

REQUERIMIENTOS FUNCIONALES

RF1	
Nombre	“Generar listado aleatorio de vuelos”
Resumen	Permite generar un listado aleatorio de vuelos con su respectiva información (fechas, horarios, aerolíneas, números (id único), ciudades de destino y puertas de embarque.
Entradas	(tamaño : el tamaño deseado de vuelos aleatorios que quiere generar el usuario)
Resultado	Se ha generado un nuevo listado de vuelos de manera aleatoria (si el usuario así lo desea) y se han desplegado los nuevos a través de la interfaz.

RF2	
Nombre	“Navegar a través de la pantalla”
Resumen	Permite que el usuario pueda visualizar los n vuelos generados y desplegados aleatoriamente en la interfaz, a través de la interacción con botones (hacia adelante y hacia atrás).
Entradas	
Resultado	Se ha avanzado/retrocedido en la interfaz para mostrar los siguientes vuelos que hacen parte del n total.

RF3	
Nombre	“Ordenar los vuelos desplegados por criterio”
Resumen	Permite que el usuario pueda ordenar los n vuelos aleatorios generados y mostrados actualmente a partir de los 3 distintos métodos de ordenamiento clásicos (inserción, selección y burbuja), a través de todos los distintos criterios de ordenamiento (por hora, por fecha, por ciudad, etc.) y calculando el tiempo que tardó el proceso.
Entradas	
Resultado	Se han ordenado los vuelos de acuerdo al criterio deseado por el usuario y se ha desplegado el tiempo que tardó la ejecución.

RF4	
Nombre	“Buscar los vuelos desplegados por criterio”
Resumen	Permite que el usuario pueda buscar los n vuelos aleatorios generados y mostrados actualmente a partir de los métodos de búsqueda clásicos (secuencial y binaria), a través de los distintos criterios de búsqueda (por hora, por fecha, por ciudad, etc.) y calculando el tiempo que tardó el proceso.
Entradas	
Resultado	Se ha resaltado el vuelo buscado de acuerdo al criterio deseado por el usuario (si los vuelos coinciden se mostrará la primera ocurrencia) y se ha desplegado el tiempo que tardó la ejecución.

REQUERIMIENTOS NO FUNCIONALES

Mostrar las funcionalidades del programa a través de una interfaz gráfica interactiva de javaFX.

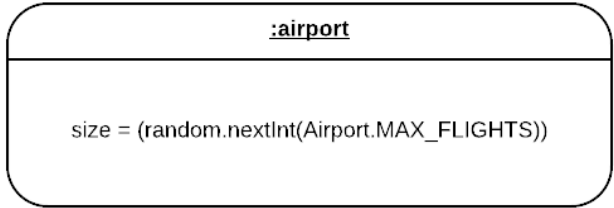
TRAZABILIDAD DEL ANALISIS AL DISEÑO

Requerimientos Funcionales	Clases	Métodos
<p>Generar listado aleatorio de vuelos</p> <p>Juan José Valencia Jaramillo</p>	<p><u>AirportScreenController</u> <u>Airport</u> <u>Flight</u></p>	<p>-Initialize () : void -InitializeTV() : void +generateRandomFlights () : void +getFlights() : ObservableList<Flight> +updateGUI() : void</p> <p>+Airport(int size) +setFlights(int size) : void +init() : void +checkIDs() : void +load(String path) : void +generateRandomAirline() : void +generateRandomDestination() : void -generateRandomGate() : void</p> <p>+Flight() -generateRandomDate() : String -generateRandomHour() : String +generateRandomID() : int -verifyDate() : void -verifyHour() : void +setId(int newid) : void</p>

Navegar a través de la pantalla	<u>AirportScreenController</u> <u>Airport</u>	
Ordenar los vuelos desplegados por criterio	<u>AirportScreenController</u> <u>Insertion</u> <u>Selection</u> <u>Bubble</u> <u>FlightComparator</u>	+ sortByHour() : void + sortByDate() : void + sortByDestination() : void + sortByAirline() : void + sortByIdentifier() : void + sortByGate() : void + updateGUI() : void +sort(T[] , Comparator <T>) : void +sortByAirline(Airport airport) : void +sortByGate(Airport airport) : void +sortByIdentifier(Airport airport) : void
Buscar los vuelos desplegados por criterio	<u>AirportScreenController</u> <u>Linear</u> <u>Binary</u>	+ searchAirline() : void + searchIdentifier() : void + searchGate() : void + searchDestination() : void + searchIdentifier(Airport airport, int id) : int

		+ searchGate(Airport airport, String gate) : int + searchDestination(Airport airport, String destination) : int + searchAirline(Airport airport, String airline) : int
--	--	--

DISEÑO DE PRUEBAS(Escenarios)

Nombre	Clase	Escenario
setUpScenary1	AirportTest	vacío
setUpScenary2	AirportTest	 <pre> classDiagram class Airport { size = (random.nextInt(Airport.MAX_FLIGHTS)) } </pre>

DISEÑO DE PRUEBAS(Pruebas)

Objetivo de la prueba: Verificar la correcta creación de un aeropuerto				
Clase	Método	Escenario	Valores de entrada	Salida
Airport	Airport	setUpScenary1	size = Airport.MAX_FLIGHTS	Se ha creado un nuevo aeropuerto con su respectivo tamaño (aleatorio) de vuelos.

Objetivo de la prueba:				
Clase	Método	Escenario	Valores de entrada	Salida
Alrport	Init()	setUpScenary2	Ninguno	Se han inicializado los vuelos con atributos aleatorios dentro del aeropuerto.

Objetivo de la prueba:				
Clase	Método	Escenario	Valores de entrada	Salida
Airport	checkIDs()	setUpScenary2()	Ninguno	Los Id's de los aviones son mutuamente excluyentes y cada uno se encuentra asignado a un número único.