Pontificia Universidad Javeriana
Departamento de Ingeniería de Sistemas
Estructuras de Datos
Proyecto del Curso, 2018-3

Por Santiago Palacios, Julián Parada, Mónica Alvarez

Diseño del sistema centralizado para apoyo de agencias de viajes.

1. Diseños de los Tipos Abstractos de Datos. (TAD) a utilizar.

TAD Sistema

Conjunto mínimo de datos:

- empresas, lista de agencias, contiene la información de todas las agencias asociadas al sistema.
- vuelos, lista de vuelos, contiene la información de los vuelos con disponibilidad de venta que ofrecen las diferentes agencias.
- tiquetes, lista de tiquetes, contiene la información de los tiquetes vendidos por las diferentes agencias.

Operaciones:

login(lista de agencias, usuario, contraseña, agencia)

- Descripción: hace la autenticación correspondiente de una agencia en el sistema, verificando que el usuario y contraseña que llegan coincidan con un usuario y contraseña que estén en la lista de agencias cargadas al sistema
- Entradas: lista de agencias del sistema, usuario correspondiente al id de la agencia, contraseña de la agencia y la agencia del sistema donde guardara la agencia autenticada.
- Salidas: booleano que dice si la agencia se autentico o no.
- Condiciones: la lista de agencias ya debe estar cargada y no debe haber una agencia ya autenticada.

llenarListaEmpresas(nombre del archivo, lista de agencias)

- Descripción: lee la información del archivo que contiene la información completa de las agencias y las almacena en la lista de agencias del sistema.
- Entradas: nombre del archivo de agencias, lista de agencias del sistema.
- Salidas: booleano que dice si la lista de agencias del sistema se pudo llenar.
- Condiciones: la lista de agencias debe estar vacía.

llenarListaVuelos(nombre del archivo, lista de vuelos)

- Descripción: lee la información del archivo que contiene la información completa de los vuelos y las almacena en la lista de vuelos del sistema.
- Entradas: nombre del archivo de vuelos, lista de vuelos del sistema.

- Salidas: booleano que dice si la lista de vuelos del sistema se pudo llenar.
- Condiciones: la lista de vuelos debe estar vacía.

llenarListaTiquetes(nombre del archivo, lista de tiquetes)

- Descripción: lee la información del archivo que contiene la información completa de los tiquetes y las almacena en una lista.
- Entradas: nombre del archivo de tiquetes, lista de tiquetes del sistema.
- Salidas: booleano que dice si la lista de tiquetes del sistema se pudo llenar.
- Condiciones: la lista de tiquetes debe estar vacía.

actualizarAsientos(lista de vuelos , lista de agencias)

- Descripción: recorre la lista de todas las agencias y sus tiquetes vendidos respectivos para luego actualizar la información de disponibilidad de sillas en los vuelos.
- Entradas: lista de vuelos del sistema, lista de agencias del sistema.
- Salidas: los vuelos de la lista de vuelos del sistema con su número de sillas actualizado.
- Condiciones: la lista de vuelos y agencias deben estar llenas.
- Operaciones auxiliares: sillasVuelos(lista de vuelos, cada vuelo de la lista vuelo).

sillasVuelo(lista de vuelos, número de un vuelo)

- Descripción: busca el número de un vuelo en la lista de vuelo y reduce en uno su cantidad de sillas disponibles.
- Entradas: lista de vuelos del sistema, id de algún vuelo.
- Salidas: booleano que dice si se le pudo actualizar el número de sillas al vuelo en la lista de vuelos del sistema.
- Condiciones: la lista de vuelos debe estar llena.

sillasVueloPositiva (lista de vuelos, id del vuelo)

- Descripción: Busca dentro de la lista de vuelos el vuelo con el id especificado, si lo encuentra agrega 1 a la cantidad de sillas disponibles.
- Entradas: lista de vuelos, id del vuelo a modificar.
- Salidas: retorna un booleano indicando si fue posible hacer la modificación.
- Condiciones: el vuelo debe existir dentro de la lista.

borrarTiquete (lista de tiquetes, tiquetes)

- Descripción: recorre la lista de ventas buscando el tiquete que recibe, si lo encuentra se comparan las fechas y si el que llega es más reciente del que ya existe se toma como modificado se borra de la lista y se activa una bandera booleana para luego agregar el tiquete nuevo.
- Entradas: lista de ventas de cada agencia del sistema, una venta nueva.
- Salidas: booleano que dice si está en la lista de ventas de la agencia, se comprobó si es más actual y se eliminó.

- Condiciones: se verifica que la venta no existe en la lista de ventas, si se encuentra se verifica que la fecha de compra de la venta nueva tenga una fecha más reciente que la que ya existe.
- Operaciones auxiliares: compararFecha(fecha de la venta que ya está, fecha de la nueva venta).

borrarTiqueteN (lista de tiquetes, tiquete)

- Descripción: Busca el tiquete que llega como parámetro en la lista de tiquetes, si lo encuentra lo elimina sin ningún criterio.
- Entradas: lista de tiquetes, el tiquete que se va a eliminar.
- Salidas: booleano que indica si la operación se pudo realizar.
- Condiciones: El tiquete debe existir en la lista de tiquetes para poderse eliminar.

buscarTiquete (lista de tiquetes, un tiquete)

- Descripción: Busca dentro de la lista de tiquetes, el tiquete que llega como parámetro y retorna si lo encuentra o no.
- Entradas: lista de tiquetes, un tiquete a buscar.
- Salidas: retorna un booleano indicando si el tiquete está dentro de la lista.

buscarTiqueteR(lista de tiquetes, tiquete)

- Descripción: Busca el tiquete que llega como parámetro dentro de la lista de tiquetes, si lo encuentra crea una copia de este y lo retorna.
- Entradas: lista de tiquetes, el tiquete que se desea buscar.
- Salidas: Estructura de tipo tiquete con la información correspondiente si lo encuentra, si no, una estructura nula.
- Condiciones: la lista de tiquetes debe contener información.

modificarVendidos(lista de tiquetes, tiquete, indicador)

- Descripción: Modifica un tiquete de la lista de tiquetes de acuerdo con un carácter identificador del tipo de tiquete (‘m’ para modificado, ‘c’ cancelado).
- Entradas: lista de tiquetes, tiquete a modificar, el carácter identificador.
- Salidas: retorna un booleano indicando si fue posible hacer la modificación.
- Condiciones: el tiquete debe existir dentro de la lista.
- Operaciones auxiliares:

comparaFecha (fecha1, fecha2)

- Descripción: recibe dos cadenas de caracteres con fechas y retorna valores enteros entre -1 y 1 dependiendo de si una fecha es mayor o menor que la otra.
- Entradas: fecha 1 y fecha 2.
- Salidas: número entero -1 si fecha1 < fecha2, 0 si fecha1 = fecha2 y 1 si fecha1 > fecha2.
- Condiciones: ambas fechas deben llegar en cadena de caracteres y en formato yyyymmaa.

buscarVuelo(lista de vuelos, código)

- Descripción: busca en la lista de vuelos un vuelo con el código que recibe como parámetro, retorna si lo encuentra o no.
- Entradas: lista de vuelos del sistema y el código del vuelo a buscar.
- Salidas: booleano que dice si el vuelo existe o no.
- Condiciones: la lista de vuelos debe estar llena y el código como cadena de caracteres.

buscarVueloR(lista de vuelos, id de vuelo)

- Descripción: Recorre la lista de vuelos en busca de un vuelo que coincida con el identificador que llega como parámetro, si lo encuentra retorna la estructura con la información correspondiente, si no, una estructura nula.
- Entradas: lista de vuelos, identificador del vuelo a buscar.
- Salidas: estructura de tipo vuelo.

borrarVuelo(lista de vuelos, vuelo)

- Descripción: Busca en la lista de vuelos, el vuelo que se recibe por paráme-tro, si lo encuentra lo borra de la lista.
- Entradas: lista de vuelos, vuelvo que se desea borrar.
- Salidas: retorna un booleano indicando si fue posible hacer la eliminación.
- Condiciones: el vuelo debe existir dentro de la lista.

consolidarl(agencia, lista de vuelos programados, lista de agencias, lista de vuelos disponibles)

- Descripción: Elimina los vuelos programados y los tiquetes anteriores a la fecha actual, a continuación, actualiza la información de las listas de la agencia que se encuentra autenticada. Finalmente, hace un llamado a la función que escribe el archivo de los tiquetes para persistir las actualizaciones.
- Entradas: agencia, lista de vuelos programados, lista de agencias, lista de vuelos disponibles.
- Salidas: La información de todas las agencias actualizada, un archivo de texto plano con la información actualizada de los tiquetes.
- Operaciones auxiliares: actualizarVuelosProgramados, actualizarTiquetesAgencia, escribirArchTickets, escribirArchPass, escribirArchFlights.

actualizarVuelosProgramados(lista de vuelos programados)

- Descripción: Recorre la lista de vuelos programados en busca de los vuelos con la fecha igual o menor a la del día actual, dichos vuelos con estas características con eliminados de la lista.
- Entradas: lista de vuelos programados.
- Salidas: lista de vuelos programados con fechas mayores a la actual.
- Condiciones: La lista de vuelos programados no debe estar vacía.
- Operaciones auxiliares: fechaActual, comparaFecha, borrarVuelo.

actualizarTiquetesAgencia(lista de agencias)

- Descripción: Para cada agencia, recorre su respectiva lista de tiquetes vendidos en busca de tiquetes con vuelos con fechas iguales o menores a la del día actual, los tiquetes con estas características son borrados de las listas en las que se encuentre (vendidos, modificados, cancelados).
- Entradas: lista de agencias del sistema.
- Salidas: Las tres listas de cada agencia con tiquetes con fecha mayores a la actual.
- Condiciones: La lista de agencias y las tres listas de tiquetes para cada una no deben estar vacías.
- Operaciones auxiliares: borrarTiqueteN, comparaFecha, fechaActual, buscarTiqueteR.

ImprimirAgencias (lista de agencias)

- Descripción: imprime en consola la información de las agencias asociadas al sistema.
- Entradas: lista de agencias del sistema.
- Salidas: todas las agencias del sistema en una vista agradable al usuario.
- Condiciones: la lista de agencias debe estar llena.

ImprimirVuelos (lista de vuelos)

- Descripción: imprime en consola la información de todos los vuelos registrados.
- Entradas: lista de vuelos del sistema.
- Salidas: todos los vuelos del sistema en una vista agradable al usuario.
- Condiciones: la lista de vuelos debe estar llena.

ImprimirTiquetes (listaTiquetes)

- Descripción: imprime en consola la información de las ventas de tiquetes.
- Entradas: lista de tiquetes del sistema.
- Salidas: todos los tiquetes del sistema en una vista agradable al usuario.
- Condiciones: la lista de tiquetes debe estar llena.

ImprimirVenta (venta)

- Descripción: imprime en consola la información de una venta de tiquete determinada.
- Entradas: venta.
- Salidas: la venta en una vista agradable al usuario.
- Condiciones: la venta no debe estar vacía.

escribirArchTickets (lista de agencias)

- Descripción: para cada agencia registrada en el sistema, escribe en un archivo de texto plano la información de sus tiquetes correspondientes.
- Entradas: lista de agencias del sistema.

- Salidas: archivo de texto plano con la información de los tiquetes.
- Condiciones: debe existir información dentro de la lista de agencias.

llenarListasArbol(lista de ciudades origen, lista de los id de los vuelos, lista de las fechas disponibles, lista de los vuelos programados)

- Descripción: Recorre la lista de vuelos programados, extrayendo la información de identificador, día y ciudad de origen, agregando cada uno en una lista diferente.
- Entradas: lista de ciudades origen, lista de los identificadores de vuelo, lista de los días de operación.
- Salidas: Tres listas con la información de id's, orígenes y fechas.

llenarAgenciasArbol(lista de agencias, lista de usuarios a llenar)

- Descripción: Recorre la lista que contiene la información de las agencias asociadas al sistema y extrae la información del nombre de usuario con la que cada una se encuentra registrada para agregarla a otra lista que será utilizada para llenar el árbol de la función autocompletar.
- Entradas: lista de agencias del sistema, lista de cadena de caracteres que se va a llenar.
- Salidas: Lista con la información de los nombres de usuario de las agencias.

llenarListaVentasArbol(lista de identificadores de tiquetes a llenar, lista de tiquetes)

- Descripción: Recorre la lista de tiquetes y agrega el id de cada tiquete a una lista de cadenas de caracteres.
- Entradas: lista de tiquetes, lista de cadenas de caracteres a llenar.
- Salidas: La lista de cadena de caracteres con la información de todos los identificadores de tiquetes existentes.

llenar(lista de cadenas de caracteres, árbol general de cadenas de caracteres)

- Descripción: Inserta la información de una lista de manera correcta en el trie.
- Entradas: lista de cadenas de caracteres, árbol general de cadenas de caracteres
- Salidas: El Trie modificado con la información de la lista en él.
- Condiciones: La lista no puede ser vacía y el árbol no debería poseer información que no le corresponda.

getAuto(Nodo, lista de caracteres)

- Descripción: A partir de un nodo de inicio modifica una lista de cadena de caracteres, recorriendo el árbol en forma de pre-orden e ingresando únicamente los nodos hoja, que representan los comandos completos.
- Entradas: Nodo, lista de caracteres
- Salidas: Una lista de cadenas de caracteres con información de los comandos completos.
- Condiciones: El nodo se debe utilizar a forma de raíz de un subárbol.

buscarTrie(cadena de caracteres, Nodo)

- Descripción: Recorre el árbol buscando el nodo que tenga la misma cadena de caracteres que llega como parámetro y luego lo retorna
- Entradas: Cadena de caracteres, Nodo
- Salidas: Un nodo.
- Condiciones: El nodo se debe utilizar a forma de raíz del árbol o de un subárbol.

imprimirVuelosR(lista de vuelos)

- Descripción: Recorre la lista de vuelos que llega como parámetro e imprime únicamente la información de identificador de vuelo, día de operación, hora de partida, duración, su origen y destino.
- Entradas: lista de vuelos a imprimir.
- Salidas: Impresión en consola.
- Condiciones: La lista de vuelos no debe estar vacía.

llenarGrafoTiempo(lista de vuelos disponibles, grafo a llenar de tiempo, grafo a llenar de costo, grafo a llenar de ruta)

- Descripción: Recorre la lista de vuelos disponibles del sistema y agrega al grafo tipo que llega como parámetro, las ciudades como vértices que se encuentran dentro de los vuelos, el peso de las aristas del grafo se calcula con la duración del vuelo. Repite la operación con los otros dos grafos, teniendo en cuenta para el peso de las aristas el costo del tiquete y para el último, el peso para todas las aristas es 1.
- Entradas: lista de vuelos, grafos a llenar.
- Salidas: grafos con la información de los vuelos del sistema.
- Condiciones: La lista de vuelos no debe estar vacía.
- Operaciones auxiliares: tiempoVuelo(duración del vuelo), insertarVertice(ciudad), insertarArista(origen, destino, peso).

buscarVueloDisponible(origen, destino, lista de vuelos)

- Descripción: Recorre la lista de vuelos disponibles del sistema en busca del vuelo con ciudades origen y destino igual a las ciudades que se reciben por parámetro, si lo encuentra retorna una estructura de tipo vuelo con la información correspondiente.
- Entradas: lista de vuelos, ciudad origen, ciudad destino.
- Salidas: estructura de tipo Vuelo con la información del vuelo o estructura de tipo Vuelo nula.
- Condiciones: La lista de vuelos no debe estar vacía.

change(lista de vuelos disponibles, lista de vuelos programados, agencia, id del pasajero, id del vuelo original, id del vuelo nuevo)

- Descripción: Se recibe el id del pasajero, y el id de su vuelo original y el del nuevo, se recorre la lista de todas las agencias buscando el tiquete que corresponde, si está en vendidos con el tipo “v” se hace el cambio a tipo “m” y se agrega a la lista de modificados con el ID del vuelo nuevo. Sí el tipo del tiquete en la lista de vendidos es “m” se va y se busca en la de modificados y se elimina el que está y se introduce el nuevo con el número de vuelo nuevo, en cualquier caso, se informa del cambio de tarifa en el precio de los tiquetes.
- Entradas: lista de vuelos programados, lista de vuelos disponibles, agencia, id del pasajero, id del vuelo original, id del vuelo nuevo.
- Salidas:
- Condiciones: Que el vuelo exista y que la fecha del vuelo coincida con su día.
- Operaciones auxiliares: buscarVueloR(lista de vuelos disponibles, id del vuelo nuevo), diaFecha(fecha del vuelo), sillasVueloPositiva(lista de vuelos programados, vuelo original), buscarVuelo(lista de vuelos programados, vuelo nuevo), sillasVuelo(lista de vuelos programados, vuelo nuevo), fechachange(fecha vuelo origianl, día fecha original, día del vuelo nuevo).

TAD Agencia

Conjunto mínimo de datos:

- tiquetesVendidos, lista<Venta>, almacena la información de la venta de tiquetes realizada por la agencia de viajes.
- usuario, cadena de caracteres, contiene el nombre de usuario con la que se identifica la agencia dentro del sistema y con el cual debe hacer el proceso de autenticación.
- contraseña, cadena de caracteres, almacena la contraseña con la que la agencia puede hacer su respectiva autenticación en el sistema.

Operaciones:

generarReporteDisponibles (lista de vuelos)

- Descripción: retorna una lista con la información de los vuelos que aún tienen disponibilidad de asientos.
- Entradas: la lista que contiene la información de los vuelos ofrecidos por la agencia.
- Salidas: retorna una lista con los vuelos que aún tienen disponibilidad de asientos.
- Condiciones: la información de los vuelos debe estar cargada en memoria.

generarReporteOrigenFecha (lista de vuelos, origen, fecha)

- Descripción: retorna una lista con la información de los vuelos que aún tienen disponibilidad de asientos filtrando por un origen y una fecha dada.
- Entradas: lista de lista de vuelos ofrecidos por la agencia, una ciudad de origen y una fecha de operación.

- Salidas: una lista de los vuelos que aún tienen disponibilidad de asientos que operan desde la ciudad de origen y fecha especificada.
- Condiciones: la información de los vuelos debe estar cargada en memoria.
- Operaciones auxiliares: diaFecha(fecha).

generarReporteOrigen (lista de vuelos, origen)

- Descripción: retorna una lista con la información de los vuelos que aún tienen disponibilidad de asientos filtrando por un origen dado.
- Entradas: la lista que contiene la información de los vuelos y una ciudad de origen.
- Salidas: lista de los vuelos disponibles que operan para la ciudad de origen especificada.
- Condiciones: la información de los vuelos ofrecidos por la agencia debe estar cargada en memoria.

venderVueloSencillo (lista de vuelos, id, fecha, agencia, lista de ventas)

- Descripción: retorna una información de una venta (tiquete), durante el proceso se pide información adicional por consola y se hacen las verificaciones necesarias, asegurando que el id del vuelo y el dia de la fecha de salida coincidan.
- Entradas: lista de vuelos, identificador del vuelo del que se quiere hacer la venta, la fecha de para la que se dispone a viajar, la agencia que desea realizar la venta, lista de tiquetes vendidos.
- Salidas: estructura de tipo Venta con la información propia y la correspondiente al cliente.
- Condiciones: El vuelo del que se quiere hacer la venta debe tener disponibilidad de asientos. La fecha especificada debe coincidir con el día de operación del vuelo.
- Operaciones auxiliares: generarReporteDisponibles(lista de vuelos), diaFecha(fecha), encontrarVuelo(lista de vuelos disponibles, identificador del vuelo, día).

encontrarVuelo (lista de vuelos, id de vuelo, día)

- Descripción: busca dentro de la lista de vuelos el vuelo que coincida con el id y el día especificados.
- Entradas: identificador del vuelo y el día de operación del vuelo.
- Salidas: estructura de tipo Vuelo con la información que coincide con el identificador de vuelo y la fecha.
- Condiciones: El vuelo esté contenido dentro de la lista.
- La lista de vuelos no esté vacía.
- El día de operación especificado debe coincidir con el día de operación del vuelo que se señala con el id.

reportMoney (agencia, lista de vuelos programados)

- Descripción: Recorre las listas de la agencia que está autenticada para realizar un informe financiero de acuerdo con los tiquetes que ésta ha vendido. Primero recorre las listas de vendidos para calcular el total del dinero que la agencia recibirá por tales ventas. A continuación, realiza el mismo calculo con la lista de modificados, teniendo en cuenta los tiquetes originales y los tiquetes nuevos para determinar si la agencia debe reembolsar dinero o recibirá dinero por una modificación, luego realiza el cálculo para la lista de vuelos cancelados, teniendo en cuenta la regla de negocio de que la agencia solo hará el reembolso del 85% del valor del tiquete cancelado. Finalmente imprime en pantalla la información calculada.
- Entradas: agencia autenticada, lista de vuelos programados.
- Operaciones auxiliares: buscarVuelloR, buscarTiquete, imprimirTiquetes.

cancelar (id de vuelo, agencia, lista de vuelos programados, lista de agencias)
- Descripción: Recorre la lista de vuelos de la agencia que está autenticada buscando el vuelo correspondiente al identificador que llega como parámetro, si el vuelo es de tipo vendido, cambia su tipo a cancelado y lo agrega a la lista de vuelos cancelados de la agencia luego busca el vuelo en la lista de vuelos programados para agregar 1 a la cantidad de asientos disponibles, de igual forma sucede si el vuelo es modificado.
- Entradas: identificador de vuelo que se desea cancelar, agencia autenticada, lista de los vuelos programados, lista de las agencias del sistema.
- Salidas: booleano que indica si la operación se pudo realizar, las listas de la agencia con la información actualizada.
- Operaciones auxiliares: buscarTiqueteR(id del tiquete), sillasVueloPositiva(vuelo), borrarTiquete(id del tiquete).

seleccionarVendidos(lista de tiquetes)
- Descripción: Recorre la lista de tiquetes vendidos de la agencia y agrega en una nueva la lista de los tiquetes con tipo ‘v’.
- Entradas: lista de tiquetes que se va a modificar.
- Salidas: lista actualizada únicamente con los vuelos vendidos.

TAD Nodo

Conjunto mínimo de datos:
- Dato del nodo.
- Lista de apuntadores a nodo (descendientes).

Operaciones:
- Asignar dato.
- Obtener dato del nodo.
- Obtener la lista de descendientes.

TAD Árbol General

Conjunto mínimo de datos:

- Nodo raíz.

Operaciones:

- Constructor.
- Destructor.
- Asignar raíz.
- Obtener raíz.
- Es vacío.
- Tamaño.
- Altura.
- Insertar nodo.
- Eliminar nodo.
- Buscar nodo.
- Preorden.

TAD Grafo

Conjunto mínimo de datos:

- Lista de vértices.
- Matriz de adyacencia.

Operaciones:

- Constructor.
- Destructor.
- Agregar vértice.
- Agregar arista.
- Existe arista.
- Obtener vértice.
- Obtener índice.
- Obtener número de vértices.
- Obtener vector de vértices.
- Obtener Dijkstra.
- Obtener camino más corto.
- Búsqueda en anchura (BFS).

2. Diseño de las estructuras a utilizar.

Estructura Venta

- agencia, cadena de caracteres, representa el nombre de usuario con la que se identifica una agencia dentro del sistema.
- entero Id de venta, número único identificador para la venta del tiquete.

- vuelo, cadena de caracteres, representa el identificador del vuelo del tiquete.
- cadena de caracteres, cédula, contiene la información del número de identificación personal del cliente del tiquete.
- apellidos, cadena de caracteres, contiene la información del/los apellido(s) del cliente.
- cadena de caracteres, nombres, contiene la información del/los nombre(s) del cliente.
- cadena de caracteres, fecha de vuelo, representa la fecha de operación del vuelo de la venta.
- fecha de compra, cadena de caracteres, contiene la información de la fecha en la que el cliente realizó la respectiva compra.
- Hora compra, cadena de caracteres, contiene la información de la hora en la que el cliente realizó la respectiva compra.

Estructura Vuelo

- código, cadena de caracteres, representa un identificador único para el vuelo.
- dia, cadena de caracteres, contiene la información del día de la semana en la que opera el vuelo.
- origen, cadena de caracteres, contiene la información de la ciudad de origen.
- destino, cadena de caracteres, contiene la información de la ciudad destino.
- hora, cadena de caracteres, representa la hora de partida.
- duración, cadena de caracteres, cantidad de minutos que representa la duración del vuelo.
- sillas, entero, cantidad de sillas que representa la capacidad del avión.
- costo, real, representa el valor monetario del tiquete para este vuelo.

3. Diseño de las operaciones auxiliares a utilizar.

Operaciones:
HoraActual ()

- Descripción: toma la información de fecha y hora actual del sistema, la procesa y luego retorna únicamente la hora en formato (HHMM)
- Salidas: cadena de caracteres que representa la hora y minutos obtenida desde el sistema.

FechaActual ()

- Descripción: toma la información de fecha y hora actual del sistema, la procesa y luego retorna únicamente la fecha en formato (YYYYMMDD).
- Salidas: cadena de caracteres que contiene la información de año, mes y día obtenido desde el sistema.

DiaFecha(fecha)

- Descripción: recibe una cadena de caracteres en forma de fecha y retorna el día de la semana al que corresponde la fecha ingresada.

- Entradas: la fecha especificada en formato (YYYYMMDD).
- Salidas: retorna una cadena de caracteres con el día de la semana que coincide con la fecha.
- Condiciones: la fecha en cadena de caracteres en formato (YYYYMMDD).

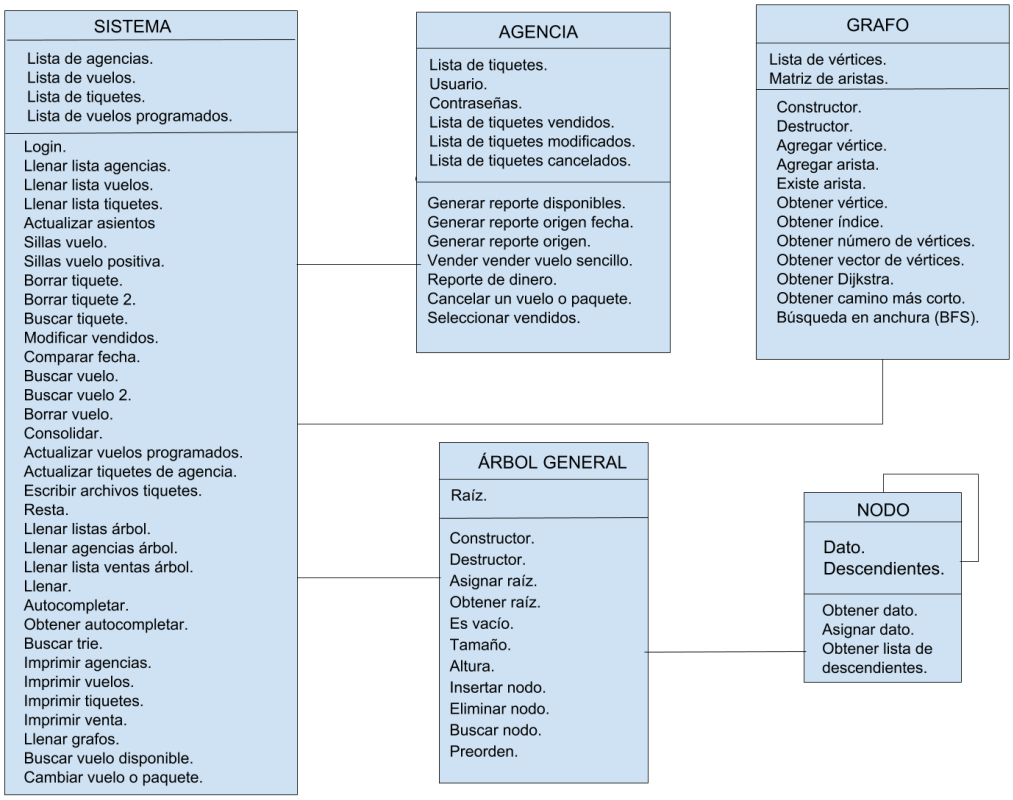
TiempoVuelo(hora)

- Descripción: recibe una cadena de caracteres en con la duración del vuelo dada en formato “HH:MM”, halla su equivalencia en minutos.
- Entradas: la duración especificada en formato (HHMM).
- Salidas: retorna un entero largo indicando la cantidad de minutos.
- Condiciones: la fecha en cadena de caracteres en formato (HHMM).
- Operaciones auxiliares: stol(hora).

FechaChange(fecha, dia origen, dia destino)

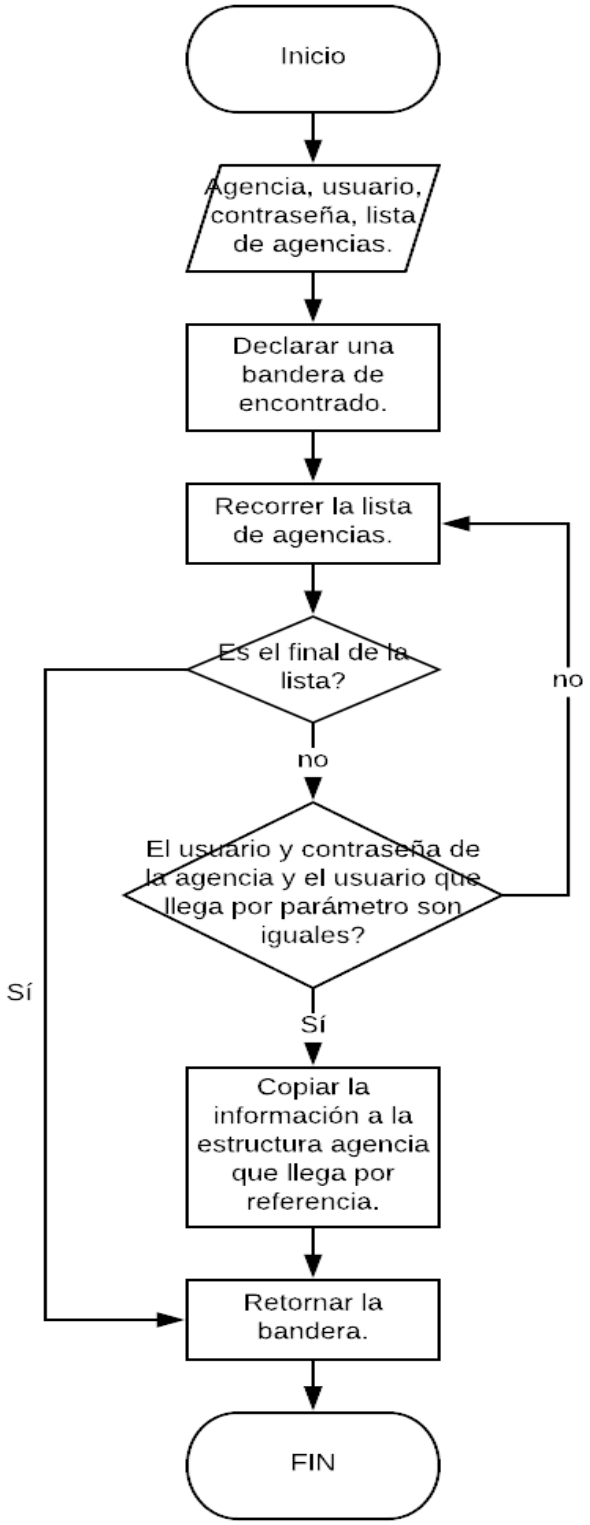
- Descripción: Encuentra una fecha en formato (DDMMYYYY) de acuerdo al día de la semana destino que llega como parámetro, para esto se tiene en cuenta la fecha que llega como parámetro y el día de la semana al que corresponde la fecha.
- Entradas: cadena de caracteres con la fecha en formato (DDMMYYYY), dos cadenas de caracteres con días de la semana.
- Salidas: retorna una cadena de caracteres con la fecha resultante en formato (DDMMYYYY).
- Condiciones: la fecha en cadena de caracteres en formato (HHMM).

4. Diagrama de relación entre TAD’s.

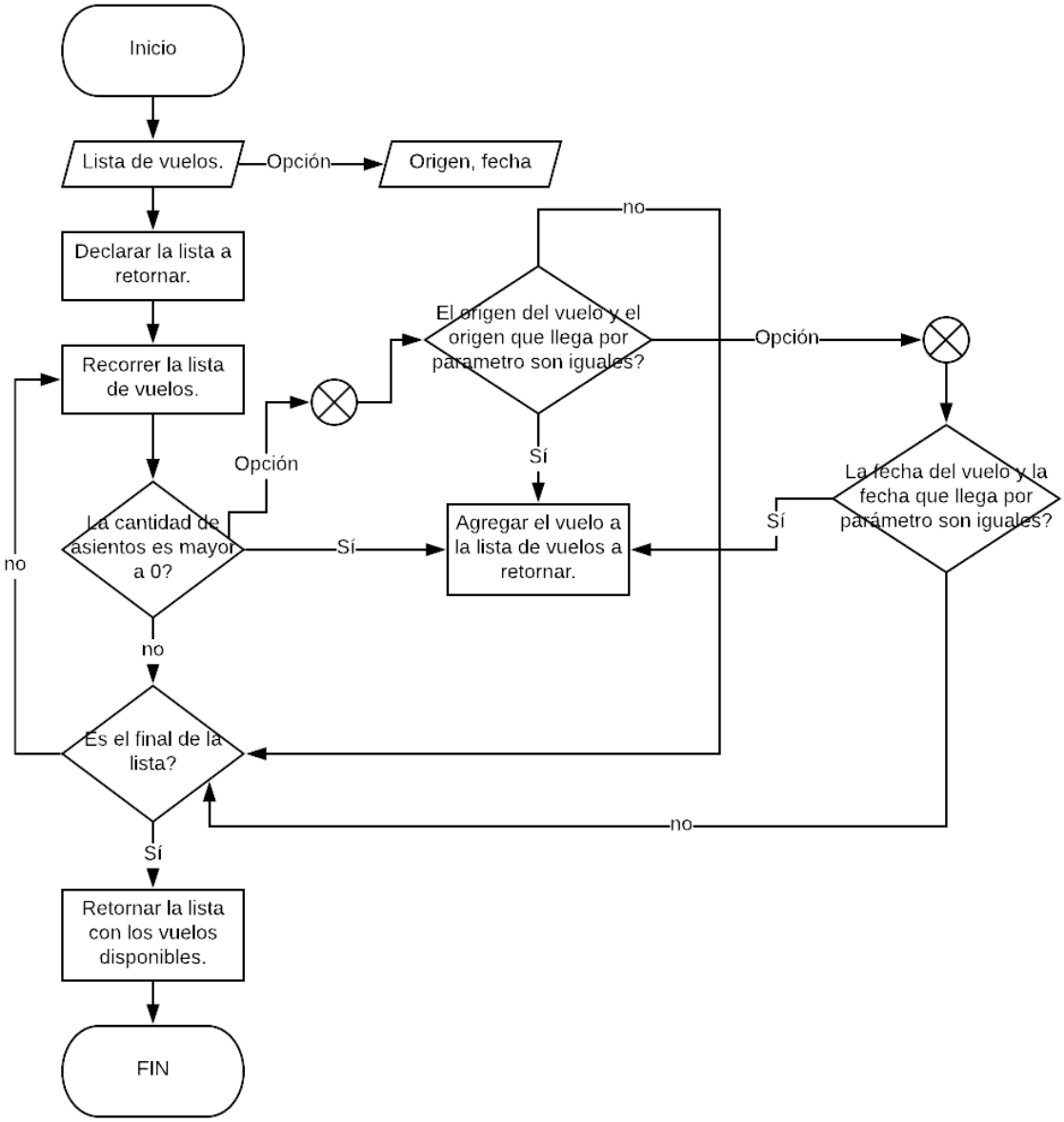


5. Diagramas de flujo de los comandos disponibles.

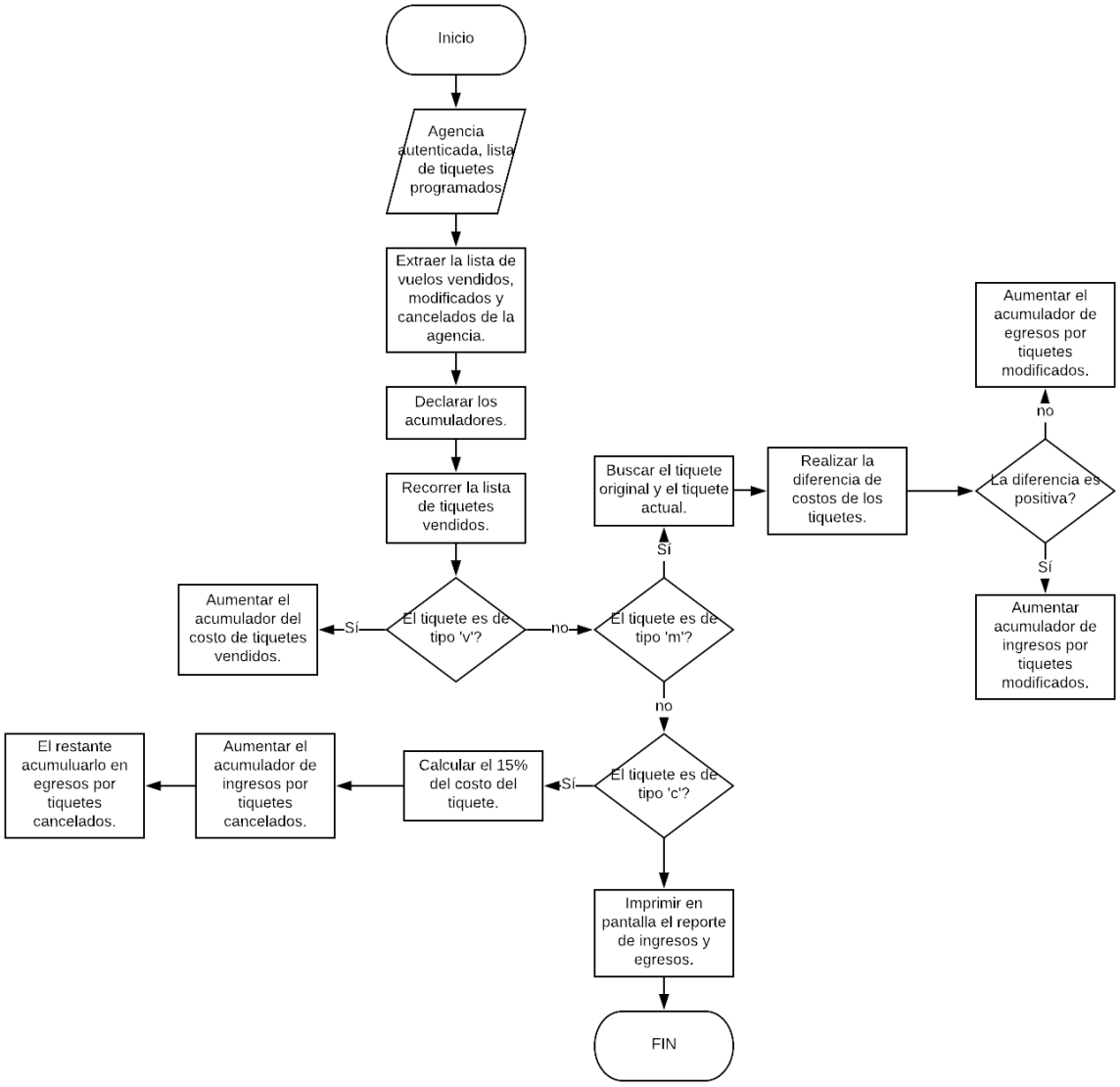
Login



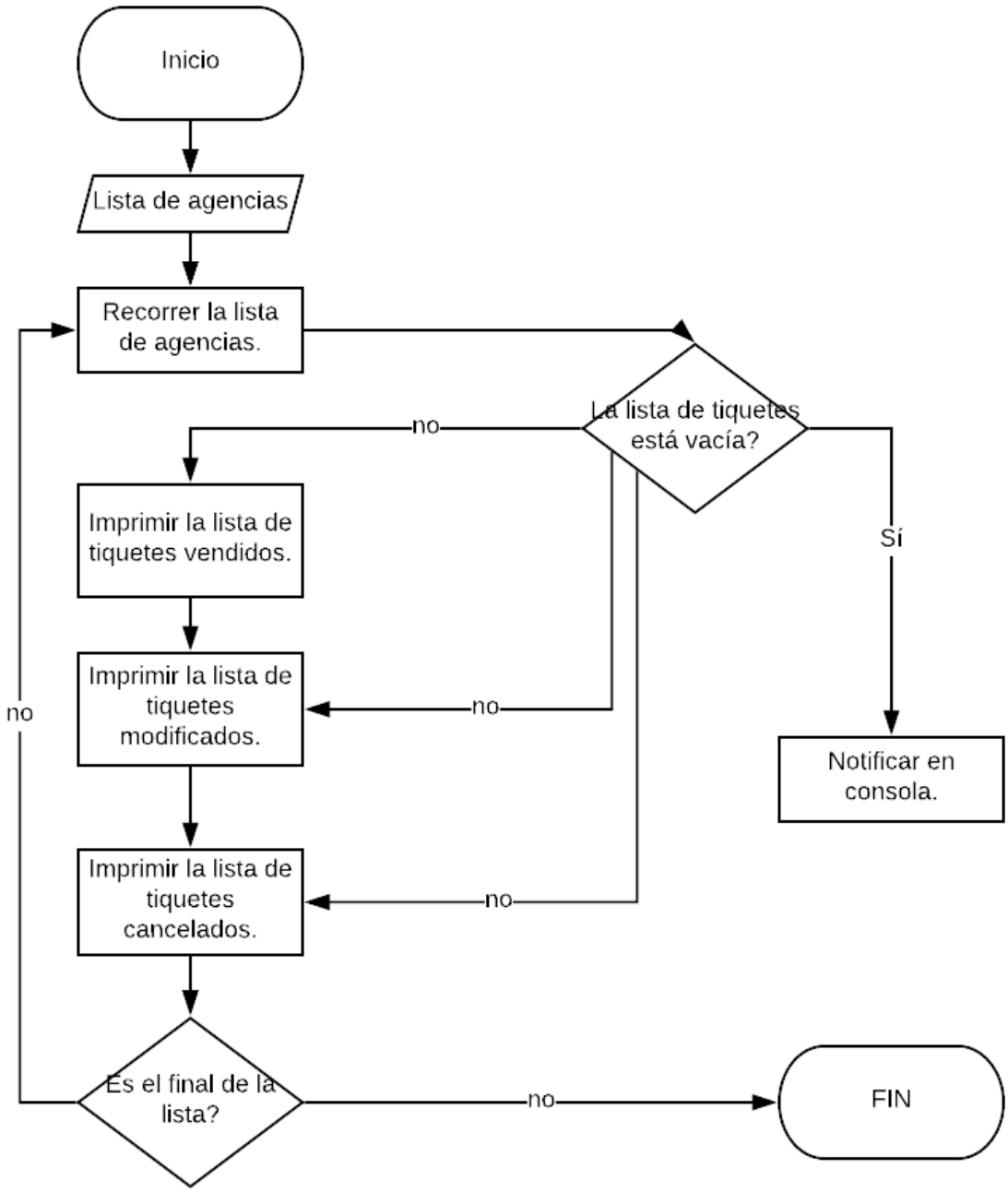
Report flights



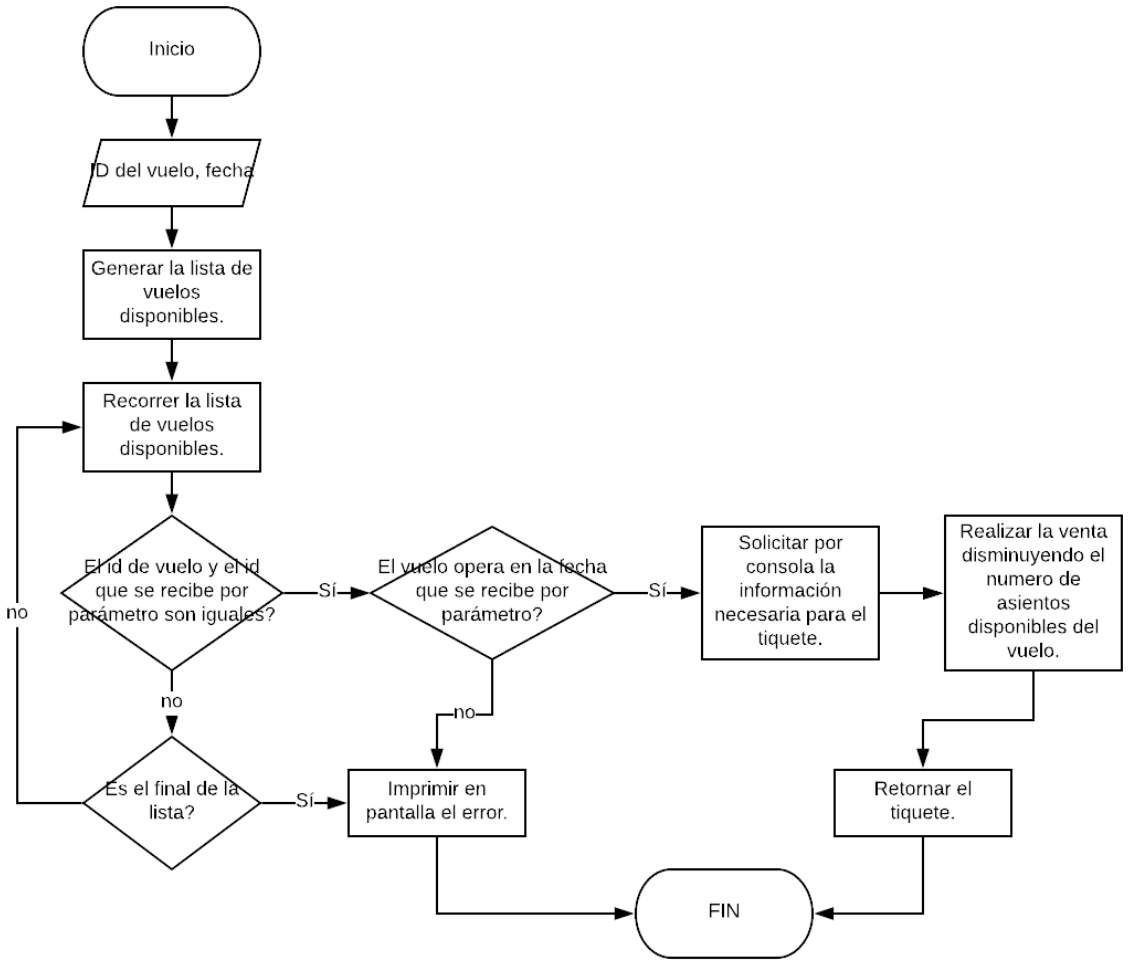
Report money



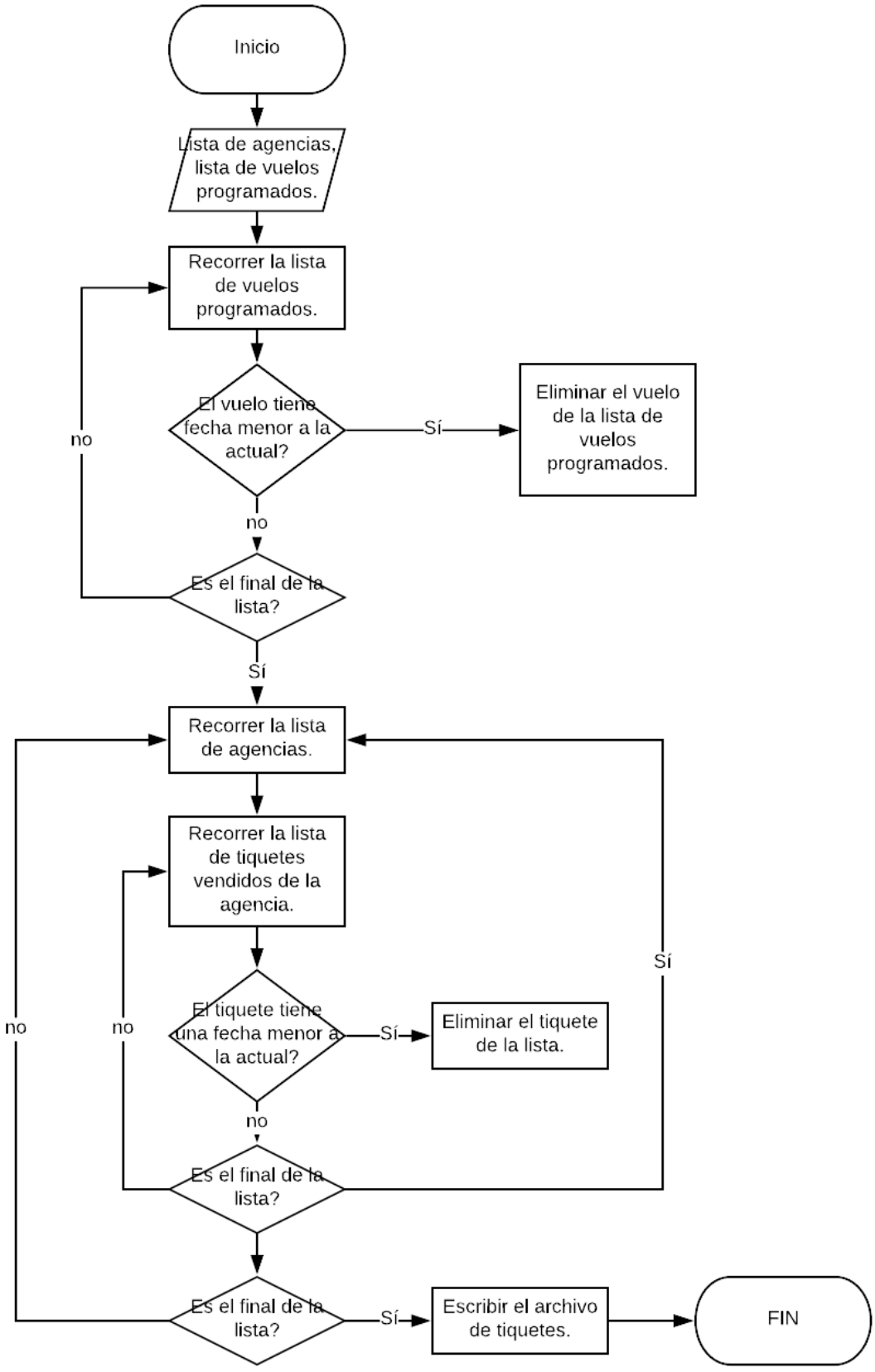
Report inventory



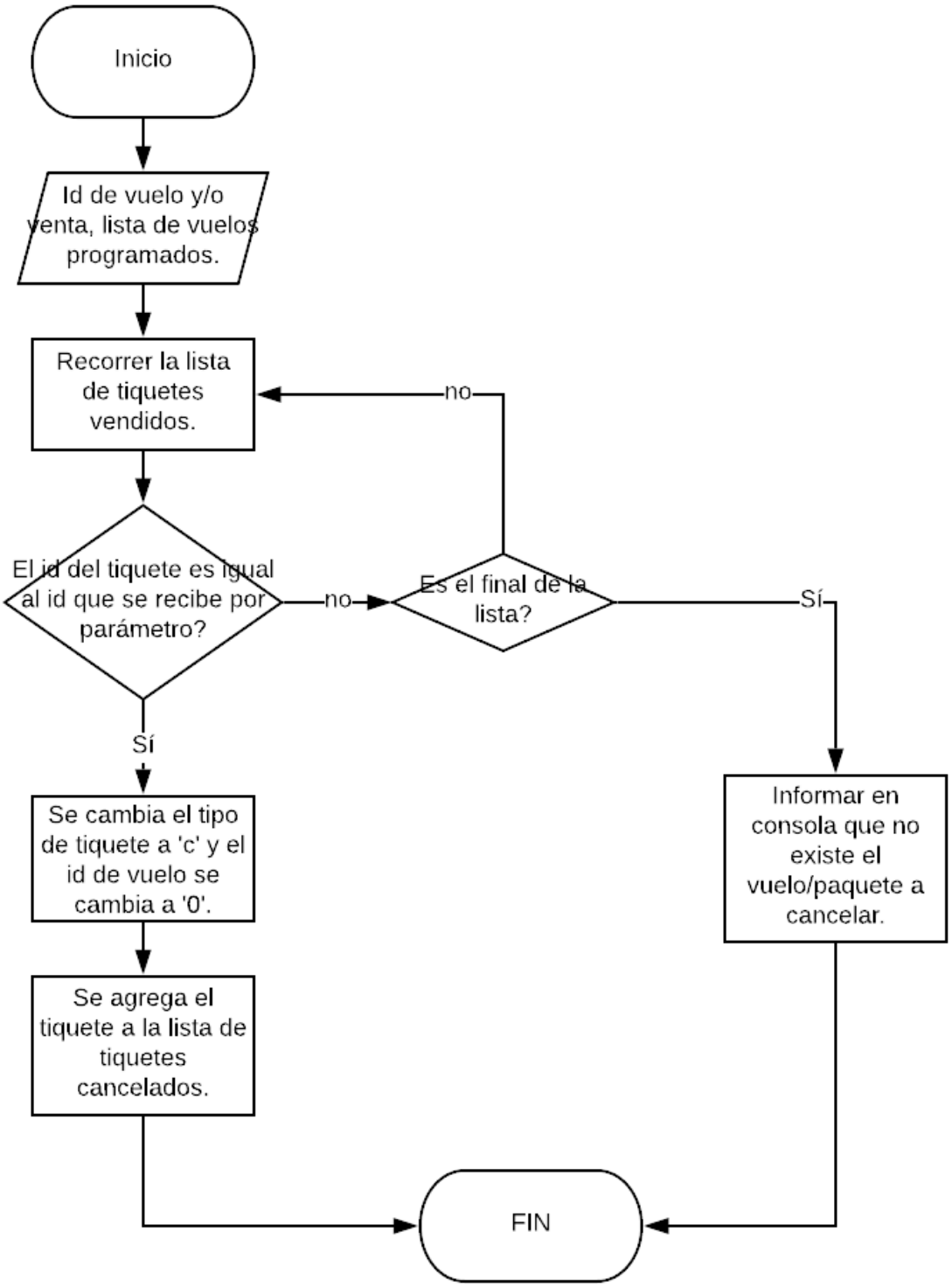
Sell


Inicio
ID del vuelo, fecha
Generar la lista de vuelos disponibles.
Recorrer la lista de vuelos disponibles.
El id de vuelo y el id que se recibe por parámetro son iguales?
Sí
El vuelo opera en la fecha que se recibe por parámetro?
Sí
Solicitar por consola la información necesaria para el tiquete.
Realizar la venta disminuyendo el numero de asientos disponibles del vuelo.
no
no
no
Es el final de la lista?
Sí
Imprimir en pantalla el error.
Retornar el tiquete.
FIN

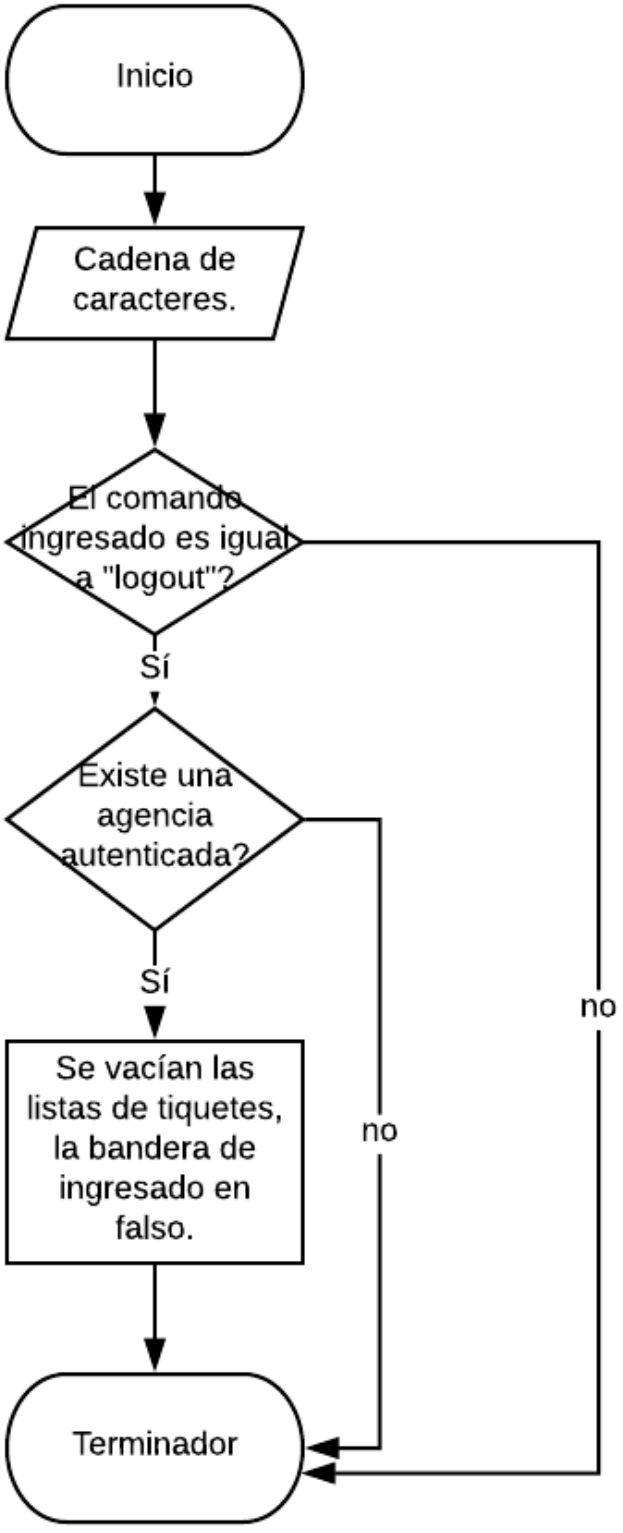
Consolidate



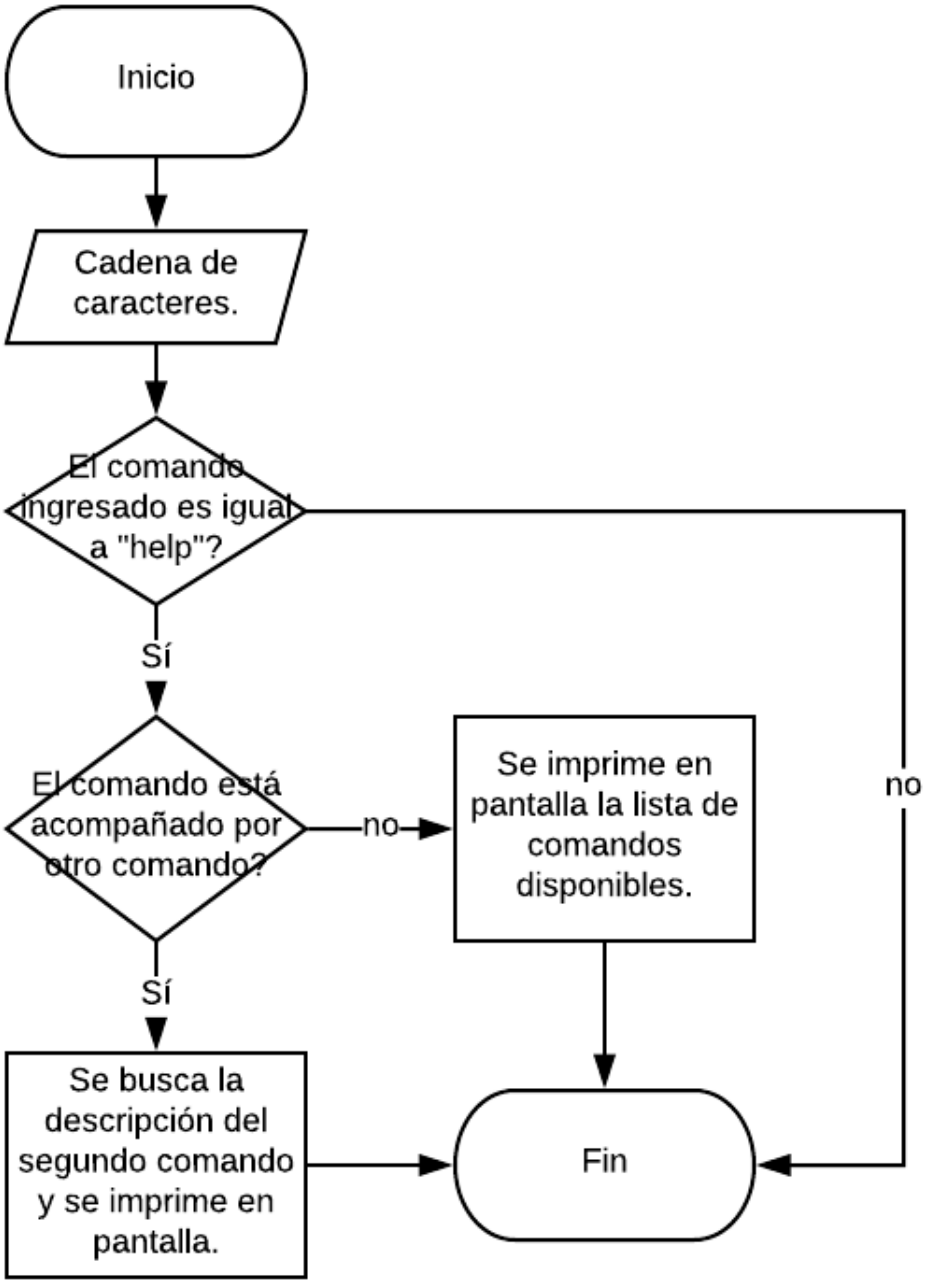
Cancel


Inicio
Id de vuelo y/o venta, lista de vuelos programados.
Recorrer la lista de tiquetes vendidos.
no
El id del tiquete es igual al id que se recibe por parámetro?
no
Es el final de la lista?
Sí
Sí
Se cambia el tipo de tiquete a 'c' y el id de vuelo se cambia a '0'.
Informar en consola que no existe el vuelo/paquete a cancelar.
Se agrega el tiquete a la lista de tiquetes cancelados.
FIN

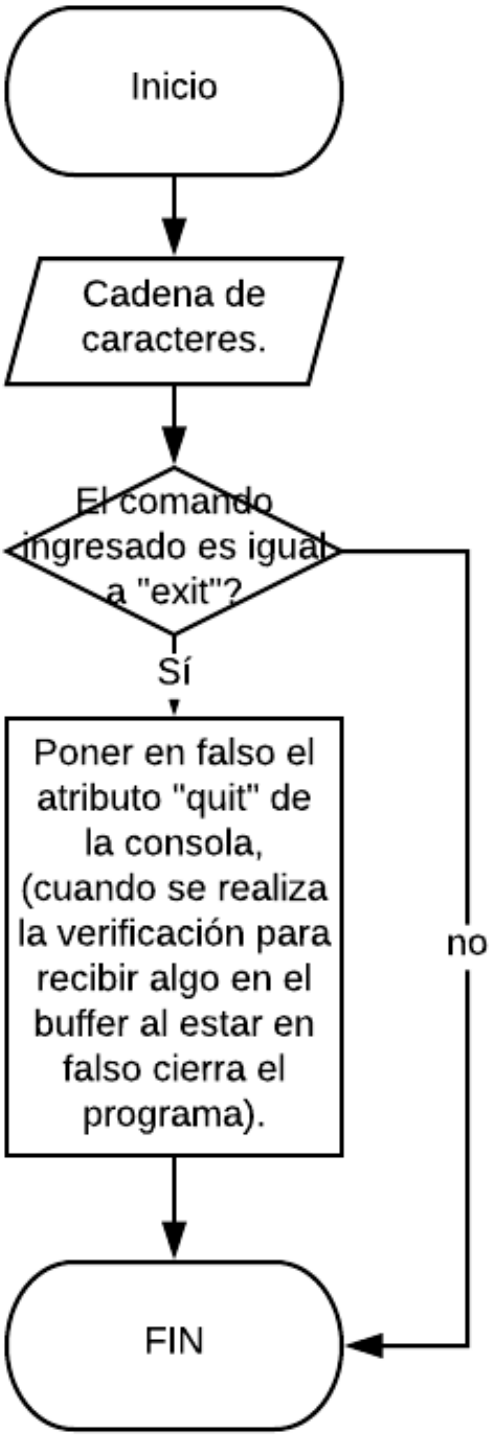
Logout


Inicio
Cadena de caracteres.
El comando ingresado es igual a "logout"?
Sí
no
Existe una agencia autenticada?
Sí
no
Se vacían las listas de tiquetes, la bandera de ingresado en falso.
Terminador

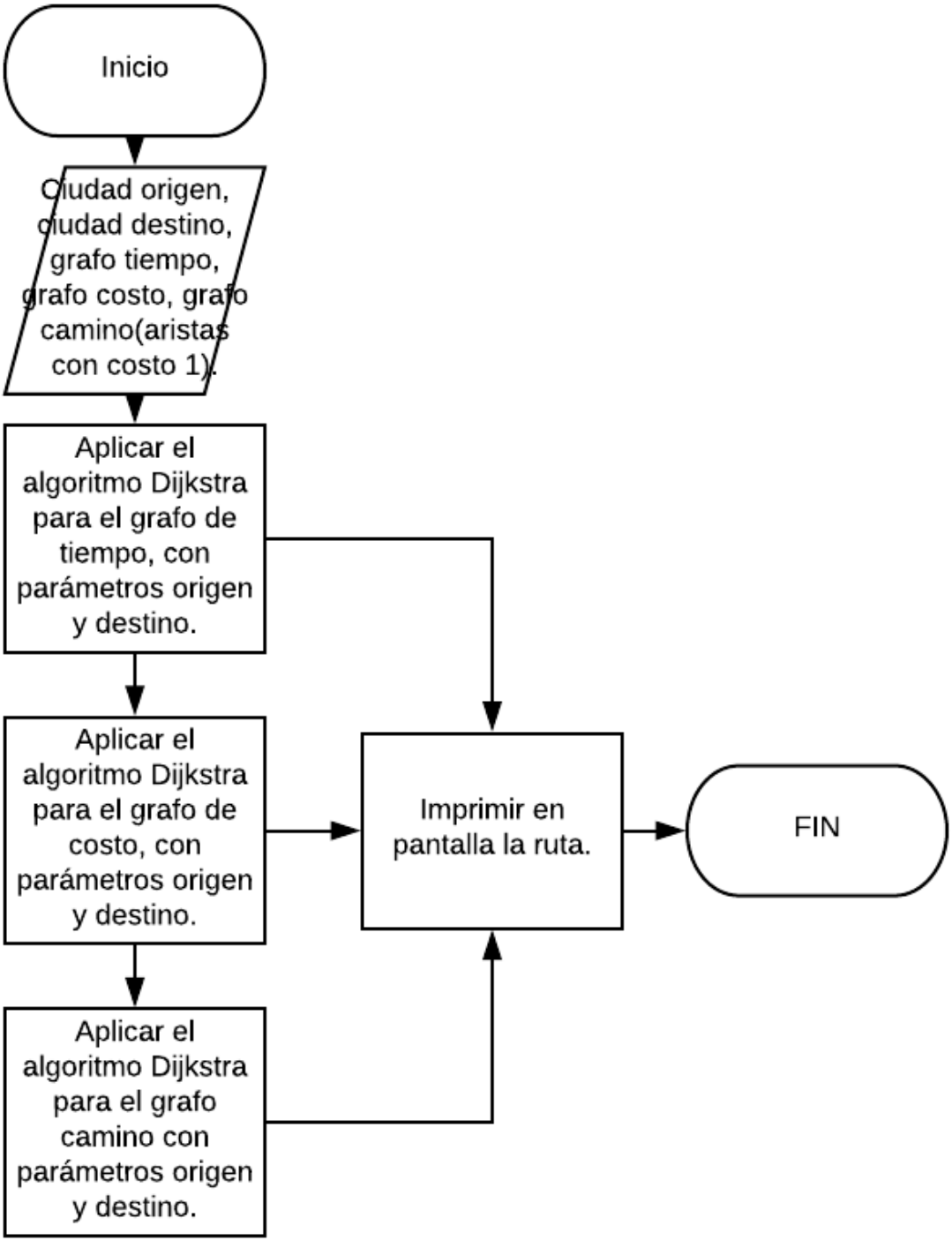
Help


Inicio
Cadena de caracteres.
El comando ingresado es igual a "help"?
Sí
El comando está acompañado por otro comando?
no
Se imprime en pantalla la lista de comandos disponibles.
Sí
Se busca la descripción del segundo comando y se imprime en pantalla.
Fin
no

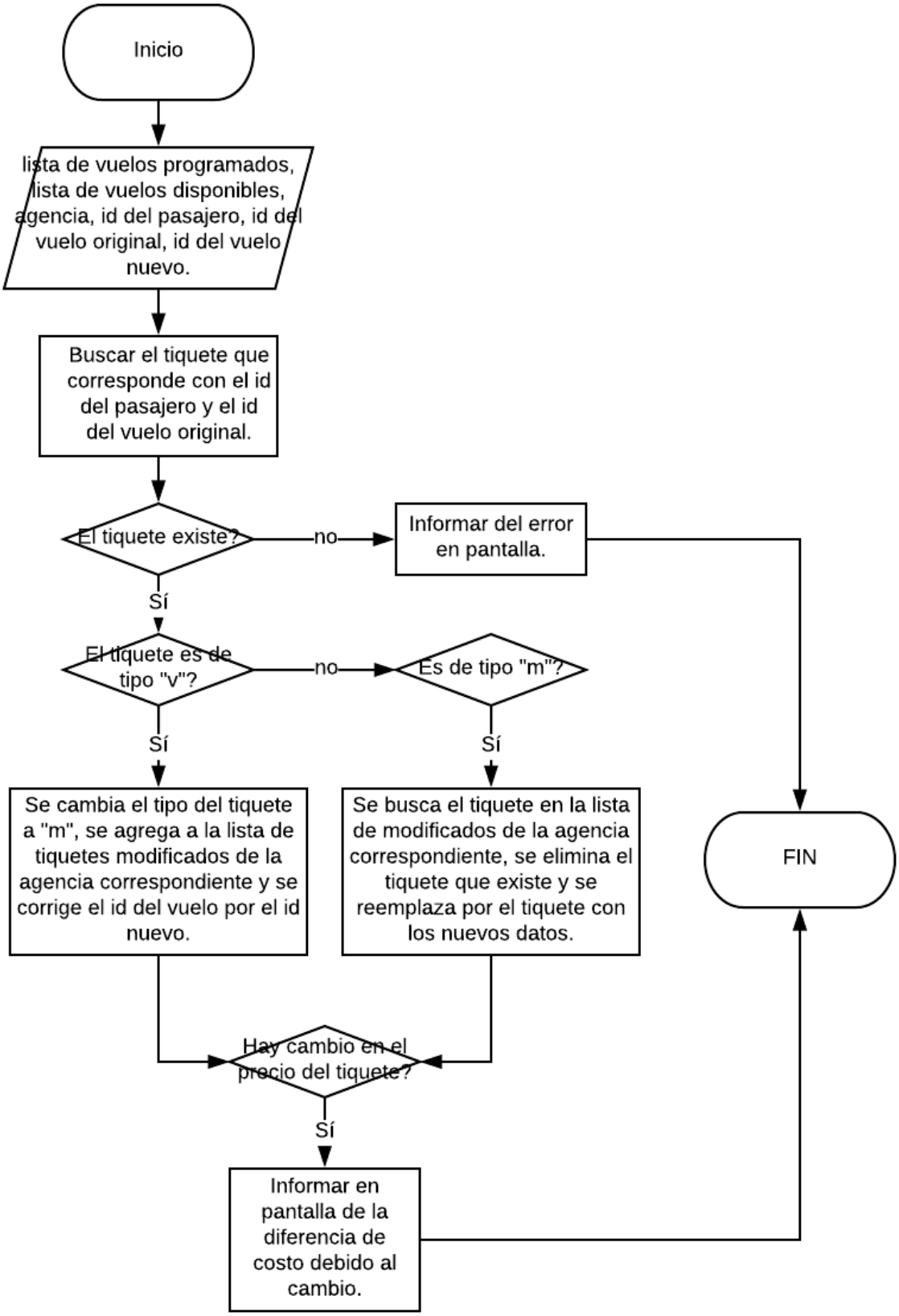
Exit


Inicio
Cadena de caracteres.
El comando ingresado es igual a "exit"?
Sí
no
Poner en falso el atributo "quit" de la consola, (cuando se realiza la verificación para recibir algo en el buffer al estar en falso cierra el programa).
FIN

Path


Inicio
Ciudad origen, ciudad destino, grafo tiempo, grafo costo, grafo camino(aristas con costo 1).
Aplicar el algoritmo Dijkstra para el grafo de tiempo, con parámetros origen y destino.
Aplicar el algoritmo Dijkstra para el grafo de costo, con parámetros origen y destino.
Aplicar el algoritmo Dijkstra para el grafo camino con parámetros origen y destino.
Imprimir en pantalla la ruta.
FIN

Change


Inicio
lista de vuelos programados, lista de vuelos disponibles, agencia, id del pasajero, id del vuelo original, id del vuelo nuevo.
Buscar el tiquete que corresponde con el id del pasajero y el id del vuelo original.
El tiquete existe?
no
Informar del error en pantalla.
Sí
El tiquete es de tipo "v"?
no
Es de tipo "m"?
Sí
Sí
Se cambia el tipo del tiquete a "m", se agrega a la lista de tiquetes modificados de la agencia correspondiente y se corrige el id del vuelo por el id nuevo.
Se busca el tiquete en la lista de modificados de la agencia correspondiente, se elimina el tiquete que existe y se reemplaza por el tiquete con los nuevos datos.
FIN
Hay cambio en el precio del tiquete?
Sí
Informar en pantalla de la diferencia de costo debido al cambio.

Acta de evaluación.

Corrección de errores:

-Si el programa se ejecuta sin parámetros adicionales para los nombres de los archivos, el programa no debería generar violación de segmento, sino con un mensaje indicar que parámetros se requieren.

Para corregir esta violación de segmento se agregó una nueva verificación a la hora de recibir los parámetros de ejecución del sistema. Si el número de parámetros es menor que 3, entonces se informa del error en pantalla y se especifica los archivos necesarios para la correcta ejecución.

-El reporte de vuelos sigue usando mucho espacio para cada entrada, y si son demasiadas el reporte se alarga innecesariamente.

En este caso, agregamos una nueva función que imprime la información más relevante del vuelo, tal como lo es el código, el día y hora de operación, las ciudades origen y destino y la duración.

- En el reporte de inventarios, si el tiquete ya ha sido cancelado, ¿por qué sale en los vendidos también? Cada tiquete sólo debería aparecer en uno de los 3 bloques: o vendido, o modificado, o cancelado.

Como necesitamos tener un registro de los vuelos cuando han sido modificados y/o cancelados, para este caso decidimos agregar una función que selecciona únicamente los vuelos de tipo modificado (“m”) agregándolos a una lista que se imprime a la hora de hacer el reporte de inventarios.

Documento de diseño:

- Revisar de nuevo la definición de un TAD, debe tener datos y operaciones para que pueda ser considerado TAD. Si sólo tengo datos es una estructura auxiliar, si sólo tengo operaciones son funciones auxiliares. Se siguen describiendo, pero no se consideran un TAD.

Se corrigió las inconsistencias con respecto a las estructuras auxiliares y se eliminaron del diagrama de relación entre TAD’s, también se revisó FechaYHora que estaba definido como TAD para redefinirlo como un apartado con operaciones auxiliares.

- Harían falta esquemáticos para logout, exit y help.

Se agregó al documento de diseño los diagramas de flujo para logout, exit y help,

además de agregar los esquemáticos para los nuevos comandos path y change.

Autocompletar:

- El autocompletar debe dar las opciones dependiendo del nivel en el que me encuentro. Si estoy al principio de la línea debería permitirme completar solo comandos, una vez completo un comando que necesita parámetros me debería ayudar a completar el parámetro adecuado.

En este caso, almacenamos la información del buffer en un arreglo, para realizar una revisión detallada de lo recibido y así determinar si lo recibido necesita autocompletar un comando o necesita autocompletar un parámetro. De acuerdo con esto, se realiza la correspondiente búsqueda en los diferentes árboles.

- Hay que revisar la inserción de espacios adicionales, y el buffer que se envía cuando se oprime TAB, en algunos casos no completa nada, pero se fastidia el proceso de reconocer el comando.

Se garantiza que solo se realice la inserción de espacios o completar palabras solo cuando se deba completar, seguidamente después de encontrar si hay alguna palabra para completar se reescribe por medio de la impresión en pantalla como quedaba ya completada y se setea el buffer, dado el caso que no se deba o no se pueda autocompletar no se hace nada.

- También debería ayudarme a completar cosas que ya conoce, por ejemplo, si escribo "rep" debería completar hasta "report" por sí solo.

Se corrigió este error agregando unas nuevas verificaciones a la hora de usar la funcionalidad de autocompletar el comando "report" en específico, para que a la hora de presentarse tal caso se autocomplete como: "report ".

Report money:

- El reporte podría usar menos espacio por cada entrada cuando son muchas, no es fácil leer toda la información.

Se redujo la información que se imprime en el reporte, para la venta realizada se muestra únicamente el número único, los nombres, apellidos y documento de identidad de la persona que realizó la compra y los ingresos generados a la agencia por dicha venta.

Cancel:

- El comando debería permitir cancelar ventas hechas en otras agencias, para eso

debería pedir el ID completo, no solo la parte final.

Para solucionar el problema de las cancelaciones lo que hicimos fue adaptar la lectura del parámetro para tener la agencia y el id del tiquete, luego recorrimos las agencias para obtener la estructura que recibía la función como parámetro, finalmente llamamos a la función como lo hacíamos antes pero con los parámetros arreglados.

Consolidate:

- La idea del comando es dejar solo una línea por cada tiquete: si luego de la compra ha sido modificado, dejar solo la línea de la modificación; si ha sido cancelado, no incluir ese tiquete en el archivo.

Se realizó una nueva verificación en la función de escribir archivos, para que solo se incluyan en el archivo los tiquetes vendidos y los tiquetes nuevos de los que han sido modificados.