TAD Sistema_Agencias

Estado:

Sesion_Activa: Referencia a la Agencia de Viajes activa
Rutas: Vector de rutas aéreas
Agencias: Lista de agencias de viajes
Comandos: Lista de comandos
Autocompletar: Árbol con palabras para autocompletar

Interface:

hacer_login(nombre_agencia, contrasena) : bool
    Pre: nombre_agencia debe existir y la contraseña debe ser alfanumérica. La Sesion_Activa debe ser igual a null.
    Post: Si el nombre_agencia son correctas, se accede al sistema y se permite trabajar.
hacer_logout( ) : bool
    Pre: La Sesion_Activa debe ser diferente de null
    Post: Se cierra la Sesion_Activa
salir( )
    Pre: El programa debe estar activo
    Post: Se cierra la aplicación
reportar_vuelos([origen] , [fecha])
    Pre: La lista de vuelos no puede estar vacía, debe haber una sesión activa.
    Post: Reporte visualmente agradable de los vuelos disponibles.
reportar_ventas( )
    Pre: La lista de ventas no puede estar vacía, debe haber una sesión activa.
    Post: Reporte visualmente agradable del inventario de ventas.
vender_vuelo_sencillo(id_vuelo, fecha, num_identificacion, nombre, apellidos)
    Pre: El número de vuelo debe existir en la lista de rutas aéreas
    Post: Se genera una venta
buscar_ruta_aerea(id_vuelo) : bool
    Post: Se retorna si existe o no la ruta aérea
agregar_vuelo(codigo, dia, ciudad_origen, ciudad_destino, hora_vuelo, duracion, sillas_disponibles, costo_silla)
    Pre: El vuelo a agregarse no debe estar todavía en el vector de rutas aéreas
    Post: Un vuelo agregado al vector de rutas aéreas
agregar_agencia(nombre,contrasena)
    Pre: Pre: La agencia a agregarse no debe estar todavía en la lista de agencias de viajes
fillWord(dato)
    Pre: La palabra debe estar incompleta
    Post: La palabra autocompletada
consolidate()
    Pre: Debe haber ventas cuyas fechas de vuelo ya hayan pasado
    Post: Lista de ventas con solo las ventas de vuelos todavía no realizados
reportar_dinero()
    Pre: Que haya ventas realizadas por la agencia activa

Post: Información con las ventas y totales recibidos por la empresa y por pagar

buscar_venta(id)

Pre: El id debe ser válido

Post: La venta encontrada

cancelar_venta(id)

Pre: El id debe ser válido y la venta que corresponda no debe estar cancelada anteriormente

Post: Venta cancelada y agregada al vector de ventas

agregar_venta(id, id_vuelo, apellidos, nombre, identificacion, fecha_vuelo, fecha_venta, hora_venta)

Post: Venta agregada exitosamente a su correspondiente agencia

agregar_comando(comando, parametro)

Pre: Los comandos sean correctos

Post: Comandos con su respectiva información agregadas a la lista de comandos

llenarArbol()

Pre:Los comandos sean correctos y las palabras sean concisas

Post: Árbol y sub-arboles llenados con todas las palabras correctamente

llenarGrafo()

Pre:Deben existir vuelos en el sistema

Post:Grafo llenado correctamente

GetPaths(ciudad origen, ciudad destino) Vector de deque de cadena de caracteres

Pre: GetShorestPath ya implementado

Post: Los tres caminos mas cortos bajo los criterios especificados

CambiarVuelo(id pasajero, id vuelo original, id vuelo nuevo, fecha_vuelo, diferencia precio) Apuntador a la venta original

Pre: Que el vuelo nuevo y la venta del vuelo original exista.

Post: Vuelo correctamente cambiado

TAD Ruta_Aerea

Estado:

Codigo: Cadena de Caracteres

Dia: Cadena de Caracteres

Ciudad_Origen: Cadena de Caracteres

Ciudad_Destino: Cadena de Caracteres

Hora_Vuelo : Hora

Sillas_Disponibles: Numero natural

sillas >= 0

Costo_Silla: Numero Natural

costo >= 0

Interface:

Disminuir_sillas_disponibles( )

Pre: Las sillas disponibles deben ser mayor a 0

Post: Las sillas se disminuyen en 1

TAD Agencia

Estado:

Ventas: Vector de tipo ventas que contiene las ventas de la aerolínea.
Nombre Agencia: Cadena de caracteres.
Contraseña: Cadena de caracteres.

Interface:

agregar_venta(identificador, cod_vuelo, num_identificacion, apellidos, nombres, fecha_vuelo, fecha_venta, hora_venta)

Post: Venta agregada al vector de ventas

imprimir_ventas()

Pre: Deben existir ventas
Post: Información de las ventas organizadas correctamente para imprimir

consolidate()

Pre: Debe haber ventas cuyas fechas de vuelo ya hayan pasado
Post: Lista de ventas con solo las ventas de vuelos todavía no realizados

eraseFromFile(identificador, fechaVuelo)

Pre: Debe haber ventas cuyas fechas de vuelo ya hayan pasado
Post: Ventas consolidadas en el archivo de tickets

buscar_venta(id)

Pre: Deben existir ventas por parte de la Agencia
Post: Venta encontrada por id

buscar_cedula(id)

Pre: Deben existir ventas por parte de la Agencia
Post: Venta encontrada por cédula

TAD Venta

Estado:

Identificador: Número natural.
Cod_vuelo: Cadena de Caracteres.
Num_Identificacion: Cadena de Caracteres.
Apellidos: Cadena de Caracteres.
Nombres: Cadena de Caracteres.
Fecha_vuelo: Fecha
Fecha_venta: Fecha
Hora: Hora

Interface:

TAD Fecha

Dia: Numero natural
0 < Dia <= 31
Mes: Numero natural

0 < Mes <= 12

Anno: Numero natural

año > 0

Interface:

TAD Hora

Estado:

Hora: Numero natural

0 <= Hora < 24

Minuto: Numero natural

0<= Minuto < 60

Interface:

TAD Comando

Nombre: Cadena de Caracteres

Parámetros: Cadena de Caracteres

Interface:

TAD Trie

Estado:

m_root: raíz del árbol trie

Interface:

Insert(valor)

Pre: La palabra no debe estar en el árbol anteriormente y debe existir raíz

Post: Palabra agregada al árbol

InsertInto(key, valor)

Pre: La palabra no debe estar en el sub-árbol anteriormente

Post: Palabra agregada al sub-árbol

Find(valor1, valor2)

Pre: La palabras o parte de alguna palabra debe estar en el árbol o en los sub-arboles y debe existir raíz

Post: Retorna la palabra autocompletada o las sugerencias para autocompletar

Weight()

Post: Cuenta el número de nodos del árbol

PrintAsPNG( )

Pre: Debe existir raiz

Post: Genera un archivo .png con el árbol ilustrado

TAD Node

Estado:

m_Valor: carácter
m_Children: Lista de hijos de nodos
m_Trie: Sub-árbol que la posee cada hoja del árbol

Interface:

Insert(valor)
Pre: La palabra no debe estar en el árbol anteriormente
Post: Palabra agregada al árbol
InsertInto(valor)
Pre: La palabra no debe estar en el sub-árbol anteriormente
Post: Palabra agregada al sub-árbol
Find(valor)
Pre: La palabra o parte de la palabra debe estar en el árbol
Post: Retorna la palabra autocompletada o las sugerencias para autocompletar
Find(valor, contador)
Pre: La palabra debe estar en el árbol y/o en los sub-árboles
Post: Retorna el nodo de la palabra encontrada

AddChild(valor)
Post: Agregado un nuevo hijo a un nodo
IsLeaf()
Pre: Debe haber nodos en el árbol
Post: Rectifica si un nodo tiene más hijos o no
Weight()
Post: Cuenta el número de nodos del árbol
PrintAsPNG( )
Pre: Debe haber nodos en el árbol
Post: Genera un archivo .png con el árbol ilustrado

TAD Graph

Estado:

Vertices: Vector de valores que representa cada nodo
Costs: Matriz de costos que hay entre un vértice y otro
TCosto: Vector de Rutas Aéreas

Interface:

AddNode(Vertice) : Natural
Pre: El vértice no debe estar añadido previamente
Post: Vector con un nuevo vértice
AddArc (Indice Vertice a, Indice Vertice b, Costo)
Pre: La arista no debe tener un valor todavía
Post: Arista con el costo añadido a la matriz de costos
HasArc (Indice Vertice a, Indice Vertice b) : booleano
Pre: La matriz de costos debe existir

Post: Retorna si existe o no la arista

GetNode(Indice Vértice) : Cadena de caracteres

Pre: El vértice debe existir

Post: Retorna el vértice asociado

GetDijkstra (Indice Vértice semilla, criterio) : Vector de números largos

Pre: Debe existir el vértice asociado al indice

Post: Vector que representa un árbol de recubrimiento

GetDijkstraDirect (Indice Vértice semilla) : Vector de números enteros largos

Pre: Debe existir el vértice asociado al indice

Post: Vector que representa un árbol de recubrimiento

GetShortestPath(ndice Vértice a, Indice Vértice b, criterio) Deque de cadena de caracteres

Pre: Árbol de recubrimiento mínimo

Post: Camino mas corto según criterio especificado

GetShortesPath (v, v, criterio) Deque de cadena de caracteres

Pre: Árbol de recubrimiento mínimo

Post: Camino mas corto según criterio especificado

GetSize() Entero Largo

Pre: Vector de vértices creado

Post: Tamaño de vector de vértices