

SISTEMA

DE RUTAS

POI DE DELIVERY

OBJETIVO



Crear un sistema capaz de encontrar rutas optimas para la entrega de productos .

Implementar el algoritmo de Dijkstra para calcular la ruta más corta y económica.

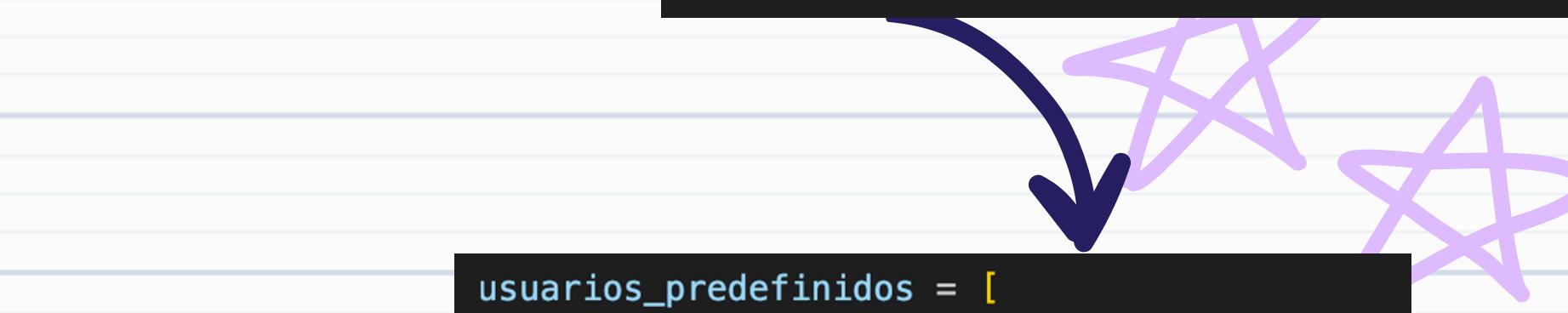


INICIO

```
def iniciar_sesion():
    usuario = input("Usuario: ").strip()
    contrasena = input("Contraseña: ").strip()
    for u in usuarios_predefinidos:
        if usuario == u["correo"] and contrasena == u["contrasena"]:
            print("Inicio sesión exitoso.")
            return u["rol"], u["nombre"].split()[0]
    try:
        with open("usuarios.txt", "r") as f:
            for linea in f:
                nombre, iden, edad, correo, clave, rol = linea.strip().split(",")
                if usuario == correo and contrasena == clave:
                    print("Inicio sesión exitoso.")
                    return rol, nombre.split()[0]
    except FileNotFoundError:
        pass
    print("Credenciales incorrectas.")
    return None, None
```

```
# ----- MAIN -----
if __name__ == "__main__":
    rutas = leer_datos()
    grafo = lista_adyacencia_para_dijkstra(rutas)

while True:
    print("\n--- BIENVENIDO AL SISTEMA ---")
    print("1. Iniciar sesión")
    print("2. Registrarse")
    print("3. Salir")
    opcion = input("Seleccione opción: ").strip()
    if opcion == "1":
        rol,nombre = iniciar_sesion()
        if rol=="Administrador":
            menu_administrador()
        elif rol=="Cliente":
            menu_cliente(nombre)
    elif opcion=="2":
        registro_usuario()
    elif opcion=="3":
        print("Gracias por usar el sistema. ¡Hasta pronto!")
        break
    else:
        print("Opción inválida.")
```



```
usuarios_predefinidos = [
    {
        "nombre": "Alex Anguie",
        "identificacion": "1234567890",
        "edad": "20",
        "correo": "alexanguie@gmail.com",
        "contrasena": "Admin1234",
        "rol": "Administrador"
    }
]
```

REGISTRA

```
def registro_usuario():
    nombre = input("Ingrese su nombre y apellido: ").strip()
    while True:
        identificacion = input("Ingrese su identificación (10 dígitos): ").strip()
        if identificacion.isdigit() and len(identificacion) == 10:
            break
        print("Identificación inválida. Debe tener 10 dígitos.")
    while True:
        edad = input("Ingrese su edad (1-99): ").strip()
        if edad.isdigit() and 1 <= int(edad) <= 99:
            break
        print("Edad inválida.")
    while True:
        correo = input("Ingrese su correo (@gmail.com o @epn.edu.ec): ").strip()
        if (correo.endswith("@gmail.com") or correo.endswith("@epn.edu.ec")):
            try:
                with open("usuarios.txt", "r") as f:
                    if correo in f.read():
                        print("Correo ya registrado.")
                        continue
            except FileNotFoundError:
                pass
            break
        print("Correo inválido.")
    contrasena = input("Ingrese contraseña segura: ").strip()
    while not validar_contrasena(contrasena):
        print("Contraseña inválida (mínimo 8, mayúscula y número).")
        contrasena = input("Ingrese contraseña segura: ").strip()
    with open("usuarios.txt", "a") as f:
        f.write(f"{nombre},{identificacion},{edad},{correo},{contrasena},Cliente\n")
    print("Registro exitoso.")
```

MÓDULAR

AGREGAR

```
def agregarConexion(origen, destino, distancia, costo):
    if origen not in grafo:
        grafo[origen] = {}
    if destino not in grafo:
        grafo[destino] = {}
    grafo[origen][destino] = {'distancia': distancia, 'costo': costo}
    grafo[destino][origen] = {'distancia': distancia, 'costo': costo}
```

ELIMINAR

```
def actualizar_o_eliminar_ciudad():
    if not seleccion_cliente:
        print("No hay ciudades seleccionadas.")
        return
```

MODIFICAR

```
if nombre not in seleccion_cliente:
    print("No encontrada.")
    return
accion = input("Actualizar (A) o Eliminar (E): ").upper()
if accion == "E":
    seleccion_cliente.remove(nombre)
elif accion == "A":
    nueva = input("Nuevo nombre: ").strip()
    if nueva in grafo:
        idx = seleccion_cliente.index(nombre)
        seleccion_cliente[idx] = nueva
    else:
        print("Ciudad no registrada.")
```

GUARDAR DATOS

CUENTA-REGISTRO PROPIO

```
def guardar_seleccion_cliente(nombre_cliente):
    if not seleccion_cliente:
        print("Nada que guardar.")
        return
    with open(f"rutas-{nombre_cliente.lower()}.txt", "w") as f:
        for c in seleccion_cliente:
            f.write(c+"\n")
    print("Guardado.")
def cargar_seleccion_cliente(nombre_cliente):
    seleccion_cliente.clear()
    try:
        with open(f"rutas-{nombre_cliente.lower()}.txt", "r") as f:
            for linea in f:
                ciudad = linea.strip()
                if ciudad in grafo:
                    seleccion_cliente.append(ciudad)
    except FileNotFoundError:
        pass
```

RUTAS

AGREGA UNA IDEA PRINCIPAL

RUTAS PRINCIPALES DEL
SISTEMA

AGREGA UNA IDEA PRINCIPAL

COSTO Y DISTANCIA DE
ENTRE CADA CIUDAD

```
zonas_de_entrega = {
    "Costa": {
        "Guayas": ["Guayaquil", "Playas"],
        "Manabí": ["Manta", "Puerto López"]
    },
    "Sierra": {
        "Pichincha": ["Quito", "Panecillo"],
        "Tungurahua": ["Baños de Agua Santa"],
        "Ibarra": ["Otavalo", "Yahuarcocha"]
    },
    "Oriente": {
        "Napo": ["Tena"],
        "Pastaza": ["Puyo"]
    }
}
```

```
def crear_archivo_centros():
    with open("centros.txt", "w") as archivo:
        archivo.write("Guayaquil,Panecillo, 300.1, 50\n")
        archivo.write("Quito, Cuenca, 360.3, 60\n")
        archivo.write("Napo, Loja, 286.9, 70\n")
        archivo.write("Ibarra, Yahuarcocha, 408.5, 40\n")
        archivo.write("Manta, Tena, 267.7, 65\n")
        archivo.write("Otavalo, Latacunga, 353, 30\n")
        archivo.write("Santo Domingo, Manta, 566, 35\n")
        archivo.write("Carchi, Quito, 149.5, 25\n")
        archivo.write("Puyo, Riobamba, 250.4, 20\n")
        archivo.write("Guayaquil, Machala, 123, 28\n")
```

RUTAS

```
for op in ["1","2","3"]:  
    if op == "1":  
        for o,destinos in grafo.items():  
            for d,datos in destinos.items():  
                print(f"{o} -> {d}: {datos['distancia']} km | ${datos['costo']}")  
  
    elif op == "2":  
        ciudades = list(grafo.keys())  
        metodo = input("Ordenar: 1-Burbuja 2-Selección 3-Merge: ").strip()  
        if metodo == "1":  
            ciudades = burbuja(ciudades)  
        elif metodo == "2":  
            ciudades = seleccion(ciudades)  
        elif metodo == "3":  
            ciudades = merge_sort(ciudades)  
        for i,c in enumerate(ciudades,1):  
            print(f"{i}. {c}")
```

0. cerrar sesión

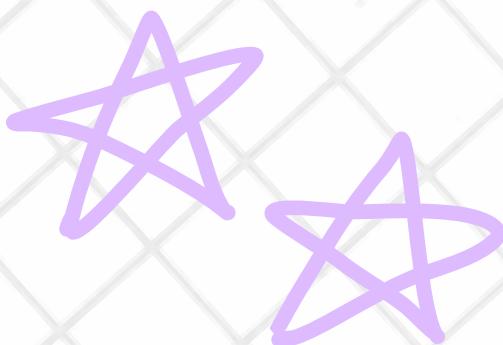
Opción: 1

Guayaquil -> Panecillo: 435.1 km | \$50.0
Guayaquil -> Machala: 183.0 km | \$28.0
Panecillo -> Guayaquil: 435.1 km | \$50.0
Machala -> Guayaquil: 183.0 km | \$28.0
Quito -> Cuenca: 460.3 km | \$60.0
Quito -> Carchi: 184.5 km | \$25.0
Cuenca -> Quito: 460.3 km | \$60.0
Carchi -> Quito: 184.5 km | \$25.0
Napo -> Loja: 586.9 km | \$70.0
Loja -> Napo: 586.9 km | \$70.0
Ibarra -> Yahuarcocha: 308.5 km | \$40.0
Yahuarcocha -> Ibarra: 308.5 km | \$40.0
Manta -> Tena: 567.7 km | \$65.0
Manta -> Santo Domingo: 246.0 km | \$35.0
Tena -> Manta: 567.7 km | \$65.0
Santo Domingo -> Manta: 246.0 km | \$35.0
Otavalo -> Latacunga: 203.0 km | \$30.0
Latacunga -> Otavalo: 203.0 km | \$30.0
Puyo -> Riobamba: 136.4 km | \$20.0
Riobamba -> Puyo: 136.4 km | \$20.0

MENÚ CLIENTE

Dijkstra

COSTO-DISTANCIA



```
def dijkstra(inicio, tipo_costo='costo'):
    distancias = {nodo: float('inf') for nodo in grafo}
    predecesores = {nodo: None for nodo in grafo}
    distancias[inicio] = 0
    cola_prioridad = [(0, inicio)]
    while cola_prioridad:
        costo_actual, nodo_actual = heapq.heappop(cola_prioridad)
        if costo_actual > distancias[nodo_actual]:
            continue
        for vecino, datos in grafo[nodo_actual].items():
            costo_arista = datos[tipo_costo]
            nuevo_costo = costo_actual + costo_arista
            if nuevo_costo < distancias[vecino]:
                distancias[vecino] = nuevo_costo
                predecesores[vecino] = nodo_actual
                heapq.heappush(cola_prioridad, (nuevo_costo, vecino))
    return distancias, predecesores
```

MENSAJE



Guayaquil -> Manta: 45.1 km	\$30.0
Guayaquil -> Quito: 309.5 km	\$80.0
Guayaquil -> Machala: 123.0 km	\$25.0
Manta -> Guayaquil: 45.1 km	\$30.0
Manta -> Quito: 40.3 km	\$40.0
Quito -> Manta: 40.3 km	\$40.0
Quito -> Latacunga: 353.0 km	\$30.0
Quito -> Santo Domingo: 566.0 km	\$55.0
Quito -> Guayaquil: 309.5 km	\$80.0
Santo Domingo -> Quito: 566.0 km	\$55.0
Latacunga -> Quito: 353.0 km	\$30.0

ORIGEN Y DESTINO

QUITO A GUAYAQUIL

Origen: Quito
Destino: Guayaquil
La mejor ruta es:
Ruta: Quito -> Manta -> Guayaquil
Distancia total: 85.40 km
Costo total: \$70.00 USD

Convención

AHORA YA COMPRENDIMOS EL USO DE LAS ESTRUCTURAS
DE ARBOLES , GRAFOS , DICICONARIOS ENTRE OTRAS,
GRACIAS A ESTO YA PODEMOS CREAR NUESTRO PROPIO
MAPA