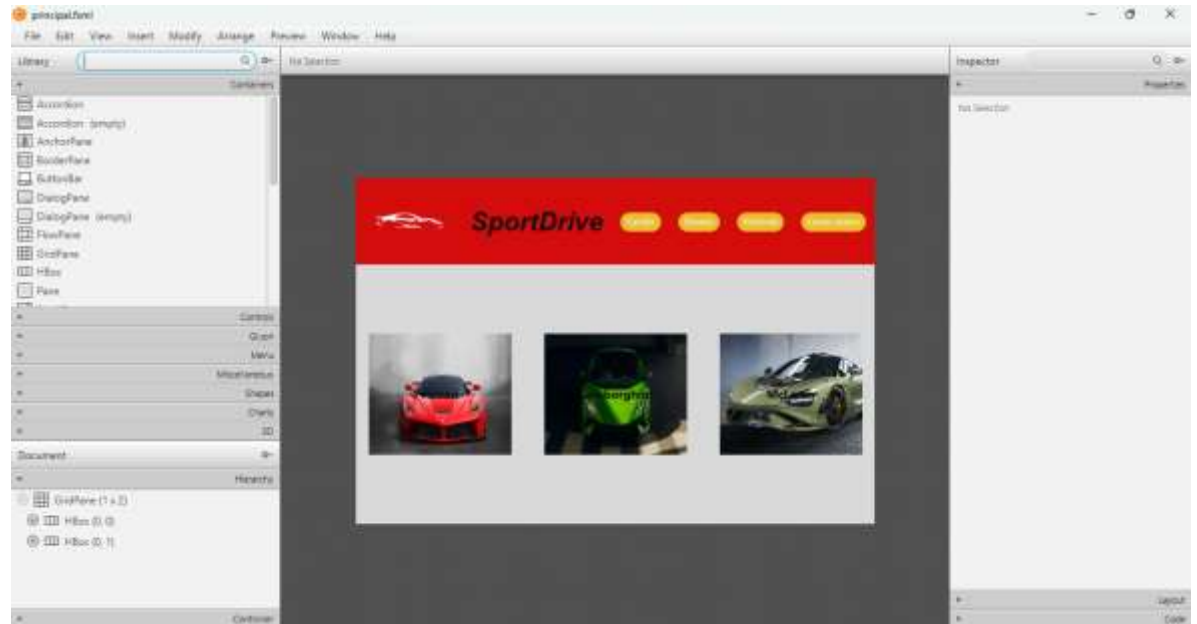
3. Ahora se crean las interfaces del proyecto usando archivos fxml



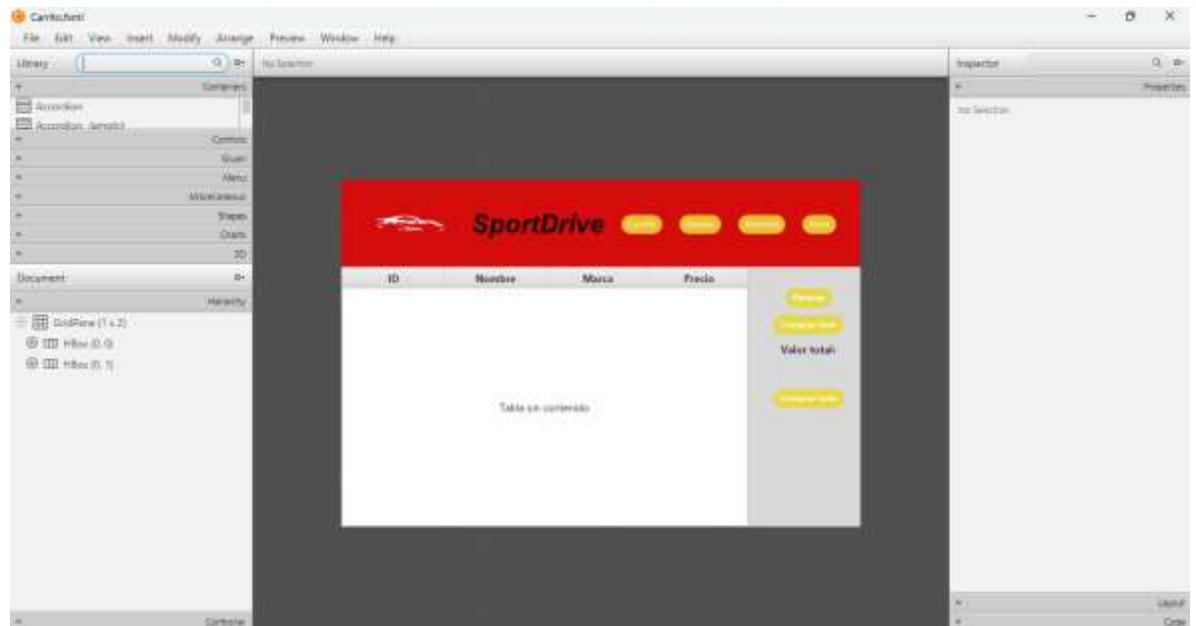
-En el inicio usamos un stackpanel para sobreponer el formulario de registro o de login



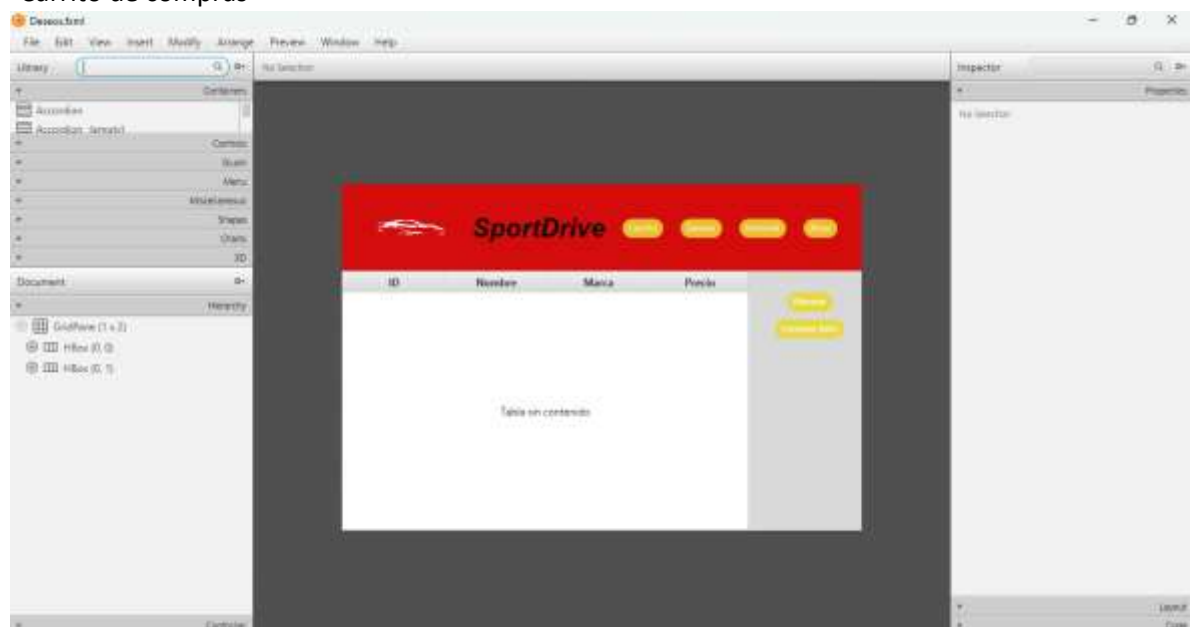
-Ahora se crean las demás interfaces



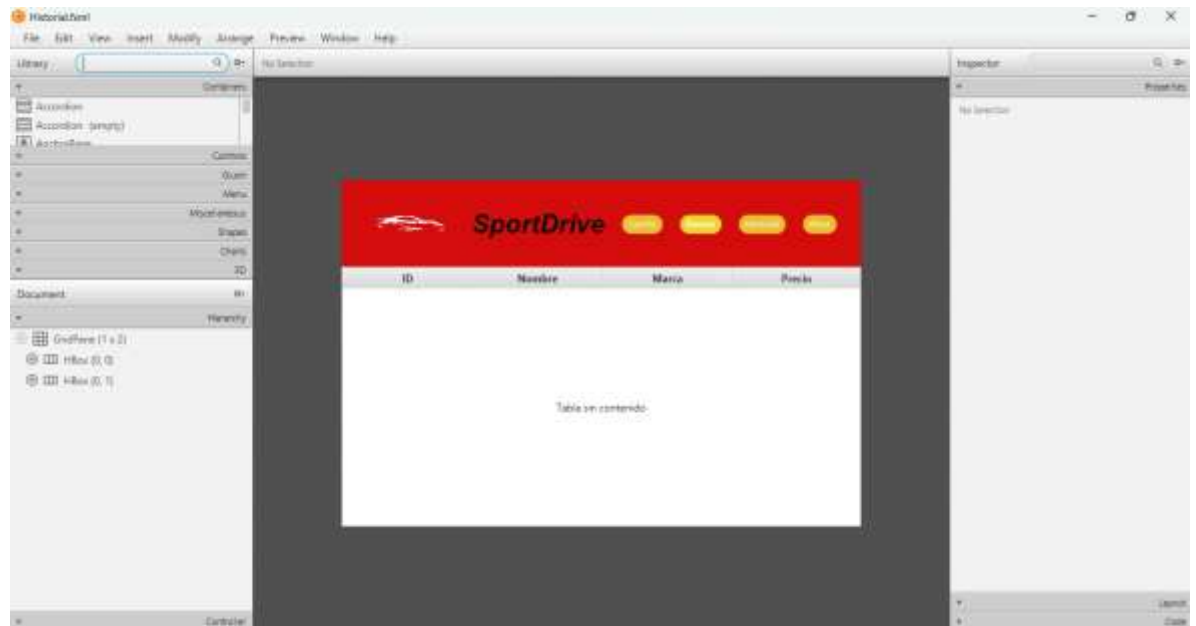
-Aquí los espacios vacíos se llenan dependiendo de la marca que seleccione el usuario.



-Carrito de compras



-Lista de deseos



-Historial de compras

4. Ahora se crean los controladores para los archivos fxml

```
25 public class InicioController implements Initializable {
26
27     @FXML
28     private Button registrarse, iniciar;
29
30     @FXML
31     private StackPane stack;
32
33     private VBox log, reg;
34
35     @FXML
36     private void handleEvent(ActionEvent e) {
37
38         Object evt = e.getSource();
39
40         if(evt.equals(iniciar)){
41             log.setVisible(true);
42             reg.setVisible(false);
43         }else if(evt.equals(registrarse)){
44             log.setVisible(false);
45             reg.setVisible(true);
46         }
47     }
48
49     /**
50      * Initializes the controller class.
51      */
52     @Override
53     public void initialize(URL url, ResourceBundle rb) {
54         try {
55             log = LoadForm("/metodos/login.fxml");
56             reg = LoadForm("/metodos/registro.fxml");
57             stack.getChildren().addAll(log, reg);
58
59             log.setVisible(true);
60             reg.setVisible(false);
61
62         } catch (IOException ex) {
63             Logger.getLogger(InicioController.class.getName()).log(Level.SEVERE, null, ex);
64         }
65         // TODO
66     }
67
68     private VBox LoadForm(String url) throws IOException{
69         return (VBox)FXMLLoader.load(getClass().getResource(url));
70     }
71
72 }
```

-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

```

26 public class LoginController implements Initializable {
27     public static user usuario;
28     public static String error;
29
30     @FXML
31     private TextField user;
32
33     @FXML
34     private PasswordField pass;
35
36     @FXML
37     private Button ingresar, limpiar;
38
39     public void actionEvent(ActionEvent e){
40
41         Object evt = e.getSource();
42
43         if(evt.equals(ingresar)){
44             String nombre = user.getText();
45             String contraseña = pass.getText();
46             usuario = RegistroController.lista.loguear(nombre, contraseña);
47             if(usuario == null){
48                 JOptionPane.showMessageDialog(null, "Datos incorrectos");
49             }
50             else{
51                 user.setText("");
52                 loadStage("welcome/principal.fxml", 4);
53             }
54         }
55         if(evt.equals(limpiar)){
56             user.setText("");
57             pass.setText("");
58         }
59     }
60
61     /**
62      * Initialize the controller class.
63      */
64     @Override
65     public void initialize(URL url, ResourceBundle rb) {
66         // TODO
67     }
68 }

```

-Ese es el controlador del formulario del login

```

public class RegistroController implements Initializable {
    public static listasCuentas lista = new listasCuentas();

    @FXML
    private TextField user;

    @FXML
    private PasswordField pass;

    @FXML
    private Button ingresar, limpiar;

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

        if(evt.equals(ingresar)){
            String nombre = user.getText();
            String contraseña = pass.getText();
            lista.crearCuenta(nombre, contraseña);
            user.setText("");
            pass.setText("");
        }

        if(evt.equals(limpiar)){
            user.setText("");
            pass.setText("");
        }
    }

    /**
     * Initialize the controller class.
     */
    @Override
    public void initialize(URL url, ResourceBundle rb) {
        // TODO
    }
}

```

-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.


```

39 public class CarritoController implements Initializable {
40
41     @FXML
42     private TableView<vehiculo> tabla;
43     @FXML
44     private TableColumn<vehiculo, String> id;
45     @FXML
46     private TableColumn<vehiculo, String> marca;
47     @FXML
48     private TableColumn<vehiculo, String> precio;
49     @FXML
50     private TableColumn<vehiculo, String> peso;
51
52     public static ObservableList<vehiculo> tabList = FXCollections.observableArrayList();
53
54     @FXML
55     private Button btnBack, btnCarrito, btnHistorial, btnEliminar, btnComprar, btnComprarT, btnDesena;
56
57     @FXML
58     private Label lbTotal;
59
60     public void actionEvent (ActionEvent e){
61         Object evt = e.getSource();
62
63         if(evt.equals(btnEliminar)){
64             int id = tabla.getSelectionModel().getSelectedItem().getId();
65             PrincipalController.pila.eliminarProducto(id);
66             tabla.clear();
67             PrincipalController.pila.cargarTabla();
68             float total = PrincipalController.pila.valorTotal();
69             lbTotal.setText(Float.toString(total));
70         }
71         if(evt.equals(btnComprarT)){
72             tabla.clear();
73             PrincipalController.pila.comprarTodo();
74             float total = PrincipalController.pila.valorTotal();
75             lbTotal.setText(Float.toString(total));
76         }
77         if(evt.equals(btnComprar)){
78             int id = tabla.getSelectionModel().getSelectedItem().getId();
79             PrincipalController.pila.comprarProducto(id);
80             tabla.clear();
81             PrincipalController.pila.cargarTabla();
82             float total = PrincipalController.pila.valorTotal();
83             lbTotal.setText(Float.toString(total));
84         }
85
86         if(evt.equals(btnHistorial)){
87             tabList.clear();
88             loadStage("/resources/Historial.fxml", e);
89         }
90         if(evt.equals(btnCarrito)){
91             JOptionPane.showMessageDialog(null, "Se encuentra en la pagina del carrito!");
92         }
93         if(evt.equals(btnBack)){
94             tabList.clear();
95             loadStage("/resources/principal.fxml", e);
96         }
97         if(evt.equals(btnDesena)){
98             loadStage("/resources/Desena.fxml", e);
99         }
100
101         /**
102          * Inicializa the controller class.
103          */
104         @Override
105         public void initialize(URL url, ResourceBundle rb) {
106             // TODO
107             PrincipalController.pila.cargarTabla();
108             float total = PrincipalController.pila.valorTotal();
109             lbTotal.setText(Float.toString(total));
110             id.setCellValueFactory(new PropertyValueFactory<vehiculo, String>("id"));
111             marca.setCellValueFactory(new PropertyValueFactory<vehiculo, String>("marca"));
112             precio.setCellValueFactory(new PropertyValueFactory<vehiculo, String>("precio"));
113             peso.setCellValueFactory(new PropertyValueFactory<vehiculo, String>("peso"));
114             tabla.setItems(tabList);
115         }
116     }

```

-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.


```

80 public class DeseosController implements Initializable {
81
82     @FXML
83     private TableView<vehiculo> tabla;
84     @FXML
85     private TableColumn<vehiculo, String> id;
86     @FXML
87     private TableColumn<vehiculo, String> nom;
88     @FXML
89     private TableColumn<vehiculo, String> mar;
90     @FXML
91     private TableColumn<vehiculo, String> pre;
92
93     public static ObservableList<vehiculo> tabList = FXCollections.observableArrayList();
94
95     @FXML
96     private Button btnBask, btnCarrito, btnHistorial, btnEliminar, btnComparar, btnDeseos;
97
98     public void actionEvent (ActionEvent e){
99         Object evt = e.getSource();
100         if(evt.equals(btnEliminar)){
101             int id = tabla.getSelectionModel().getSelectedItem().getId();
102             PrincipalController.colas2.eliminarProducto(id);
103             tabList.clear();
104             PrincipalController.colas2.cargarTabla();
105         }
106         if(evt.equals(btnComparar)){
107             int id = tabla.getSelectionModel().getSelectedItem().getId();
108             PrincipalController.colas2.compararProducto(id);
109             tabList.clear();
110             PrincipalController.colas2.cargarTabla();
111         }
112         if(evt.equals(btnHistorial)){
113             tabList.clear();
114             loadStage("/resources/Historial.fxml", e);
115         }
116         if(evt.equals(btnCarrito)){
117             loadStage("/resources/Carrito.fxml", e);
118         }
119         if(evt.equals(btnBask)){
120             tabList.clear();
121             loadStage("/resources/principal.fxml", e);
122         }
123         if(evt.equals(btnDeseos)){
124             JOptionPane.showMessageDialog(null, "Ya se encuentra en la página de deseos!");
125         }
126     }
127 }
128
129
130 @Override
131 public void initialize(URL url, ResourceBundle rb) {
132     PrincipalController.colas2.cargarTabla();
133     id.setCellValueFactory(new PropertyValueFactory<vehiculo, String>("id"));
134     nom.setCellValueFactory(new PropertyValueFactory<vehiculo, String>("nombre"));
135     mar.setCellValueFactory(new PropertyValueFactory<vehiculo, String>("marca"));
136     pre.setCellValueFactory(new PropertyValueFactory<vehiculo, String>("precio"));
137     tabla.setItems(tabList);
138     // TODO
139 }

```

-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.

```

38 public class HistorialController implements Initializable {
39
40     @FXML
41     private Button btnBack, btnCarrito, btnHistorial, btnDescos;
42
43     @FXML
44     private TableView<vehiculo> tabla;
45     @FXML
46     private TableColumn<vehiculo, String> id;
47     @FXML
48     private TableColumn<vehiculo, String> nom;
49     @FXML
50     private TableColumn<vehiculo, String> mar;
51     @FXML
52     private TableColumn<vehiculo, String> pre;
53
54     public static ObservableList<vehiculo> tablist = FXCollections.observableArrayList();
55
56
57     public void actionEvent (ActionEvent e){
58         Object evt = e.getSource();
59
60         if(evt.equals(btnHistorial)){
61             JOptionPane.showMessageDialog(null, "Ya se encuentra en la pagina del historial");
62         }
63         if(evt.equals(btnCarrito)){
64             tablist.clear();
65             loadStage("/metodos/Carrito.fxml", e);
66         }
67         if(evt.equals(btnBack)){
68             tablist.clear();
69             loadStage("/metodos/principal.fxml", e);
70         }
71         if(evt.equals(btnDescos)){
72             loadStage("/metodos/Descos.fxml", e);
73         }
74     }
75
76
77
78
79     @Override
80     public void initialize(URL url, ResourceBundle rb) {
81         // TODO
82         PrincipalController.cola.cargarCola();
83         id.setCellValueFactory(new PropertyValueFactory<vehiculo, String>("id"));
84         nom.setCellValueFactory(new PropertyValueFactory<vehiculo, String>("nombre"));
85         mar.setCellValueFactory(new PropertyValueFactory<vehiculo, String>("marca"));
86         pre.setCellValueFactory(new PropertyValueFactory<vehiculo, String>("precio"));
87         tabla.setItems(tablist);
88     }
89

```

-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>