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

Diagrama de clases App

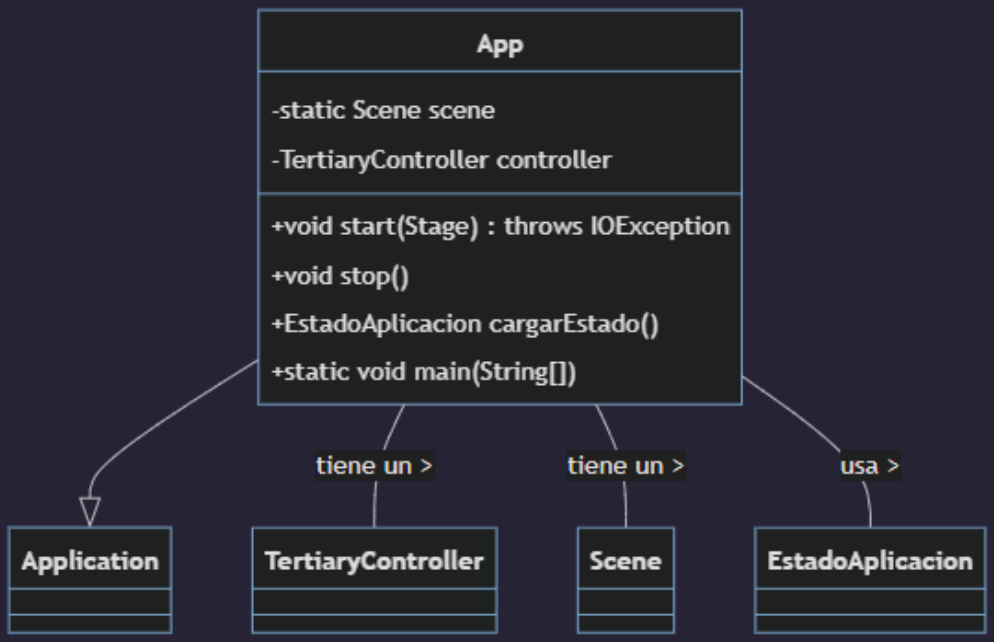
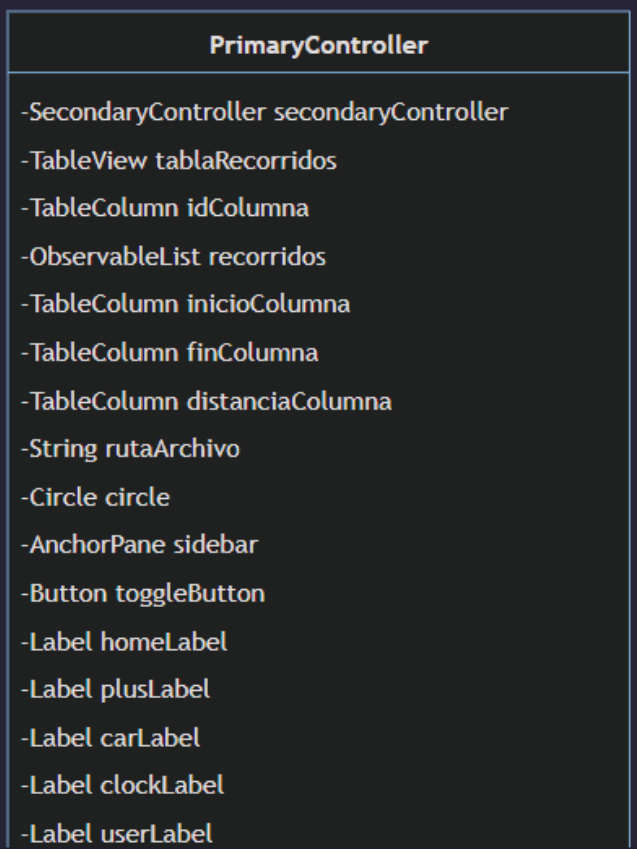


Diagrama de clases PrimaryController



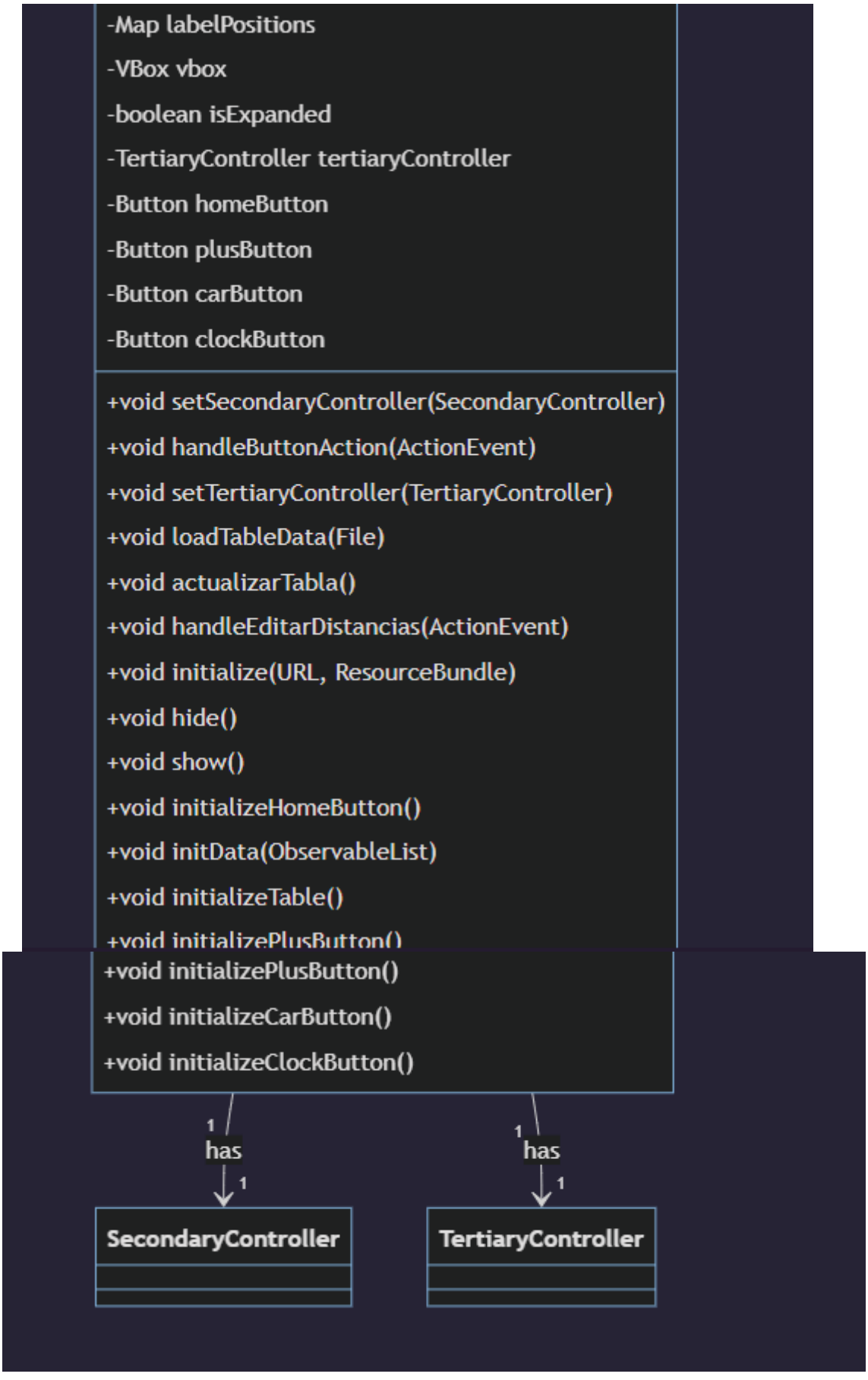
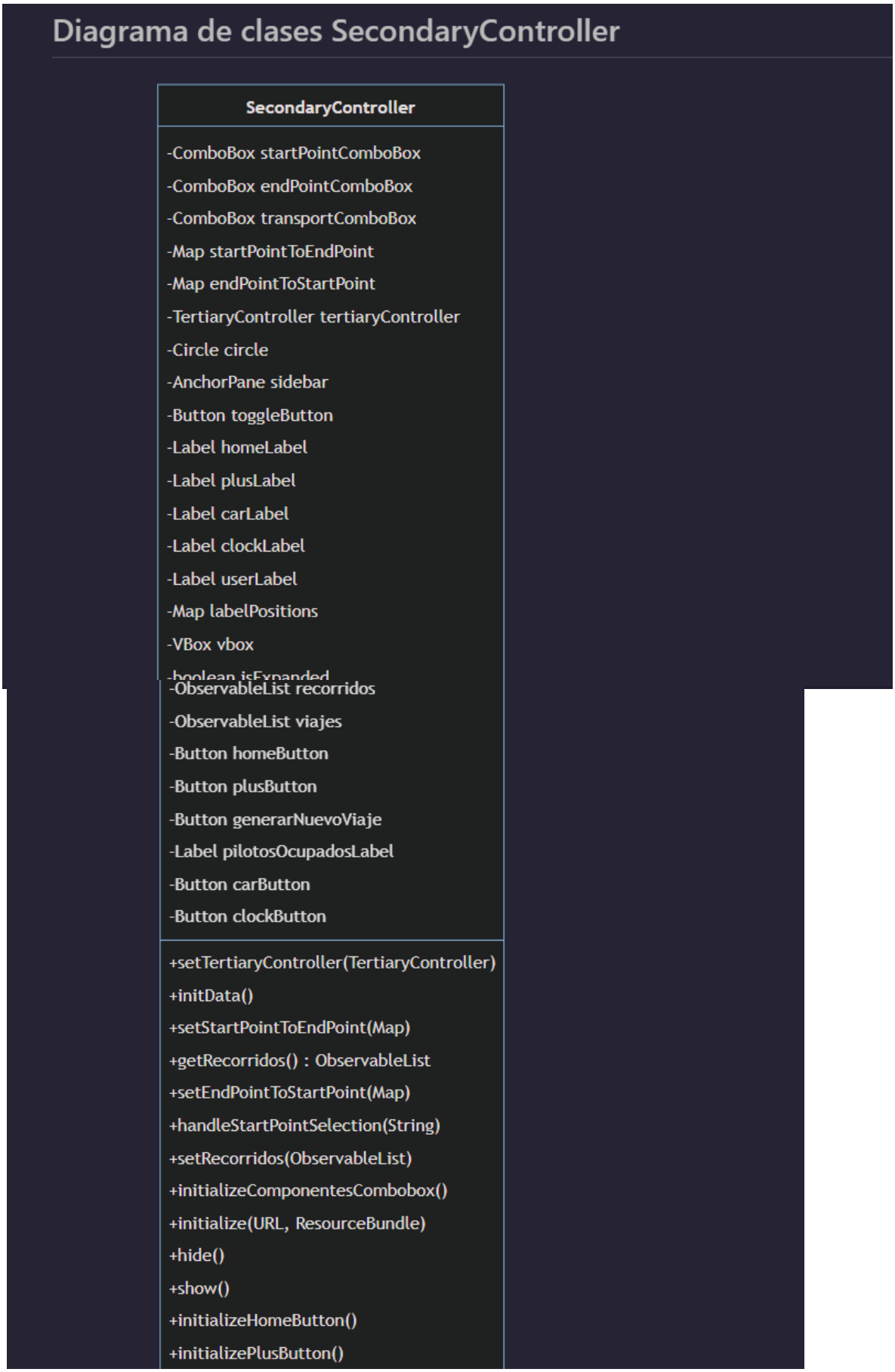

-Map labelPositions
-VBox vbox
-boolean isExpanded
-TertiaryController tertiaryController
-Button homeButton
-Button plusButton
-Button carButton
-Button clockButton
+void setSecondaryController(SecondaryController)
+void handleButtonAction(ActionEvent)
+void setTertiaryController(TertiaryController)
+void loadTableData(File)
+void actualizarTabla()
+void handleEditarDistancias(ActionEvent)
+void initialize(URL, ResourceBundle)
+void hide()
+void show()
+void initializeHomeButton()
+void initData(ObservableList)
+void initializeTable()
+void initializePlusButton()
+void initializePlusButton()
+void initializeCarButton()
+void initializeClockButton()
1
has
1
1
has
1
SecondaryController
TertiaryController

Diagrama de clases SecondaryController



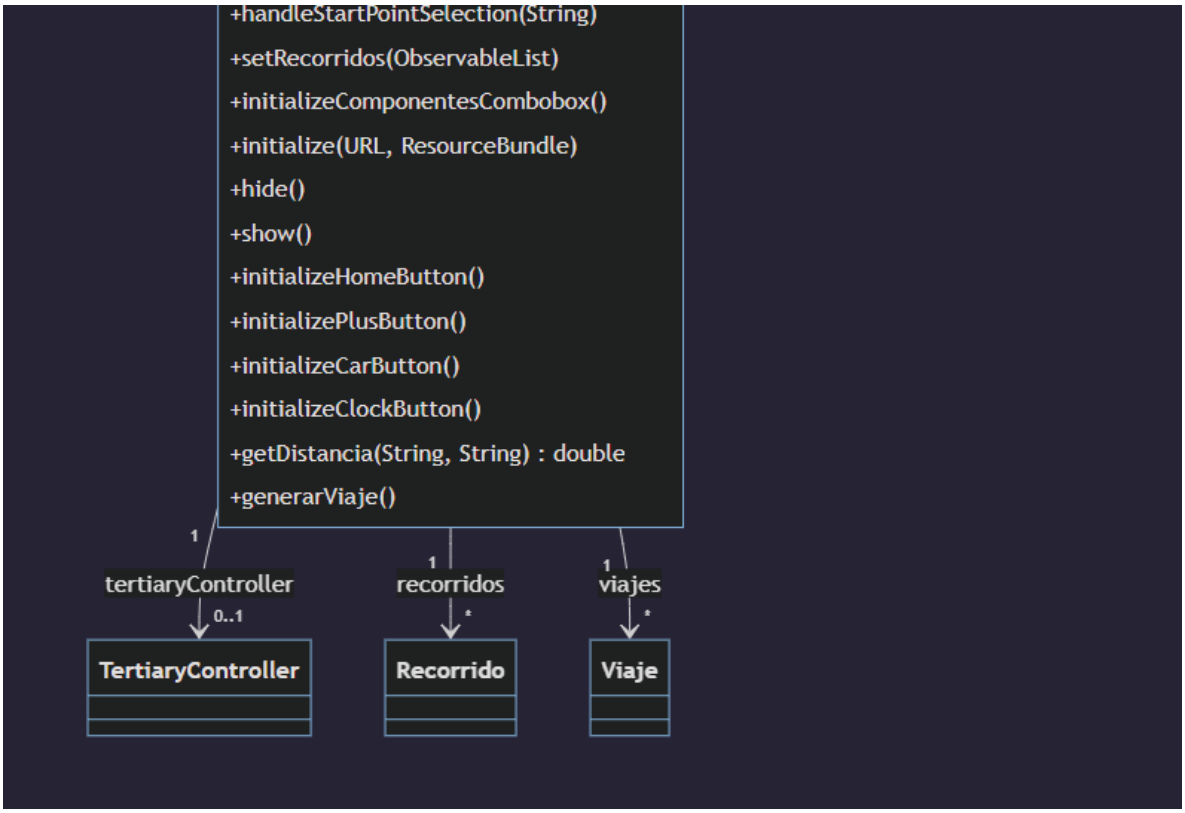

+handleStartPointSelection(String)
+setRecorridos(ObservableList)
+initializeComponentesCombobox()
+initialize(URL, ResourceBundle)
+hide()
+show()
+initializeHomeButton()
+initializePlusButton()
+initializeCarButton()
+initializeClockButton()
+getDistancia(String, String) : double
+generarViaje()
1
tertiaryController
0..1
TertiaryController
1
recorridos
*
Recorrido
1
viajes
*
Viaje

Diagrama de clases TertiaryController

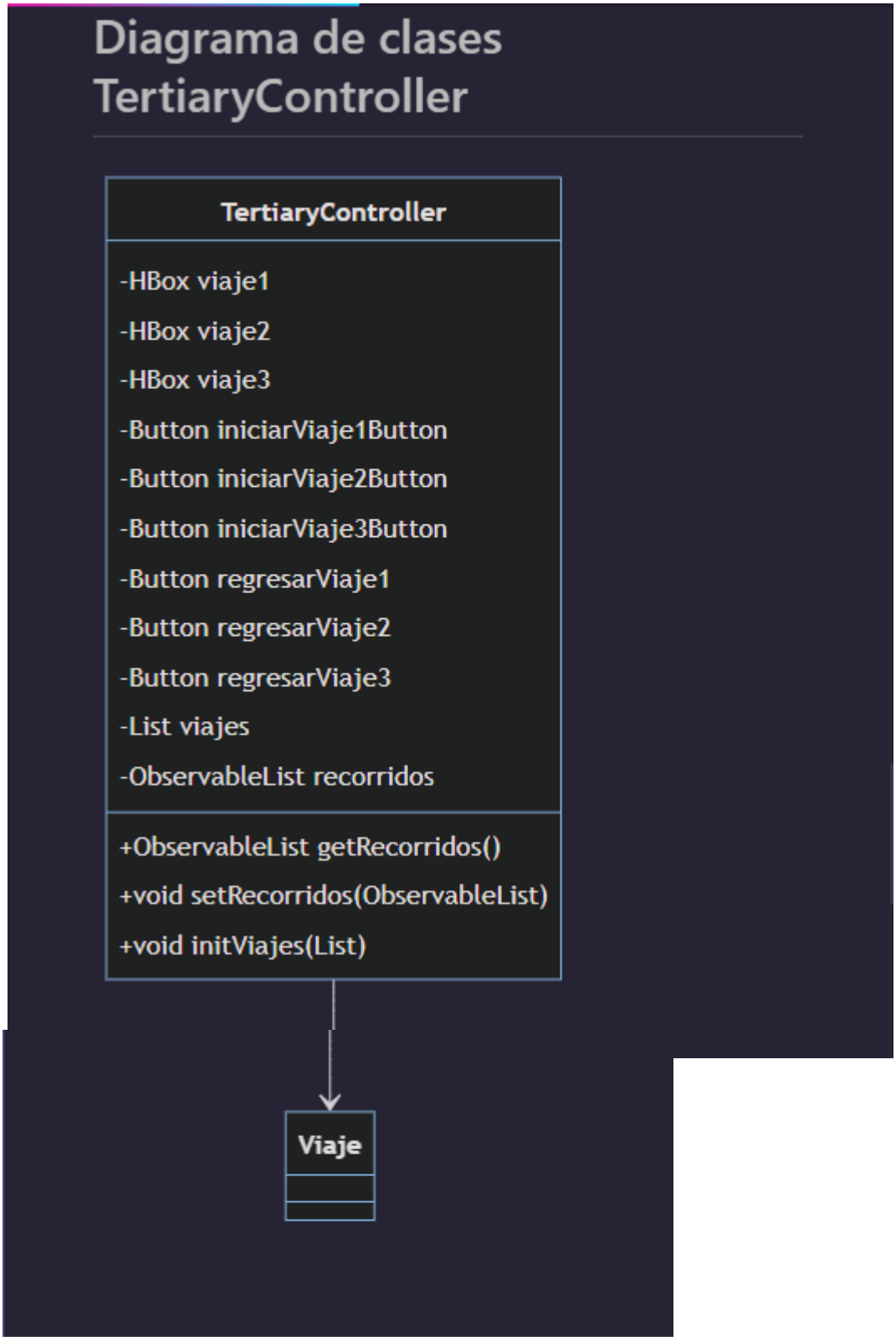


Diagrama de clases QuaternaryController

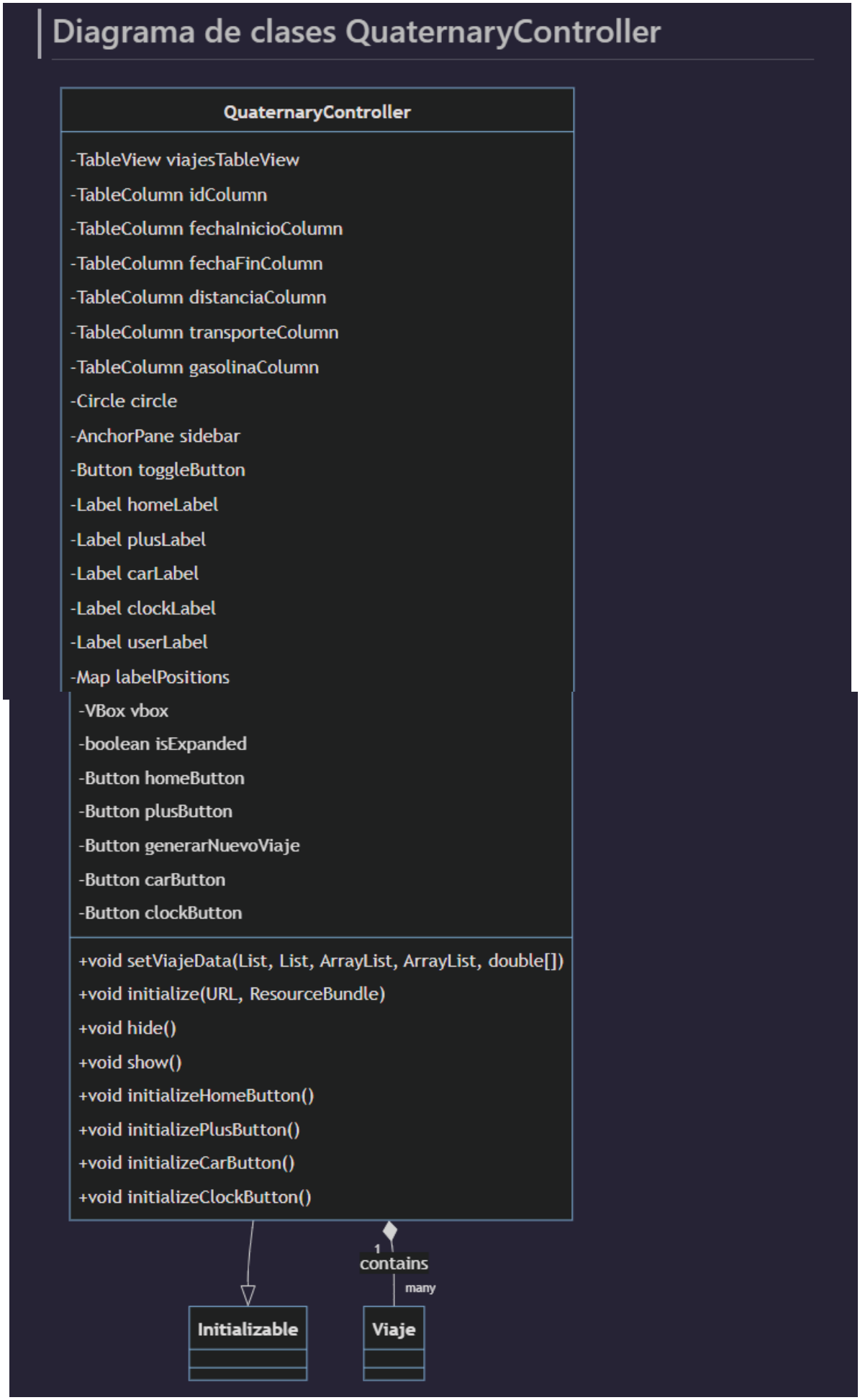


Diagrama de clases EditarDistanciaController

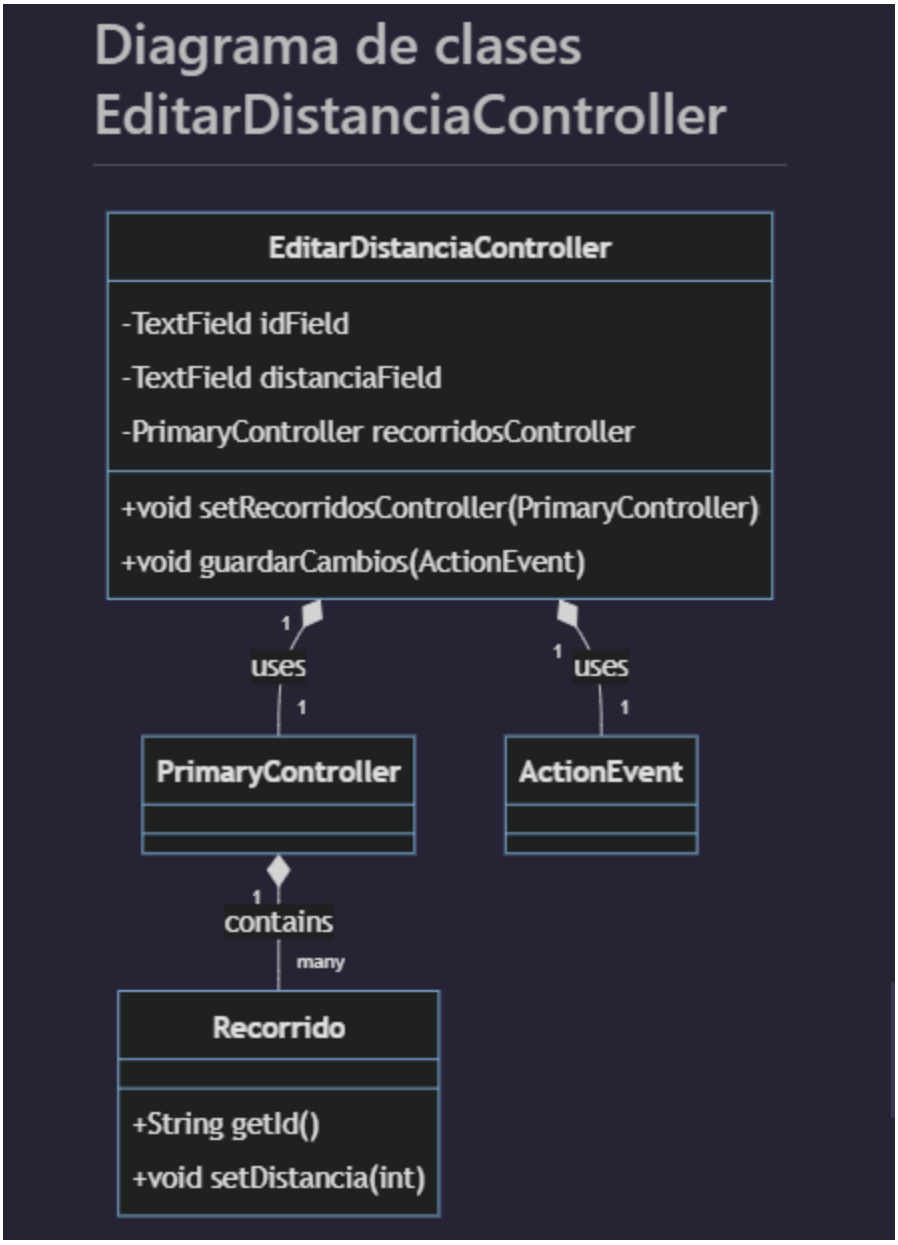


Diagrama de clases Recorrido

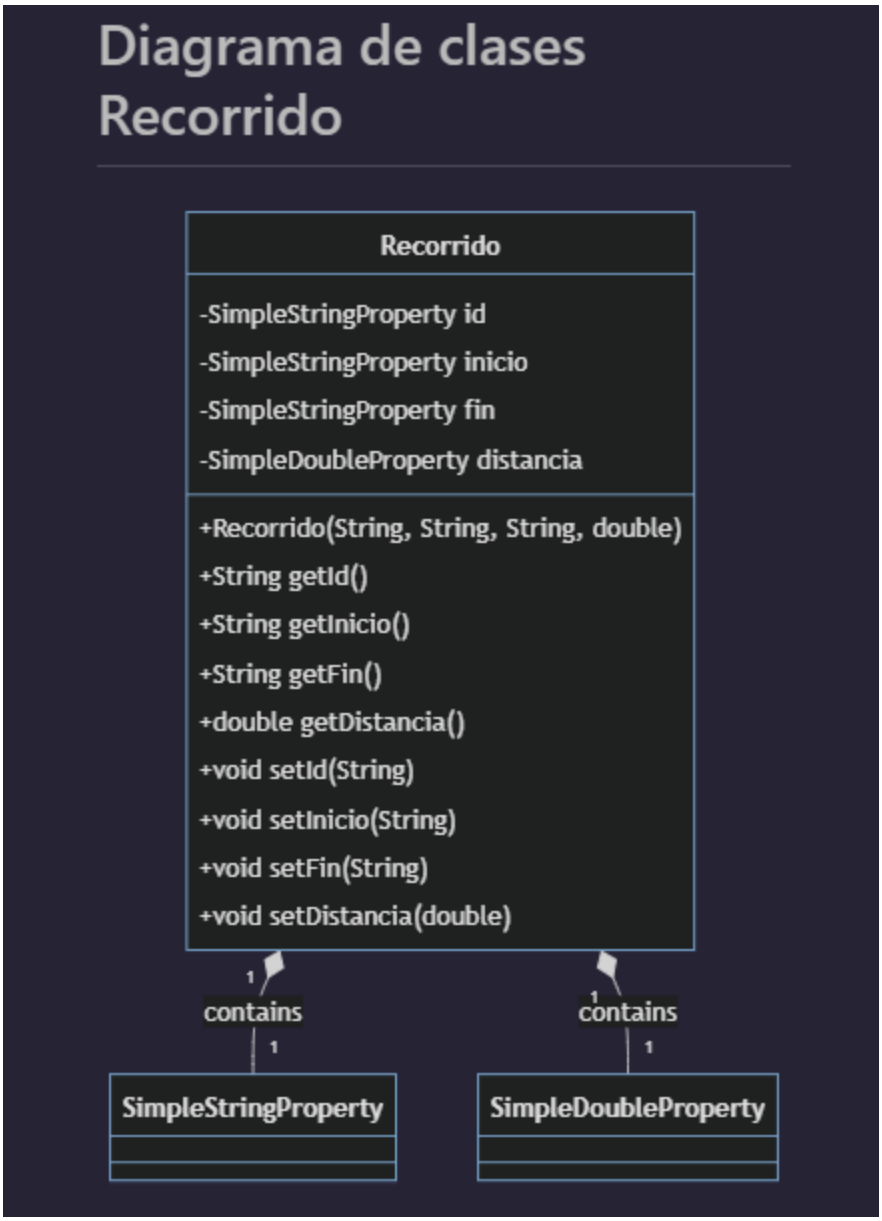
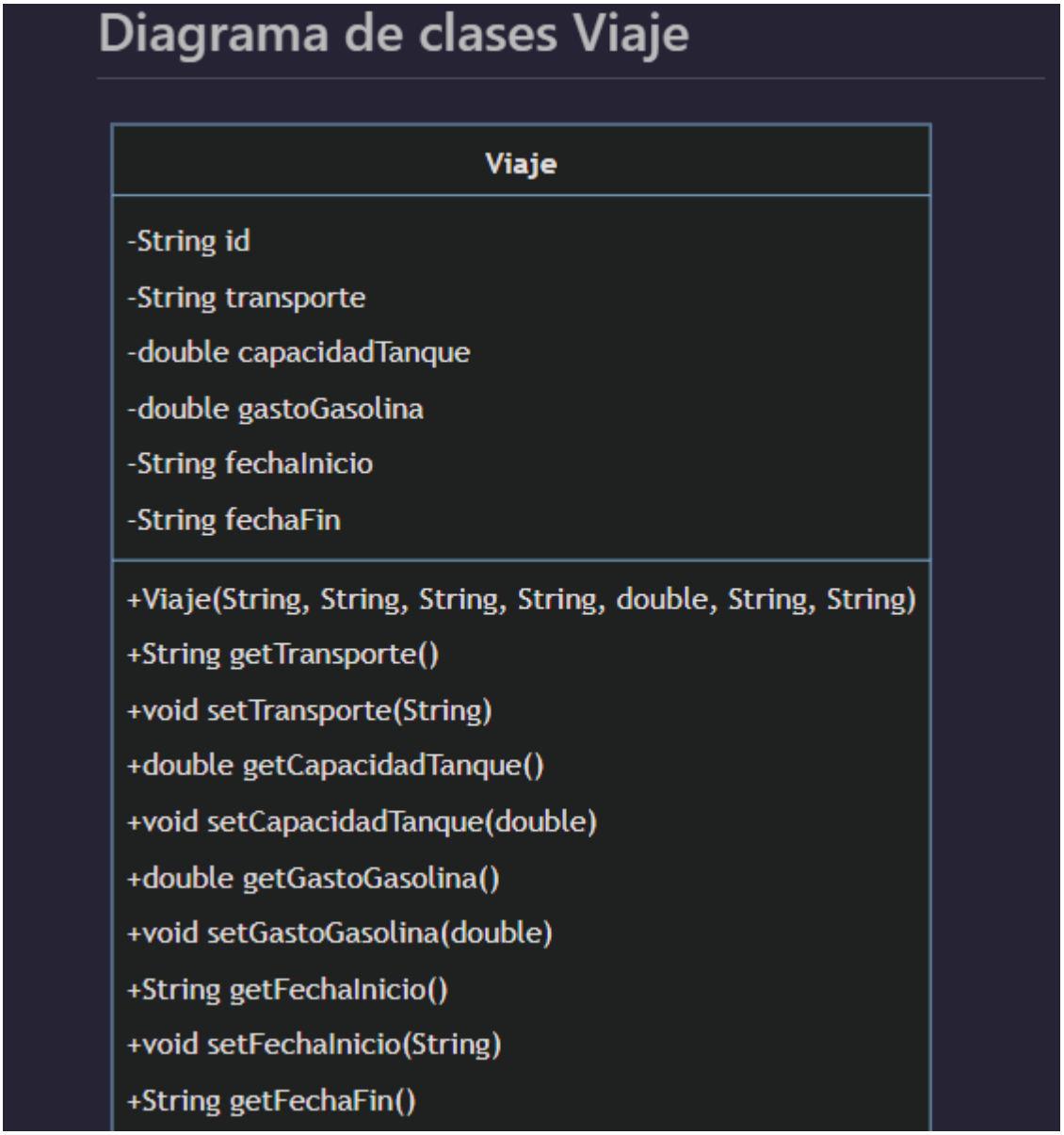


Diagrama de clases Viaje




+void setFechaFin(String)
+String toString()
+String getId()
+void setId(String)
extends
implements
Recorrido
Serializable

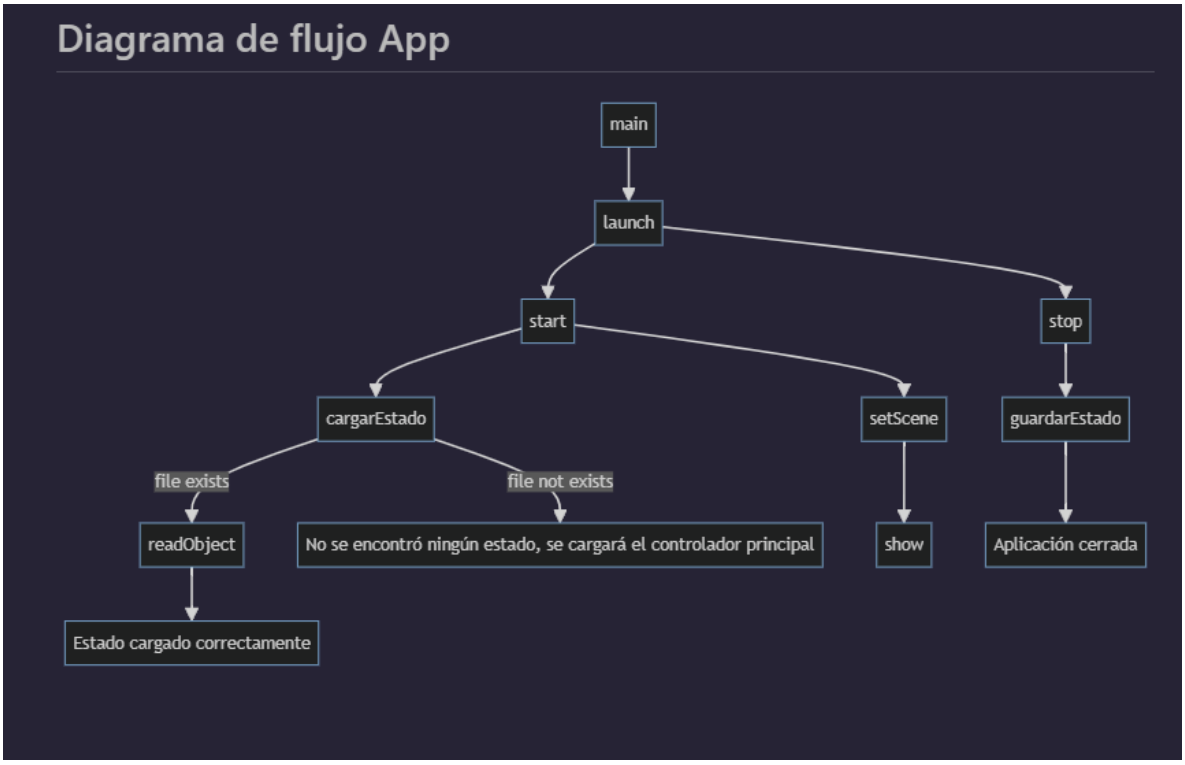


Diagrama de flujo App
main
launch
start
stop
cargarEstado
setScene
guardarEstado
file exists
file not exists
readObject
No se encontró ningún estado, se cargará el controlador principal
show
Aplicación cerrada
Estado cargado correctamente

Diagrama de flujo PrincipalController

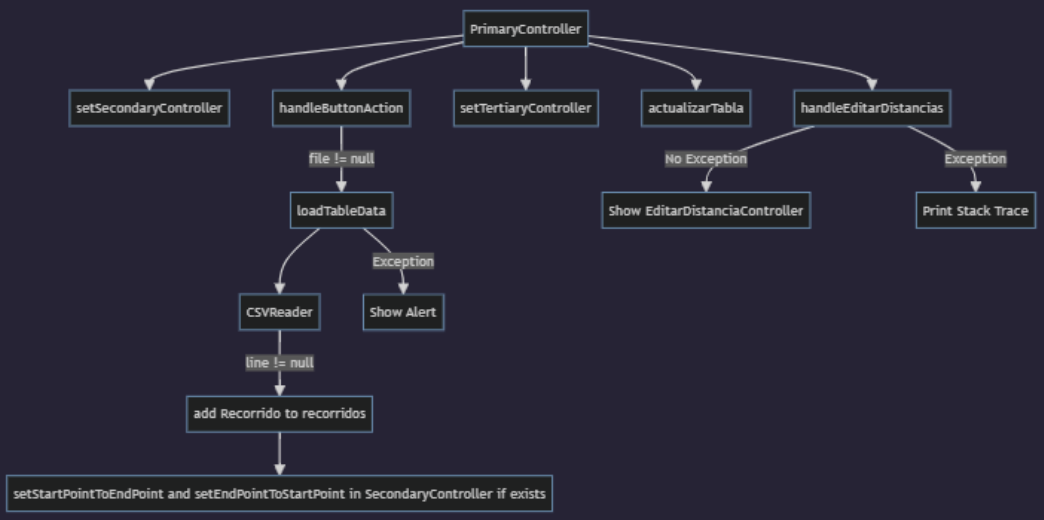


Diagrama de flujo SecondaryController

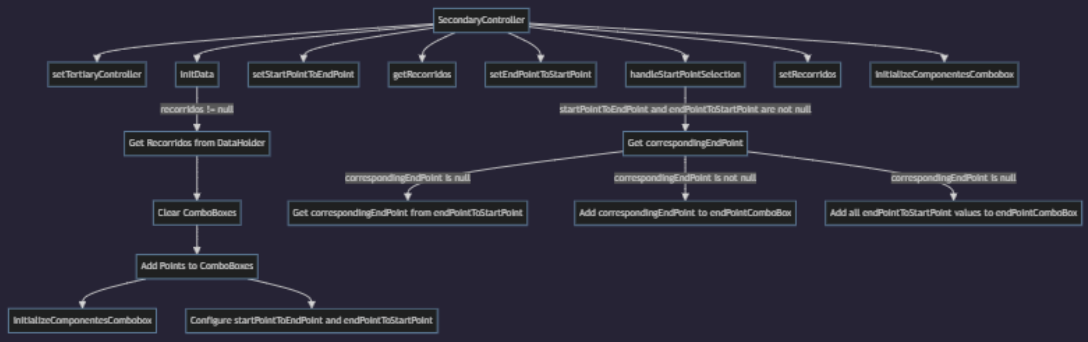


Diagrama de flujo TertiaryController

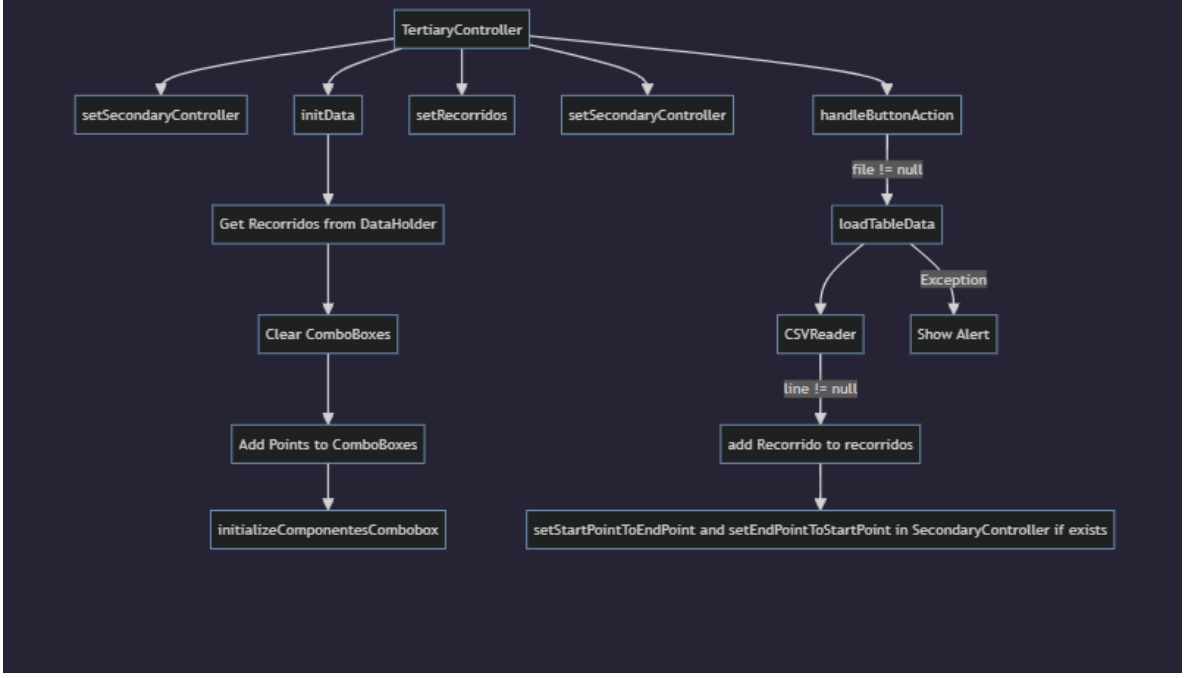


Diagrama de flujo QuaternaryController