Universidad de San Carlos de Guatemala

Manual Técnico

“Practica 2”

Introducción a la Programación y Computación 1

Desarrollado por:
Daniel Hernandez

Descripción general:

El proyecto es una aplicación de gestión de viajes que permite a los usuarios planificar, registrar y visualizar información sobre los viajes realizados. La aplicación consta de varias clases que desempeñan roles específicos en su funcionamiento. Aquí está una descripción general de las principales funcionalidades y componentes del proyecto:

1. Clase GenerateTripController:

- Esta clase controla la generación de viajes, permitiendo a los usuarios seleccionar las rutas y vehículos para planificar un viaje.
- Gestiona la interfaz de usuario relacionada con la generación de viajes, incluyendo la selección de rutas y vehículos.

```
1 usage   Daniel Hernandez
public void onGenerateAnewTripPressed() {
    pressCount++;
    if (pressCount <= 3) {
        try {
            FXMLLoader loader = new FXMLLoader(getClass().getResource( name: "Trip.fxml"));
            Parent root;

            // Si es la primera vez que se presiona el botón, cargar Trip.fxml y crear una nueva pestaña
            if (pressCount == 1) {
                root = loader.load();
                tripController = loader.getController();

                // Crear una nueva pestaña y establecer su contenido
                Tab newTripTab = new Tab( s: "Nuevo Viaje");
                newTripTab.setContent(root);

                // Agregar la nueva pestaña al TabPane
                mainWindowsController.getTabPane().getTabs().add(newTripTab);
            }
            // Si no es la primera vez que se presiona el botón, obtener el controlador de la pestaña existente
            else {
                Tab existingTab = mainWindowsController.getTabPane().getTabs().get(0); // Asume que la pestaña de viaje es la primera pestaña
                root = (Parent) existingTab.getContent();
            }

            String selectedPlace1 = Places1.getValue();
            String selectedPlace2 = Places2.getValue();
            String selectedVehicle = SelectedVehicles.getValue();
            if (selectedVehicle != null) {
                SelectedVehicles.getItems().remove(selectedVehicle);
            }
            String selectedRouteDistance = getSelectedRouteDistance(selectedPlace1, selectedPlace2);

            Image vehicleImage = vehicleService.getVehicleImage(selectedVehicle);

            // Llenar los campos correspondientes en TripController dependiendo del valor de pressCount
            switch (pressCount) {
                case 1:
                    tripController.setBeg1(selectedPlace1);
                    tripController.setFinal1(selectedPlace2);
                    tripController.setKM1(selectedRouteDistance);
                    tripController.setNameV1(selectedVehicle);
```

```
                case 2:
                    tripController.setBeg2(selectedPlace1);
                    tripController.setFinal2(selectedPlace2);
                    tripController.setKM2(selectedRouteDistance);
                    tripController.setNameV2(selectedVehicle);
                    tripController.setImage2(vehicleImage);
                    break;
                case 3:
                    tripController.setBeg3(selectedPlace1);
                    tripController.setFinal3(selectedPlace2);
                    tripController.setKM3(selectedRouteDistance);
                    tripController.setNameV3(selectedVehicle);
                    tripController.setImage3(vehicleImage);
                    break;
            }
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
    else {
        GenerateAnewTrip.setDisable(true);
        try {
            // Cargar GenerateTripAlert.fxml
            FXMLLoader loader = new FXMLLoader(getClass().getResource( name: "/Panels/GenerateTripAlert.fxml"));
            Parent alertRoot = loader.load();

            Scene scene = new Scene(alertRoot);
            Stage stage = new Stage();
            stage.setScene(scene);
            stage.show();
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}
```

2. **Clase MainWindowsController:**

- Actúa como el controlador principal de la ventana principal de la aplicación.
- Coordina la interacción entre diferentes componentes de la interfaz de usuario, como los controladores de carga de rutas y generación de viajes.

```
@FXML
public void initialize() {
    // Establecer MainWindowsController para LoadRoutesController y GenerateTripController
    if (loadRoutesController != null) {
        loadRoutesController.setMainWindowsController(this);
    }

    if (generateTripController != null) {
        generateTripController.setMainWindowsController(this);
    }

    Platform.runLater(this::postInitialize);
}

1 usage  Daniel Hernandez
public void postInitialize() {
    if (generateTripController != null && tabPane != null && generateTripTab != null) {
        generateTripController.setTabPane(tabPane);
        generateTripController.setGenerateTripTab(generateTripTab);
        generateTripController.setGenerateTripTabContent(generateTripTab.getContent());
    }
}

2 usages  Daniel Hernandez
public LoadRoutesController getLoadRoutesController() { return loadRoutesController; }

no usages  Daniel Hernandez
public void setLoadRoutesController(LoadRoutesController loadRoutesController) {
    this.loadRoutesController = loadRoutesController;
}

no usages  Daniel Hernandez
public void setGenerateTripController(GenerateTripController generateTripController) {
    this.generateTripController = generateTripController;
}

1 usage  Daniel Hernandez
public GenerateTripController getGenerateTripController() { return generateTripController; }

2 usages  Daniel Hernandez
public TabPane getTabPane() { return tabPane; }

no usages  Daniel Hernandez
public Tab getGenerateTripTab() { return generateTripTab; }
```

3. **Clase LoadRoutesController:**
    - Controla la carga de rutas disponibles en la aplicación, permitiendo a los usuarios seleccionar una ruta para su viaje.
    - Gestiona la interfaz de usuario relacionada con la carga de rutas, como la visualización de la lista de rutas disponibles.
    -

```
    this.colDistance.setCellValueFactory(cellData -> cellData.getValue().distanceProperty());
}

Daniel Hernandez
@FXML
private void loadRoutes(ActionEvent event) throws IOException {
    // Comprueba si mainWindowsController es null y, si es así, crea una nueva instancia
    if (mainWindowsController == null) {
        mainWindowsController = new MainWindowsController();
    }

    FileChooser fileChooser = new FileChooser();
    fileChooser.getExtensionFilters().addAll(
            new FileChooser.ExtensionFilter( s: "CSV files",  ...strings: "*.csv"));
    File csvFile = fileChooser.showOpenDialog( window: null);

    if (csvFile != null) {
        Path path = csvFile.toPath();

        try (BufferedReader reader = Files.newBufferedReader(path);
             ObjectOutputStream out = new ObjectOutputStream(
                     new FileOutputStream( name: "routes.dat")))
        {
            String line;
            int id = 1;
            while ((line = reader.readLine()) != null) {
                String[] data = line.split( regex: ",");
                String startLocation = data[0]; // assuming this is start
                String endLocation = data[1]; // assuming this is end
                String distance = data[2]; // assuming this is distance

                Route route = new Route(Integer.toString(id++),
                        startLocation, endLocation, distance);

                out.writeObject(route);
                ShowInformation.getItems().add(route);
            }
            mainWindowsController.getGenerateTripController().fillComboBoxes(ShowInformation.getItems());
        }
    }
}
```

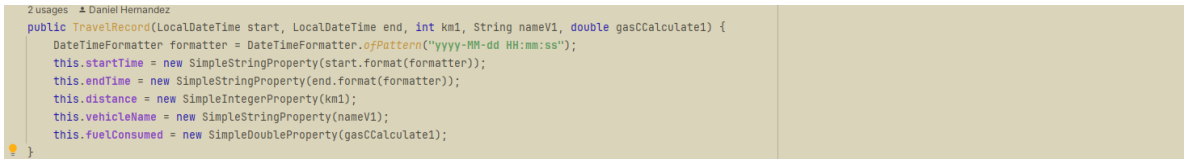
4. **Clase RecordController:**
    - Gestiona el registro de los detalles de los viajes realizados, incluyendo la persistencia de los datos de los viajes en archivos binarios.
    - Controla la carga y visualización de los registros de viajes anteriores en la aplicación.

```
    }

    1 usage   Daniel Hernandez
    public void loadRecords(String filename) {
        try {
            FileInputStream fileIn = new FileInputStream(filename);
            ObjectInputStream in = new ObjectInputStream(fileIn);
            List<TravelRecordData> recordDataList = (List<TravelRecordData>) in.readObject();

            travelRecords = FXCollections.observableArrayList();
            for (TravelRecordData recordData : recordDataList) {
                TravelRecord record = new TravelRecord(
                        LocalDateTime.parse(recordData.getStartTime(), DateTimeFormatter.ofPattern("yyyy-MM-dd HH:mm:ss")),
                        LocalDateTime.parse(recordData.getEndTime(), DateTimeFormatter.ofPattern("yyyy-MM-dd HH:mm:ss")),
                        recordData.getDistance(),
                        recordData.getVehicleName(),
                        recordData.getFuelConsumed()
                );
                record.setId(recordData.getId());
                record.setColor(recordData.getColor());
                travelRecords.add(record);
            }

            table.setItems(travelRecords);
            in.close();
            fileIn.close();
        } catch (IOException i) {
            i.printStackTrace();
        } catch (ClassNotFoundException c) {
            System.out.println("Class not found");
            c.printStackTrace();
        }
    }
```

5. **Clase Route:**
    - Representa una ruta en la aplicación, con propiedades como el inicio, el fin y la distancia de la ruta.

```
1 usage   Daniel Hernandez
public void setDistance(String distance) { this.distance.set(distance); }

1 usage   Daniel Hernandez
public Route(String id, String start, String end, String distance) {
    this.id = new SimpleStringProperty(id);
    this.start = new SimpleStringProperty(start);
    this.end = new SimpleStringProperty(end);
    this.distance = new SimpleStringProperty(distance);
}

no usages   Daniel Hernandez
private void writeObject(ObjectOutputStream s) throws IOException {
    s.defaultWriteObject();

    s.writeUTF(id.get());
    s.writeUTF(start.get());
    s.writeUTF(end.get());
    s.writeUTF(distance.get());
}

no usages   Daniel Hernandez
private void readObject(ObjectInputStream s) throws IOException, ClassNotFoundException {
    s.defaultReadObject();

    id = new SimpleStringProperty(s.readUTF());
    start = new SimpleStringProperty(s.readUTF());
    end = new SimpleStringProperty(s.readUTF());
    distance = new SimpleStringProperty(s.readUTF());
}
```

6. **Clase TravelRecord:**
    - Representa un registro de viaje, que contiene información sobre el inicio y fin del viaje, la distancia recorrida, el vehículo utilizado y el consumo de combustible.
    - Se utiliza para registrar los detalles de los viajes realizados por los usuarios.

- 

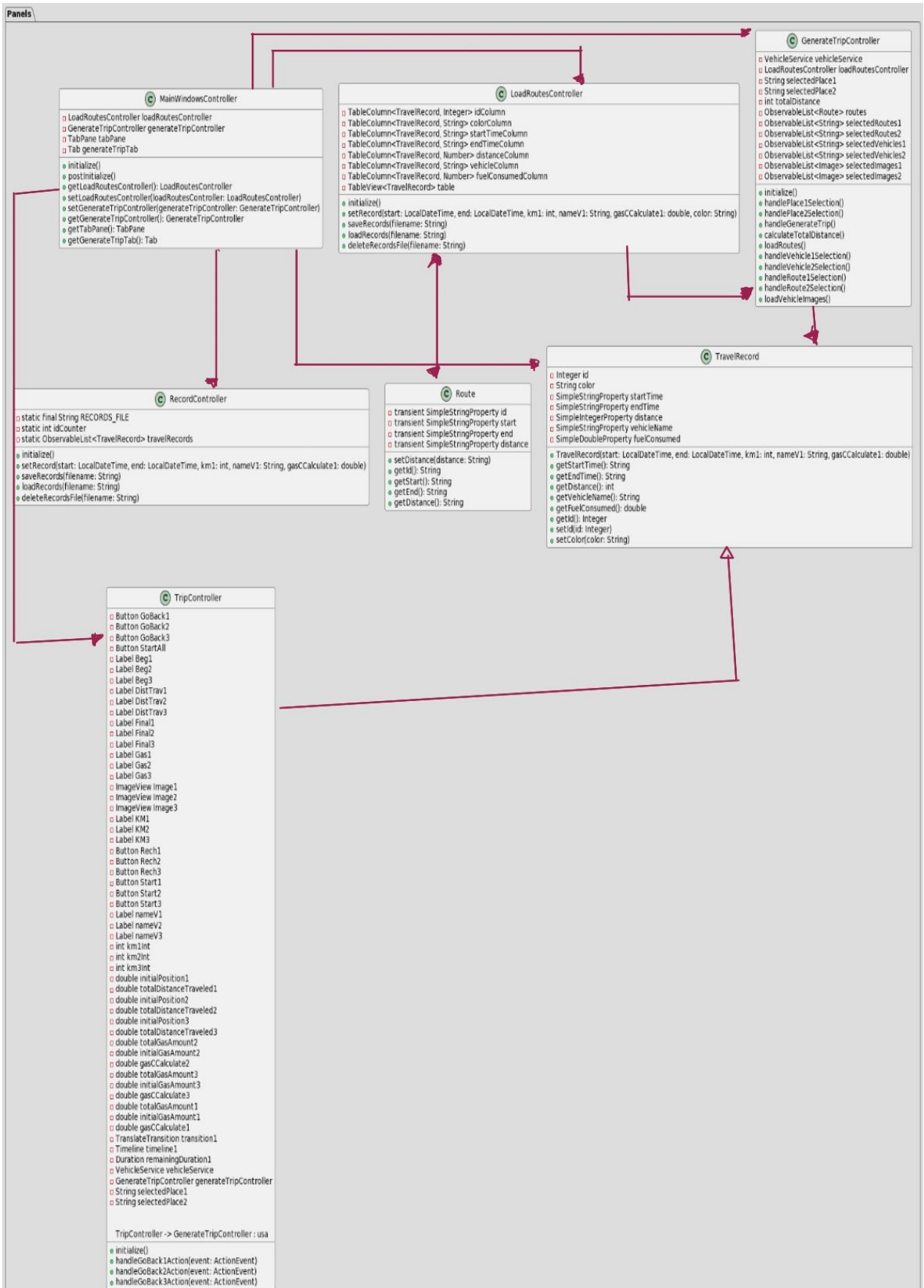
```
2 usages  Daniel Hernandez
public TravelRecord(LocalDateTime start, LocalDateTime end, int km1, String nameV1, double gasCCalculate1) {
    DateTimeFormatter formatter = DateTimeFormatter.ofPattern("yyyy-MM-dd HH:mm:ss");
    this.startTime = new SimpleStringProperty(start.format(formatter));
    this.endTime = new SimpleStringProperty(end.format(formatter));
    this.distance = new SimpleIntegerProperty(km1);
    this.vehicleName = new SimpleStringProperty(nameV1);
    this.fuelConsumed = new SimpleDoubleProperty(gasCCalculate1);
}
```

7. **Clase TripController:**
    - Controla la lógica relacionada con los viajes en la aplicación, incluyendo el inicio, el regreso y la representación visual de los mismos.
    - Gestiona la interacción del usuario con la simulación de los viajes, actualizando los valores de distancia recorrida y consumo de combustible en tiempo real.
    - 

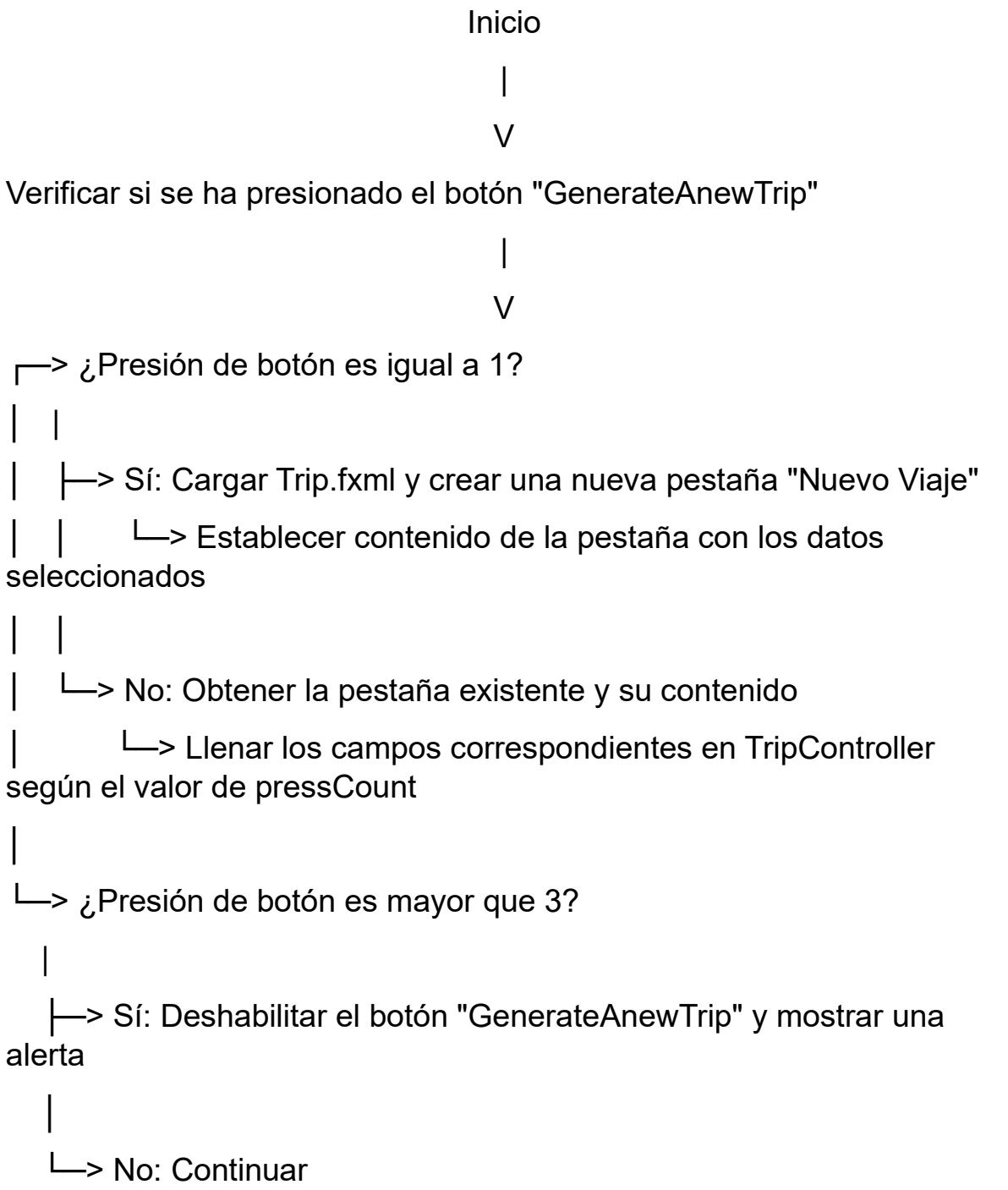
```
Daniel Hernandez
@Override
public void handle(ActionEvent event) {
    LocalDateTime startTime = LocalDateTime.now();
    // Reset initial position when start
    initialPosition1 = Image1.getTranslateX();

    transition1.setNode(Image1);
    remainingDuration1 = Duration.seconds(km1Int);
    transition1.setDuration(remainingDuration1); // Set the duration to be the remaining duration
    transition1.setToX(Image1.getTranslateX() - 286.0000458); // Desplace 286.0000458 px a la izquierda

    double kmPerPixel = km1Int / 286.0000458;

    timeline1.getKeyFrames().add(new KeyFrame(Duration.millis( v: 100), e -> {
        double localDistanceTraveled =
                Math.abs((Image1.getTranslateX() - initialPosition1) * kmPerPixel);
        DistTrav1.setText(String.format("%.2f", totalDistanceTraveled1 + localDistanceTraveled));

        String vehicleName = nameV1.getText();
        double gasConsumption = vehicleService.getGasConsumption(vehicleName);
        gasCCalculate1= localDistanceTraveled * gasConsumption;
        totalGasAmount1 = initialGasAmount1- localDistanceTraveled * gasConsumption;
        Gas1.setText(String.format("%.2f", totalGasAmount1));

        if (totalGasAmount1 <= 0) {
            Gas1.setText("Sin gasolina");
            transition1.stop();
            timeline1.stop();
            Rech1.setVisible(true);
        }

    }));
    timeline1.setCycleCount(Animation.INDEFINITE);

    transition1.play();
    timeline1.play();

    transition1.setOnFinished(e -> {
        LocalDateTime endTime = LocalDateTime.now();
        timeline1.stop();
        totalDistanceTraveled1 += km1Int;
        GoBack1.setVisible(true);
```

En conjunto, estas clases forman parte de una aplicación de gestión de viajes que ofrece funcionalidades para planificar, registrar y visualizar información sobre los viajes realizados por los usuarios. La aplicación utiliza una interfaz de usuario intuitiva y proporciona una experiencia interactiva para los usuarios al gestionar sus viajes.

Diagrama de clases:



Diagramas de flujo:



Inicio

|

V

Mostrar ventana principal

|

V

Inicializar componentes de la ventana principal

|

V

Esperar acción del usuario

|

V

Si se presiona el botón "LoadRoutesButton" entonces

| |

| V

| Mostrar cuadro de diálogo para seleccionar un archivo CSV

| |

| V

| Si se selecciona un archivo entonces

| | |

| | V

| | Leer el archivo CSV línea por línea

| | |

| | V

| | Para cada línea en el archivo CSV

| | | |

| | | V

| | | Obtener datos de inicio, fin y distancia

| | | |

| | | V

| | | Crear una nueva ruta con los datos obtenidos

| | | |

| | | V

| | | Agregar la ruta a la lista de rutas mostradas

| | | |

| | V

| | Actualizar las ComboBoxes en la ventana principal

| |

| V

| Fin

|

V

Si se presiona el botón "EditDistanceButton" entonces

| |

| V

| Mostrar ventana de edición de distancias de rutas

| |

| V

| Esperar acción del usuario en la ventana de edición

| |

| V

| Si se actualiza la distancia de una ruta entonces

| | |

| | V

| | Actualizar la distancia de la ruta en la lista mostrada

| |

| V

| Fin

|

V

Fin

Inicio

|

V

Inicializar componentes de la tabla

|

V

Cargar registros desde el archivo

|

V

Mostrar registros en la tabla

|

V

Esperar acción del usuario

|

V

Si se agrega un nuevo registro entonces

| |

| V

| Crear un nuevo objeto de registro con los datos proporcionados

| |

| V

| Asignar un ID al nuevo registro

| |

| V

| Añadir el nuevo registro a la lista de registros

| |

| V

| Guardar los registros en el archivo

| |

| V

| Mostrar los registros actualizados en la tabla

|

V

Si se elimina el archivo de registros entonces

| |

| V

| Eliminar el archivo de registros

|

V

Fin

Inicio

|

V

Inicializar componentes de la interfaz de usuario

|

V

Esperar acciones del usuario

|

|--- Si se presiona el botón "Start1" entonces

| |

| V

| Iniciar viaje para el vehículo 1

| |

| V

| Realizar animación de desplazamiento del vehículo 1

| |

| V

| Calcular distancia recorrida y consumo de combustible para el vehículo 1

| |

| V

| Actualizar etiquetas de distancia y combustible para el vehículo 1

| |

| V

| Si el combustible del vehículo 1 es insuficiente entonces

| | |

| | V

| | Detener animación y mostrar botón de recargar para el vehículo 1

| |

| V

| Si se completa el viaje del vehículo 1 entonces

| | |

| | V

| | Registrar el viaje del vehículo 1

| |

| V

| Si se presiona el botón "GoBack1" entonces

| | |

| | V

| | Iniciar viaje de regreso para el vehículo 1

| | |

| | V

| | Realizar animación de desplazamiento de regreso para el vehículo 1

| | |

| | V

| | Calcular distancia recorrida y consumo de combustible para el vehículo 1 en el regreso

| | |

| | V

| | Actualizar etiquetas de distancia y combustible para el vehículo 1 en el regreso

| | |

| | V

| | Si el combustible del vehículo 1 es insuficiente para el regreso entonces

| | | |

| | | V

| | | Detener animación de regreso y mostrar botón de recargar para el vehículo 1

| | |

| | V

| | Si se completa el viaje de regreso del vehículo 1 entonces

| | | |

| | | V

| | | Registrar el viaje de regreso del vehículo 1

| | |

| | V

| | Fin

| |

| V

| Fin

|

|--- Si se presiona el botón "Start2" o "Start3" entonces (acciones similares para vehículo 2 y 3)

| |

| V

| Iniciar viaje para el vehículo respectivo

| |

| V

| Realizar animación de desplazamiento del vehículo respectivo

| |

| V

| Calcular distancia recorrida y consumo de combustible para el vehículo respectivo

| |

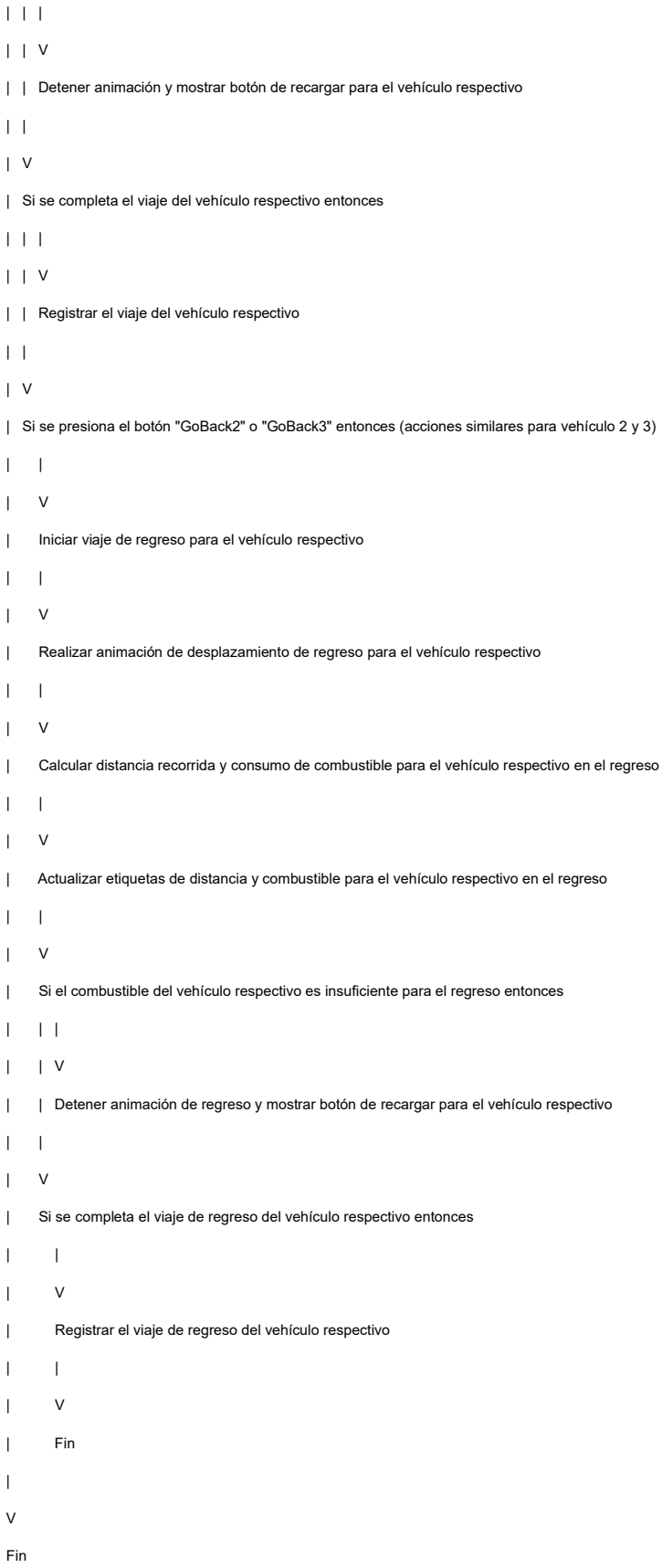
| V

| Actualizar etiquetas de distancia y combustible para el vehículo respectivo

| |

| V

| Si el combustible del vehículo respectivo es insuficiente entonces


Detener animación y mostrar botón de recargar para el vehículo respectivo
Si se completa el viaje del vehículo respectivo entonces
Registrar el viaje del vehículo respectivo
Si se presiona el botón "GoBack2" o "GoBack3" entonces (acciones similares para vehículo 2 y 3)
Iniciar viaje de regreso para el vehículo respectivo
Realizar animación de desplazamiento de regreso para el vehículo respectivo
Calcular distancia recorrida y consumo de combustible para el vehículo respectivo en el regreso
Actualizar etiquetas de distancia y combustible para el vehículo respectivo en el regreso
Si el combustible del vehículo respectivo es insuficiente para el regreso entonces
Detener animación de regreso y mostrar botón de recargar para el vehículo respectivo
Si se completa el viaje de regreso del vehículo respectivo entonces
Registrar el viaje de regreso del vehículo respectivo
Fin
Fin