**Requerimientos funcionales**

|  |  |
| --- | --- |
| Nombre | RF#1 Mostrar la información de los vuelos |
| Resumen | Mostrar la información de un vuelo correspondiente a: hora de salida, aerolínea a la que pertenece el vuelo, código del vuelo, destino y puerta de embarque. |
| Entradas | Ninguna |
| Salidas | Información de los vuelos |

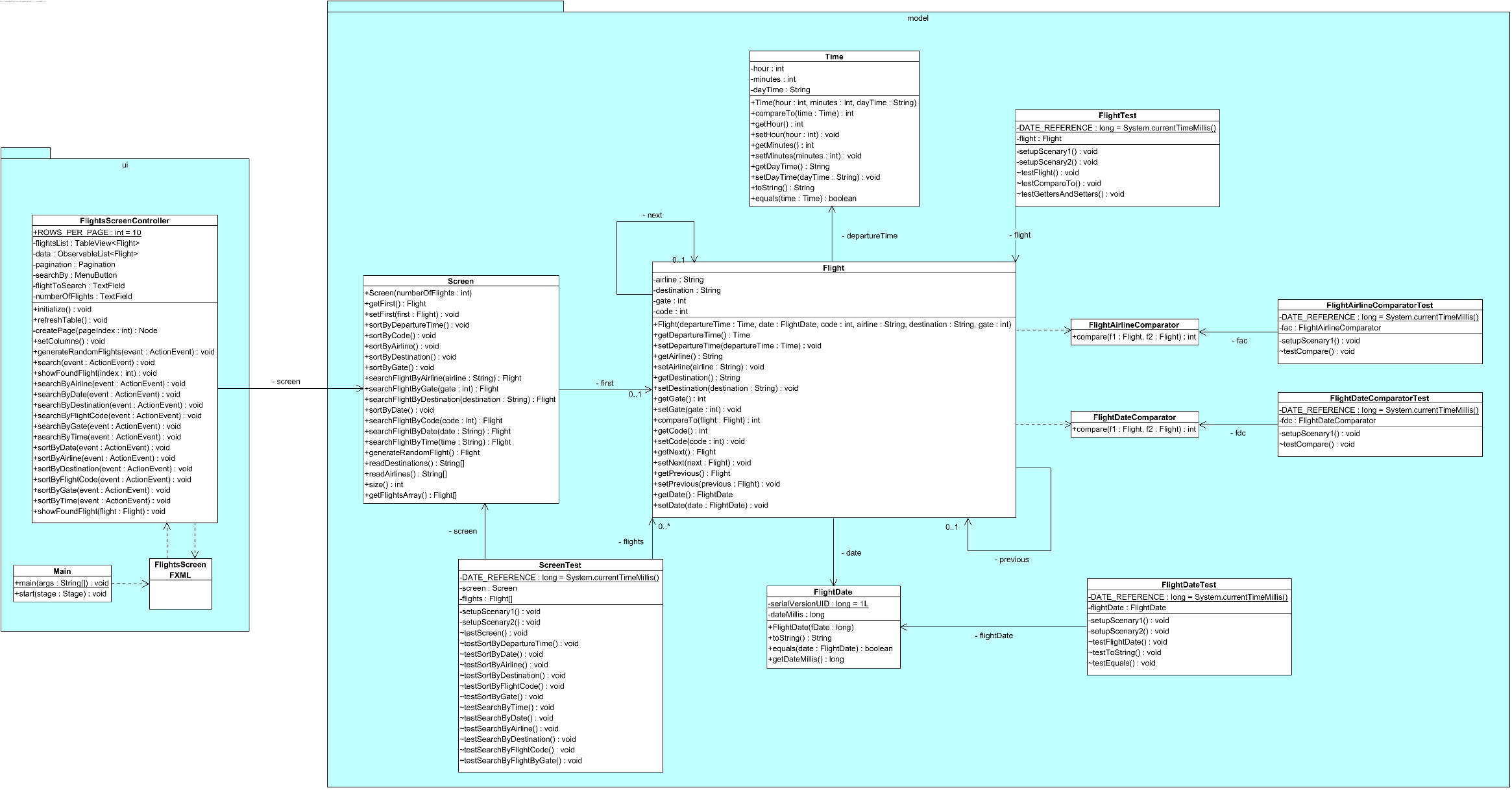
|  |  |
| --- | --- |
| Nombre | RF#2 Generar un listado aleatorio de vuelos |
| Resumen | Se debe poder generar un aleatorio de vuelos con información aleatoria. Por defecto el programa muestra un listado aleatorio y se debe poder generar uno nuevo cada vez que el usuario lo pida. |
| Entradas | Número de vuelos |
| Salidas | Listado nuevo de vuelos |

|  |  |
| --- | --- |
| Nombre | RF#3 Buscar un vuelo |
| Resumen | Se debe poder buscar un vuelo por un criterio escogido por el usuario, ya sea por fecha, por hora de salida, por número de vuelo, por ciudad de destino y por puerta de embarque. Si hay más de un vuelo que concuerde con el criterio buscado, entonces se muestra el primer vuelo encontrado. |
| Entradas | Se define según cada criterio de búsqueda. |
| Salidas | Si la entrada para cada criterio de vuelo concuerda con al menos un vuelo, se muestra el primer vuelo encontrado, de lo contrario se informa al usuario que no hay un vuelo correspondiente a la entrada y criterio de búsqueda escogidos. |

|  |  |
| --- | --- |
| Nombre | RF#4 Ordenar los vuelos |
| Resumen | Se debe poder ordenar los vuelos por los diferentes criterios (hora de salida, nombre de aerolínea, código de vuelo, destino y puerta de desembarque), el criterio debe ser escogido por el usuario. Por defecto el programa ordena los vuelos por hora de salida. |
| Entradas | Criterio de ordenamiento |
| Salidas | Vuelos ordenados |

**Requerimientos no funcionales**

|  |  |
| --- | --- |
| Nombre | RNF#1 Uso de JavaFX para la interfaz gráfica. |
| Resumen | Se hace uso de las clases de JavaFX y el software de SceneBuilder para construir la interfaz gráfica que se muestra al usuario. |

**Diagrama de clases**

**Trazabilidad**

|  |  |  |
| --- | --- | --- |
| **Requerimiento** | **Método(s)** | **Clase(s)** |
| Mostrar la información de los vuelos | setColumns  createPage  refreshTable  initialize | FlightsScreenController |
| Generar un listado aleatorio de vuelos | generateRandomFlights  generateRandomFlight | FlightsScreenController  Screen |
| Buscar un vuelo | search  searchBy\*  searchBy\* | FlightsScreenController  Screen |
| Ordenar los vuelos | sortBy\*  sortBy\* | FlightsScreenController  Screen |

**Escenarios**

Ilustración 1

: Time

: Flight

* code = 50
* airline = “Avianca”
* destination = “Pereira”
* gate = 5
* hour = 3
* minutes = 15
* daytime = “AM”

: FlightDate

* dateMillis = System.currentTimeMillis

Ilustración 2

: Time

: Flight

* code = 50
* airline = “SATENA”
* destination = “Pereira”
* gate = 5
* hour = 3
* minutes = 15
* daytime = “AM”

: FlightDate

* dateMillis = System.currentTimeMillis + 50000000L

Ilustración 3

: Screen

first

Next

Next4

: Flight

: Flight

* code = 200
* airline = “SATENA”
* destination = “Medellín”
* gate = 9

: Flight

* code = 55
* airline = “Avianca”
* destination = “Manizales”
* gate = 7

prev

prevv

* code = 30
* airline = “Wingo”
* destination = “Bogotá”
* gate = 4

: FlightDate

* dateMillis = System.currentTimeMillis + 6000L
* dateMillis = System.currentTimeMillis + 50000000L

: FlightDate

* dateMillis = System.currentTimeMillis

: FlightDate

* hour = 6
* minutes = 30
* daytime = “AM”

: Time

* hour = 4
* minutes = 30
* daytime = “AM”

: Time

* hour = 5
* minutes = 30
* daytime = “AM”

: Time

|  |  |  |
| --- | --- | --- |
| **Nombre** | **Clase** | **Escenario** |
| setupScenary1 | Flight | vacío |
| setupScenary2 | Flight | Ilustración 1 |
| setupScenary1 | Screen | vacío |
| setupScenary2 | Screen | Ilustración 3 |

**Pruebas**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Objetivo de la prueba:** Probar que el constructor asigna los valores correctamente a los atributos | | | | |
| **Clase** | **Método** | **Escenario** | **Valores de entrada** | **Resultado** |
| Flight | Flight | setupScenary1 | Objeto Time (3,15,”AM”) , Objeto FlightDate (System.currentMillis()), 50, “Avianca”, “Pereira”, 5 | Los atributos se inicializan correctamente. |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Objetivo de la prueba:** Probar que el método compareTo hace bien el trabajo de comparar a la instancia actual con el objeto tipo Flight que se pasa como parámetro, con el criterio por defecto que es el de departure time (hora de salida). | | | | |
| **Clase** | **Método** | **Escenario** | **Valores de entrada** | **Resultado** |
| Flight | compareTo | setupScenary2 | 3 objetos tipo flight con los siguientes atributos  [Lupa](https://correoicesiedu-my.sharepoint.com/personal/jesus_villota1_correo_icesi_edu_co/Documents/Tercer%20Semestre/APO%20II/Lab%204/flights-screen/Objetos%20prueba%20compareTo.PNG) | Los objetos se comparan correctamente. |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Objetivo de la prueba:** Probar que los métodos get devuelven los valores de los atributos correctamente y los métodos set modifican los valores de sus respectivos atributos. | | | | |
| **Clase** | **Método** | **Escenario** | **Valores de entrada** | **Resultado** |
| Flight | get\*  set\* | setupScenary2 | Se modifican los valores de los atributos de la instancia de Flight de la siguiente manera:  [Lupa](https://correoicesiedu-my.sharepoint.com/personal/jesus_villota1_correo_icesi_edu_co/Documents/Tercer%20Semestre/APO%20II/Lab%204/flights-screen/Atributos%20modificados%20prueba%20getters%20y%20setters.PNG) | Los métodos get devuelven los valores de los atributos que han sido previamente modificados con los métodos set. |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Objetivo de la prueba:** Probar que el constructor asigna correctamente los valores pasados como parámetros a los atributos de la instancia. | | | | |
| **Clase** | **Método** | **Escenario** | **Valores de entrada** | **Resultado** |
| Screen | Screen | setupScenary1 | Número de vuelos a crear: 3 | La instancia de la clase Flight se crea correctamente y sin errores, se crean aleatoriamente 3 vuelos y se agregan a la lista doblemente enlazada de vuelos que está como atributo de la clase. |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Objetivo de la prueba:** Probar que los métodos de búsqueda devuelven una referencia al objeto dentro de la lista que corresponde al primer objeto tipo Flight según el criterio de búsqueda. | | | | |
| **Clase** | **Método** | **Escenario** | **Valores de entrada** | **Resultado** |
| Screen | searchByTime | setupScenary2 | “5:30 AM” | El método devuelve un índice que referencia al objeto que corresponde al primer objeto encontrado que corresponde al criterio de hora de salida. |
| Screen | searchByDate | setupScenary2 | “Apr 12 2019” | El método devuelve un índice que referencia al objeto que corresponde al primer objeto encontrado que corresponde al criterio de fecha del vuelo. |
| Screen | searchByAirline | setupScenary2 | “Avianca” | El método devuelve una referencia al objeto que corresponde al primer objeto encontrado que corresponde al criterio de aerolínea. |
| Screen | searchByDestination | setupScenary2 | “Medellín” | El método devuelve una referencia al objeto que corresponde al primer objeto encontrado que corresponde al criterio de ciudad de destino. |
| Screen | searchByCode | setupScenary2 | 30 | El método devuelve una referencia al objeto que corresponde al primer objeto encontrado que corresponde al criterio de código de vuelo. |
| Screen | searchByGate | setupScenary2 | 7 | El método devuelve una referencia al objeto que corresponde al primer objeto encontrado que corresponde al criterio de puerta de embarque. |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Objetivo de la prueba:** Probar que los métodos de ordenamiento sitúan a cada objeto del arreglo de la instancia de la clase Screen en el lugar adecuado según sus respectivos criterios. | | | | |
| **Clase** | **Método** | **Escenario** | **Valores de entrada** | **Resultado** |
| Screen | sortByDepartureTime | setupScenary2 | Ninguno | El método ordena de manera correcta y ascendentemente a cada objeto de la lista de vuelos de acuerdo con el criterio de hora de salida. |
| Screen | sortByDate | setupScenary2 | Ninguno | El método ordena de manera correcta y ascendentemente a cada objeto de la lista de vuelos de acuerdo con el criterio de fecha de vuelo. |
| Screen | sortByAirline | setupScenary2 | Ninguno | El método ordena de manera correcta y ascendentemente a cada objeto de la lista de vuelos de acuerdo con el criterio de aerolínea. |
| Screen | sortByDestination | setupScenary2 | Ninguno | El método ordena de manera correcta y ascendentemente a cada objeto de la lista de vuelos de acuerdo con el criterio de ciudad de destino. |
| Screen | sortByCode | setupScenary2 | Ninguno | El método ordena de manera correcta y ascendentemente a cada objeto de la lista de vuelos de acuerdo con el criterio de código de vuelo. |
| Screen | sortByGate | setupScenary2 | Ninguno | El método ordena de manera correcta y ascendentemente a cada objeto de la lista de vuelos de acuerdo con el criterio de puerta de embarque. |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Objetivo de la prueba:** Probar que los métodos “compare” de las clases comparadoras retornan un valor adecuado para cada caso. | | | | |
| **Clase** | **Método** | **Escenario** | **Valores de entrada** | **Resultado** |
| FlightAirlineComparator | compare | setupScenary1 | [Ilustración 1](#Ilustración1)  [Ilustración 2](#Ilustración2) | El método retorna un valor menor que 0 ya que al compararse por una cadena de texto “Avianca” representa un valor menor que “SATENA” |
| FlightDateComparator | compare | setupScenary1 | [Ilustración 1](#Ilustración1)  [Ilustración 2](#Ilustración2) | El método retorna un valor menor que 0 ya que se compara por una fecha entonces la fecha del objeto de la [Ilustración 1](#Ilustración1) debe ser menor que la del objeto de la [Ilustración 2](#Ilustración2) |