Creación de la interfaz
CARROCÓMICO

Índice

Introducción

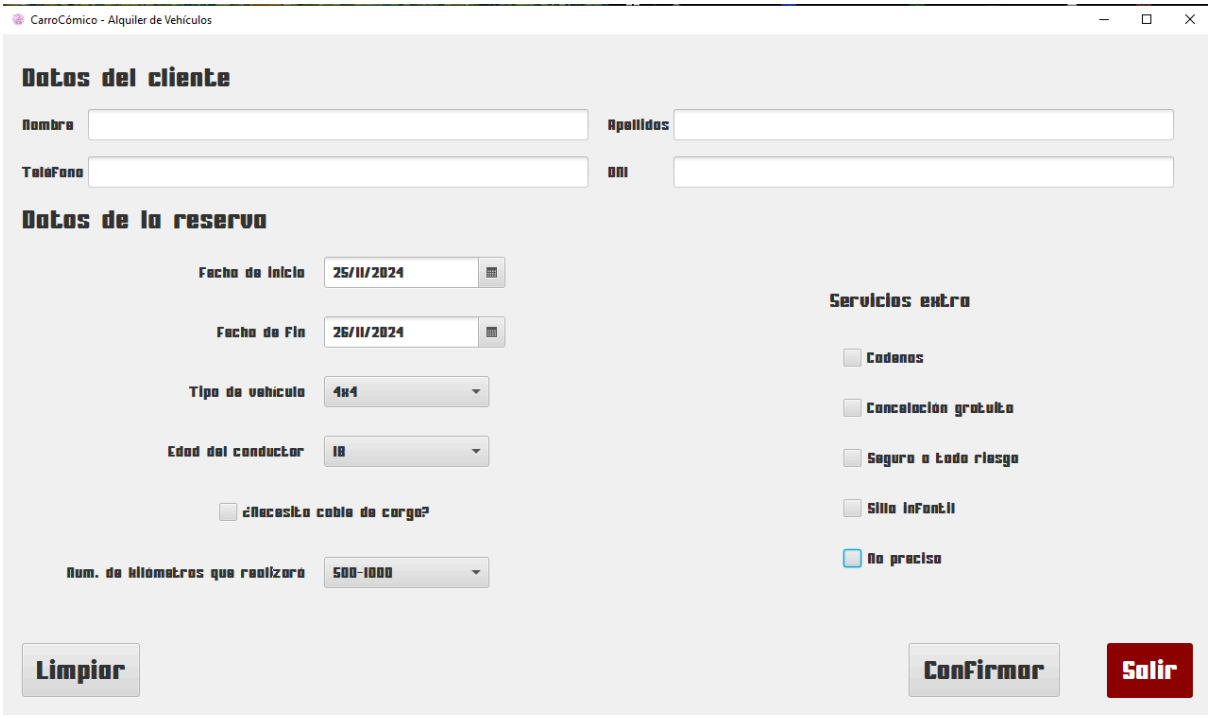
Este trabajo tiene como objetivo el diseño y desarrollo de una interfaz gráfica de usuario (GUI) para la gestión de alquileres de vehículos en CarroCómico, un innovador servicio de movilidad ubicado en TurboLandia. CarroCómico ofrece una experiencia única al poner a disposición de los usuarios una flota de coches y bicicletas eléctricas, promoviendo la movilidad sostenible y el cuidado del medio ambiente. El desafío ha sido crear una interfaz amigable y funcional que permita a los usuarios realizar reservas de manera intuitiva, gestionar sus preferencias y comunicarse con el servicio de manera eficiente. Para ello, se han empleado componentes adecuados para asegurar una experiencia visual atractiva y una distribución práctica de los elementos en la interfaz, cumpliendo con los requisitos establecidos para este proyecto. Además, se ha integrado la gestión de eventos, accesibilidad mediante combinaciones de teclas y una estética visual que refuerza el compromiso con la sostenibilidad, brindando una interacción fluida y coherente con el concepto de CarroCómico.

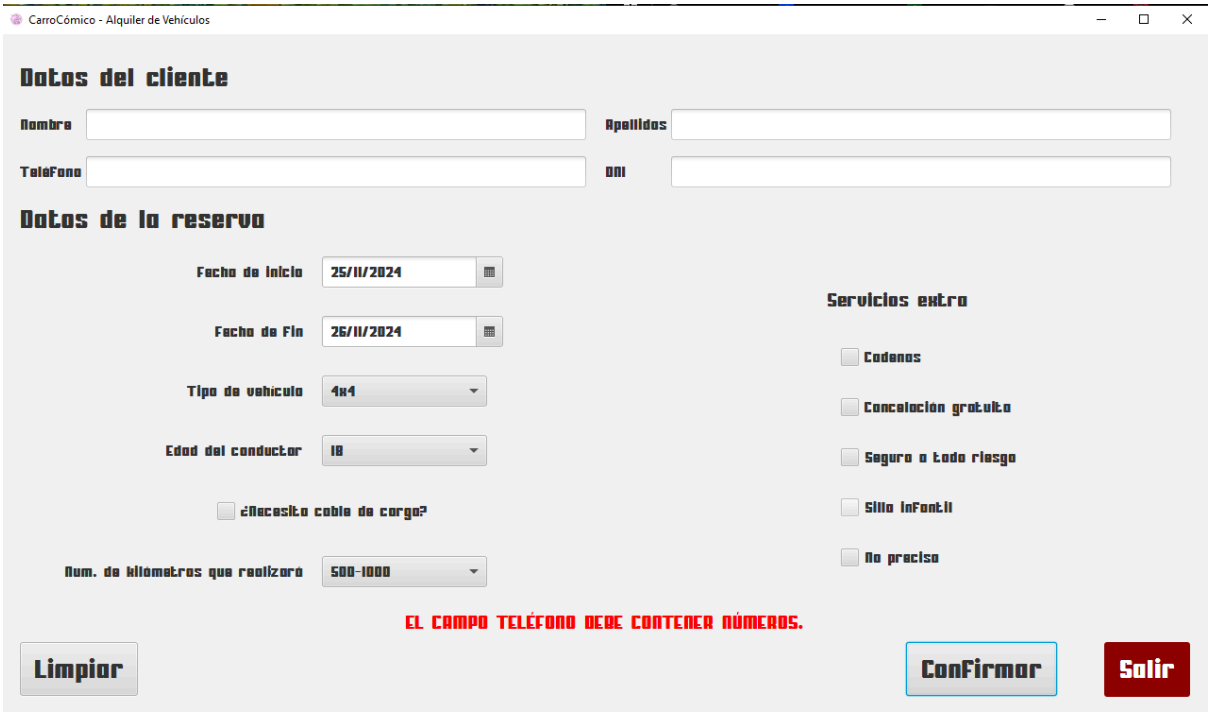
1- Creación de la interfaz empleando los componentes más adecuados.

Se han creado dos interfaces, la del menú principal y la de las reservas. Este proyecto cuenta con la misma estructura de VBOX y HBOX entrelazados para conseguir una estructura que permita crear una visual centrada y correctamente alineada.

- Menú principal: cuenta con un MenuBar con las opciones, con dos imágenes y tres botones. El MenuBar nos dirige a los coches eléctricos o a las bicicletas eléctricas (que generarán un mensaje de error al no haber disponibles). Tanto desde el MenuBar como desde los botones bajo las imágenes se puede acceder a los mismos sitios. El botón de Salir cierra la aplicación.

- Coche eléctrico: cuenta con un formulario completo dividido en dos secciones (Datos del cliente y Datos de reserva). Abajo del todo tiene 3 botones, el de aceptar, que comprueba que todos los inputs estén correctos y envía el formulario limpiando después el formulario, el de limpiar, que limpia todos los campos, y el de salir que nos devuelve al menú principal.


CarroCómico - Alquiler de Vehículos
Datos del cliente
Nombre
Apellidos
Teléfono
DNI
Datos de la reserva
Fecha de inicio
25/11/2024
Fecha de fin
26/11/2024
Tipo de vehículo
4x4
Edad del conductor
18
¿Necesita cable de carga?
Num. de kilómetros que realizará
500-1000
Servicios extra
Cadenas
Cancelación gratuita
Seguro a todo riesgo
Silla infantil
No preciso
Limpiar
Confirmar
Salir


CarroCómico - Alquiler de Vehículos
Datos del cliente
Nombre
Apellidos
Teléfono
DNI
Datos de la reserva
Fecha de inicio
25/11/2024
Fecha de fin
26/11/2024
Tipo de vehículo
4x4
Edad del conductor
18
¿Necesita cable de carga?
Num. de kilómetros que realizará
500-1000
Servicios extra
Cadenas
Cancelación gratuita
Seguro a todo riesgo
Silla infantil
No preciso
EL CAMPO TELÉFONO DEBE CONTENER NÚMEROS.
Limpiar
Confirmar
Salir

2- Cambiar los nombres y ToolTip para los elementos de formulario adecuados

Todos los elementos de la interfaz tienen los nombres cambiados y los campos tienen ToolTips

Menú principal:

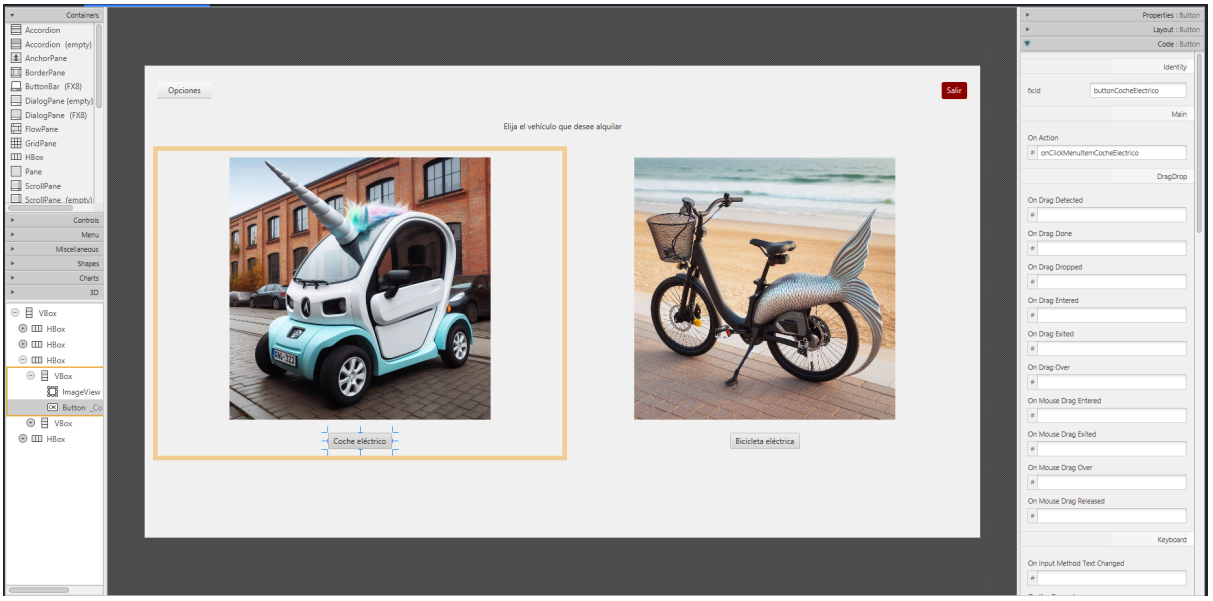
```
@FXML
private Label labelError;
@FXML
private Label labelElegir;
@FXML
private MenuBar menuBarOpciones;
@FXML
private Button buttonSalir;
@FXML
private Button buttonCocheElectrico;
@FXML
private Button buttonBicicletaElectrica;
```

Coche eléctrico:

```
@FXML
private Label labelDatosCliente;
@FXML
private Label labelNombre;
@FXML
private Label labelApellidos;
@FXML
private Label labelTelefono;
@FXML
private Label labelDNI;
@FXML
private Label labelDatosReserva;
@FXML
private Label labelFechaInicio;
@FXML
private Label labelFechaFin;
@FXML
private Label labelTipoVehiculo;
@FXML
private Label labelEdadConductor;
@FXML
private Label labelKM;
@FXML
private Label labelError;
@FXML
private TextField textFieldNombre;
@FXML
private TextField textFieldApellidos;
@FXML
private TextField textFieldTelefono;
```

```java
@FXML
private TextField textFieldDNI;
@FXML
private DatePicker datePickerFechaInicio;
@FXML
private DatePicker datePickerFechaFin;
@FXML
private ChoiceBox<String> choiceBoxTipoVehiculo;
@FXML
private ChoiceBox<Integer> choiceBoxEdadConductor;
@FXML
private CheckBox checkBoxCableCarga;
@FXML
private ChoiceBox<String> choiceBoxKM;
@FXML
private Label labelServiciosExtra;
@FXML
private CheckBox checkBoxNoPrecisa;
@FXML
private CheckBox checkBoxSillaInfantil;
@FXML
private CheckBox checkBoxSeguroTodoRiesgo;
@FXML
private CheckBox checkBoxCancelacionGratuita;
@FXML
private CheckBox checkBoxCadenas;
@FXML
private Button buttonConfirmar;
@FXML
private Button buttonLimpiar;
```

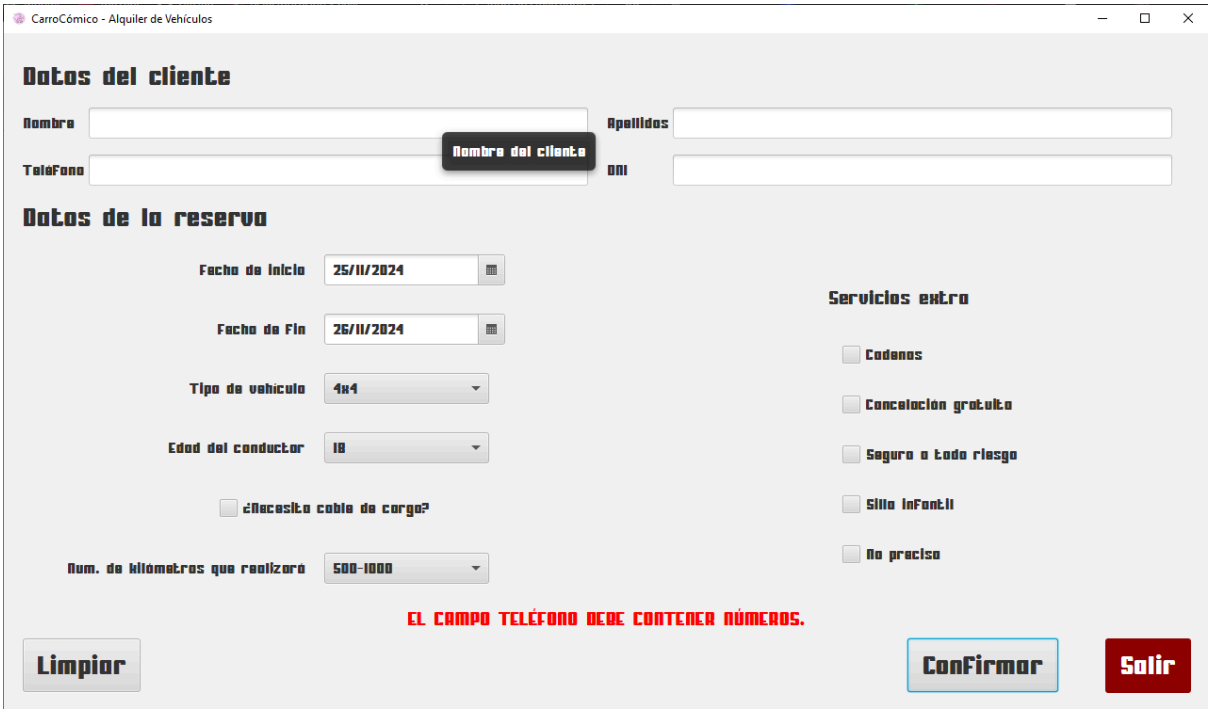
```
@FXML
private Button buttonLimpiar;
@FXML
private Button buttonSalir;
@FXML
private Tooltip toolTipNombre;
@FXML
private Tooltip toolTipApellidos;
@FXML
private Tooltip toolTipTelefono;
@FXML
private Tooltip toolTipDNI;
```



(Lo mismo que en la última imagen se repite en cada uno de los elementos necesarios)

Para comenzar, podemos ver como se han cambiado los nombres de los diferentes elementos que componen la visual en ambos controladores y en los elementos en el xml.

Por otro lado, se han incluído los ToolTips como se pedía.



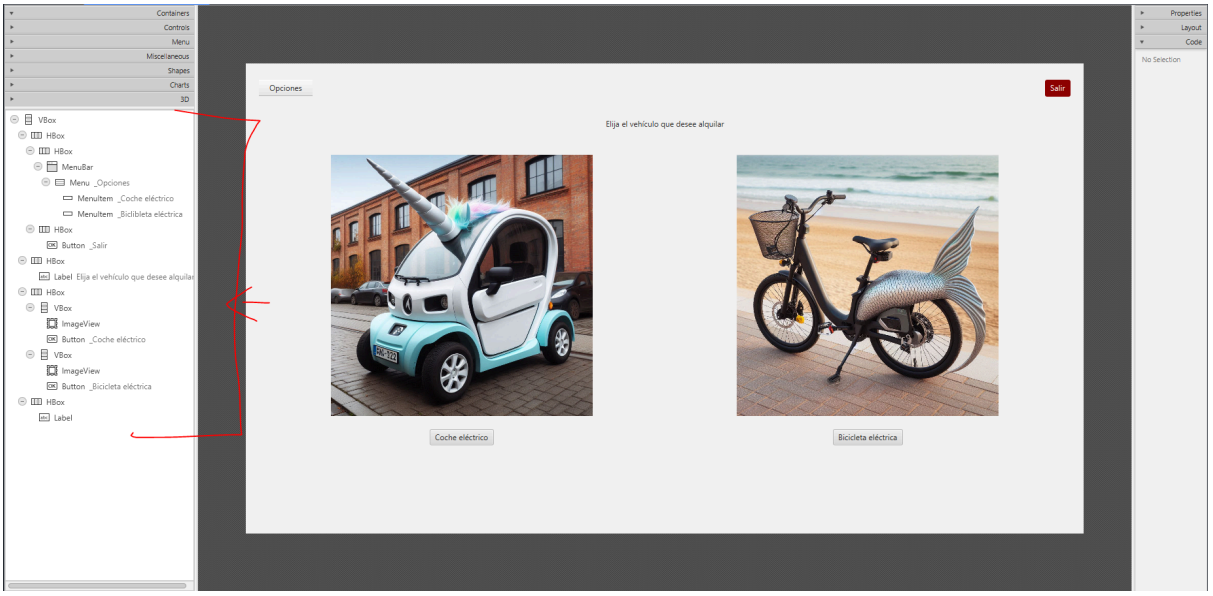
(Esto se repite en todos los inputs necesarios)

```
<TextField fx:id="textFieldNombre">
   <tooltip>
      <Tooltip fx:id="toolTipNombre" text="Nombre del cliente" />
   </tooltip>
   <VBox.margin>
      <Insets right="20.0" />
   </VBox.margin>
</TextField>
```

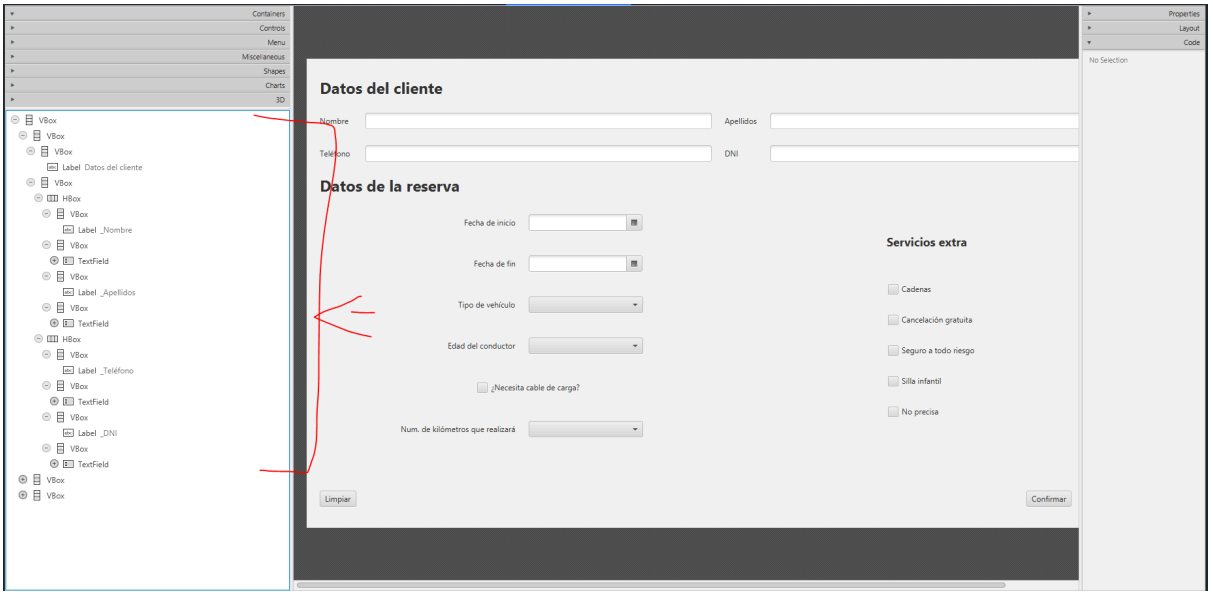
Para añadirlo, se ha hecho todo desde el xml de la forma que se muestra en la captura, agregando directamente el tooltip con su propio id(para ponerle la fuente) y su texto. Esto se ha repetido en todos los inputs y otros elementos necesarios.

3- Distribuir los elementos adecuadamente usando el modo de distribución libre.

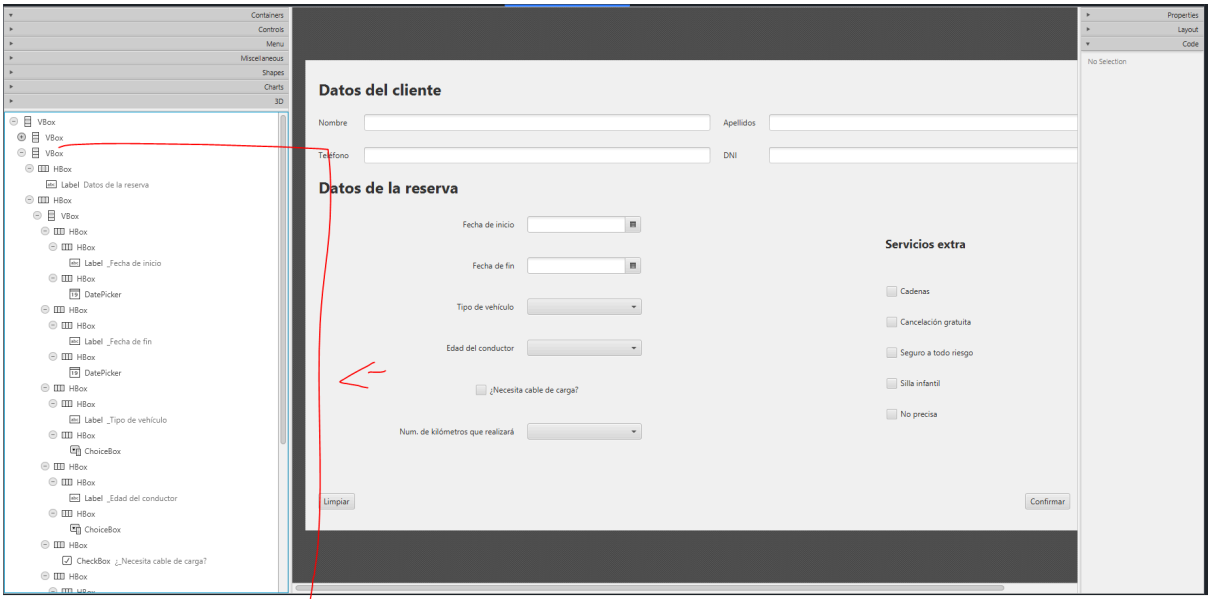
En este caso, como ya se ha explicado anteriormente, se han dispuesto armónicamente los elementos usando HBOX y VBOX entrelazados para colocar y alinear correctamente cada elemento. Se adjuntan fotos de la estructura.

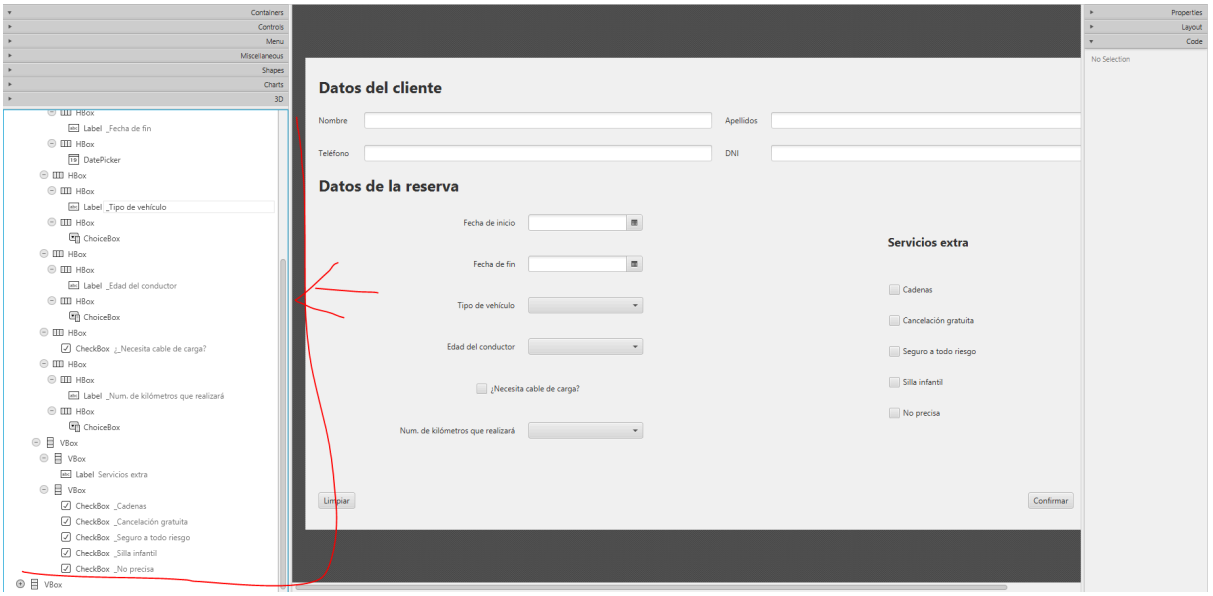


Menú principal.

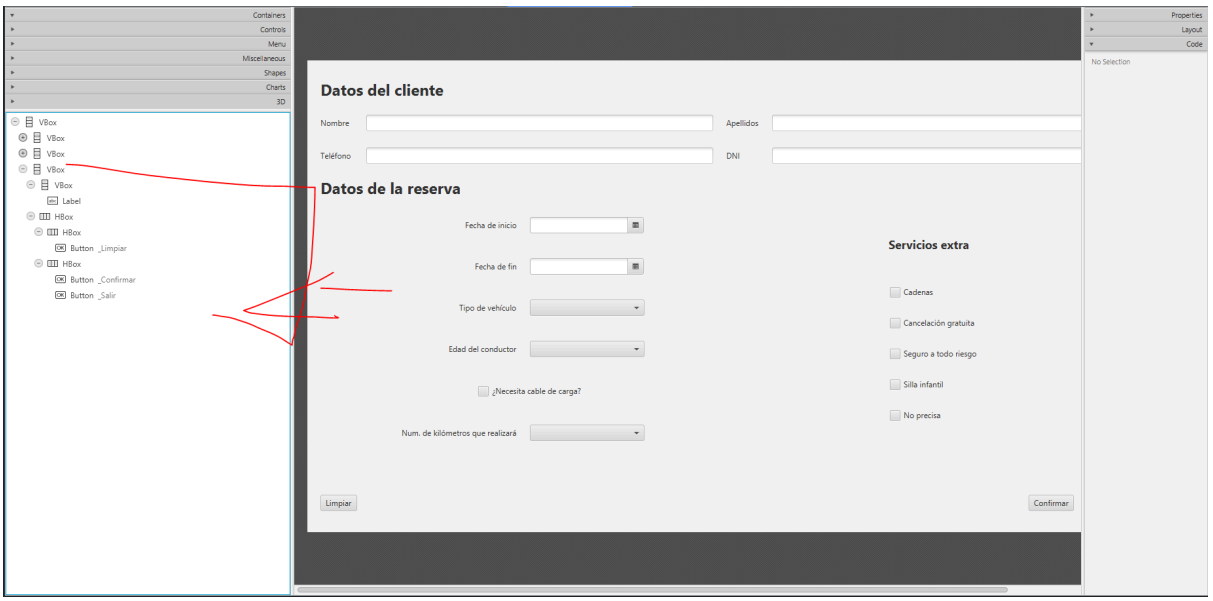


Coche eléctrico - Datos del cliente





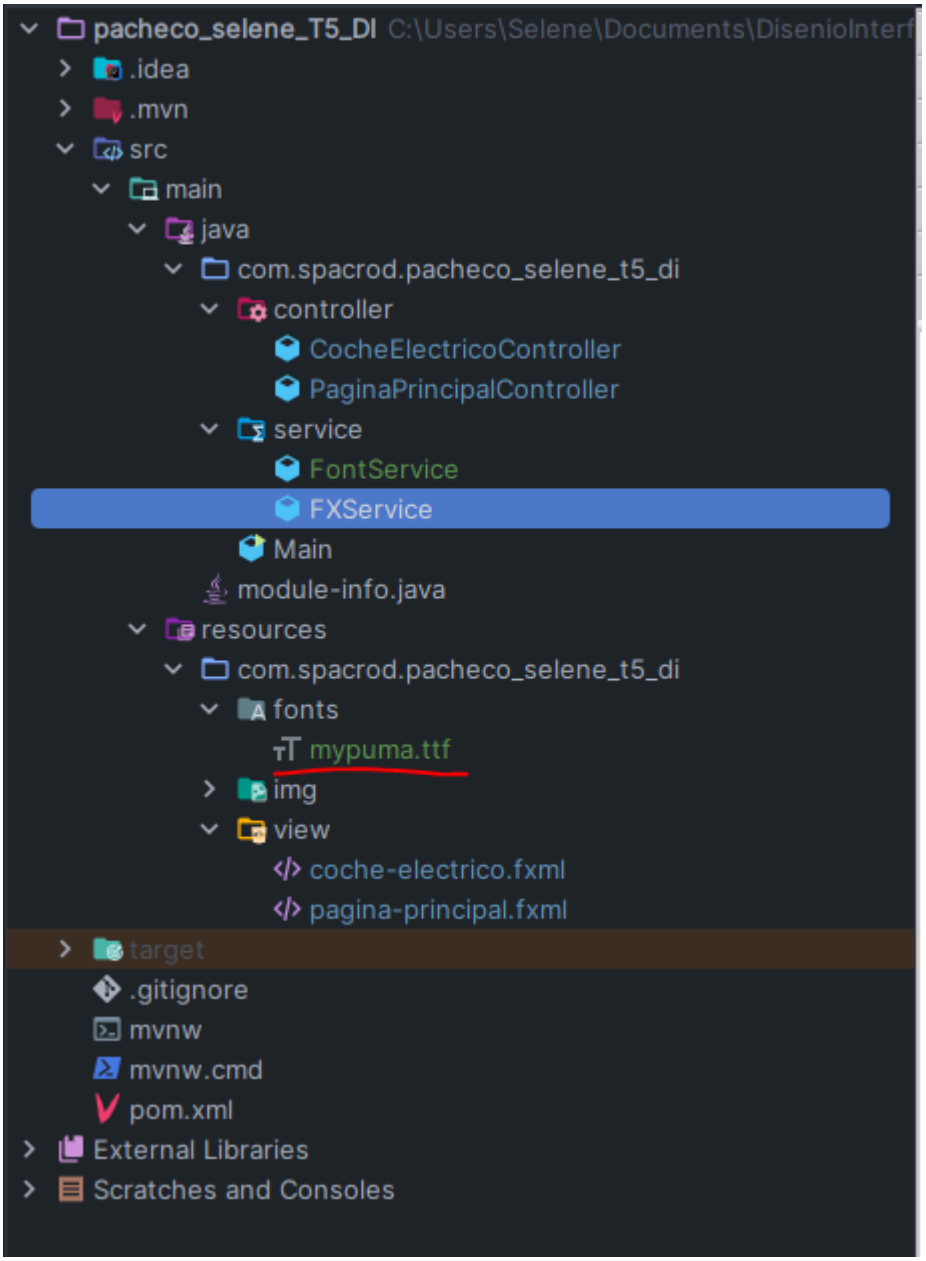
Coche eléctrico - Datos de la reserva Partes I y II



Coche eléctrico - Botones finales

4- Modificar la fuente de las etiquetas para dar vistosidad a la interfaz.

Para cambiar la fuente debemos que seguir los siguientes pasos:



Para mi proyecto he elegido la fuente mypuma que es como la marca Puma. Cogemos el fichero .ttf y lo pegamos en una carpeta fonts en nuestra carpeta de recursos.

```java
public class FontService {  15 usages  new *
    public static final Font CUSTOM_PUMA_FONT_NORMAL = Font.loadFont(FontService.class.getResourceAsStream( name: "/com/spacrod/pacheco_selene_t5_di/fonts/mypuma.ttf"),  v: 14);  1 usage
    public static final Font CUSTOM_PUMA_FONT_MEDIUM = Font.loadFont(FontService.class.getResourceAsStream( name: "/com/spacrod/pacheco_selene_t5_di/fonts/mypuma.ttf"),  v: 18);  2 usages
    public static final Font CUSTOM_PUMA_FONT_BIG = Font.loadFont(FontService.class.getResourceAsStream( name: "/com/spacrod/pacheco_selene_t5_di/fonts/mypuma.ttf"),  v: 24);  2 usages

    public static void applyFont(Font font, Object... nodes) {  5 usages  new *
        for(Object node : nodes) {
            if (node == null || font == null) {
                throw new IllegalArgumentException("El nodo o la fuente no pueden ser nulos.");
            }
            //verificamos el tipo del nodo y aplica la fuente correspondiente
            switch (node) {
                case Labeled labeled ->  //Labels, Buttons, CheckBox, RadioButton, etc.
                        labeled.setFont(font);
                case TextInputControl textInputControl ->  //TextField, TextArea, etc.
                        textInputControl.setFont(font);
                case ChoiceBox<?> choiceBox ->  //ChoiceBox
                        choiceBox.setStyle("-fx-font-family: '" + font.getFamily() + "'; -fx-font-size: " + font.getSize() + "px;");
                case ComboBox<?> comboBox ->  //ComboBox
                        comboBox.setStyle("-fx-font-family: '" + font.getFamily() + "'; -fx-font-size: " + font.getSize() + "px;");
                case Tooltip tooltip ->  //Tooltip
                        tooltip.setFont(font);
                case DatePicker datePicker ->  //DatePicker
                        datePicker.setStyle("-fx-font-family: '" + font.getFamily() + "'; -fx-font-size: " + font.getSize() + "px;");
                case MenuBar menuBar ->
                        menuBar.setStyle("-fx-font-family: '" + font.getFamily() + "'; -fx-font-size: " + font.getSize() + "px;");
                default ->
                        throw new IllegalArgumentException("El tipo de nodo no es compatible: " + node.getClass().getSimpleName());
            }
        }
    }
```

Luego creamos una clase FontService que englobe los métodos y las variables que necesitemos como aparece en la imagen.

```java
@FXML  new *
public void initialize() {
    FontService.applyFont(//fuente grande
            FontService.CUSTOM_PUMA_FONT_BIG,
            menuBarOpciones,
            buttonSalir,
            buttonCocheElectrico,
            buttonBicicletaElectrica,
            labelError
    );
    FontService.applyFont(
            FontService.CUSTOM_PUMA_FONT_MEDIUM,
            labelElegir
    );
}
```

Finalmente, aplicamos la fuente en los elementos que lo necesiten con el método definido en FontService. Aquí vemos los elementos del menú principal.

```
//aplicamos las fuentes
FontService.applyFont(//fuente normal
        FontService.CUSTOM_PUMA_FONT_NORMAL,
        labelNombre,
        textFieldNombre,
        labelApellidos,
        textFieldApellidos,
        labelTelefono,
        textFieldTelefono,
        labelDNI,
        textFieldDNI,
        labelFechaInicio,
        datePickerFechaInicio,
        labelFechaFin,
        datePickerFechaFin,
        labelTipoVehiculo,
        choiceBoxTipoVehiculo,
        labelEdadConductor,
        choiceBoxEdadConductor,
        checkBoxCableCarga,
        labelKM,
        choiceBoxKM,
        checkBoxCadenas,
        checkBoxCancelacionGratuita,
        checkBoxSeguroTodoRiesgo,
        checkBoxSillaInfantil,
        checkBoxNoPrecisa,
        toolTipNombre,
        toolTipApellidos,
```

```
        checkBoxNoPrecisa,
        toolTipNombre,
        toolTipApellidos,
        toolTipTelefono,
        toolTipDNI
);
FontService.applyFont(//fuente mediana
        FontService.CUSTOM_PUMA_FONT_MEDIUM,
        labelServiciosExtra,
        labelError
);
FontService.applyFont(//fuente grande
        FontService.CUSTOM_PUMA_FONT_BIG,
        labelDatosCliente,
        labelDatosReserva,
        buttonConfirmar,
        buttonLimpiar,
        buttonSalir
);
```

En la interfaz del Coche eléctrico.

5- Añadir el código para poder abrir el diálogo desde la ventana principal usando el menú y usando un botón a que se gestione el evento.

```
public class FXService {  18 usages  Selene
    public static final String VENTANA_MAIN;  3 usages
    public static final String COCHE_ELECTRICO;  2 usages
    static {
        VENTANA_MAIN = "pagina-principal.fxml";
        COCHE_ELECTRICO = "coche-electrico.fxml";
    }
    public static void cambiarVentana(String ventana){  3 usages  Selene
        try{
            FXMLLoader fxmlLoader = new FXMLLoader(Main.class.getResource( name: "view/"+ventana));
            Scene scene = new Scene(fxmlLoader.load(),  v: 1280,  v1: 720);
            Main.stage.setTitle("CarroCómico - Alquiler de Vehículos");
            Main.stage.getIcons().add(new Image(Objects.requireNonNull(Main.class.getResourceAsStream( name: "/com/spacrod/pacheco_selene_t5_di/img/icono.png"))));
            Main.stage.setScene(scene);
            Main.stage.show();
        }catch(IOException e){
            System.out.println("No se puede cambiar la ventana: "+e.getMessage());
            System.exit( status: 1);
        }
    }
    public static void configureLabelWithMnemonic(Label label, Control control) {label.setLabelFor(control);}  9 usages  Selene
}
```

Para cambiar de ventana usamos el método cambiarVentana() definido en FXService como se muestra en la captura.

```
@FXML  Selene
protected void onClickMenuItemCocheElectrico() {
    FXService.cambiarVentana(FXService.COCHE_ELECTRICO);
    limpiarLabelError();
}
```

Luego, en el controlador del menú principal llamamos al método para que cambie a la pantalla del coche eléctrico.

```
@FXML  Selene
protected void onClickButtonSalir() { FXService.cambiarVentana(FXService.VENTANA_MAIN); }
```

Finalmente, el la pantalla del coche eléctrico, en el botón salir, definimos que vuelva el menú principal.

6- Añadir las combinaciones de teclas para que el cursor se coloque sobre el campo de texto correspondiente a la combinación (Alt + tecla).

Para aplicar las combinaciones de teclas lo hacemos de la siguiente forma:

```
<MenuBar fx:id="menuBarOpciones">
  <menus>
    <Menu mnemonicParsing="true" text="_Opciones">
      <items>
        <MenuItem mnemonicParsing="true" onAction="#onClickMenuItemCocheElectrico" text="_Coche eléctrico" />
            <MenuItem mnemonicParsing="true" onAction="#onClickMenuItemBicicletaElectrica" text="_Biclibleta eléctrica" />
      </items>
    </Menu>
  </menus>
</MenuBar>
```

(En todos los elementos que no están vinculados con labels se ha aplicado la siguiente forma de actuar, en el xml directamente. Esto engloba checkboxs, botones, menús, etc.)

```
<VBox alignment="CENTER_LEFT" prefHeight="50.0" prefWidth="100.0">
    <children>
        <Label fx:id="labelTelefono" mnemonicParsing="true" text="_Teléfono" />
    </children>
</VBox>
<VBox alignment="CENTER_LEFT" prefHeight="50.0" prefWidth="580.0">
    <children>
        <TextField fx:id="textFieldTelefono">
            <tooltip>
                <Tooltip fx:id="toolTipTelefono" text="Teléfono del cliente" />
            </tooltip>
            <VBox.margin>
                <Insets right="20.0" />
            </VBox.margin>
        </TextField>
```

(En los inputs vinculados a labels, se activa el mnemonicParsing y se asigna correctamente el nombre con la _)

```
public static void configureLabelWithMnemonic(Label label, Control control) {label.setLabelFor(control);}  9 usages  Selene
```

```
//ponemos los tooltips en los campos
FXService.configureLabelWithMnemonic(labelNombre, textFieldNombre);
FXService.configureLabelWithMnemonic(labelApellidos, textFieldApellidos);
FXService.configureLabelWithMnemonic(labelTelefono, textFieldTelefono);
FXService.configureLabelWithMnemonic(labelDNI, textFieldDNI);
FXService.configureLabelWithMnemonic(labelFechaInicio, datePickerFechaInicio);
FXService.configureLabelWithMnemonic(labelFechaFin, datePickerFechaFin);
FXService.configureLabelWithMnemonic(labelTipoVehiculo, choiceBoxTipoVehiculo);
FXService.configureLabelWithMnemonic(labelEdadConductor, choiceBoxEdadConductor);
FXService.configureLabelWithMnemonic(labelKM, choiceBoxKM);
```

(Y terminamos vinculándolo todo con código)

Conclusiones

A lo largo de este trabajo se ha logrado crear una interfaz de usuario adecuada para el contexto del alquiler de vehículos CarroCómico. Se ha empleado una distribución intuitiva de los elementos, asegurando que todos los componentes sean fácilmente accesibles y visualmente atractivos para los usuarios. Se han incorporado detalles como la personalización de los nombres y ToolTips, la modificación de fuentes para mejorar la estética, y la integración de funcionalidades clave como la apertura de diálogos desde la ventana principal mediante botones y menús. Además, se ha incluido la gestión de eventos, como el control dinámico de las opciones de alquiler según el tipo de vehículo seleccionado y la posibilidad de asociar combinaciones de teclas para mejorar la accesibilidad. Este enfoque no solo ha permitido cumplir con los requisitos técnicos del proyecto, sino también ofrecer una interfaz que refleje el compromiso de CarroCómico con la movilidad sostenible y proporcione una experiencia de usuario agradable, fomentando el uso de vehículos eléctricos y el cuidado del medio ambiente.