Mavigo - Documentation Technique

Table of Contents

1. Presentation de l'equipe

Membre
Marie BIBI
Raphael MEIMOUN
Seyed Amineddin MIRI
Malado SOW

2. Presentation thematique

2.1. Contexte

Mavigo est un assistant personnel de transport en commun pour la ville de Paris. L'application aide les utilisateurs a naviguer dans le reseau de transport parisien avec une planification intelligente des trajets et des fonctionnalites ameliorant la qualite de vie.

2.2. Objectifs

- Simplifier la planification des trajets en Ile-de-France
- Fournir des informations en temps reel sur les perturbations
- Integrer la gestion des taches avec Google Tasks
- Offrir des suggestions de trajets adaptees aux preferences utilisateurs

2.3. Fonctionnalites principales

- **Planification de trajets** : Recherche d'itineraires optimaux via l'API PRIM
- **Alertes en temps reel** : Notifications sur les perturbations du reseau
- **Integration Google Tasks** : Synchronisation des trajets avec les taches
- **Reroutage automatique** : Suggestions alternatives en cas de perturbation
- **Suivi de position** : Progression en temps reel pendant le trajet

3. Technologies et methodologies

3.1. Stack technique

3.1.1. Backend

Technologie	Version/Details
Java	21 (LTS)
Spring Boot	3.5.7
Spring Data JPA	Persistance des donnees
Spring Security	OAuth2 avec Google
H2 Database	Base de donnees de developpement
Gradle	9.1 (systeme de build)

3.1.2. Frontend

- HTML5
- CSS3
- JavaScript (Vanilla, sans framework)

3.1.3. APIs externes

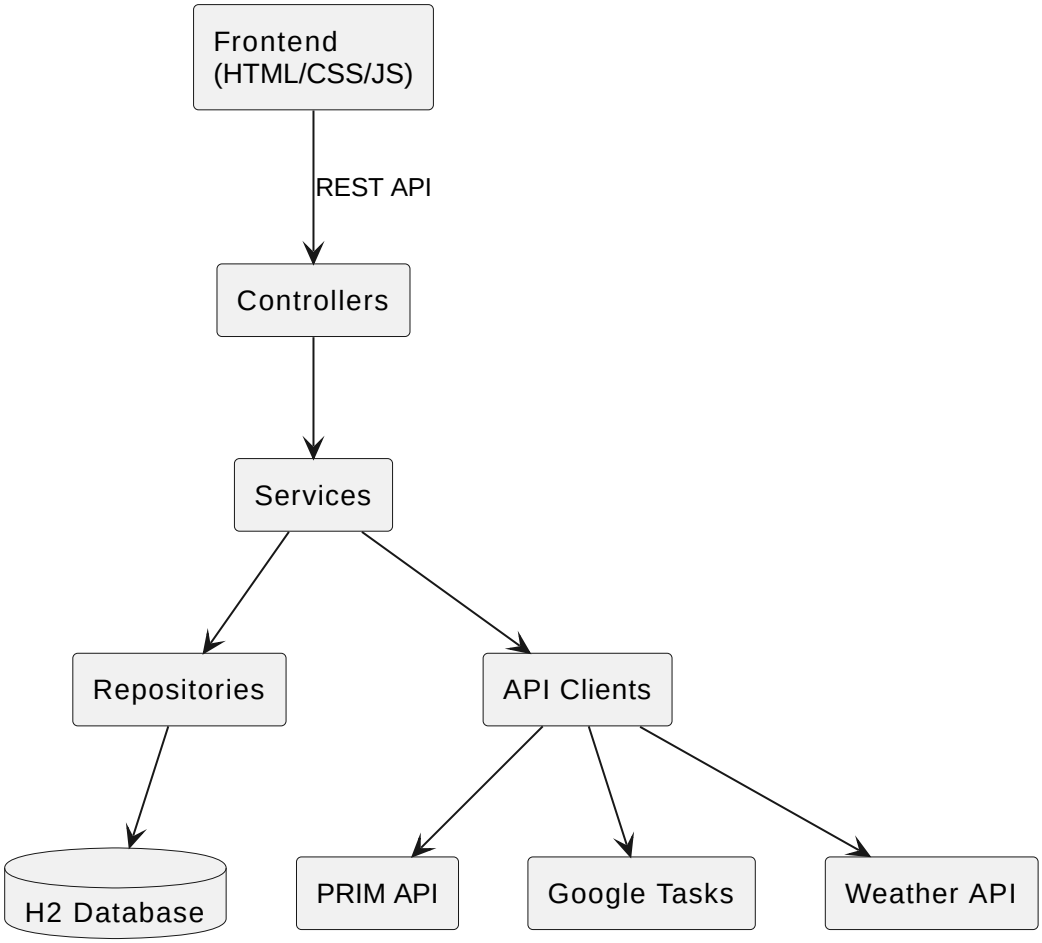
- **PRIM API** : Ile-de-France Mobilites pour la planification de trajets
- **Google Tasks API** : Gestion des taches utilisateur
- **Weather API** : Donnees meteorologiques

3.2. Methodologies

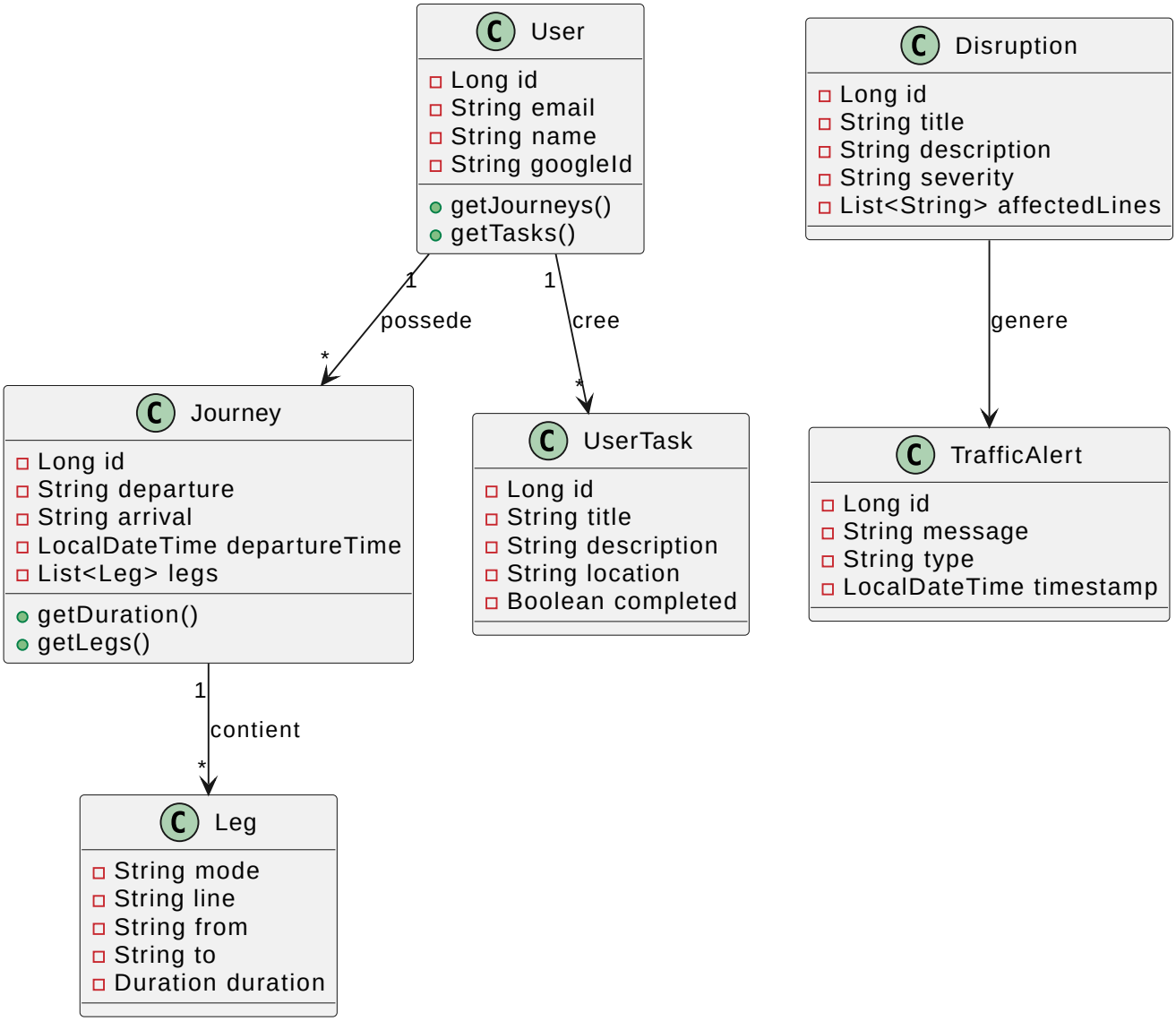
- **Git Flow** : Branches feature, dev, main
- **CI/CD** : GitHub Actions pour l'integration continue
- **Code Review** : Pull requests obligatoires

4. Architecture

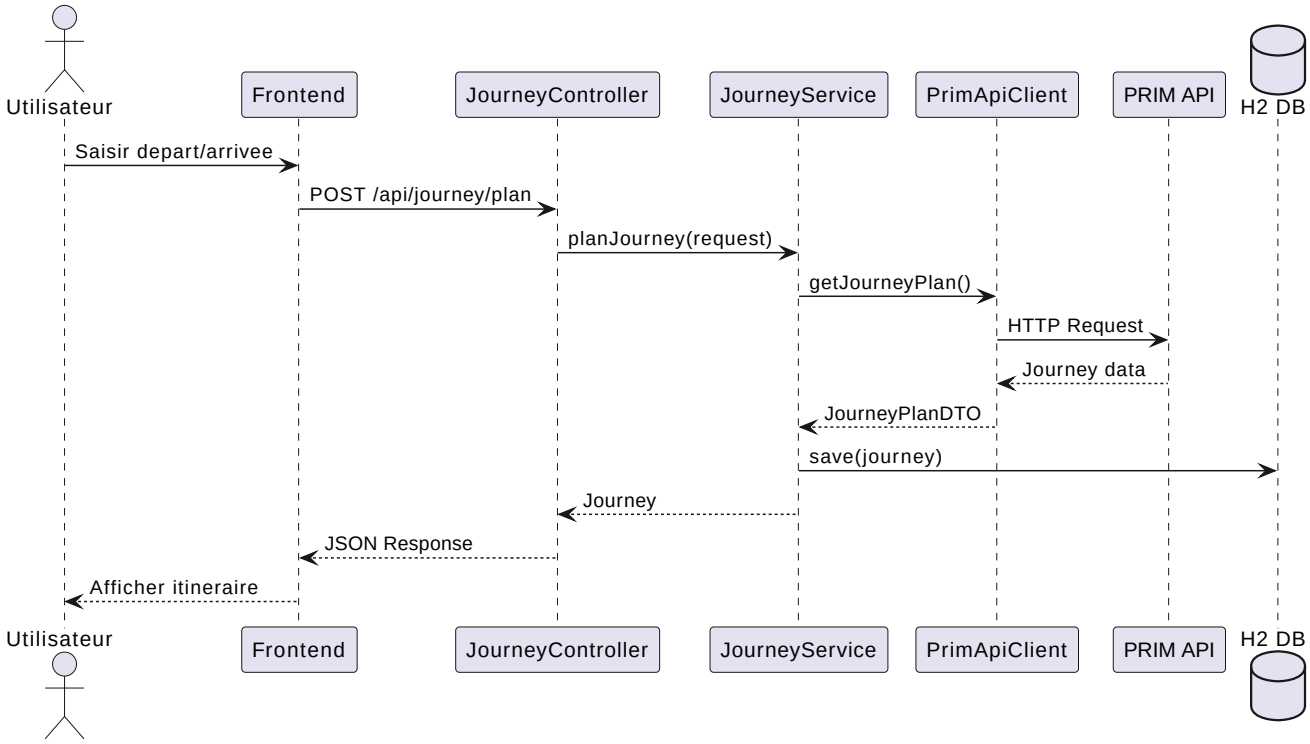
4.1. Architecture globale



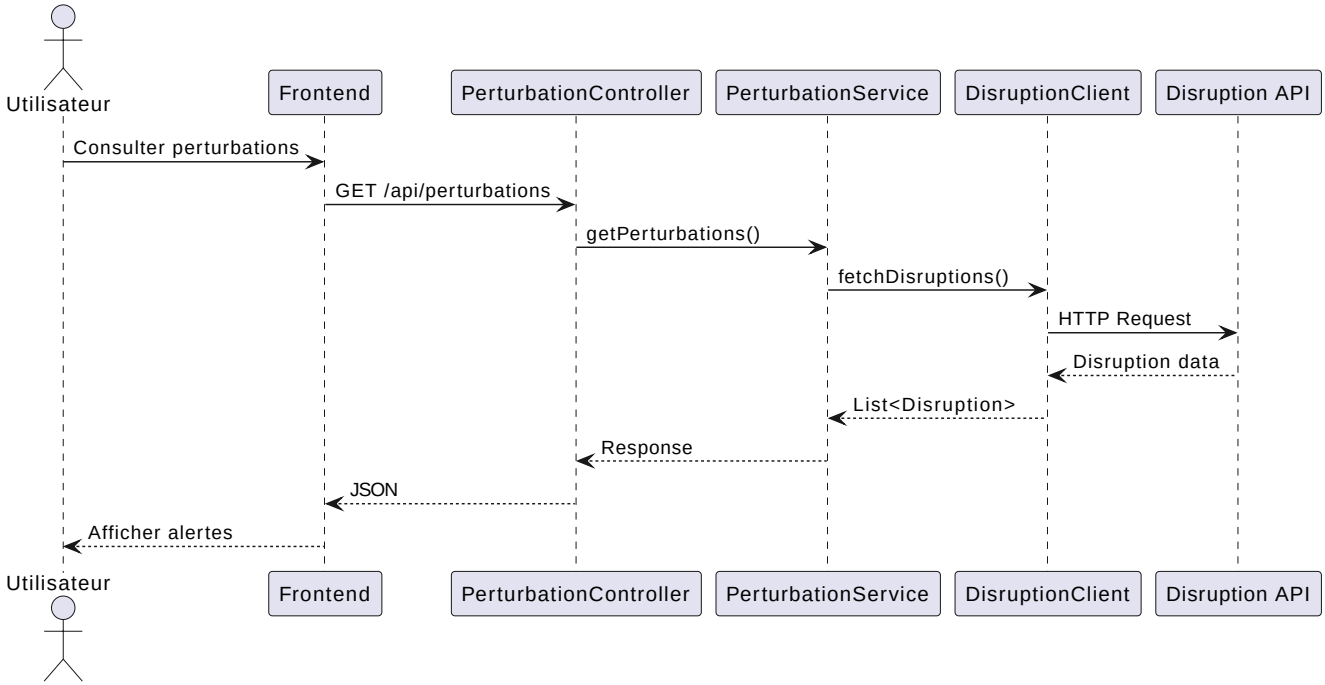
4.2. Diagramme de classes - Modeles principaux



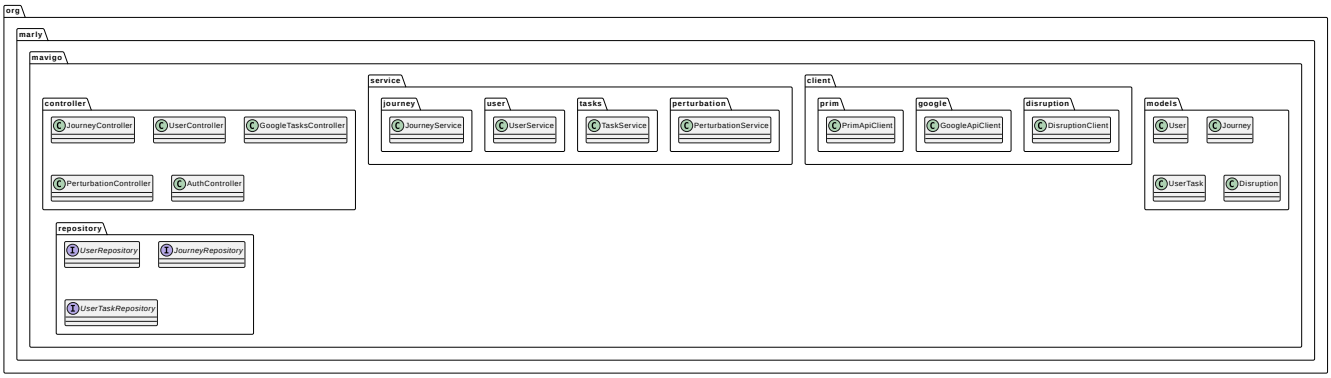
4.3. Diagramme de sequence - Planification de trajet



4.4. Diagramme de sequence - Gestion des perturbations



4.5. Structure des packages



5. Endpoints API

5.1. Journey API

Methode	Endpoint	Description
POST	`/api/journey/plan`	Planifier un nouveau trajet
GET	`/api/journey/{id}`	Obtenir les details d'un trajet
GET	`/api/journey/user`	Lister les trajets de l'utilisateur

5.2. User API

Methode	Endpoint	Description
GET	`/api/user/profile`	Obtenir le profil utilisateur
PUT	`/api/user/preferences`	Mettre a jour les preferences

5.3. Tasks API

Methode	Endpoint	Description
GET	`/api/tasks`	Lister les taches
POST	`/api/tasks`	Creer une nouvelle tache
PUT	`/api/tasks/{id}`	Modifier une tache
DELETE	`/api/tasks/{id}`	Supprimer une tache

5.4. Perturbations API

Methode	Endpoint	Description
GET	`/api/perturbations`	Obtenir les perturbations actuelles

Methode	Endpoint	Description
GET	`/api/perturbations/line/{lineId}`	Perturbations pour une ligne specifique

6. Guide d'installation

6.1. Prerequis

- Java 21 ou superieur
- Gradle 9.1+ (ou utiliser le wrapper inclus)
- Compte Google pour OAuth2 (optionnel)
- Cle API PRIM

6.2. Installation

```
# Cloner le repository
git clone https://github.com/Aminmiri82/devops-project-MARLY.git
cd devops-project-MARLY

# Configurer les variables d'environnement
cd Mavigo
cp local.env.example local.env
# Editer local.env avec vos cles API

# Lancer l'application
./gradlew bootRun
```

L'application sera disponible sur http://localhost:8080

6.3. Build

```
# Construire le JAR
./gradlew build

# Le JAR sera dans build/libs/
```

6.4. Tests

```
# Executer les tests
./gradlew test
```

7. Licence

Ce projet est sous licence Apache 2.0. Voir le fichier LICENSE pour plus de details.