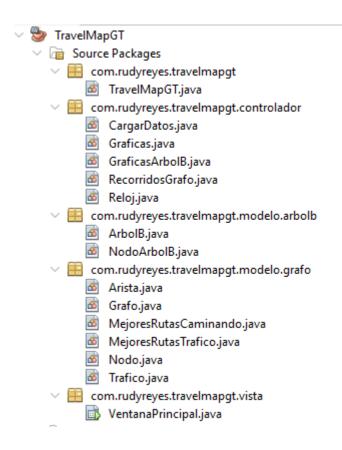
MANUAL TÉCNICO

1. Introducción y Propósito:

El siguiente proyecto contiene un software desarrollado en java, cuya función permite a las personas de Guatemala organizar sus viajes desde un punto A a un punto B por medio de las siguientes opciones: Vehículo y Caminando.

Arquitectura y Diseño:



Esta es la estructura del proyecto que esta basada en el modelo arquitectónico MVC (Modelo-Vista-Controlador).

Explicación de los paquetes:

controlador: Aqui se encuentran todos los paquetes que nos permitirá obtener las acciones del usuario en la Vista y procesarla aquí haciendo uso del modelo.

modelo: Son las clases necesarias para el funcionamiento del sistema, y para guardar los datos de este.

vista: Contiene la ventana principal, que será mostrada al usuario

Detalle de las Clases:

Modelo:

∨
☐ com.rudyreyes.travelmapgt.modelo.arbolb

ArbolB.java

ModoArbolB,java

— E com.rudyreyes.travelmapgt.modelo.grafo

Arista.java

Grafo.java

MejoresRutasCaminando.java

MejoresRutasTrafico.java

Modo.java

Trafico.java

Como ya se había mencionado anteriormente estas clases nos permiten el funcionamiento del software, estas son necesarias para definir los objetos del sistema, como lo son los árboles, grafos y nodos.

Clase Nodo:

Nodo Nodo

Nodo(String nombreOrigen, ArrayList<Arista> destinos)

Nodo()

Nodo(String nombreOrigen)

destinoExiste(String nombreDestino): boolean

getDestino(String nombreDestino): Arista

getDestinos(): ArrayList<Arista>

getNombreOrigen(): String

imprimirDatos()

setDestino(Arista destino)

setDestinos(ArrayList<Arista> destinos)

setNombreOrigen(String nombreOrigen)

Esta clase contiene los atributos de un nombreOrigen y un arreglo de destinos, este arreglo contiene objetos del tipo Arista, más adelante se explicará que elementos contiene esta arista, los demás métodos conformado por esta clase, son métodos getters y setters.

Clase Arista:

/ 🖍 Arista

Arista(Nodo destino, int tiempoVehiculo, int tiempoPie, int consumoGas, int desgastePersona, int distancia)

Arista()

getConsumoGas(): int

getDesgastePersona(): int

getDestino(): Nodo

getDistancia(): int

getTiempoPie(): int

getTiempoVehiculo(): int

getTrafico(): List<Trafico>

setConsumoGas(int consumoGas)

setDesgastePersona(int desgastePersona)

setDestino(Nodo destino)

Los atributos que contiene esta clase son un nodo Destino y atributos del tipo entero como lo son el tiempo,, el desgaste y etc. métodos getters y setters.

Clase Grafo:



- Grafo()
- agregarNodo(Nodo nodo)
- agregarNodoNoDirigido(Nodo nodo)
- buscarNodo(String nombreLugar): Nodo
- buscarNodoNoDirigio(String nombreLugar): Nodo
- getNodosOrigen(): HashMap<String, Nodo>
- getNodosOrigenNoDirigido(): HashMap<String, Nodo>
- imprimirGrafo(): String
- imprimirGrafoNoDirigido(): String
- imprimirNodos()
- imprimirNodosNoDirigido()
- verificarNodo(String nombreLugar): boolean
- nodosOrigen: HashMap<String, Nodo>
- nodosOrigenNoDirigido : HashMap<String, Nodo>

atributos Los de esta clase prácticamente son dos, y mapa de NodosOrigen У otro mapa NodosOrigenNoDirigido, el primero tiene la función de mapear un grafo dirigido y el segundo un grafo no dirigido, los demás son getters y setters además clases para imprimir los datos.

Clase ArbolB:



- ArbolB(int t)
- buscarClaveMayor(): int
- buscarNodoMinimo(NodoArbolB nodoActual): NodoArbolB
- buscarNodoPorClave(int num)
- claveMayorPorNodo(NodoArbolB current): int
- getClaveMayor(NodoArbolB current): int
- insertar(int key)
- mostrarClavesNodoMinimo()
- nonFullInsert(NodoArbolB x, int key)
- print(NodoArbolB n)
- search(NodoArbolB actual, int key): NodoArbolB
- showBTree()
- 🦥 split(NodoArbolB x, int i, NodoArbolB y)
- root: NodoArbolB

Esta clase nos permite manejar el árbol b del sistema, contiene todos los metodos necesarios, para la insercion y busqueda de datos, ademas del balanceo automatico para que siempre se mantenga ordenado.

Controlador:



E com.rudyreyes.travelmapgt.controlador

CargarDatos.java

Graficas.java

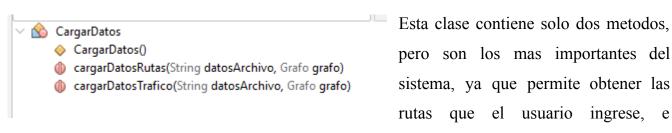
GraficasArbolB.java

RecorridosGrafo.java

Reloj.java

Clases que nos permiten estructurar los datos que el usuario nos manda desde la vista y agregarlos a los grafos.

Clase CargarDatos:



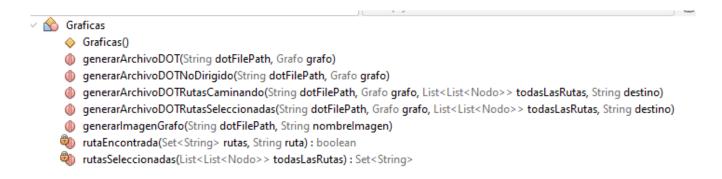
ingresarlas al grafo, prácticamente divide los datos, obtiene los Nodos Origen y Destino, y los ingresa al grafo con sus respectivas aristas.

Clase RecorridosGrafo:



Esta sin duda es la segunda clase mas importante del software ya que nos permite obtener los recorridos de un NodoInicio y un NodoFinal, los dos metodos importantes son: encontrarCaminosVehiculo: Nos devuelve un listado de rutas posibles si el usuario va en vehiculo.

Clase Graficas:



Estas clase nos genera las imagenes de los grafos, nos puede generar 4 tipos de imagenes:

generarArchivoDot: Nos genera el Dot del grafo de todo el sistema con direcciones.

generar Archivio Dot No Dirigido: Nos genera el dot del grafo completo no dirigido.

generarArchivoDotRutasCaminando: Nos genera el grafo con la diferencia de que pueden apreciar todas las rutas de un punto inicial a otro, este dot generado es un grafo no dirigido.

generarArchivoDotRutasSeleccionadas: Este es igual al anterior solo que es un grafo dirigido.

Clase Graficas ArbolB:

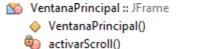


- GraficasArbolB()
- bucarPaginas(NodoArbolB n, FileWriter writer, List<String> rutas)
- generarDOT(String archivoDot, NodoArbolB root, List<String> ruta del árbol
- generarlmagenGrafo(String dotFilePath, String nombrelmagen)

Clase que contiene los métodos necesarios para generar la imagen del árbol

Vista:

Clase VentanaPrincipal:



- botonAvanzarActionPerformed(ActionEvent evt)
- botonEmpezarViajeActionPerformed(ActionEvent evt)
- botonPararHoraActionPerformed(ActionEvent evt)
- cargarRutasActionPerformed(ActionEvent evt)
- cargarTraficoActionPerformed(ActionEvent evt)
- generarArbolB()
- generarGrafoPrincipal()
- generarGrafoRutas()
- initComponents()
- main(String[] args)
- obtenerHora(): int
- recorrerViaje(String origen, String destino)

Esta clase contiene los métodos que nos permitirán obtener las acciones del usuario desde la interfaz gráfica.