Livret technique smartgraph backend

Ce livret technique détaille l'architecture et les points d'entrée de l'API SmartGraph, une application Quarkus conçue pour la gestion de données agricoles (parcelles, capteurs, robots) stockées dans une base de données orientée graphe Neo4j.

1. Présentation Générale

L'API SmartGraph est une interface RESTful permettant de monitorer et de gérer des infrastructures agricoles connectées. Elle utilise des transactions asynchrones pour interagir avec Neo4j et implémente des standards de l'Internet des Objets (IoT) via des labels comme SSN_Sensor et SAREF_Device.

Stack Technique

- **Framework :** Quarkus
- **Base de données :** Neo4j (Graphe)
- **Langage :** Java 17+
- **Communication :** JAX-RS (JSON)
- **Asynchronisme :** CompletionStage et ThreadContext de MicroProfile.

2. Architecture des Données

L'API manipule quatre entités principales représentées sous forme de nœuds dans Neo4j :

- **Parcelle :** Zones géographiques de culture (nom, superficie, culture, latitude, longitude).
- **Capteur :** Dispositifs de mesure (type, statut) liés à une parcelle.
- **Robot :** Unités mobiles d'intervention (modele, seuil de détection).
- **Observation :** Données temporelles récoltées (valeur numérique, unité, horodatage).

3. Points d'Entrée de l'API (Endpoints)

Toutes les routes sont préfixées par /api/v1.

A. Gestion des Parcelles (/parcelles)

Méthode	Path	Description
GET	/	Liste toutes les parcelles.
GET	/{id}	Détails d'une parcelle spécifique.
POST	/	Créer une nouvelle parcelle (ID généré automatiquement).
PUT	/{id}	Mettre à jour les informations d'une parcelle.
DELETE	/{id}	Supprimer une parcelle.

B. Gestion des Capteurs (/capteurs)

Les capteurs portent le label secondaire :SSN_Sensor.

- **GET** / : Liste tous les capteurs.
- **GET** /parcelles/{parcelle_id} : Liste les capteurs installés sur une parcelle précise.
- **POST** / : Enregistre un capteur.

C. Gestion des Robots (/robots)

Les robots sont enregistrés avec les labels :Robot:AgriculturalDevice:SAREF_Device:Sensor.

- **POST** / : Permet de définir le modèle et le weed_detection_threshold (seuil de détection des mauvaises herbes).

D. Gestion des Observations (/observations)

- **GET** / : Récupère l'historique des relevés.
- **POST** / : Enregistre une nouvelle donnée (numericValue) liée à un capteur (madeBySensor).

E. Dashboard & Statistiques

- **GET** /dashboard/stats : Retourne un résumé global (nombre total de parcelles, capteurs, robots et observations).

---

4. Configuration & Sécurité

CORS (Cross-Origin Resource Sharing)

Un filtre global (CorsFilter) est implémenté pour permettre les requêtes provenant de n'importe quelle origine (*).

- **Headers autorisés :** origin, content-type, accept, authorization.
- **Méthodes autorisées :** GET, POST, PUT, DELETE, OPTIONS, HEAD.

Gestion de l'Asynchronisme

L'API utilise AsyncSession de Neo4j. Le ThreadContext de MicroProfile est injecté pour garantir que le contexte de sécurité et de propagation des threads est conservé lors des opérations de rappel (thenApply, thenCompose).