Trouver les Pharmacies les Plus Proches de Votre Localisation Actuelle

Réalisé par :

Maazouz AbdElAziz , Mehdaoui Othman ,
Essalmi Adnane

Encadré par :

Pr. AHERRAHROU NOURA

FACULTÉ DES SCIENCES DHAR EL MAHRAZ
UNIVERSITÉ SIDI MOHAMED BEN ABDELLAH

PLAN

1- Introduction

2-Utilitaires et Bibliothèques

3-Étapes du Projet

4-Conclusion

1-Introduction

Objectif :

- Développer une application pour identifier et visualiser la pharmacie la plus proche de votre localisation actuelle

Technologies Utilisées :

- ***Neo4j*** : Base de données graphes pour stocker et interroger les données routières et de pharmacie.
- ***OSMnx*** : Bibliothèque Python pour télécharger les données géographiques **OpenStreetMap**.
- ***Folium*** : Bibliothèque Python pour créer des cartes interactives.

2-Utilitaires et Bibliothèques:

Neo4j :

- Base de données orientée graphe.
- Utilisée pour stocker les données routières et de pharmacie.

OSMnx :

- Téléchargement et manipulation des données géographiques.
- Extraction des réseaux routiers et des points d'intérêt (pharmacies).

Folium :

- Création de cartes interactives.
- Visualisation des points d'intérêt et des chemins les plus courts

3-Étapes du Projet

```
# Connexion à la base de données Neo4j
uri = "bolt://localhost:7687"
driver = GraphDatabase.driver(uri, auth=("neo4j", "12345678"))
start_point = (34.047608901290324, -4.959777988230495)

# Test de connexion
with driver.session() as session:
    result = session.run("RETURN 1")
    print("Connexion réussie" if result.single()[0] == 1 else "Connexion échouée")
```

Connexion à Neo4j

- Configuration de la connexion à la base de données Neo4j.
- Test de la connexion pour s'assurer qu'elle fonctionne correctement.

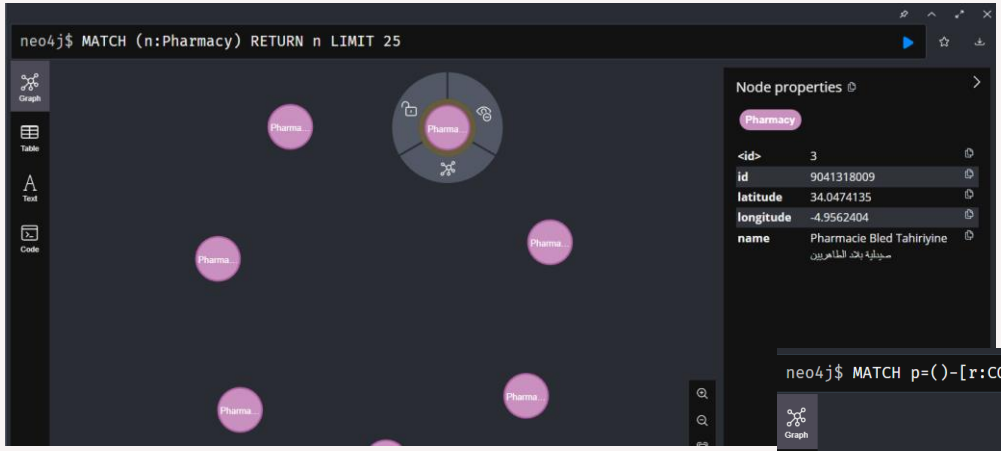
3-Étapes du Projet

```python
def import_data_to_neo4j(driver, start_point, distance=1000):
    # Récupération des pharmacies
    pharmacies = ox.geometries_from_point(start_point, tags={'amenity': 'pharmacy'}, dist=distance)
    # Récupération du réseau routier
    G = ox.graph_from_point(start_point, dist=distance, network_type='drive')

    def create_pharmacy_node(tx, id, name, lat, lon):
        query = (
            "CREATE (p:Pharmacy {id: $id, name: $name, latitude: $lat, longitude: $lon})"
        )
        tx.run(query, id=id[1], name=name, lat=lat, lon=lon)

    def create_road_node(tx, osmid, lat, lon):
        query = (
            "CREATE (r:Road {osmid: $osmid, latitude: $lat, longitude: $lon})"
        )
        tx.run(query, osmid=osmid, lat=lat, lon=lon)

    def create_road_relationship(tx, from_osmid, to_osmid, length):
        query = (
            "MATCH (r1:Road {osmid: $from_osmid}), (r2:Road {osmid: $to_osmid}) "
            "CREATE (r1)-[:CONNECTED {length: $length}]->(r2)"
        )
        tx.run(query, from_osmid=from_osmid, to_osmid=to_osmid, length=length)
```

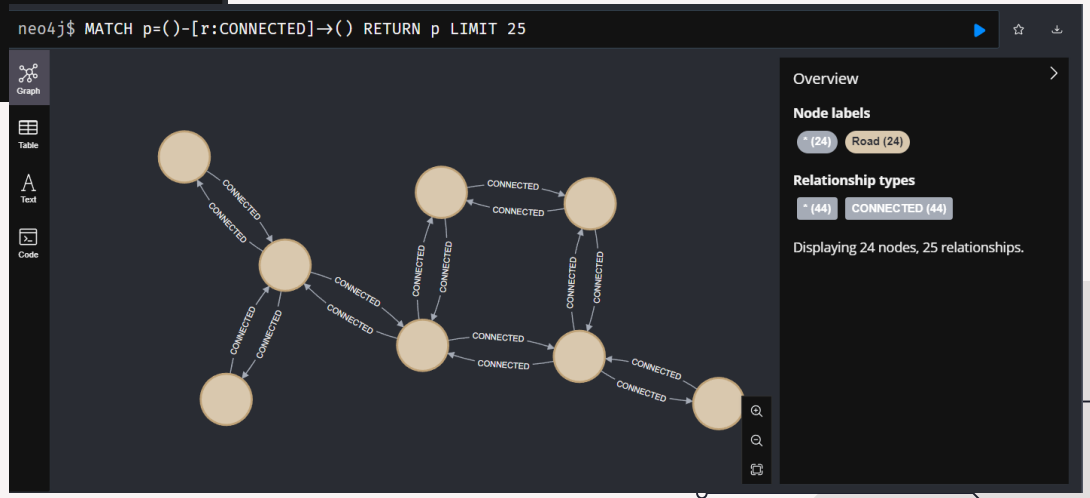
Importation des Données

- Récupération des données de pharmacies et du réseau routier à partir d'OSMnx.
- Importation de ces données dans Neo4j en créant des nœuds et des relations correspondants.

3-Étapes du Projet



- Les pharmacies



- réseau routier

3-Étapes du Projet

```python
def find_nearest_pharmacy(driver, start_point):
    query = (
        "MATCH (p:Pharmacy) "
        "RETURN p.id AS id, p.name AS name, p.latitude AS lat, p.longitude AS lon, "
        "point.distance(point({latitude: $lat, longitude: $lon}), point({latitude: p.latitude, longitude: p.longit
        "ORDER BY distance ASC LIMIT 1"
    )

    with driver.session() as session:
        result = session.run(query, lat=start_point[0], lon=start_point[1])
        return result.single()

# Test de la fonction de recherche de la pharmacie la plus proche
nearest_pharmacy = find_nearest_pharmacy(driver, start_point)
print(f"La pharmacie la plus proche est : {nearest_pharmacy['name']} à {nearest_pharmacy['distance']} mètres")
```

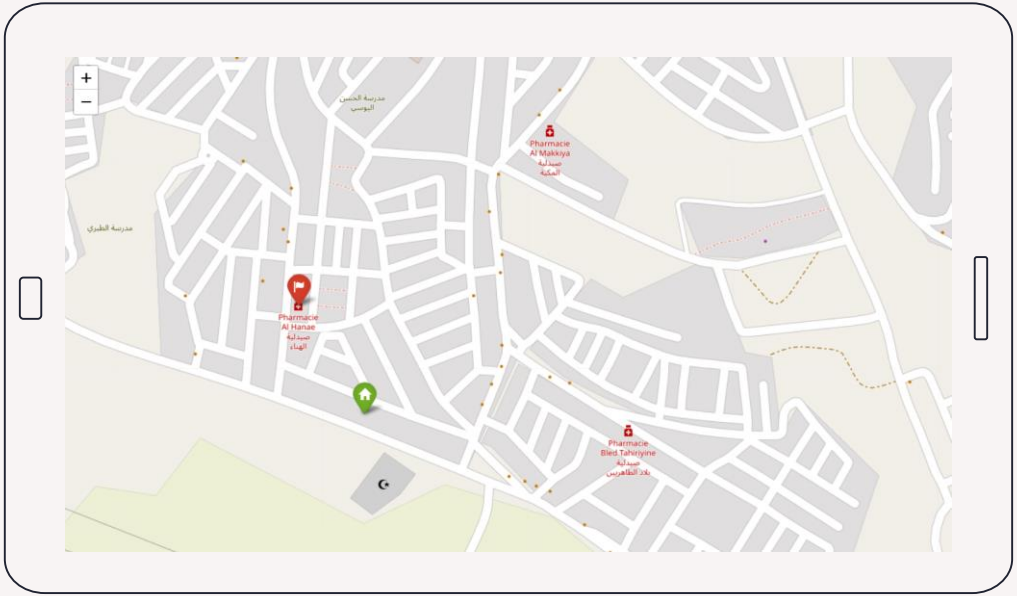
Executed at 2024.05.29 11:10:30 in 627ms

```
La pharmacie la plus proche est : Pharmacie Al Hanae صيدلية الهناء à 156.41452209294732 mètres
```

Recherche de la Pharmacie la Plus Proche

- Utilisation d'une requête Cypher pour trouver la pharmacie la plus proche du point de départ spécifié.

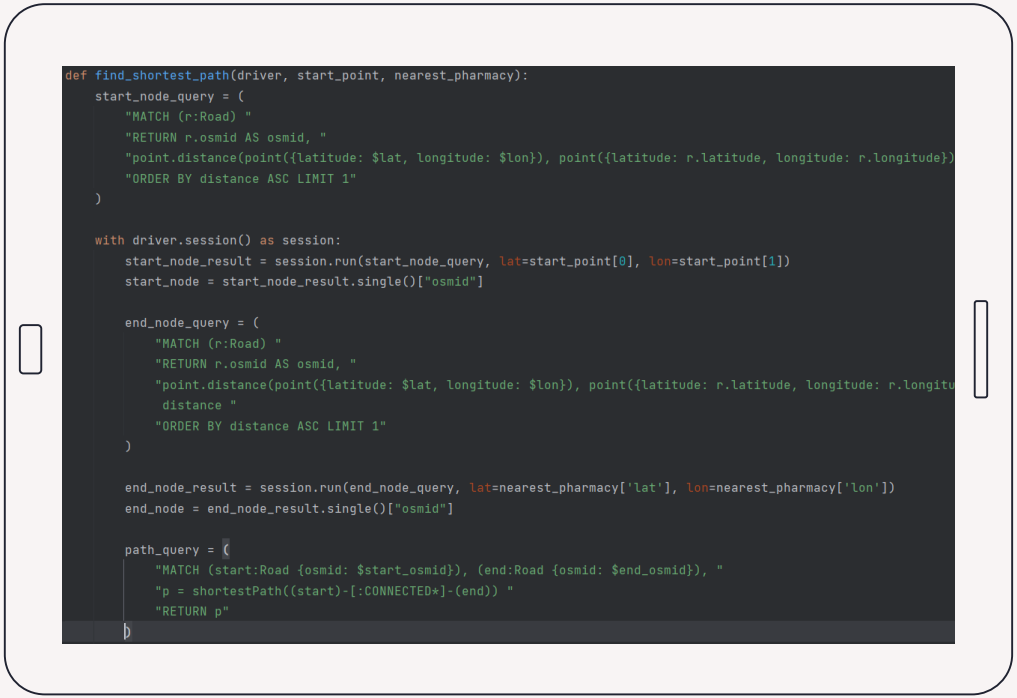
3-Étapes du Projet



Création d'une Carte Interactive

- Utilisation de Folium pour créer une carte interactive affichant le point de départ et la pharmacie la plus proche.

3-Étapes du Projet

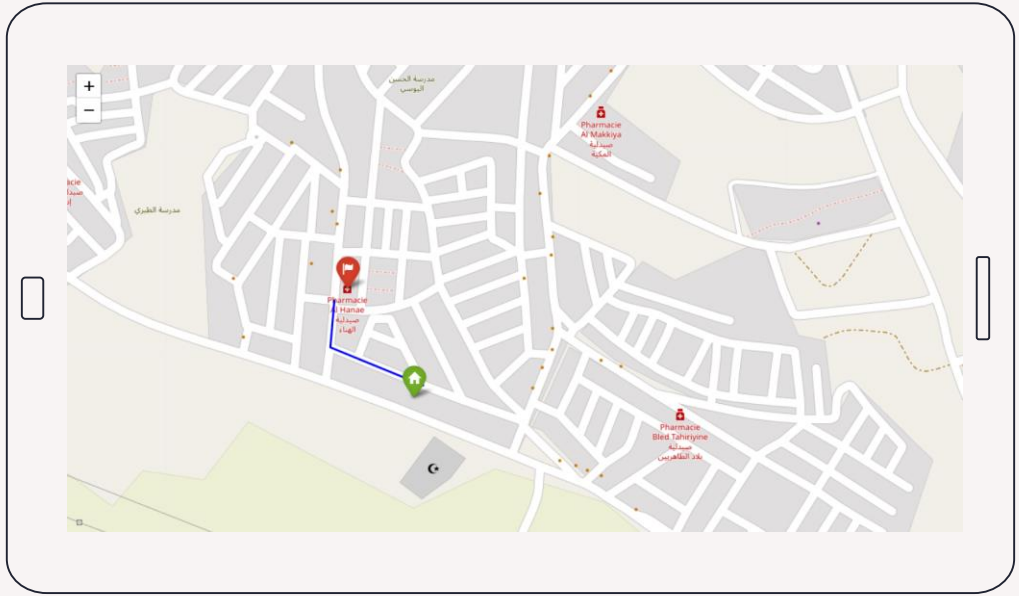


```
def find_shortest_path(driver, start_point, nearest_pharmacy):
    start_node_query = (
        "MATCH (r:Road) "
        "RETURN r.osmid AS osmid, "
        "point.distance(point({latitude: $lat, longitude: $lon}), point({latitude: r.latitude, longitude: r.longitude})
        "ORDER BY distance ASC LIMIT 1"
    )

    with driver.session() as session:
        start_node_result = session.run(start_node_query, lat=start_point[0], lon=start_point[1])
        start_node = start_node_result.single()["osmid"]

        end_node_query = (
            "MATCH (r:Road) "
            "RETURN r.osmid AS osmid, "
            "point.distance(point({latitude: $lat, longitude: $lon}), point({latitude: r.latitude, longitude: r.longitu
             distance "
            "ORDER BY distance ASC LIMIT 1"
        )

        end_node_result = session.run(end_node_query, lat=nearest_pharmacy['lat'], lon=nearest_pharmacy['lon'])
        end_node = end_node_result.single()["osmid"]

        path_query = (
            "MATCH (start:Road {osmid: $start_osmid}), (end:Road {osmid: $end_osmid}), "
            "p = shortestPath((start)-[:CONNECTED*]-(end)) "
            "RETURN p"
        )
```

Calcul du Chemin le Plus Court

- Utilisation de l'algorithme de plus court chemin **'shortestPath'** pour trouver le chemin le plus rapide vers la pharmacie la plus proche.

3-Étapes du Projet



- **Affichage du chemin sur la carte interactive.**

Conclusion!

- Ce projet a démontré comment utiliser diverses technologies pour résoudre un problème de localisation en temps réel.
- En combinant les capacités de Neo4j pour la gestion des graphes, OSMnx pour les données géospatiales, et Folium pour la visualisation, nous avons créé un système efficace pour localiser la pharmacie la plus proche et calculer le chemin le plus court.

Merci pour votre attention!