Universidad de San Carlos de Guatemala

Facultad de Ingeniería

Escuela de Ciencias y Sistemas

Estructuras de Datos

Catedrático:

Ing. Edgar Rene Ornelis Hoil

Tutor académico:

Elian Saúl Estrada Urbina

“MANUAL TECNICO PROYECTO 2”

Melvin Eduardo Ordoñez Sapón RA | 3256215860802

Guatemala, diciembre del 2024

“Id y enseñad a todos”

ACERCA DEL PROGRAMA:

Editor utilizado: Visual Studio Code

Versión de python: 3.12.3

Programado en Sistema Operativo: Ubuntu 24.04 LTS

Terminal Utilizada: Terminal de Linux

Lenguaje de Programación Utilizado: Python

Compatibilidad: Windows, Linux, macOS

Versión Del Programa: 1.0

Herramientas Utilizadas: Graphviz

ESTRUCTURAS UTILIZADAS

ArbolB:

```
# Constructor de la clase

def __init__(self, orden: int):
self.raiz: NodoArbolB = NodoArbolB(True) # Nodo raíz inicial
self.orden: int = orden # Orden del árbol B
```

Esta estructura que implementa un árbol B, que es una estructura de datos equilibrada para manejar grandes cantidades de datos. En este proyecto se utiliza para almacenar vehiculos

```
def insertar_valor(self, vehiculo: Vehiculo):
```

Inserta un objeto `Vehiculo` en el árbol. Si la raíz se llena, la divide y crea una nueva raíz.

```
def insertar_valor_no_completo(self, nodo: NodoArbolB, vehiculo: Vehiculo):
```

Inserta un vehículo en un nodo que no está lleno. Si el nodo es hoja, lo inserta directamente; si no, busca el hijo adecuado para continuar.

```
def dividir_pagina(self, raiz: NodoArbolB, posicion: int):
```

Divide un nodo lleno en dos. Mueve la clave media al nodo padre y ajusta los hijos.

Grafo:
Esta estructura utiliza lo siguiente.

```
def __init__(self):
self.vertices: Lista[Vertice] = Lista();
```

Esto de la clase lista de adyacencia y lo que hace es inicializar la lista que contendrá los diferentes vértices.

```
def insertar(self, ruta: Ruta):
```

Este es el metodo principal para agregar una ruta al grafo, y lo que hace es es que busca si el vertice origen ya existe en la lista, si existe agrega el destino como vecino de ese vertice. Si no existe crea un nuevo vertice origen, lo agrega a la lista de vertices, y luego agrega el destino como su vecino

Lista Circular Doblemente Enlazada

```
class NodoDoble:

def __init__(self, cliente: Cliente):
self.cliente = cliente
self.siguiente = None
self.anterior = None
```

Esta es la clase del nodo que es la principal para la lista doble enlazada, lo que hacer es que maneja un nodo siguiente y anterior. con la lógica de que al momento de llegar al final este apuntara de nuevo al inicio y viceversa

Aplicacion:

```
from src.frontend.Aplicacion import Aplicacion

from tkinter import Tk

def main() -> None:
root = Tk();
app = Aplicacion(root);
root.mainloop();
if __name__ == "__main__":
main()
```

Esta es la clase que tiene el main y lo que hace es llamar a la clase aplicación en donde están todos los objetos necesarios para la lógica del proyecto y con la interfaz gráfica para el usuario