### Manual Técnico de UDrive

### 1. Requerimientos funcionales:

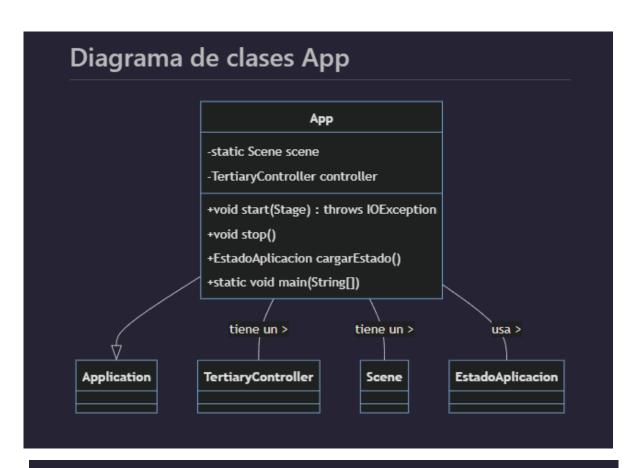
- Lanzar una aplicación orientada al manejo de viajes y rutas, en el cual se pueden cargar rutas a una tabla por medio de un archivo CSV a una tabla, y a partir de esos datos, generar rutas de viajes.
- Al momento de generar las rutas, poder visualizar el estado de cada viaje por medio de imágenes que representan los vehículos seleccionados por viaje anteriormente.
   Ver en tiempo real la posición de cada uno de ellos, la distancia que ha recorrido cada uno, cuánto han consumido de gasolina y cuánto les queda, y si se les agota, pausar el recorrido, recargar combustible, y continuar con el trayecto del viaje
- Visualizar en una tabla el historial de cada viaje, representado cada uno por un ID único, y mostrar información como la fecha y hora en la que se iniciaron los viajes, fecha y hora en la que finalizaron, qué vehículos se utilizaron, las distancias que recorrieron cada uno, y cuánta gasolina consumió cada vehículo.
- Utilizar la serialización en Java para que, cuando se cierre la aplicación durante la
  ejecución de viajes, estos al momento de que se vuelva a abrir la aplicación, los
  viajes se queden como se dejaron en la sesión previo al cierre de la aplicación, lo
  mismo con el historial, que se mantenga la información de la tabla aunque se cierre
  la aplicación, y toda la información serializada almacenarla como información
  binaria.

#### 2. Atributos del Sistema:

- Aplicación interactiva para el manejo de rutas y viajes.
- De fácil utilización para cualquier tipo de usuario
- Una interfaz agradable a la vista e intuitiva.

### 3. Requerimientos mínimos de entorno de Desarrollo:

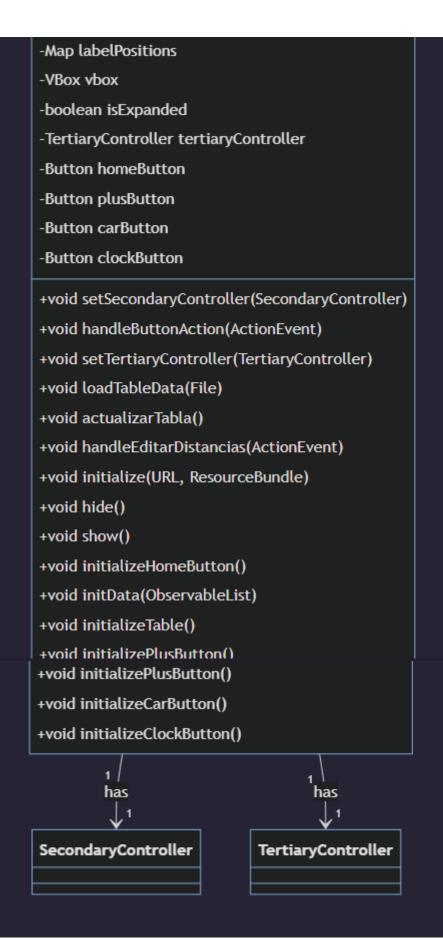
- Visual Studio Code
  - o Procesador con velocidad o 1.6 GHz o superior
  - o Memoria RAM de 1GB
- Windows 10/11, macOS 10.15 o superior, Linux
- Java 21 y JavaFX 21



# Diagrama de clases PrimaryController

### PrimaryController

- -SecondaryController secondaryController
- -TableView tablaRecorridos
- -TableColumn idColumna
- -ObservableList recorridos
- -TableColumn inicioColumna
- -TableColumn finColumna
- -TableColumn distanciaColumna
- -String rutaArchivo
- -Circle circle
- -AnchorPane sidebar
- -Button toggleButton
- -Label homeLabel
- -Label plusLabel
- -Label carLabel
- -Label clockLabel
- -Label userLabel



## Diagrama de clases SecondaryController

### SecondaryController

- -ComboBox startPointComboBox
- -ComboBox endPointComboBox
- -ComboBox transportComboBox
- -Map startPointToEndPoint
- -Map endPointToStartPoint
- $\hbox{-} Tertiary Controller tertiary Controller \\$
- -Circle circle
- -AnchorPane sidebar
- -Button toggleButton
- -Label homeLabel
- -Label plusLabel
- -Label carLabel
- -Label clockLabel
- -Label userLabel
- -Map labelPositions
- -VBox vbox
- -hoolean is Expanded -Observable List recorridos
- -ObservableList viajes
- -Button homeButton
- -Button plusButton
- -Button generarNuevoViaje
- -Label pilotosOcupadosLabel
- -Button carButton
- -Button clockButton
- +setTertiaryController(TertiaryController)
- +initData()
- +setStartPointToEndPoint(Map)
- +getRecorridos() : ObservableList
- +setEndPointToStartPoint(Map)
- +handleStartPointSelection(String)
- +setRecorridos(ObservableList)
- +initializeComponentesCombobox()
- +initialize(URL, ResourceBundle)
- +hide()
- +show()
- +initializeHomeButton()
- +initializePlusButton()



# Diagrama de clases TertiaryController

## **TertiaryController**

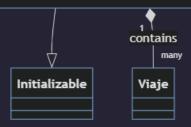
- -HBox viaje1
- -HBox viaje2
- -HBox viaje3
- -Button iniciarViaje1Button
- -Button iniciarViaje2Button
- -Button iniciarViaje3Button
- -Button regresarViaje1
- -Button regresarViaje2
- -Button regresarViaje3
- -List viajes
- -ObservableList recorridos
- +ObservableList getRecorridos()
- +void setRecorridos(ObservableList)
- +void initViajes(List)

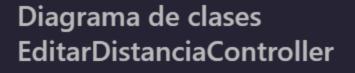
Viaje

# Diagrama de clases QuaternaryController

### QuaternaryController

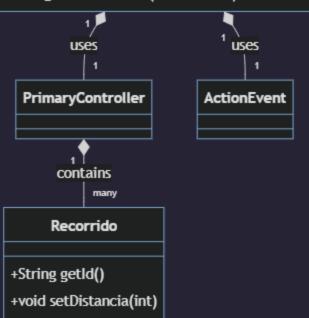
- -TableView viajesTableView
- -TableColumn idColumn
- -TableColumn fechalnicioColumn
- -TableColumn fechaFinColumn
- -TableColumn distanciaColumn
- -TableColumn transporteColumn
- -TableColumn gasolinaColumn
- -Circle circle
- -AnchorPane sidebar
- -Button toggleButton
- -Label homeLabel
- -Label plusLabel
- -Label carLabel
- -Label clockLabel
- -Label userLabel
- -Map labelPositions
- -VBox vbox
- -boolean isExpanded
- -Button homeButton
- -Button plusButton
- -Button generarNuevoViaje
- -Button carButton
- -Button clockButton
- +void setViajeData(List, List, ArrayList, ArrayList, double[])
- +void initialize(URL, ResourceBundle)
- +void hide()
- +void show()
- +void initializeHomeButton()
- +void initializePlusButton()
- +void initializeCarButton()
- +void initializeClockButton()

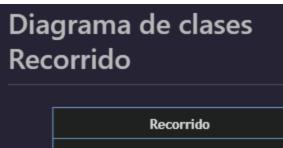




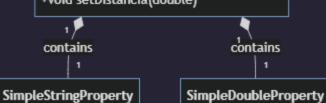
## EditarDistanciaController

- -TextField idField
- -TextField distanciaField
- -PrimaryController recorridosController
- +void setRecorridosController(PrimaryController)
- +void guardarCambios(ActionEvent)





- -SimpleStringProperty id
- -SimpleStringProperty inicio
- -SimpleStringProperty fin
- -SimpleDoubleProperty distancia
- +Recorrido(String, String, String, double)
- +String getId()
- +String getInicio()
- +String getFin()
- +double getDistancia()
- +void setId(String)
- +void setInicio(String)
- +void setFin(String)
- +void setDistancia(double)



# Diagrama de clases Viaje

### Viaje

- -String id
- -String transporte
- -double capacidadTanque
- -double gastoGasolina
- -String fechalnicio
- -String fechaFin
- +Viaje(String, String, String, String, double, String, String)
- +String getTransporte()
- +void setTransporte(String)
- +double getCapacidadTanque()
- +void setCapacidadTanque(double)
- +double getGastoGasolina()
- +void setGastoGasolina(double)
- +String getFechalnicio()
- +void setFechalnicio(String)
- +String getFechaFin()

