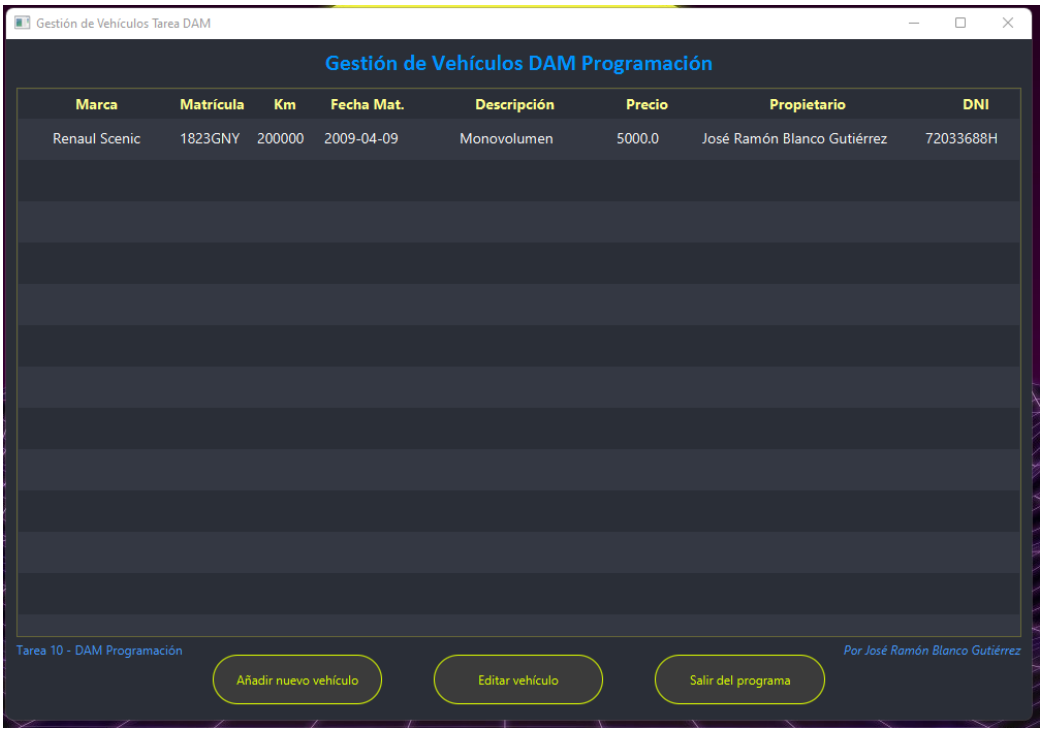
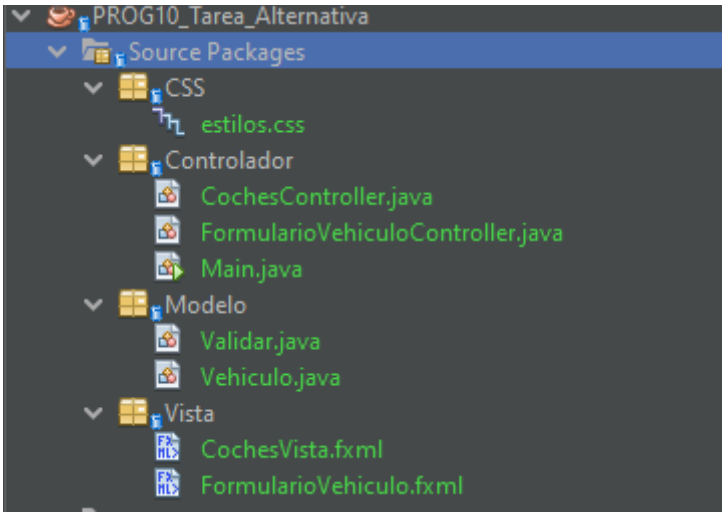
Informe de la PROG10 Tarea Alternativa

Siguiendo con los ejercicios que hemos venido desarrollando a lo largo del curso, sería interesante crear una interfaz que permita gestionar de forma mínima los vehículos de la tarea 5, por ejemplo para listar vehículos y poder añadir nuevos. Tienes libertad para pensar cómo se podría implementar esa interfaz. Una idea podría ser utilizar una tabla para mostrar los vehículos y un formulario para añadir nuevos vehículos. Sobre las filas de la tabla, podríamos mostrar un menú para eliminar los vehículos.



El programa esta estructurado siguiendo el patrón Modelo- Vista-Controlador

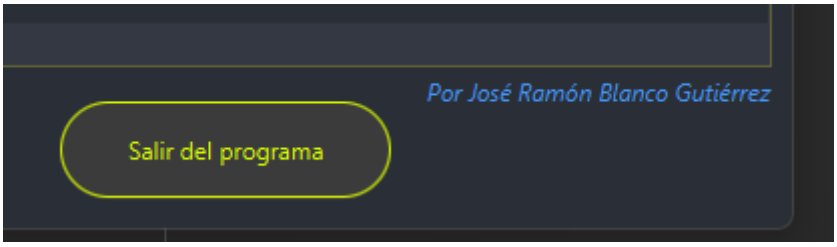


El inicio de la aplicación donde lee el fichero de fxml y lo carga e inicia el programa.

```
 */
public class Main extends Application{

    @Override
    public void start(Stage stage) throws Exception {
        Parent root = FXMLLoader.load(getClass().getResource("/Vista/CochesVista.fxml"));

        Scene scene = new Scene(root);
        stage.setScene(scene);
        stage.setTitle("Gestión de Vehículos Tarea DAM");
        stage.show();
    }

    public static void main(String[] args) {
        System.out.println("Iniciando Gestión de vehiculos en JavaFX...");
        launch(args);
    }

}
```

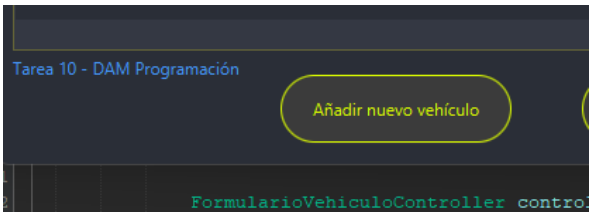
En el inicio de la ventana de la tabla lo primero que hace el programa es iniciar la tabla y vincular las columnas con el atributo de modelo Vehiculo.

```
@Override
public void initialize(URL url, ResourceBundle rb) {
    vehiculos = FXCollections.observableArrayList();    //Inicializa el ObservableList con un ArrayList de JavaFX
    this.tablaVehiculos.setItems(vehiculos);            //Vincula a la tabla el Arraylist

    //-- Asocia cada atributo de Vehiculo con las columnas de la tabla
    this.columMarca.setCellValueFactory(new PropertyValueFactory("marca"));
    this.columMatricula.setCellValueFactory(new PropertyValueFactory("matricula"));
    this.columKm.setCellValueFactory(new PropertyValueFactory("kilometros"));
    this.columFecha.setCellValueFactory(new PropertyValueFactory("fechaMatriculacion"));
    this.columDescrip.setCellValueFactory(new PropertyValueFactory("descripcion"));
    this.columPrecio.setCellValueFactory(new PropertyValueFactory("precio"));
    this.columPropietario.setCellValueFactory(new PropertyValueFactory("nombrePropietaro"));
    this.columDni.setCellValueFactory(new PropertyValueFactory("dniPropietario"));
}
```



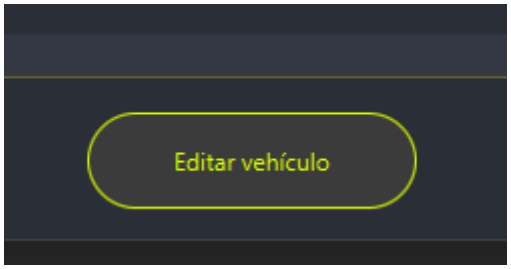
Para Salir del programa crea un stage basándose en un objeto de la ventana y cierra

```
 * Método para cerrar el programa
 * @param event
 */
@FXML
private void clickSalir(ActionEvent event) {
    Stage stage = (Stage) this.tablaVehiculos.getScene().getWindow();
    stage.close(); //Cerrar
}
```



El método Nuevo Vehículo llama a otro formulario para introducir los datos y después de que el usuario introduce los datos devuelve el objeto vehículo que es el que añade a la tabla.

```java
@FXML
private void clickNuevoVehiculo(ActionEvent event) {
    try {
        FXMLLoader loader = new FXMLLoader(getClass().getResource("/Vista/FormularioVehiculo.fxml"));

        Parent root = loader.load();

        FormularioVehiculoController controlador = loader.getController();
        controlador.iniciarAtributos(vehiculos);

        Scene scene = new Scene(root);
        Stage stage = new Stage();
        stage.initModality(Modality.APPLICATION_MODAL);
        stage.setTitle("Nuevo vehiculo");
        stage.setResizable(false);
        stage.initStyle(StageStyle.UTILITY);
        stage.setScene(scene);
        stage.showAndWait();

        Vehiculo vehiculoTemp = controlador.getVehiculo();

        if (vehiculoTemp != null) {
            this.vehiculos.add(vehiculoTemp);
            this.tablaVehiculos.refresh();
        }

    } catch (IOException ex) {
        Alert alert = new Alert(Alert.AlertType.ERROR);
        alert.setHeaderText(null);
        alert.setTitle("Error" );
        alert.setContentText(ex.getMessage());
        alert.showAndWait();
    }
}
```



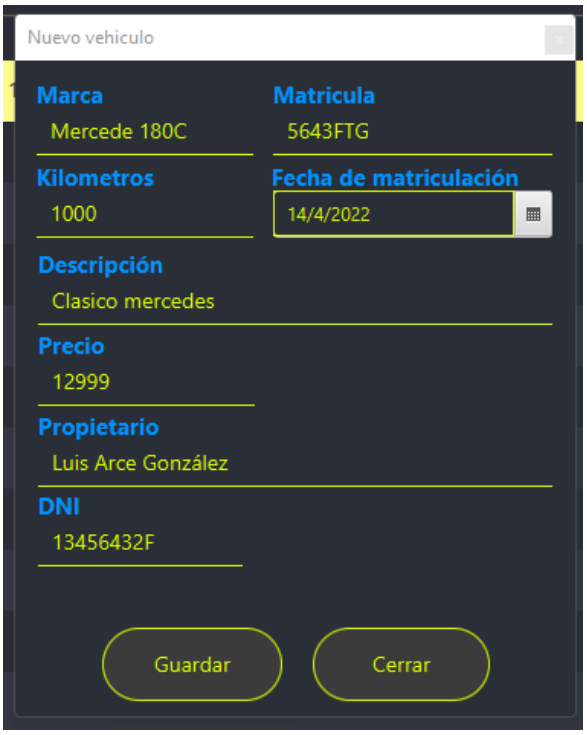
Método Editar es muy parecido al anterior, salvo que comprueba si esta seleccionando en la lista algún elemento, si es así lo enviá al nuevo formulario para poner editarlo.

Un vez editado actualiza el dato, la curiosidad de editar esta en que en todo momento estamos referenciando con “punteros” el dato el arraylist de la tabla por lo tanto con guardar en esa referencia los datos, actualizamos la tabla.

```java
private void clickEditarVehiculo(ActionEvent event) {

    Vehiculo vehiculotemporal = this.tablaVehiculos.getSelectionModel().getSelectedItem();

    if (vehiculotemporal == null) {
        Alert alert = new Alert(Alert.AlertType.INFORMATION);
        alert.setHeaderText(null);
        alert.setTitle("Información" );
        alert.setContentText("Se tiene que seleccionar un vehiculo para poderle borrar");
        alert.showAndWait();
    }else {
        try {
            FXMLLoader loader = new FXMLLoader(getClass().getResource("/Vista/FormularioVehiculo.fxml"));

            Parent root = loader.load();

            FormularioVehiculoController controlador = loader.getController();
            controlador.iniciarAtributos(vehiculos,vehiculotemporal);

            controlador.getTxtMatricula().setDisable(true);

            Scene scene = new Scene(root);
            Stage stage = new Stage();
            stage.initModality(Modality.APPLICATION_MODAL);
            stage.setTitle("Editar Vehiculo");
            stage.setResizable(false);
            stage.initStyle(StageStyle.UTILITY);
            stage.setScene(scene);
            stage.showAndWait();

            vehiculotemporal = controlador.getVehiculo();

            if (vehiculotemporal != null) {
                this.tablaVehiculos.refresh();

            }

        } catch (IOException ex) {
            Alert alert = new Alert(Alert.AlertType.ERROR);
            alert.setHeaderText(null);
            alert.setTitle("Error" );
            alert.setContentText(ex.getMessage());
            alert.showAndWait();
        }
    }
}
```

Para Borrar pulsado con el botón derecho lo que comprueba es que tengas una fila seleccionada y la borra.

```java
@FXML
private void clickBorrarVehiculo(ActionEvent event) {
    //Crea un objeto temporal con la fila seleccionada
    Vehiculo vehiculotemporal = this.tablaVehiculos.getSelectionModel().getSelectedItem();

    if (vehiculotemporal == null) {
        Alert alert = new Alert(Alert.AlertType.INFORMATION);
        alert.setHeaderText(null);
        alert.setTitle("Información" );
        alert.setContentText("Se tiene que seleccionar un vehiculo para poderle borrar");
        alert.showAndWait();
    } else {
        this.vehiculos.remove(vehiculotemporal); //borra el elemento selecionado
        this.tablaVehiculos.refresh(); //Actualiza la tabla

        Alert alert = new Alert(Alert.AlertType.INFORMATION);
        alert.setHeaderText(null);
        alert.setTitle("Información" );
        alert.setContentText("Se ha borrado el vehiculo " + vehiculotemporal.getMatricula());
        alert.showAndWait();
    }
}
```



Para iniciar el formulario tenemos dos métodos sobrecargados que ambos pasan al formulario el ArrayList para que se tenga la lista de vehiculos y el segundo iniciar tambien se le pasa el opbjeto que se quiere editar.S

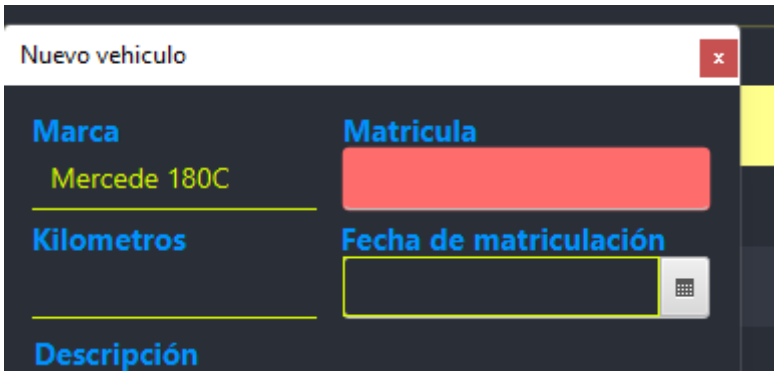
```java
/**
 * Método para pasar al formulario de nuevo el arraylist
 * @param v
 */
public void iniciarAtributos(ObservableList<Vehiculo> v) {
    this.vehiculos = v;
}
/**
 * Método para pasar al formulario en modo editar el arralist y el objeto seleccionado
 * para modificar.
 * @param v
 */
public void iniciarAtributos(ObservableList<Vehiculo> vList, Vehiculo veh ) {
    //Iniciamos los objetos
    this.vehiculos = vList;
    this.vehiculo = veh;
    //Llena los textfield con los datos pasados
    this.txtMarca.setText(veh.getMarca());
    this.txtMatricula.setText(veh.getMatricula());
    this.txtKm.setText(veh.getKilometros()+"");
    this.dpFecha.setValue(veh.getFechaMatriculacion());
    this.txtDescripcion.setText(veh.getDescripcion());
    this.txtPrecio.setText(veh.getPrecio()+"");
    this.txtPropietario.setText(veh.getNombrePropietaro());
    this.txtDni.setText(veh.getDniPropietario());
}
```

Cuando el usuario pulsa guardar llama al método Guardar y este pasa los datos de los textfield a un objeto vehículo que sera el objeto que se pase a la tabla.

```
int kilometros = 0;
LocalDate fechaMatriculacion = null;
String descripcion = null;
float precio = 0;
String nombrePropietaro = null;
String dniPropietario = null;

//Pasa los datos de los textfiels a las variables para crear un objeto temporal
try {
    marca = this.txtMarca.getText();
    matricula = this.txtMatricula.getText();
    kilometros = Integer.parseInt(this.txtKm.getText());
    fechaMatriculacion = this.dpFecha.getValue();
    descripcion = this.txtDescripcion.getText();
    precio = Float.parseFloat(this.txtPrecio.getText());
    nombrePropietaro = this.txtPropietario.getText();
    dniPropietario = this.txtDni.getText();

    vehiculoTemporal = new Vehiculo(marca, matricula, kilometros,
                                    fechaMatriculacion, descripcion,
                                    precio, nombrePropietaro, dniPropietario);
} catch (NumberFormatException ex) {
    Alert alert = new Alert(Alert.AlertType.ERROR);
    alert.setHeaderText(null);
    alert.setTitle("ERROR");
    alert.setContentText("Error en los datos, posiblemente tengas datos en blanco o números incorrectos");
    alert.showAndWait();
    return;
}


if (!this.vehiculos.contains(vehiculoTemporal)) {   //Comprueba si existe en el arraylist el nuevo vehiculo
    if (this.vehiculo != null) {    //Estamos modificando
        this.vehiculo.setMarca(marca);
        this.vehiculo.setMatricula(matricula);
        this.vehiculo.setKilometros(kilometros);
        this.vehiculo.setFechaMatriculacion(fechaMatriculacion);
        this.vehiculo.setDescripcion(descripcion);
        this.vehiculo.setPrecio(precio);
        this.vehiculo.setNombrePropietaro(nombrePropietaro);
        this.vehiculo.setDniPropietario(dniPropietario);
    }else {                         //Es nuevo
        this.vehiculo = vehiculoTemporal;
        Alert alert = new Alert(Alert.AlertType.INFORMATION);
        alert.setHeaderText(null);
        alert.setTitle("Info");
        alert.setContentText("Se ha añadido un nuevo vehiculo");
        alert.showAndWait();
```

Un dato es que el campo Matricula verifica si es correcto el formato

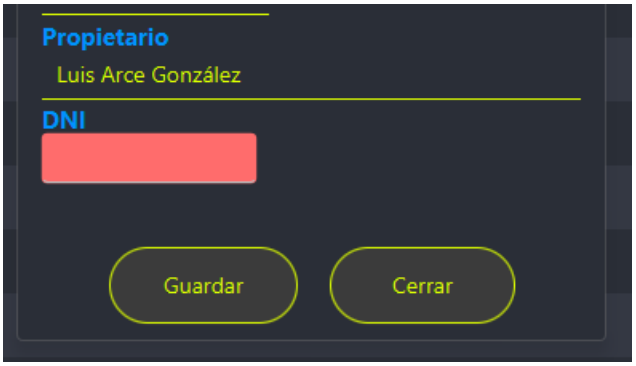
El método que se lanza cuando el usuario escribe en la matricula y cambia el estilo con CSS.




```
 * Método que valida eñ formato de una matricula 0000LLL
 *
 * @param matricula Matricula que se desea chequear el formato
 * @return
 */
public static boolean checkMatricula(String matricula) {
    return matricula.matches("^[0-9]{4}[A-Z]{3}$");
}

/**
```

```
@FXML
private void verificarMatricula(KeyEvent event) {
    TextField txt = (TextField) event.getSource();

    if (!Validar.checkMatricula(txt.getText())) {
        txt.setStyle("-fx-background-color: #ff6c6c;"
                   + "-fx-border-color: #bfbfbf;"
                   + "-fx-border-radius: 4px;");
    } else {
        txt.setStyle("-fx-background-color: #2A2E37;"
                   +"-fx-text-fill: #ddff00;"
                   +"-fx-border-width: 0 0 1px 0;"
                   +"-fx-border-color: #ddff00;");
    }
}
```

Para el DNI el programa hace exactamente lo mismo que con matricula verifica si tiene el formato correcto y cambia el color con CSS




```
     * @param dni DNI que hay que chequear el formato
     * @return
     */
    public static boolean checkDNI(String dni) {
        return dni.matches("^[0-9]{8}[TRWAGMYFPDXBNJZSQVHLCKE]{1}$");
    }
```

```
@FXML
private void verificarDNI(KeyEvent event) {
    TextField txt = (TextField) event.getSource();

    if (!Validar.checkDNI(txt.getText())) {
        txt.setStyle("-fx-background-color: #ff6c6c;"
                   + "-fx-border-color: #bfbfbf;"
                   + "-fx-border-radius: 4px;");
    } else {
        txt.setStyle("-fx-background-color: #2A2E37;"
                   +"-fx-text-fill: #ddff00;"
                   +"-fx-border-width: 0 0 1px 0;"
                   +"-fx-border-color: #ddff00;");
    }
}
```

Método que cierra la aplicación al pulsar en cerrar.

```
/**
 * Método que se llama para cerrar la ventana de dialogo.
 * @param event
 */
@FXML
private void ClickBtnCerrar(ActionEvent event) {
    this.vehiculo = null;
    Stage stage = (Stage) this.btnCerrar.getScene().getWindow();
    stage.close();
}
```

NOTA: Para los estilos de la tabla me he basado en este codigo CSS open source:
https://github.com/afsalashyana/Library-Assistant.git