**Requerimientos Funcionales**

|  |  |
| --- | --- |
| Nombre | R1: Mostrar la hora en formato AM/PM |
| Descripción | Muestra la hora actual y de salida de un vuelo en formato AM/PM, dado que es más fácil de entender para el usuario. |
| Entradas | Ninguna. |
| Resultados | Se muestra en pantalla la hora en el formato AM/PM. |

|  |  |
| --- | --- |
| Nombre | R2: Generar aleatoriamente el listado de vuelos |
| Descripción | Se generan los datos de cada vuelo de manera aleatoria. Estos son:   * Fecha * Horario * Aerolínea * Numero de vuelo (único) * Ciudad destino * Puerta de embarque * Estado |
| Entradas | Número de vuelos |
| Resultados | Se han generado los datos de cada vuelo de manera aleatoria. |

|  |  |
| --- | --- |
| Nombre | R4: Avanzar página en el listado de vuelos. |
| Descripción | Avanza y muestra la siguiente página del listado de vuelos |
| Entradas | Ninguna. |
| Resultados | Se avanzó a la siguiente página del listado de vuelos. |

|  |  |
| --- | --- |
| Nombre | R5: Devolver página en el listado de vuelos. |
| Descripción | Devuelve y muestra la página anterior en el listado de vuelos. |
| Entradas | Ninguna. |
| Resultados | Se retrocedió a la página anterior del listado de vuelos. |

|  |  |
| --- | --- |
| Nombre | R3: Ordenar la lista de vuelos de manera natural con Comparable |
| Descripción | Ordena el listado de vuelos en el orden natural, es decir, desde la menor a la mayor hora de salida de cada vuelo. |
| Entradas | Ninguna. |
| Resultados | Se ha ordenado el listado de vuelos en el orden natural, de menor a mayor hora y de salida. |

|  |  |
| --- | --- |
| Nombre | R7: Ordenar la lista de vuelos por fecha con Burbuja |
| Descripción | Ordena la lista de vuelos por el criterio de fecha, de menor a mayor, utilizando el algoritmo de ordenamiento clásico burbuja. |
| Entradas | Ninguna. |
| Resultados | Se ha ordenado la lista de vuelos por la fecha, con el método de burbuja. |

|  |  |
| --- | --- |
| Nombre | R7: Ordenar la lista de vuelos por aerolínea con Selección |
| Descripción | Ordena la lista de vuelos por el criterio de nombre de aerolínea, de menor a mayor, utilizando el algoritmo de ordenamiento clásico selección. |
| Entradas | Ninguna. |
| Resultados | Se ha ordenado la lista de vuelos por la aerolínea, con el método de selección. |

|  |  |
| --- | --- |
| Nombre | R7: Ordenar la lista de vuelos por número de vuelo con Comparator |
| Descripción | Ordena la lista de vuelos por el criterio número de vuelo, utilizando la interface Comparable. |
| Entradas | Ninguna. |
| Resultados | Se ha ordenado la lista de vuelos por el número de vuelo, con la interface Comparable. |

|  |  |
| --- | --- |
| Nombre | R7: Ordenar la lista de vuelos por ciudad de destino con inserción |
| Descripción | Ordena la lista de vuelos por el criterio de ciudad de destino, de menor a mayor, utilizando el algoritmo de ordenamiento clásico de inserción. |
| Entradas | Ninguna. |
| Resultados | Se ha ordenado la lista de vuelos por la ciudad de destino con el método de inserción. |

|  |  |
| --- | --- |
| Nombre | R7: Ordenar la lista de vuelos por puerta de embarque con Comparator de Arrays |
| Descripción | Ordena la lista de vuelos por el criterio de puerta de embarque, de menor a mayor, utilizando el método sort(\*) de la clase Arrays, con la interface Comparator |
| Entradas | Ninguna. |
| Resultados | Se ha ordenado la lista de vuelos por puerta de embarque con la clase Arrays |

|  |  |
| --- | --- |
| Nombre | R7: Ordenar la lista de vuelos por estado con Comparable de Arrays |
| Descripción | Ordena la lista de vuelos por el criterio de estado, de menor a mayor, utilizando el método sort(\*) de la clase Arrays, con la interface Comparable |
| Entradas | Ninguna. |
| Resultados | Se ha ordenado la lista de vuelos por la puerta de embarque, con la clase Arrays. |

|  |  |
| --- | --- |
| Nombre | R6: Buscar un vuelo con búsqueda secuencial. |
| Descripción | Busca un vuelo de forma secuencial por cualquiera de sus criterios de búsqueda. Los criterios son:   * Fecha * Horario * Aerolínea * Numero de vuelo (único) * Ciudad destino * Puerta de embarque * Estado |
| Entradas | Un criterio de búsqueda, un valor a buscar. |
| Resultados | Se ha buscado de forma secuencial el vuelo solicitado a través del criterio de búsqueda ingresado. |

|  |  |
| --- | --- |
| Nombre | R7: Buscar un vuelo con búsqueda binaria. |
| Descripción | Busca un vuelo con búsqueda binaria por cualquiera de sus criterios de búsqueda. Los criterios son:   * Fecha * Horario * Aerolínea * Numero de vuelo (único) * Ciudad destino * Puerta de embarque * Estado |
| Entradas | Una cadena el caracteres que contiene el criterio de búsqueda. |
| Resultados | Se ha buscado de forma binaria el vuelo solicitado a través del criterio de búsqueda ingresado. |

|  |  |
| --- | --- |
| Nombre | R7: Calcular tiempo de búsqueda |
| Descripción | Calcula el tiempo que se tarda en realizar una búsqueda de un vuelo a partir de un criterio |
| Entradas | Ninguna. |
| Resultados | Se ha calculado en tiempo de búsqueda de un vuelo a partir de un criterio. |

|  |  |
| --- | --- |
| Nombre | R7: |
| Descripción |  |
| Entradas |  |
| Resultados |  |

|  |  |
| --- | --- |
| Nombre | R7: |
| Descripción |  |
| Entradas |  |
| Resultados |  |

**Tabla de Trazabilidad**

|  |  |  |
| --- | --- | --- |
| Clase | Requerimiento | Método |
|  |  |  |
|  |  |  |
|  |  |  |

**Diseño de Casos de Pruebas Unitarias Automáticas**

**Configuración de escenarios**

|  |  |  |
| --- | --- | --- |
| Clase | Nombre | Escenario |
| TestFligth | setupScenary1() | La clase Fligth es nula |
|  | setupScenary2() | La clase Fligth se ha instanciado con los siguientes datos.  date = “2019-07-23”  time=”04:34 AM”  airline=”AVIANCA”  fligth=”AV 3213”  city=”Madrid”  gate=6  state=”Exit” |
|  | setupScenary3() |  |
| TestAirport | setupScenary1() | La clase Airport es nula |
|  | setupScenary2() | La clase Airport se ha instanciado |
|  | setupScenary3() |  |
|  |  |  |

**Casos de pruebas unitarias automáticas**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Objetivo de la prueba 1: Verificar que la clase Flight es nula y que no se podrán realizar operaciones sobre ella. | | | | |
| Clase | **Método** | **Escenario** | **Valores de entrada** | **Resultados** |
| Fligth | testFligth2() | setupScenary1() | Ninguna | La clase es nula. |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Objetivo de la prueba 2: Verificar que se asignan correctamente los valores a cada atributo de la clase Fligth | | | | |
| Clase | **Método** | **Escenario** | **Valores de entrada** | **Resultados** |
| Fligth | testFligth2() | setupScenary2() | Ninguna | Los valores se asignan correctamente a sus atributos correspondientes. |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Objetivo de la prueba 3: Verificar que la clase Airport es nula y que no se podrán realizar operaciones sobre ella. | | | | |
| Clase | **Método** | **Escenario** | **Valores de entrada** | **Resultados** |
| Airport | testAirport1() | setupScenary1() | Ninguna | La clase Airport es nula. |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Objetivo de la prueba 4: Verificar que se leen y cargan correctamente las fechas de los vuelos | | | | |
| Clase | **Método** | **Escenario** | **Valores de entrada** | **Resultados** |
| Airport | testLoadDates() | setupScenary2() | String = “data/test-dates.txt” | Las fechas se leen y cargar correctamente |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Objetivo de la prueba 5: Verificar que se leen y cargar correctamente los horarios de los vuelos. | | | | |
| Clase | **Método** | **Escenario** | **Valores de entrada** | **Resultados** |
| Airport | testLoadTimes() | setupScenary2() | String = “data/test-times.txt” | Los horarios se leen y se cargan correctamente. |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Objetivo de la prueba 6: Verificar que se leen y cargar correctamente las aerolíneas de los vuelos. | | | | |
| Clase | **Método** | **Escenario** | **Valores de entrada** | **Resultados** |
| Airport | testLoadAirlines() | setupScenary2() | String = “data/test-airlines.txt” | Las aerolíneas se leen y se cargan correctamente. |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Objetivo de la prueba 7: Verificar que se leen y cargan correctamente las ciudades de destino de los vuelos. | | | | |
| Clase | **Método** | **Escenario** | **Valores de entrada** | **Resultados** |
| Airport | testLoadCities() | setupScenary2() | String = “data/test-cities.txt” | Las ciudades se leen y se cargan correctamente. |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Objetivo de la prueba 8: Verificar que se leen y cargan correctamente los estados de los vuelos. | | | | |
| Clase | **Método** | **Escenario** | **Valores de entrada** | **Resultados** |
| Airport | testLoadStates() | setupScenary2() | String = “data/test-states.txt” | Los estados de los vuelos se leen y se cargan correctamente. |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Objetivo de la prueba 9: Verificar que se genera correctamente la lista de vuelos aleatorios. | | | | |
| Clase | **Método** | **Escenario** | **Valores de entrada** | **Resultados** |
| Airport | testGenerateRandomFligths(int) | setupScenary2() | numberFligths = 20 | La lista de vuelos aleatorios se crea correctamente. |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Objetivo de la prueba 10: Verificar que se crea correctamente la lista observable de vuelos. | | | | |
| Clase | **Método** | **Escenario** | **Valores de entrada** | **Resultados** |
| Airport | testCreateObservableFligths() | setupScenary2() | Ninguna | La lista observable de vuelos se creó correctamente. |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Objetivo de la prueba 11: Verificar que los vuelos se ordenan correctamente por orden natural. | | | | |
| Clase | **Método** | **Escenario** | **Valores de entrada** | **Resultados** |
| Airport | testSortBynaturalOrder() | setupScenary2() | Ninguna | La lista de vuelos se ordena por orden natural correctamente. |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Objetivo de la prueba 12: Verificar que los vuelos se ordenan correctamente por fechas. | | | | |
| Clase | **Método** | **Escenario** | **Valores de entrada** | **Resultados** |
| Airport | testSortByDate() | setupScenary2() | Ninguna | La lista de vuelos se ordena por fechas correctamente. |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Objetivo de la prueba 13: Verificar que los vuelos se ordenan correctamente por aerolíneas. | | | | |
| Clase | **Método** | **Escenario** | **Valores de entrada** | **Resultados** |
| Airport | testSortByAirline() | setupScenary2() | Ninguna | La lista de vuelos se ordena por aerolíneas correctamente. |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Objetivo de la prueba 14: Verificar que los vuelos se ordenan correctamente por código de vuelo | | | | |
| Clase | **Método** | **Escenario** | **Valores de entrada** | **Resultados** |
| Airport | testSortByFligth() | setupScenary2() | Ninguna | La lista de vuelos se ordena por código de vuelo correctamente. |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Objetivo de la prueba 15: Verificar que los vuelos se ordenan correctamente por ciudad de destino. | | | | |
| Clase | **Método** | **Escenario** | **Valores de entrada** | **Resultados** |
| Airport | testSortByCity() | setupScenary2() | Ninguna | La lista de vuelos se ordena por ciudad de destino correctamente. |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Objetivo de la prueba 16: Verificar que los vuelos se ordenan correctamente por número de puerta de embarque. | | | | |
| Clase | **Método** | **Escenario** | **Valores de entrada** | **Resultados** |
| Airport | testSortByGate() | setupScenary2() | Ninguna | La lista de vuelos se ordena por numero de puerta de embarque correctamente. |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Objetivo de la prueba 17: Verificar que los vuelos se ordenan correctamente por el estado del vuelo | | | | |
| Clase | **Método** | **Escenario** | **Valores de entrada** | **Resultados** |
| Airport | testSortByState() | setupScenary2() | Ninguna | La lista de vuelos se ordena por el estado del vuelo correctamente. |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Objetivo de la prueba 18: Verificar que dados un criterio de búsqueda y un valor a buscar, se busca mediante búsqueda secuencial correctamente un vuelo. | | | | |
| Clase | **Método** | **Escenario** | **Valores de entrada** | **Resultados** |
| Airport | testSearchBySequentialSearch(String,String) | setupScenary2() | criteria=”DATE”  valueSearch= | Se encontró el vuelo buscado por fecha. |
|  |  |  | criterio=”DATE”  valueSearch=” | El vuelo buscado por fecha no existe |
|  |  |  |  |  |
|  |  |  | criteria=”TIME”  valueSearch=” | Se encontró el vuelo buscado por hora. |
|  |  |  | criterio=”TIME”  valueSearch | El vuelo buscado por hora no existe. |
|  |  |  |  |  |
|  |  |  | criteria=”AIRLINE”  valueSearch | Se encontró el vuelo buscado por aerolínea. |
|  |  |  | criterio=”AIRLINE”  valueSearch=” | El vuelo buscado por aerolínea no existe. |
|  |  |  |  |  |
|  |  |  | criteria=”FLIGTH”  valueSearch | Se encontró el vuelo buscado por código de vuelo. |
|  |  |  | criterio=”FLIGTH”  valueSearch=” | El vuelo buscado por código de vuelo no existe. |
|  |  |  |  |  |
|  |  |  | criteria=”CITY”  valueSearch= | Se encontró el vuelo buscado por ciudad. |
|  |  |  | criterio=”CITY”  valueSearch= | El vuelo buscado por ciudad no existe. |
|  |  |  |  |  |
|  |  |  | criteria=”GATE”  valueSearch= | Se encontró el vuelo buscado por puerta de embarque. |
|  |  |  | criterio=”GATE”  valueSearch= | El vuelo buscado por puerta de embarque no existe. |
|  |  |  |  |  |
|  |  |  | criteria=”STATE”  valueSearch=” | Se encontró el vuelo buscado por estado. |
|  |  |  | criterio=”STATE”  valueSearch=” | El vuelo buscado por estado no existe. |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Objetivo de la prueba 19: Verificar que dados un criterio de búsqueda y un valor a buscar, se busca mediante búsqueda binaria correctamente un vuelo. | | | | |
| Clase | **Método** | **Escenario** | **Valores de entrada** | **Resultados** |
| Airport | testSearchBySequentialSearch(String,String) | setupScenary2() | criteria=”DATE”  valueSearch=” | Se encontró el vuelo buscado por fecha. |
|  |  |  | criterio=”DATE”  valueSearch=” | El vuelo buscado por fecha no existe |
|  |  |  |  |  |
|  |  |  | criteria=”TIME”  valueSearch=” | Se encontró el vuelo buscado por hora. |
|  |  |  | criterio=”TIME”  valueSearch=” | El vuelo buscado por hora no existe. |
|  |  |  |  |  |
|  |  |  | criteria=”AIRLINE”  valueSearch=” | Se encontró el vuelo buscado por aerolínea. |
|  |  |  | criterio=”AIRLINE”  valueSearch=” | El vuelo buscado por aerolínea no existe. |
|  |  |  |  |  |
|  |  |  | criteria=”FLIGTH”  valueSearch=” | Se encontró el vuelo buscado por código de vuelo. |
|  |  |  | criterio=”FLIGTH”  valueSearch=” | El vuelo buscado por código de vuelo no existe. |
|  |  |  |  |  |
|  |  |  | criteria=”CITY”  valueSearch=” | Se encontró el vuelo buscado por ciudad. |
|  |  |  | criterio=”CITY”  valueSearch=” | El vuelo buscado por ciudad no existe. |
|  |  |  |  |  |
|  |  |  | criteria=”GATE”  valueSearch=” | Se encontró el vuelo buscado por puerta de embarque. |
|  |  |  | criterio=”GATE”  valueSearch=” | El vuelo buscado por puerta de embarque no existe. |
|  |  |  |  |  |
|  |  |  | criteria=”STATE”  valueSearch=” | Se encontró el vuelo buscado por estado. |
|  |  |  | criterio=”STATE”  valueSearch=” | El vuelo buscado por estado no existe. |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Objetivo de la prueba 1: Verificar que se calcula correctamente el tiempo de búsqueda. | | | | |
| Clase | **Método** | **Escenario** | **Valores de entrada** | **Resultados** |
|  |  |  |  |  |