ROUTELOGIC APP

"Donde la ingeniería de software encuentra la eficiencia empresarial."

OsDeCo presenta el desarrollo de la aplicación ROUTELOGIC APP que ayuda a la optimización de las actividades logísticas dentro de las empresas, facilitando las decisiones de los involucrados en la logística sin perder la eficacia o precisión de los resultados

En nuestro mundo, nuestra sociedad, cada estado tiene un factor muy importante que determina el tipo de calidad de vida de sus habitantes el cual es la economía y ¿Cómo es que se desarrolla la economía dentro de cada país? Se desarrolla a través del intercambio de bienes o servicios a través de la comercialización entre países, la comercialización es el punto clave que ayuda a que el nivel de la economía de cierto país aumente o en su defecto disminuya con la importación o exportación de bienes o servicios dentro de este contexto entra una ciencia humana que nació en 1838 por el teórico militar Barón de Jomini llamada "logística" surgiendo de los campos de batalla debido al movimiento de capital humano de un punto geográfico a otro, junto con suministros, armamento para el ejercito en la contabilidad de las armas, suministros, su conceptualización no solo se derivaba del movimiento de los ya mencionados también de la contabilidad, administración y organización, es decir también se encargaba de que todos los movimientos y recursos ocupados fuesen lo suficiente para poder combatir las necesidades que surgieran y que fuesen lo mejor optimizados para ahorrar tiempo, recursos económicos y incluso capital humano, la logística antes era conocida como "administración y organización" hasta 1838 en que Barón de Jomini elevo la logística como una de las 3 ramas principales en el arte de la guerra.

En la actualidad la logística es la ciencia que permite el movimiento de bienes y servicios de manera global, es la ciencia que permite a las grandes empresas poder movilizar sus productos ya sean tangibles o intangibles en una red de comercialización global y no solo les permite hacer llegar los productos al punto final, también les permite optimizar tiempos, recursos, costos, este punto es muy importante ya que es este arte lo que hace posible que un grupo socioeconómico(empresa) logre que su producto llegue a manos del consumidor y asi lograr el objetivo principal de toda empresa, satisfacer las necesidades de la sociedad, dentro de las empresas en su actividades realizadas para hacer esto posible existe un algoritmo o serie de pasos que llevan a cabo para logar a su cometido llamada cadena de suministro en la que se engloban las actividades desde la adquisición de la materia prima hasta que el producto llega a manos del consumidor.

Nos adentraremos en un punto esencial de la cadena de suministro, la etapa de distribución, esta etapa es muy esencial ya que de esta depende la percepción que el cliente pueda tener de nuestra empresa ya que es el momento en el que el producto pueda sufrir algún daño en su integridad física, sufrir extravío, o percances que modifiquen su tiempo de entrega hacia al cliente, faltando a la fecha estimada que se le otorgo o incluso al producto que fue adquirido, la empresa debe de estar preparada para ofrecer la entrega mas optima, eficaz y precisa y estar preparada para poder dar frente ante todo tipo de percance, esto no solo dará una imagen buena hacia nuestros consumidores, también generaríamos imagen ante el mercado, desarrollaríamos una competitividad significativa ante nuestra competencia y con ello un aumento de ganancia debido a la superioridad ante la competencia, debido a que se pretende tener una preferencia de parte de los consumidores.

Los ingenieros en logística, se enfrentan a varios factores a tener en cuenta en el momento de seleccionar las rutas de distribución, no solamente se trata en tomar la ruta mas corta o con cualquier tipo de transporte, se trata de analizar el tipo de producto, si es perecedero, no perecedero, si es frágil o no, si es pesado, rígido o si ocupa algún estándar de temperatura para guardar su integridad durante el transporte del producto, también tomar en cuenta riesgos dependiendo la modalidad de transporte y el clima y ya tomando en cuenta tales factores determinar la ruta mas optima para la distribución de los productos, tener un registro de toda ruta para ante cualquier percance la empresa pueda actuar de inmediato de una manera precisa y eficaz para garantizar la entrega del producto en optimas condiciones y cumpliendo con la fecha establecida al consumidor, por otra parte también obtener los totales de costos en la transportación de los productos, saber cuanto le costara a la empresa y poder reducir costos con las elecciones adecuadas dependiendo el tipo de producto al reducir costos de manera en que no se pierda la eficacia de la empresa estamos consiguiendo mayor porcentaje de utilidades económicas para la empresa, lo cual también es uno de los objetivos principales de toda empresa ya que incluso en la definición de empresa se define como: grupo socioeconómico que logra fines determinados por los mismos para satisfacer las necesidades de la sociedad y obtener beneficio en utilidades económicas.

Para tales necesidades o factores que toman en cuenta los involucrados en la comercialización de las empresas nosotros nos hemos planteado el proyecto de desarrollar un software que ayude a que tales decisiones que toman otros profesionales en un ámbito tan importante como lo es en la comercialización global sean de manera mas rápida pero no menos efectiva, garantizando toda la complejidad de la selección de ruta para la distribución de productos, un programa que tome en cuenta factores como: clima, distancia, tipo de producto, transporte, costos de transporte con el algoritmo de búsqueda de anchura(BFS, por sus siglas en inglés Breadth-First Search). La Búsqueda en Amplitud (BFS) sirve para encontrar el camino más corto en un grafo no ponderado (unweighted graph) debido a su naturaleza de exploración por niveles, aplicándolo en un registro de cada nodo como centro de distribución en una red ya sea nacional o multinacional , e incluso la salida de mercancías de los almacenes dependiendo de la modalidad de la empresa sin la necesidad de tener que usar tarjetas de almacén que son documentos con lo que los encargados determinan la salida de las mercancías dependiendo la modalidad que la empresa maneje como FIFO (First In, First Out), es un sistema de gestión de inventario que establece que los productos o mercancías que ingresan primero al almacén son los primeros en salir para ser vendidos utilizados con la estructura de cola en programación y Primeras entradas, últimas salidas" se refiere a un método de gestión de inventarios conocido como UEPS (Últimas Entradas, Primeras Salidas) o LIFO (Last In, First Out) con la estructura de pila en programación.

Garantizando una mayor facilidad en tales tomas de decisiones sin perder la precisión y eficacia en tales procesos

presentamos la construcción y desarrollo del programa

Usaremos la importación de networkx, matplotib.pyplot y tkinter.

```python
import networkx as Nx; import matplotlib.pyplot as plt; import tkinter as Tk

class main():
    def __init__(self):
        self.G=Nx.Graph()
        self.t=plt.figure(figsize=(8, 6))   # Tamaño de la ventana
        self.v=Tk.Tk()
        self.v.geometry('400x400+250+250')
        self.v.title('RouteLigic')
        self.ventana()
        self.csv()
        self.v.mainloop()
    def ventana(self):
        titulo=Tk.Label(self.v, text='Mejor Ruta', font=('Arial', 12)); titulo.pack(pady=5)
        n_name=Tk.Label(self.v, text='Origen: ', font=('Arial', 12)); n_name.pack(pady=5)
        i_name=Tk.Entry(self.v, width=10); i_name.pack(pady=5)
        n2_name=Tk.Label(self.v, text='Destino: ', font=('Arial', 12)); n2_name.pack(pady=5)
        i2_name=Tk.Entry(self.v, width=10); i2_name.pack(pady=5)
        boton_g=Tk.Button(self.v, text='Ver grafo', command=lambda: self.Grafo()); boton_g.pack(pady=5)
        boton=Tk.Button(self.v, text='Encontrar la mejor ruta: ', command=lambda: self.Grafo_ruta(i_name.get(), i2_name.get())); boton.pack(pady=5)
```

Usaremos una clase llamada “main” para poder correr el programa creando un grafo con networkx creamos el grafo con un tamaño de 8, 6 pulgadas y la creación de la ventana con tkinter con un tamaño de 400x400 en la posición de 250+250 el cual lleva por nombre “RouteLogic” en donde el usuario pueda visualizar el grafo completo de los centros de distribución a su vez de poder ingresar el punto de partida y el destino y el cual será resaltada la ruta más corta por medio de la utilización de botones.

Los datos se recogen desde un archivo .csv en donde se en listara los nombres de centros de distribución, la distancia entre ellos, la capacidad de carga y el tiempo de traslado.

```
def csv(self):
    with open('Centros.csv', 'r') as archivo:
        f=archivo.readlines()
        nodos=set()         # Cojunto para evitar nodos duplicados
        for i in f[1:]:     # split(','): Divide cada dato separados por comas y los guarda en una lista
            N1, N2, Distancia, Capacidad, Tiempo=[x for x in i.split(',')] # Da el valor a cada uno de las variables
            print(N1)
            if N1 not in nodos:
                self.G.add_node(N1, distancia=Distancia, capacidad=Capacidad)    # Crea el N1
                nodos.add(N1)
            if N2 not in nodos:
                self.G.add_node(N2, distancia=Distancia, tiempo=Tiempo)      # Crea el N2
                nodos.add(N2)
            self.G.add_edge(N1, N2, distancia=float(Distancia), tiempo=float(Tiempo))     # Hace el grafo dirigido
            self.pos=Nx.spring_layout(self.G, seed=20)    # Calcula las cordenadas y mantenerlas evitando que cambien
            self.etiquetas={(origen, destino): f'{atributo['distancia']} km\n{atributo['tiempo']} h' for origen, destino, atributo in self.G.edges(data=True)
```

La función de ”csv” leerá cada línea de la basa de datos, al que los nombres de los centros de distribución quitando espacios y guardando los datos en variables de ”N1”, ”N2” , “Distancia“, “Capacidad“ y “ Tiempo“ con su respectivo dato respectivamente y guardando los nombre en un conjunto para evitar centros repetidos para poder crear los nodos con sus respectivas uniones, teniendo pesos la distancia, el tiempo.

```
]       def Grafo(self):
            Nx.draw(self.G, self.pos, with_labels=True, node_color='lightblue', edge_color='black', arrows=True)   # Diseño del grafo
            Nx.draw_networkx_edge_labels(self.G, self.pos, edge_labels=self.etiquetas) # Ver las etiquetas en el grafo
            plt.title('Grafo'); self.t
            plt.show()
        def Grafo_Ruta(self, ruta):
            r=list(zip(ruta[:-1], ruta[1:]))
            Nx.draw(self.G, self.pos, with_labels=True, node_color='lightgray', edge_color='lightgray', arrows=True)  # Diseño de los nodes y aristas completo
            Nx.draw_networkx_edges(self.G, self.pos, edgelist=r, edge_color='red', width=2.5, arrows=True)    # Diseño de los nodos de la ruta
            Nx.draw_networkx_nodes(self.G, self.pos, nodelist=ruta, node_color='red')     # Diseño de las aristas de la ruta
            Nx.draw_networkx_edge_labels(self.G, self.pos, edge_labels=self.etiquetas)
            plt.title('Mejor ruta'); self.t
            plt.show()
        def Ruta(self, origen, destino):
            if origen in self.G.nodes and destino in self.G.nodes:
                if Nx.has_path(self.G, origen, destino):     # Revisa si existe una ruta
                    ruta = Nx.dijkstra_path(self.G, origen, destino, weight='distancia')     # Busca la ruta mas corta con base a la distancia
                    distancia_total=Nx.dijkstra_path_length(self.G, origen, destino, weight='distancia')     # Calcula la distancia total
                    tiempo_total=Nx.dijkstra_path_length(self.G, origen, destino, weight='tiempo')       # Calcula el tiempo total
                    print(f'\nRuta más corta de {origen} a {destino}: {" → ".join(ruta)}')
                    print(f'Distancia total: {distancia_total} km')
                    print(f'Tiempo total: {tiempo_total} h')
                    self.Grafo_Ruta(ruta)
                else:
                    print("No hay ruta.")
            else:
                print("Uno o ambos nodos no existen.")

    if __name__=='__main__':
        main()
```

La función ”Grafo” le ponemos diseño al grafo completo con todos los centros de distribución, para poder visualizarlo con sus nombres a cada nodo, con un color ”lightblue” a los nodo, nombres de los centros, poniendo un color ”black” a las aristas, mostrando las etiquetas con la distancia y tiempo.

La función “Grafo_Ruta” pide como atributo la ruta mas corta para poder cambiar el diseño a esos nodos el cual los guarda en una lista, donde guardara por pares de nodos del recorrido, cambiándolos a un color “lightgray” a todos

los nodos y aristas pero resaltando la ruta con un color “red” a los nodos y aristas.

La función “Ruta” pide como atributos el origen y el destino el cual verificara la exixtencia, de no encontrarse debuelve un mensaje al usuraio de “Uno oo anbos nodos no existen”. Despues verifica si existe una ruta. Al encontrar la existencia de los nodos y que exista una ruta busca la ruta más corta usando dijkstra_path, cualculando la distancia todal recorrida y el tiempo total con dijkstra_path_length. Para poder mostrarlo al usuario la ruta separada por flechas, distancia total recorrida, el tiempo total. De no encontrar una ruta le manda un mensaje al usuario “No hay ruta”.

Al final llamamos a la Clase “main” para poder ejecutar el programa directamente.

Si bien es cierto que existen ciertas aplicaciones que también ayudan a procesos logísticos en la actualidad, son aplicaciones que como tal solo cumplen con muy pocas funciones, lo hemos buscado en el desarrollo de nuestra app