|  |  |  |  |
| --- | --- | --- | --- |
| A description... | | Diseñadores | Camilo Vargas Cabrera |
| **Análisis y Diseño Detallado** | | | |
| **Requerimientos funcionales** | | | |
| **RF 1:**  **Inicializar el sistema con la información suministrada** | **Resumen: Parser archivo CSV y generación del grafo** | | |
| **Entradas: Archivo CSV con la información de los vuelos** | | |
| **Resultados: Se ha inicializado el grafo con la información de los vuelos** | | |
| **RF 2: Agregar un nuevo aeropuerto al sistema** | **Resumen: Registra un nuevo vértice en el grafo** | | |
| **Entradas: El nombre del aeropuerto** | | |
| **Resultados: Se ha agregado un nuevo vértice en el grafo** | | |
| **RF 3: Agregar una nueva ruta de vuelo** | **Resumen: Registra un nuevo vuelo en el sistema. Si ya existe la ruta que cubre dicho vuelo, lo agrega a la ruta existente. Si la ruta no existe la crea en base a la información de origen y destino del vuelo** | | |
| **Entradas: Información del vuelo** | | |
| **Resultados: Se ha agregado un nuevo vuelo a la ruta, si la ruta no existía se ha agregado un nuevo arco al grafo.** | | |
| **RF 4: Buscar la ruta más corta** | **Resumen: Despliega la información de los vuelos a tomar tales que se minimiza el tiempo de viaje** | | |
| **Entradas: La cuidad de origen, de destino, el día y la hora aproximada de salida** | | |
| **Resultados: Si el camino existe, se despliega la información de los vuelos a tomar tales que minimizan el tiempo de viaje.** | | |
| **RF 5: Circuito más largo sin repetir ciudades** | **Resumen: Desplegar la secuencia más larga de ciudades que se puede recorrer desde una ciudad dada, sin repetir ciudades** | | |
| **Entradas: La ciudad de origen** | | |
| **Resultados: Se despliega la información de los vuelos a tomar para realizar el circuito** | | |
| **RF 6: Ciclo más largo sin repetir ciudades** | **Resumen: Desplegar el ciclo más largo de ciudades que se puede recorrer desde una ciudad dada, sin repetir ciudades** | | |
| **Entradas: La ciudad de origen** | | |
| **Resultados: Se despliega la información de los vuelos a tomar para realizar el ciclo** | | |
|  | **Resumen:** | | |
| **Entradas: N/A** | | |
| **Resultados:** | | |
|  | **Resumen:** | | |
| **Entradas:N/A** | | |
| **Resultados:** | | |
|  | **Resumen:** | | |
| **Entradas: N/A** | | |
| **Resultados:** | | |

|  |
| --- |
| **Análisis del modelo del mundo** |
| A description... |

|  |  |  |  |
| --- | --- | --- | --- |
| **Diseño de Operaciones (métodos): Para las interfaces relevantes en el diseño** | | | |
| AeroCupi | **public** **void** cargarInfoVuelosCSV(); | | Precondición: El archivo con la información de los vuelos existe |
| Postcondición: Se ha inicializado el grafo |
| Excepciones: N/A |
| AeroCupi | **public** Aeropuerto agregarAeropuerto(String nombreAeropuerto) **throws** ElementoExisteException; | | Precondición: Se ha recuperado la instancia del mundo |
| Postcondición: Se ha agregado un nuevo vértice al grafo |
| Excepciones: En caso de ya existir otro aeropuerto con el mismo nombre |
| AeroCupi | **public** **void** registrarVuelo(String aerolinea, String numero, String origen, String destino,String salida, String llegada, String equipo, String sillas, **boolean**[] disponibilidad) | | Precondición: Se ha recuperado la instancia del mundo |
| Postcondición: Se ha registrado un nuevo vuelo en el sistema, si la ruta no existía se ha agregado un nuevo arco al grafo |
| Excepciones: N/A |
| AeroCupi | **public** IListaEncadenada<String[]> darRutaMinima(String origen, String destino, String dia, String hora) | | Precondición: Se ha recuperado la instancia del mundo |
| Postcondición: Se genera una lista la información de los vuelos a tomar |
| Excepciones: N/A |
| AeroCupi |  | | Precondición: |
| Postcondición: |
| Excepciones: N/A |
| AeroCupi |  | | Precondición: Se ha recuperado la instancia del mundo |
| Postcondición: |
| Excepciones: N/A |
| AeroCupi |  | | Precondición: Se ha recuperado la instancia del mundo |
| Postcondición: |
| Excepciones: |
|  |  | |  |
|  |
|  |
|  |  | |  |
| **Diseño de Estructuras de Datos: Utilice un esquema de representación para ilustrar** | | | |
| A description... | | | |
| **Análisis de Complejidad : Para cada operación descrita en el diseño de operaciones, calcule y justifique la complejidad** | | | |
| cargarInfoVuelosCSV(); | | Donde n es el número de vuelos a cargar | |
| O(n) | |
| agregarAeropuerto() | | Solo se registra un nuevo vértice en el grafo | |
| O(1) | |
| registrarVuelo | | Se deben ubicar los vértices de origen y destino (O(1) + O(1)), generar el arco O(1) y agregar el vuelo al arco O(1) | |
| O(1) | |
| darRutaMinima() | | Donde n es el número de arcos en el grafo (Se utiliza el algoritmo de Dijkstra) | |
| O(n2) | |
|  | |  | |
| O() | |
|  | |  | |
| O() | |
|  | |  | |
| O() | |
|  | |  | |
| O() | |
|  | |  | |
|  | |
|  | |  | |
|  | |
|  | |  | |
|  | |

|  |
| --- |
| **Diagrama UML Final** |
|  |