

|  |
| --- |
| Documento de Análisis y Diseño |
| Nivel 13: [F1]  [Edgar Andrés Margffoy] |
| [Camila García] |
| Estructuras de Datos  Universidad de Los Andes  [2015-1] |
|  |

Documento de Análisis y Diseño

[Nombre del ejercicio]

|  |  |  |
| --- | --- | --- |
| **Versión del documento** | **Fecha de modificación** | **Responsables** |
| 1.0 | 18/02/15 | Edgar Andrés Margffoy  Camila García |
|  |  |  |
|  |  |  |

Contenido

[1 Requerimientos Funcionales 3](#_Toc409507633)

[2 Análisis del modelo del mundo 4](#_Toc409507634)

[3 Diseño de Interfaces 5](#_Toc409507635)

[3.1 Interfaces del mundo de la aplicación 5](#_Toc409507636)

[3.2 Interfaces de las estructuras de datos 5](#_Toc409507637)

[4 Diseño de estructuras de datos 6](#_Toc409507638)

[5 Justificación de decisiones de diseño de las estructuras de datos 7](#_Toc409507639)

[6 Diseño final de la aplicación 8](#_Toc409507640)

[7 Justificación de decisiones de diseño del mundo 9](#_Toc409507641)

[8 Análisis de complejidad de operaciones del mundo 10](#_Toc409507642)

[9 Mapa de Navegación (GUI) 11](#_Toc409507643)

# Requerimientos Funcionales

A continuación se describen los requerimientos funcionales de la aplicación. La definición de estos es independiente de la tecnología que será usada para implementarlos.

|  |  |
| --- | --- |
| **ID** | R1 |
| **Nombre** | Leer la información de vuelos para un mes/año específico |
| **Resumen** | El usuario define un mes de consulta y toda la información de los vuelos realizados en ese mes se guarda en el sistema. |
| **Entradas** | Mes de consulta |
| **Resultados** | Sistema contiene ahora información de vuelos |

|  |  |
| --- | --- |
| **ID** | R2 |
| **Nombre** | Consultar la información completa de un aeropuerto |
| **Resumen** | El usuario da el código de un aeropuerto y puede consultar su información completa. |
| **Entradas** | Código del aeropuerto |
| **Resultados** | Se muestra la información completa disponible. |

|  |  |
| --- | --- |
| **ID** | R3 |
| **Nombre** | Consultar la información completa de un vuelo en el mes de conslta |
| **Resumen** | Dado el código de un vuelo, el usuario puede ver su fecha, hora de salida y despegue programadas, y aeropuertos de origen y destino. |
| **Entradas** | Código del vuelo |
| **Resultados** | Se muestra la información de fecha, horas de despegue y aterrizaje programadas, y las ciudades y códigos de los aeropuertos. |

|  |  |
| --- | --- |
| **ID** | R4 |
| **Nombre** | Consultar la información de vuelos aterrizando en un aeropuerto e un día y periodo de tiempo determinado |
| **Resumen** | El usuario escoge un aeropuerto, una fecha, y las horas de inicio y fin de la consulta. El programa muestra todos los vuelos que aterrizaron ese día en ese aeropuerto entre los límites temporales definidos. |
| **Entradas** | Código aeropuerto. Fecha. Hora inciio, Hora fin. |
| **Resultados** | Información de los vuelos en ese lapso horario. |

|  |  |
| --- | --- |
| **ID** | R5 |
| **Nombre** | Eliminar la información de vuelos en el mes de consulta |
| **Resumen** | Elimina la información de un vuelo de los datos guardados en el sistema dado su código |
| **Entradas** | Código vuelo. |
| **Resultados** | Se elimina toda la información del vuelo. |

|  |  |
| --- | --- |
| **ID** | R6 |
| **Nombre** | Consultar vuelos directos entre dos aeropuertos en una fecha dada |
| **Resumen** | Permite saber la información de todos los vuelos directos entre dos aeropuertos en un día específico |
| **Entradas** | Fecha. Código aeropuerto origen. Código aeropuerto destino. |
| **Resultados** | Informaciñon dee los vuelos directos entre los dos aeropuertos en la fecha dada por parámetro. |

|  |  |
| --- | --- |
| **ID** | R7 |
| **Nombre** | Consultar los aeropuertos definidos en una zona geográfica |
| **Resumen** | Ver en un mapa la ubicación de los vuelos dados ciertos límites de latitud y longitud. |
| **Entradas** | Número máximo de vuelos a visualizar. Límites de longitud y latitud. |
| **Resultados** | Se muestra un mapa con la ubicación de los aeropuertos que stán dentro del límite establecido. |

|  |  |
| --- | --- |
| **ID** | R8 |
| **Nombre** | Comparar el tráfico de vuelos aterrizando entre múltiples aeropuertos |
| **Resumen** | Muestra una gráfica donde se puede comparar la cantidad de vuelos por aeropuerto, dado un máximo de aeropuertos. |
| **Entradas** | Máximo de aeropuertos a comparar. Códigos de aeropuertos. |
| **Resultados** | Se muestra una gráfica que permite visualizar la información. |

# Análisis del modelo del mundo

Diagrama de clases (UML) de la abstracción del problema que se pretende solucionar. Solo se tienen en cuenta entidades, características y relaciones relevantes para el mismo. Aún no se toman decisiones de diseño.

# Diseño de Interfaces

## Interfaces del mundo de la aplicación

A partir de los requerimientos funcionales, defina las operaciones (a través de métodos) que debe proveer el mundo del problema. Incluya parámetros, tipos de retorno y documentación completa de cada operación (incluyendo precondiciones).

cargarVuelos: Carga la información de vuelos. Llama a la clase Query, que hace la conexión cn la base de datos. Recibe un String con la información de vuelos a cargar.

cargarAeropuertos: Se encarga de leer los archivos de texto con la información de los aeropuertos y crear la lista de aeropuertos del mundo.

buscarVuelo : Retorna una lista de vuelos dados sus códigos. Si no encuentra, se devuelve lista vacía. Recibe el código del vuelo buscado por parámetro. Pre: la lista de vuelos está inicializada.

buscarVuelosDirecto: Retorna una lista con los vuelos directos entre dos aeropuertos. Recibe por parámetro los códigos de los aeropuertos de origen y destino. En caso de no haber, retorna una lista vacía.

buscarAeropuerto: busca un aeropuerto dado su código. Retorna el aeropuerto, de no encontrarlo, retorna null. Recibe por parámetro el código del aeropuerto buscado.

buscarVuelosPeriodoTiempo: recibe por parámetro una fecha, dos enteros representando la hora inicial y la hora final, y el código del aeropuerto. Retorna una lista de vuelos. En caso de no haber ninguno, devuelve lista vacía. La lista de aeropuertos y vuelos ya está inicializada.

eliminarVuelo: Elimina un vuelo del sistema. Recibe por parámetro el código del vuelo a eliminar. Retorna true si lo elimino, false si no lo encontró. La lista de vuelos esta inicializada.

darAeropuuertosLimitesGeográfico: Recibe por parámetro un máximo de aeropuertos a retornar, y las latitudes y longitudes máximas y mínimas. Devuelve una lista con máximo N elementos que cumplan las condiciones. De no encontrar ningún aeropuerto en los límites, devuelve una lista vacía.

darTraficoAeropuertos: Devuelve una lista se Strings que contienen el nombre y tráfico total de los aeropuerto solicitados. Recibe por parámetro los códigos de los aeropuertos solicitados.

## Interfaces de las estructuras de datos

Seleccione cuáles son las estructuras de datos genéricas que debe utilizar y las operaciones que estas deben proveer para minimizar la complejidad temporal de las operaciones definidas en el numeral 3.1.

Debemos utilizar la tabla de hash como estructura principal de datos. Sin embargo, utilizaremos también la lista simplemente enlazada y el arreglo dinámico implementados en el nivel anterior como apoyo a la tabla de hash. Para modelar la tabla de hash, utilizaremos el arreglo dinámico para guardar listas enlazadas de los elementos cuya llave apunte a esa dirección. Opinamos que es la manera más eficiente de manejar las colisiones en la tabla de hash, pues aunque la búsqueda es en teoría O(n), es poco probable que con una buena función de hash un gran porcentaje de los elementos terminen en la misma lista enlazada. Esto permite hacer búsquedas muy eficientes entre los vuelos, pues son demasiados dato y la tabla de hash reduce la búsqueda a O(1) en el caso promedio.

Las funcionalidades de el arreglo dinámico y la lista simplemente enlazada están dadas por la interfaz genérica de lista utilizada en el nivel anterior. Las principales funcionalidades de la tabla de hash serán la función de hash, que convierte las llaves en posiciones en el arreglo, añadir, que añade un elemento, buscar, que devuelve un elemento dada su llave y eliminar, que elimina un elemento dada su llave.

# Diseño de estructuras de datos

Describa el detalle de las Estructuras de Datos que va a utilizar en su solución, a través de un diagrama UML de las mismas. No olvide incluir invariantes si las requiere.

# Justificación de decisiones de diseño de las estructuras de datos

Describa en detalle los aspectos que tuvo en cuenta para seleccionar y diseñar las estructuras de datos descritas en el punto anterior.

# Diseño final de la aplicación

Describa en detalle el diseño final de su aplicación a través de un diagrama UML, incluya atributos y métodos. Cuando se refiera a una estructura de datos no es necesario que incluya toda su definición dentro del diagrama. Basta con referenciar su interfaz.

# Justificación de decisiones de diseño del mundo

Describa en detalle los aspectos que guiaron el diseño final de su aplicación.

De acuerdo a los objetivos propuestos en la solución del problema, fue fundamental diseccionar el funcionamiento del programa bajo dos ramas principales. La primera vertiente, corresponde al proceso de consulta en la base de datos principal del programa, esta funcionalidad se encuentra completamente desacoplada de las funcionalidades restantes, en la medida que no es necesario disponer y conocer de forma completa o parcial los mecanismos y procedimientos involucrados en la gestión posterior de la información por parte del segundo componente de la presente implementación. Es necesario agregar, que la conexión con la base de datos remota, permanece constante a lo largo del funcionamiento de la aplicación, lo cual permite evaluar algún inconveniente o error en tiempo de ejecución, así mismo, esta decisión se basa en el costo intrínseco al restablecimiento de una conexión remota a través de Internet y el tiempo de latencia asociado a una consulta específica. Adicionalmente, se pretende reducir el número de consultas y accesos requeridos, para este fin, se realizarán consultas de acuerdo a la selección previa de un mes particular y específico para el cual, es esencial recuperar la información correspondiente.

A continuación, es importante y pertinente describir el funcionamiento de las clases principales dispuestas para la gestión y organización de la información recuperada, así como la interacción de estas con la interfaz gráfica y el usuario en su entorno. De esta manera decidimos utilizar una clase principal, que extiende de una interface, con los métodos definidos en las interfaces del mundo y que permiten cumplir con los requerimientos del programa. Esta clase tiene las listas de aeropuertos y vuelos, que corresponden a las clases de Vuelo y Aeropuerto. Cada una de éstas tiene la información pertinente para cumplir con el buen funcionamiento del programa. El uso de la interface permite que la interfaz gráfica no tenga que involucrarse con el mundo, y que haya un alto desacoplamiento, permitiendo cambios futuros.

# Análisis de complejidad de operaciones del mundo

Basado en los diseños de las estructuras de datos y en el mundo de la aplicación, defina la complejidad de cada una de las operaciones. Utilice la notación O(f(n)). Justifique por qué considera que esta es la mínima posible para el problema que está resolviendo.

# Mapa de Navegación (GUI)

Basado en los requerimientos funcionales, defina un bosquejo general con los elementos que definen la interfaz gráfica (GUI) de la aplicación. Puede usar la herramienta *Balsamiq Mockups* (http://balsamiq.com/products/mockups/) u otra que permita una funcionalidad similar (NO requiere realizar la implementación de la Interfaz Gráfica)



