**UNIVERSIDAD MARIANO GÁLVEZ DE GUATEMALA**

**CENTRO UNIVERSITARIO DE ESCUINTLA**

**FACULTAD DE INGENIERÍA**

**Carrera**: Ingeniería en Sistemas de Información

**Curso**: Programación III



**Proyecto Final**

**ESTUDIANTES:**

Suarlyn Rodolfo Juárez Quevedo- 2190-22-11499

Diego Andreé López Baches- 2190-23-626

**LUGAR Y FECHA:**

Escuintla, Escuintla 20-07-2024



**Sistema de rutas de entrega**

**Introducción**

Nuestro proyecto se basa en la simulación de un sistema de rutas de entrega, en donde el usuario debe de realizar la petición de pedido de algún producto, luego realizar la petición nuestro algoritmo busca la ruta más corta hacía la ubicación del usuario. Optimizando así el tiempo y recursos para poder llegar hacia él.

**Problema**

Una empresa presenta el problema de que sus rutas de entrega se encuentran algo largas y han obtenido quejas de sus clientes, por lo cual se desea crear un sistema de creación de rutas para la optimización de una empresa encargada de la entrega de paquetes hacia clientes.

Estructura del proyecto

Clases

* Grafo
* Archivador
* Vértice
* Arista
* CargadorVecinos

Grafo

Vertice

namespace Grafos{

    public class Vertice

    {

        public int id { get; set; }

        public string departamento { get; set; }

        public string tipo { get; set; }

        public List<Arista> aristas { get; set; }

        public Vertice(int id, string departamento, string tipo)

        {

            this.id = id;

            this.departamento = departamento;

            this.tipo = tipo;

            aristas = new List<Arista>();

        }

        public void agregarArista(Arista arista)

        {

            aristas.Add(arista);

        }

    }

}

En la cual contamos con 4 atributos de clase, que son los siguientes:

Id -> Este atributo lo utilizamos para la identificación rápida de un Vértice en específico.

**CargadorVecinos**

Define una clase estática llamada CargadorVecinos que contiene un método para cargar información de departamentos vecinos desde un archivo JSON.

**Explicación paso a paso**

1. **Importación de namespaces**

using System.Text.Json;

Se importa el espacio de nombres necesario para trabajar con JSON.

1. **Clase estática**

public static class CargadorVecinos

La clase es estática, por lo que no necesita instanciarse para usar sus métodos.

1. **Método principal**

public static Dictionary<string, List<string>> CargarVecinosDesdeJson(string ruta)

Es un método estático que recibe la ruta de un archivo JSON.

Devuelve un diccionario donde la clave es el nombre de un departamento y el valor es una lista de nombres de departamentos vecinos.

1. **Lectura y deserialización**

string contenido = File.ReadAllText(ruta);

var vecinos = JsonSerializer.Deserialize<Dictionary<string, List<string>>>(contenido);

Lee todo el contenido del archivo especificado en [ruta](vscode-file://vscode-app/c:/Users/PC/AppData/Local/Programs/Microsoft%20VS%20Code/resources/app/out/vs/code/electron-sandbox/workbench/workbench.html).

Deserializa el contenido JSON a un diccionario de tipo [Dictionary<string, List<string>>](vscode-file://vscode-app/c:/Users/PC/AppData/Local/Programs/Microsoft%20VS%20Code/resources/app/out/vs/code/electron-sandbox/workbench/workbench.html" \o ").

1. **Manejo de errores**

catch (Exception ex)

{

  Console.WriteLine($"Error cargando vecinos: {ex.Message}");

    return new Dictionary<string, List<string>>();

}

* + Si ocurre algún error (por ejemplo, el archivo no existe o el JSON está mal formado), muestra un mensaje en consola y retorna un diccionario vacío.