Manual Técnico de TravelMapGT

Introducción

En esta sección, proporciona una visión general del proyecto, su propósito y los objetivos que se pretenden alcanzar con la aplicación TravelMapGT.

Arquitectura del Sistema

Describe la arquitectura general del sistema, incluyendo componentes principales, módulos y su interacción. Puedes utilizar diagramas de arquitectura para ilustrar la estructura del sistema.

Tecnologías Utilizadas

Java 21 con Maven: Explica cómo se utiliza Java 21 en el proyecto y cómo Maven facilita la gestión de dependencias y la construcción del proyecto.

Java Swing: Describe cómo se utiliza Java Swing para la interfaz de usuario y cómo se integra con el resto del sistema.

Graphviz: Explica el uso de Graphviz para la visualización de gráficos y cómo se integra en la aplicación.

Grafos y Árbol B: Describe cómo se implementan y utilizan los grafos y árboles B en el proyecto.

Java Swing

es un conjunto de herramientas y bibliotecas proporcionadas por Java para crear interfaces gráficas de usuario (GUI) para aplicaciones de escritorio. Aquí tienes una explicación técnica:

Componentes Swing: Swing proporciona una amplia gama de componentes, como botones, etiquetas, campos de texto, áreas de texto, listas, tablas, paneles, etc. Estos componentes se utilizan para construir la interfaz de usuario de una aplicación.

Modelo de eventos: Swing utiliza un modelo de eventos para manejar la interacción del usuario con los componentes de la GUI. Cuando ocurre un evento, como hacer clic en un botón o escribir en un campo de texto, Swing genera un evento que puede ser capturado y procesado por la aplicación.

Contenedores: Los contenedores son componentes que contienen otros componentes. Por ejemplo, un JFrame es un contenedor de nivel superior que puede contener varios componentes como botones, etiquetas, etc. Los contenedores se utilizan para organizar y colocar los componentes en la interfaz de usuario.

Diseño de GUI: Swing proporciona varios administradores de diseño que se utilizan para organizar los componentes dentro de un contenedor. Algunos de los administradores de

diseño comunes son BorderLayout, FlowLayout, GridLayout, etc. Estos administradores de diseño determinan cómo se distribuyen y colocan los componentes dentro de un contenedor.

Hilos de GUI: En Java, la interfaz de usuario se actualiza en un hilo separado conocido como el hilo de eventos de Swing o hilo de despacho de eventos. Es importante tener en cuenta que todas las operaciones de actualización de la interfaz de usuario deben realizarse en este hilo para evitar problemas de concurrencia.

Personalización y apariencia: Swing permite personalizar la apariencia de la interfaz de usuario mediante el uso de temas y la creación de componentes personalizados. Esto permite que las aplicaciones Swing tengan un aspecto y una sensación distintos según los requisitos del proyecto.

Internacionalización y accesibilidad: Swing proporciona soporte para la internacionalización y la accesibilidad mediante el uso de clases y métodos que permiten la traducción de texto y la mejora de la accesibilidad para usuarios con discapacidades.

Grafos

Generación de Grafo (generateGraph):

Este método toma una cadena de contenido que representa la información de los nodos y las conexiones entre ellos.

Crea un archivo .dot que define el grafo utilizando el formato DOT, un lenguaje de descripción de gráficos.

Recorre cada línea de la cadena de contenido, que parece contener información sobre las conexiones entre los nodos.

Divide cada línea para obtener los detalles de la conexión entre nodos y los agrega al archivo .dot.

Utiliza la biblioteca ProcessBuilder para ejecutar el comando dot (una herramienta de Graphviz) y generar una imagen PNG del grafo a partir del archivo .dot.

Selección de Nodos (selectGraph):

Este método permite resaltar una ruta específica o nodo en el grafo generado anteriormente.

Lee el archivo .dot generado anteriormente y guarda su contenido en un StringBuilder.

Divide el contenido en líneas para procesarlo.

Recorre cada línea del archivo .dot y verifica si contiene la relación entre los nodos seleccionados.

Si encuentra la relación entre los nodos seleccionados, modifica el color de los nodos seleccionados a rojo en el archivo .dot.

Utiliza nuevamente la biblioteca ProcessBuilder para regenerar la imagen PNG del grafo con los nodos seleccionados resaltados.

Graphviz

Graphviz es una colección de herramientas para visualizar gráficos y redes. Se basa en la representación de gráficos mediante el lenguaje DOT (Graph Description Language), que es un lenguaje de descripción de gráficos desarrollado por AT&T Labs. Aquí tienes una explicación más detallada:

Lenguaje DOT: Es un lenguaje de descripción de gráficos utilizado por Graphviz para especificar la estructura de los gráficos. Permite definir nodos, conexiones entre nodos, atributos visuales como colores y formas, y más. Es un lenguaje declarativo que proporciona una manera fácil y compacta de describir la topología de un gráfico.

Herramientas de Graphviz: Graphviz proporciona varias herramientas para trabajar con gráficos DOT:

Dot: Es el programa principal de Graphviz que toma como entrada un archivo DOT y produce como salida un gráfico renderizado en diferentes formatos, como PNG, SVG, PDF, entre otros.

Neato, Fdp, Circo, Twopi, etc.: Son programas alternativos de Graphviz que también toman archivos DOT como entrada, pero implementan diferentes algoritmos de disposición para dibujar los gráficos. Por ejemplo, Neato utiliza un algoritmo de disposición basado en fuerzas para organizar los nodos.

GVEdit: Es una herramienta gráfica que proporciona una interfaz de usuario para crear y editar archivos DOT, y visualizar los gráficos resultantes en tiempo real.

Renderizado de gráficos: Una vez que se ha definido un gráfico en el formato DOT, Graphviz utiliza algoritmos internos para renderizar el gráfico en un formato visualmente atractivo. Esto incluye la disposición de los nodos y las conexiones de manera que sean fáciles de entender y seguir.

Personalización: El lenguaje DOT y las herramientas de Graphviz permiten una amplia personalización de los gráficos. Los usuarios pueden especificar atributos visuales como colores, formas, estilos de línea, etiquetas y más, para adaptar el aspecto del gráfico a sus necesidades específicas.

Integración: Graphviz se puede integrar fácilmente en aplicaciones y flujos de trabajo existentes. Esto permite a los desarrolladores generar gráficos de manera programática a partir de datos estructurados, como resultados de algoritmos, bases de datos, archivos de configuración, entre otros.

Arbol B

Un árbol B es una estructura de datos de árbol diseñada especialmente para trabajar con datos almacenados en disco o en memoria secundaria. Está optimizado para sistemas que manejan grandes cantidades de datos y necesitan realizar operaciones eficientes de inserción, búsqueda y eliminación. Aquí tienes una explicación del código proporcionado:

Definición del Árbol B:

El árbol B se define como una clase ArbolB.

Tiene dos atributos principales: raiz, que representa el nodo raíz del árbol, y gradoMaximo, que indica el grado máximo del árbol B.

El grado máximo es el máximo número de hijos que puede tener un nodo, excepto la raíz.

Constructor:

El constructor de ArbolB inicializa la raíz como nula y establece el grado máximo del árbol.

Agregar Lista de Nodos:

El método agregarListaNodos agrega una lista de nodos al árbol.

Si la lista es demasiado grande para el grado máximo del árbol, se divide en sublistas.

Cada sublista se asigna a un nuevo nodo hijo, y estos nodos se convierten en hijos del nodo intermedio.

Si la raíz es nula, el nodo intermedio se convierte en la nueva raíz del árbol.

Si la raíz no es nula, no se implementa la lógica para agregar nodos intermedios al árbol.

Dividir Lista:

El método dividirLista divide una lista en sublistas de tamaño igual al grado máximo menos uno.

Esto se hace para preparar la lista para su inserción en el árbol B.

Clase NodoB:

La clase NodoB es una clase interna estática que representa un nodo en el árbol B.

Tiene dos atributos: hijos, que es una lista de nodos hijos (nodos intermedios), y listaNodos, que es una lista de nodos (nodos hoja).

Proporciona métodos getter y setter para acceder y modificar estos atributos.

Funcionalidades Principales

Organización de Viajes

Describe cómo los usuarios pueden organizar sus viajes desde un punto A a un punto B utilizando la aplicación.

Explica las opciones disponibles, como viajar en vehículo o caminando, y cómo se calculan las rutas más idóneas.

Detalla los elementos que se consideran en el cálculo de las rutas, como la distancia, el tiempo de recorrido y la eficiencia.

Implementación Técnica

Proporciona detalles sobre la implementación técnica de las funcionalidades principales.

Explica cómo se manejan y procesan los datos de entrada, cómo se realizan los cálculos de las rutas y cómo se presenta la información al usuario.

Incluye fragmentos de código relevantes para ilustrar la implementación.

Archivo de entrada

El archivo de entrada proporcionado es un archivo de texto plano que contiene datos relacionados con rutas entre diferentes lugares geográficos. Cada línea del archivo representa una ruta entre dos lugares, y los datos están organizados de la siguiente manera:

Origen: Es la ubicación de partida de la ruta.

Destino: Es la ubicación de llegada de la ruta.

Tiempo en Vehículo: Indica el tiempo promedio estimado de llegada en vehículo desde el origen hasta el destino.

Tiempo Caminando: Representa el tiempo aproximado que tomaría llegar caminando desde el origen hasta el destino.

Gasto en Combustible: Indica el gasto estimado en combustible para el viaje en vehículo desde el origen hasta el destino.

Gasto Físico: Representa el esfuerzo físico estimado que una persona tendría al realizar este recorrido caminando desde el origen hasta el destino.

Distancia: Es la distancia en kilómetros entre el origen y el destino de la ruta.

Pom xml

es un archivo de configuración de Maven que define la estructura y las dependencias de un proyecto. Aquí tienes una explicación de su contenido:

Cabecera XML:

La primera línea especifica la versión de XML utilizada y la codificación del archivo.

Elemento <project>:

Este elemento raíz define toda la configuración del proyecto.

Utiliza los atributos xmlns, xmlns:xsi y xsi:schemaLocation para declarar el espacio de nombres y la ubicación del esquema XML de Maven.

<modelVersion>:

Especifica la versión del modelo del POM utilizado. En este caso, se utiliza la versión 4.0.0.

<groupId>, <artifactId>, <version>:

Estos elementos definen la identificación única del proyecto. groupId identifica al grupo o entidad que creó el proyecto, artifactId es el identificador único del proyecto y version es la versión actual del proyecto.

<packaging>:

Define el tipo de empaquetado del proyecto. En este caso, se empaqueta como un archivo JAR.

<properties>:

Este elemento define propiedades personalizadas utilizadas en el proyecto. Por ejemplo, project.build.sourceEncoding define la codificación de fuente del proyecto, y maven.compiler.source y maven.compiler.target definen la versión de Java que se utilizará para compilar el código fuente.

<build>:

Define la configuración del proceso de construcción del proyecto.

Dentro de este elemento, se configuran los plugins utilizados durante la construcción del proyecto.

En este caso, se utiliza el plugin maven-jar-plugin para configurar la generación del archivo JAR.

<plugins>:

Contiene la configuración de los plugins utilizados en el proyecto.

En este caso, se configura el plugin maven-jar-plugin para generar el archivo JAR del proyecto.

Se especifica la clase principal del proyecto (main.Interfaz) que se ejecutará al lanzar el archivo JAR.

<resources>:

Define la configuración de los recursos utilizados en el proyecto.

En este caso, se especifica que los recursos se encuentran en el directorio src/main/resources y que no se aplicará ningún filtro a estos recursos.

<dependencies>:

Contiene las dependencias del proyecto.

En este caso, se especifica una dependencia externa de Maven, AbsoluteLayout, que se utilizará en el proyecto.

Pruebas

Describe las pruebas realizadas en el sistema para garantizar su calidad y fiabilidad.

Incluye información sobre las pruebas unitarias, de integración y de aceptación realizadas, así como los resultados obtenidos.

Consideraciones de Despliegue

Detalla los requisitos de hardware y software para desplegar la aplicación TravelMapGT.

Proporciona instrucciones para la instalación y configuración del sistema en un entorno de producción.

Mantenimiento y Mejoras Futuras

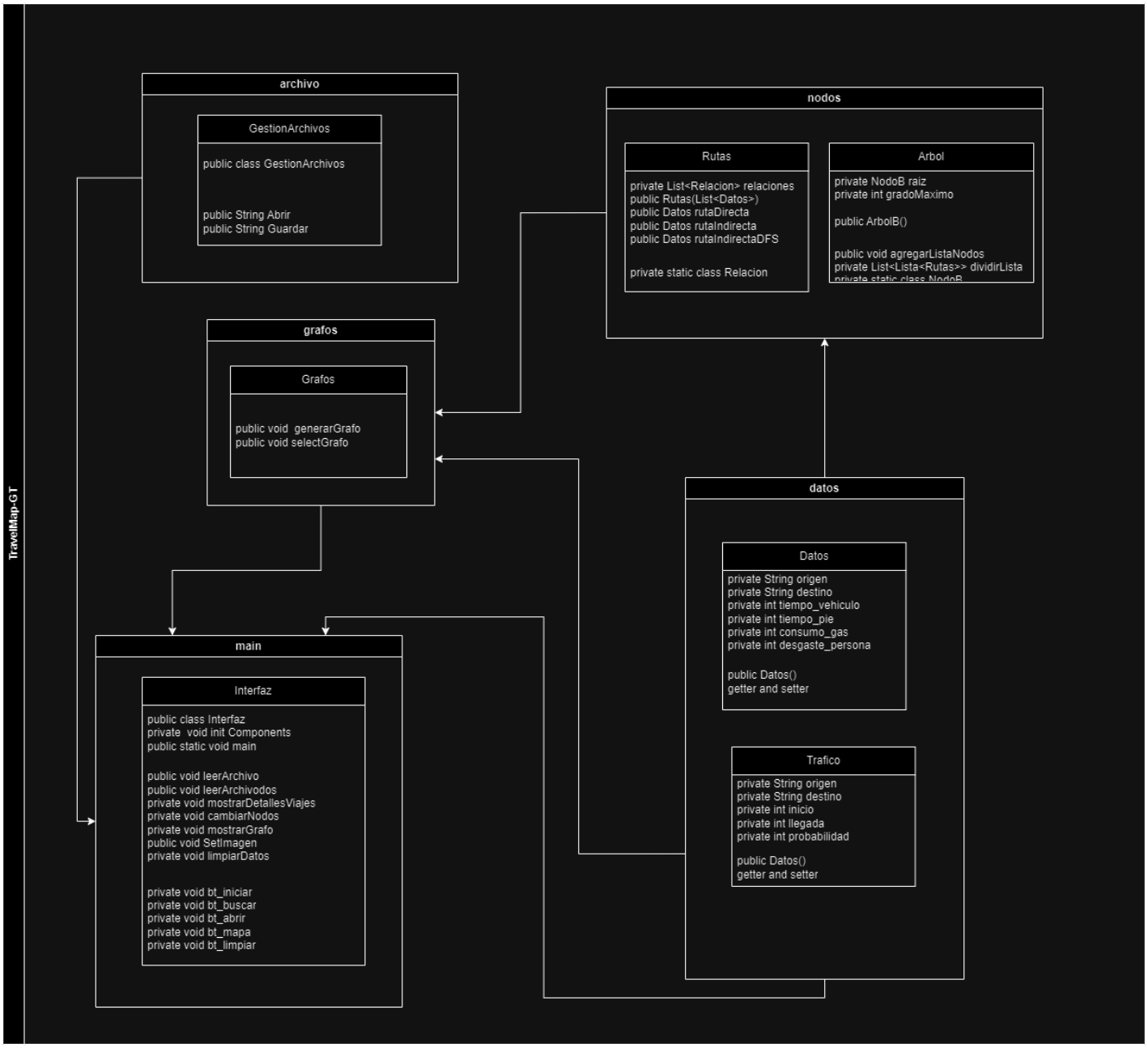
Discute las consideraciones para el mantenimiento continuo de la aplicación TravelMapGT.

Proporciona sugerencias para posibles mejoras y nuevas características que podrían implementarse en versiones futuras.

Conclusiones

Resumen de los puntos clave del manual técnico y reflexiones finales sobre el proyecto TravelMapGT.

Diagramas


TravelMap-GT
archivo
GestionArchivos
public class GestionArchivos
public String Abrir
public String Guardar
nodos
Rutas
private List<Relacion> relaciones
public Rutas(List<Datos>)
public Datos rutaDirecta
public Datos rutaIndirecta
public Datos rutaIndirectaDFS
private static class Relacion
Arbol
private NodoB raiz
private int gradoMaximo
public ArbolB()
public void agregarListaNodos
private List<Lista<Rutas>> dividirLista
grafos
Grafos
public void generarGrafo
public void selectGrafo
datos
Datos
private String origen
private String destino
private int tiempo_vehiculo
private int tiempo_pie
private int consumo_gas
private int desgaste_persona
public Datos()
getter and setter
Trafico
private String origen
private String destino
private int inicio
private int llegada
private int probabilidad
public Datos()
getter and setter
main
Interfaz
public class Interfaz
private void init Components
public static void main
public void leerArchivo
public void leerArchivodos
private void mostrarDetallesViajes
private void cambiarNodos
private void mostrarGrafo
public void SetImagen
private void limpiarDatos
private void bt_iniciar
private void bt_buscar
private void bt_abrir
private void bt_mapa
private void bt_limpiar