

LABORATORIO 5

Requerimientos funcionales

Nombre	R.# 1. Mostrar información de los vuelos.
Resumen	Permite desplegar la información de los vuelos, teniendo en cuenta que las fechas serán dadas en formato AM/PM.
Entradas	
- ninguna	
Resultados	
Se muestra toda la información de los vuelos.	

Nombre	R.# 2. Generar listado de vuelos
Resumen	Permite generar un listado de vuelos con diferentes fechas, horarios, diferentes aerolíneas, diferentes número de vuelos, diferentes destinos y puertas de embarque. Hay que mencionar además que el número de vuelo es único para cada embarque. El usuario podrá tener una vista paginada del listado de vuelos con teclas de navegación hacia adelante y hacia atrás.
Entradas	
- clic en botones de navegación	
Resultados	
Se muestra el listado de vuelos con los criterios mencionados.	

Nombre	R.# 3. Ordenar vuelos.
Resumen	Permite que el usuario ordene los vuelos por otros criterios además del por defecto que es por fecha y horario.
Entradas	
- Criterio a tener en consideración para ordenar.	
Resultados	
Se ordena el listado de vuelos teniendo en cuenta el criterio elegido por el usuario.	

Nombre	R.# 4. Buscar vuelo.
Resumen	Ofrece al usuario la posibilidad de buscar un vuelo por el criterio que este desee. Por ejemplo: por fecha, por hora, por número de vuelo, entre otros.
Entradas	
- Criterio de búsqueda.	
Resultados	
Se muestra el vuelo de acuerdo al criterio elegido por el usuario.	

Nombre	R.# 5. Mostrar tiempo
Resumen	Permite que se muestre el tiempo en que tarda el programa en buscar un vuelo o el tiempo que tarda en ordenarlos. El tiempo será mostrado en segundos.
Entradas	
- ninguno.	
Resultados	
Se genera un tipo de alerta que muestra el tiempo que tarda el programa en hacer dichas operaciones.	

Requerimientos no funcionales

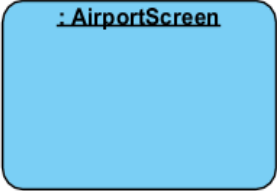
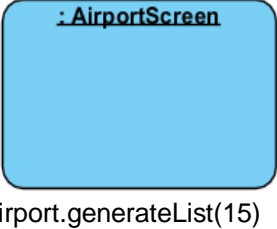
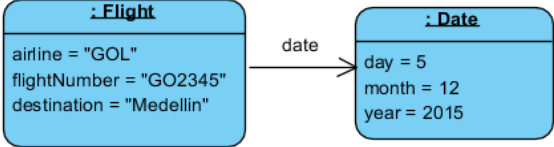

Nombre	R.# 1. Interactuar con usuario.
Resumen	Permite que, mediante una interfaz gráfica amigable, el usuario lleve a cabo los requerimientos y así mismo el desarrollo de la aplicación.
Entradas	
- Ninguna.	
Resultados	
Se muestra la interfaz gráfica al usuario.	

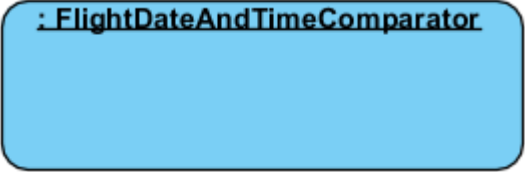
El diagrama de clases es un enlace:

[Diagrama de clases del modelo y la interfaz](#)

Diseño de casos de pruebas unitarias

Configuración de los Escenarios

Nombre	Clase	Escenario
setupScenary1	AirportScreenTest	vacío
setupScenary2	AirportScreenTest	 <pre> classDiagram class AirportScreen { } </pre>
setupScenary3	AirportScreenTest	 <pre> classDiagram class AirportScreen { } AirportScreen --> : airport.generateList(15) </pre>
setupScenary1	FlightTest	vacío
setUpScenary2	FlightTest	 <pre> classDiagram class Flight { airline = "GOL" flightNumber = "GO2345" destination = "Medellin" } class Date { day = 5 month = 12 year = 2015 } Flight --> Date : date </pre>
setUpScenary1	DateTest	vacío
setUpScenary2	DateTest	 <pre> classDiagram class Date { day = 1 month = 10 year = 2011 } </pre>

setUpScenart1	FlightDateAndTimeComparatorTest	
---------------	---------------------------------	------------------------------------------------------------------------------------

Diseño de Casos de Prueba

Objetivo de la Prueba: Verificar la correcta creación de los objetos y el correcto funcionamiento de los métodos triviales				
Clase	Método	Escenario	Valores de Entrada	Resultado
AirportScreenTest	testAirportScreen	setupScenary1	Ninguno	Se ha creado un nuevo objeto de tipo AirportScreen exitosamente. Se ha comprobado que los métodos getters funcionan correctamente.
FlightTest	testFlight	setUpScenary1	Ninguno	Se ha creado un nuevo objeto de tipo FLight exitosamente. Los métodos getters, devuelven el valor correcto.
DateTest	testDate	setUpScenary1	Ninguno	Se ha creado un nuevo objeto tipo Date exitosamente. Los métodos getters funcionan apropiadamente.

Objetivo de la Prueba: Verificar la lectura correcta de los archivos.				
Clase	Método	Escenario	Valores de Entrada	Resultado
AirportScreenTest	testLoadArchive testLoadArchive2	setupScenary2	ninguno	El método efectivamente encuentra el archivo y lo lee correctamente. Así mismo, se lanza la excepción en los momentos apropiados.

Objetivo de la Prueba: Verificar que se agrega un elemento a la lista correctamente				
Clase	Método	Escenario	Valores de Entrada	Resultado
AirportScreenTest	testAddFlight	setupScenary2	new Flight("ADA", "ADA1001", "Bogota", new Date(1,2,2018))	El método efectivamente agrega un elemento al final de la lista.

Objetivo de la Prueba: Verificar que la lista de vuelos se genera aleatoriamente y de forma correcta.				
Clase	Método	Escenario	Valores de Entrada	Resultado
AirportScreenTest	testGenerateList	setupScenary2	size = 15	El método efectivamente genera una lista de vuelos aleatorios de acuerdo con el tamaño dado.
AirportScreenTest	testRandom	setupScenary2	Ninguno	El método efectivamente devuelve un objeto de tipo Flight no nulo, el cual consta de atributos generados aleatoriamente.

Objetivo de la Prueba: Verificar que se ordena correctamente la lista de vuelos con base en los criterios entregados.				
Clase	Método	Escenario	Valores de Entrada	Resultado
AirportScreenTest	testSortingByDateAndTime	setupScenary3	Ninguno	El método efectivamente ordena la lista de vuelos de forma ascendente según la fecha y hora de salida de estos.

AirportScreenTest	testSortingByAirline	setupScenary3	Ninguno	El método efectivamente ordena la lista de vuelos de forma ascendente lexicográficamente según la aerolínea de estos.
AirportScreenTest	testSortingByFlightNumber	setupScenary3	Ninguno	El método efectivamente ordena la lista de vuelos de forma ascendente según el número de vuelo que presentan.
AirportScreenTest	testSortingByDestination	setupScenary3	Ninguno	El método efectivamente ordena la lista de vuelos de forma ascendente lexicográficamente según el destino.
AirportScreenTest	testSortingByBoarding Gate	setupScenary3	Ninguno	El método efectivamente ordena la lista de vuelos de forma ascendente según la puerta de embarque de cada vuelo

Objetivo de la Prueba: Verificar que el proceso de búsqueda de un vuelo sea correcto según el criterio de búsqueda, ya sea que se encuentre o no en la lista.				
Clase	Método	Escenario	Valores de Entrada	Resultado
AirportScreenTest	testSearchByDate	setupScenary3	year = 2025	El método efectivamente encuentre a un vuelo que está en la lista dependiendo de la fecha. Así mismo, no encuentra a los que no pertenecen a la lista. Como, por ejemplo, cuando se le pasa un año que se encuentre por fuera del dominio de los años de los vuelos.
AirportScreenTest	testSearchByTime	setupScenary3	Time = "00.00 AM"	El método efectivamente encuentre a un vuelo que está en la lista dependiendo de la hora de salida. Así mismo, no encuentra a los que no pertenecen a la lista. Como, por ejemplo, cuando se le pasa una hora que ningún vuelo tiene.
AirportScreenTest	testSearchByAirline	setupScenary3	airline = "non-existent"	El método efectivamente encuentre a un vuelo que está en la lista dependiendo de la aerolínea. Así mismo, no encuentra a los que no pertenecen a la lista. Como, por ejemplo, cuando se le pasa el nombre de una aerolínea erróneo.
AirportScreenTest	testSearchByDestination	setupScenary3	airline = "non-existent"	El método efectivamente encuentre a un vuelo que está en la lista dependiendo del destino hacia

				donde se dirige el vuelo. Así mismo, no encuentra a los que no pertenecen a la lista. Como, por ejemplo, cuando se le pasa el nombre de una destino erróneo.
AirportScreenTest	testSearchByGate	setupScenary3	Gate = 0	El método efectivamente encuentre a un vuelo que está en la lista dependiendo de la puerta de embarque. Así mismo, no encuentra a los que no pertenecen a la lista. Como, por ejemplo, cuando se le pasa una puerta inexistente.
AirportScreenTest	testSearchByFlightNumber	setupScenary3	airline = "non-existent"	El método efectivamente encuentre a un vuelo que está en la lista dependiendo de su número de vuelo. Así mismo, no encuentra a los que no pertenecen a la lista. Como, por ejemplo, cuando se le pasa un número de vuelo erróneo.

Objetivo de la Prueba: Verificar que los objetos son comparados de forma correcta entre ellos.				
Clase	Método	Escenario	Valores de Entrada	Resultado
FlightTest	testCompareTo	setupScenary2	flightNumber less= "Av1234" flightNumberHigher = "LO2545"	El método compara los objetos de forma correcta según el número de vuelo, retornando el valor de 1 cuando es mayor, -1 cuando es menor y 0 cuando son iguales.
DateTest	testCompareTo	setupScenary2	DD-MM-AAA dateless = (01,01,2010) dateHigher = (2,12,2018) dateEquals = (01,10,2011)	El método compara los objetos de forma correcta según el día, mes y año; retornando el valor de 1 cuando es mayor, -1 cuando es menor y 0 cuando son iguales.
FlightDateAndTimeComparatorTest	testCompare	setupScenary2	Los compara por fecha y hora one = (1,5,2015) les = (4,1,2014) higher = (7,12,2018)	El método compara los objetos de forma correcta según la fecha, retornando

				el valor de 1 cuando es mayor, -1 cuando es menor y 0 cuando son iguales.
--	--	--	--	---------------------------------------------------------------------------

Trazabilidad de análisis al diseño

Requerimiento funcional	Método	Clase
R1. Mostrar información de los vuelos.	Initialize() : void generateColumns(Tableview tb) : void previousList(ActionEvent event) : void nextList(ActionEvent event) : void getFirstFlight() : Flight getNextFlight() : Flight getPrevFlight() : Flight toString() : String showFlights(int size)	ScreenController ScreenController ScreenController ScreenController AirportScreen Flight Flight Date ScreenController
R2. Generar listado de vuelos	initialize() : void generateColumns(Tableview tb) : void generateList(ActionEvent event) : void generateList(int size) : void getFirstFlight() : Flight getNextFlight() : Flight getPrevFlight() : Flight loadArchive(String path) random(int n, String[] airlines, String[] destination) : Flight addFlight(Flight newFlight) : void	ScreenController ScreenController ScreenController AirportScreen Flight Flight AirportScreen AirportScreen AirportScreen

R3. Ordenar vuelos	sort(ActionEvent event) :void getFirstFlight() : Flight getNextFlight() : Flight getPrevFlight() : Flight setFirstFlight(Flight first) : void setNextFlight(Flight next) : void setPrevFlight(Flight prev) : void sortingByDateAndTime() : void sortingByAirline() : void sortingByFlightNumber() : void sortingByDestination() : void sortingByBoardingGate() : void compareTo() : int compareTo() : int getBoardingGate() : int getAirline() : String getFlightNumber() : String getDate() : Date getDepartureTime() : String getDestination() : String	ScreenController AirportScreen Flight Flight Flight Flight Flight AirportScreen AirportScreen AirportScreen AirportScreen AirportScreen Date Flight Flight Flight Flight Flight Flight Flight
R4. Buscar vuelo	searchFlight(ActionEvent event) : void searchByDate(int year, int month, int day) : Flight searchByTime(String timeKey) : Flight searchByAirline(String airlineKey) : Flight searchByDestination(String destinationKey) : Flight searchByGate(int gateKey) : Flight searchByFlightNumber(String numberKey) : Flight getDate() : Date compareTo() : int compareTo() : int getBoardingGate() : int getAirline() : String getFlightNumber() : String getDepartureTime() : String getDestination() : String getFirstFlight() : Flight getNextFlight() : Flight	ScreenController AirportScreen AirportScreen AirportScreen AirportScreen AirportScreen AirportScreen Flight Flight Date Flight Flight Flight Flight Flight Flight Flight Flight Flight
<u>R5. Mostrar tiempo</u>	<u>alertTime(long time) : void</u>	<u>ScreenController</u>