Devops-noah

Team

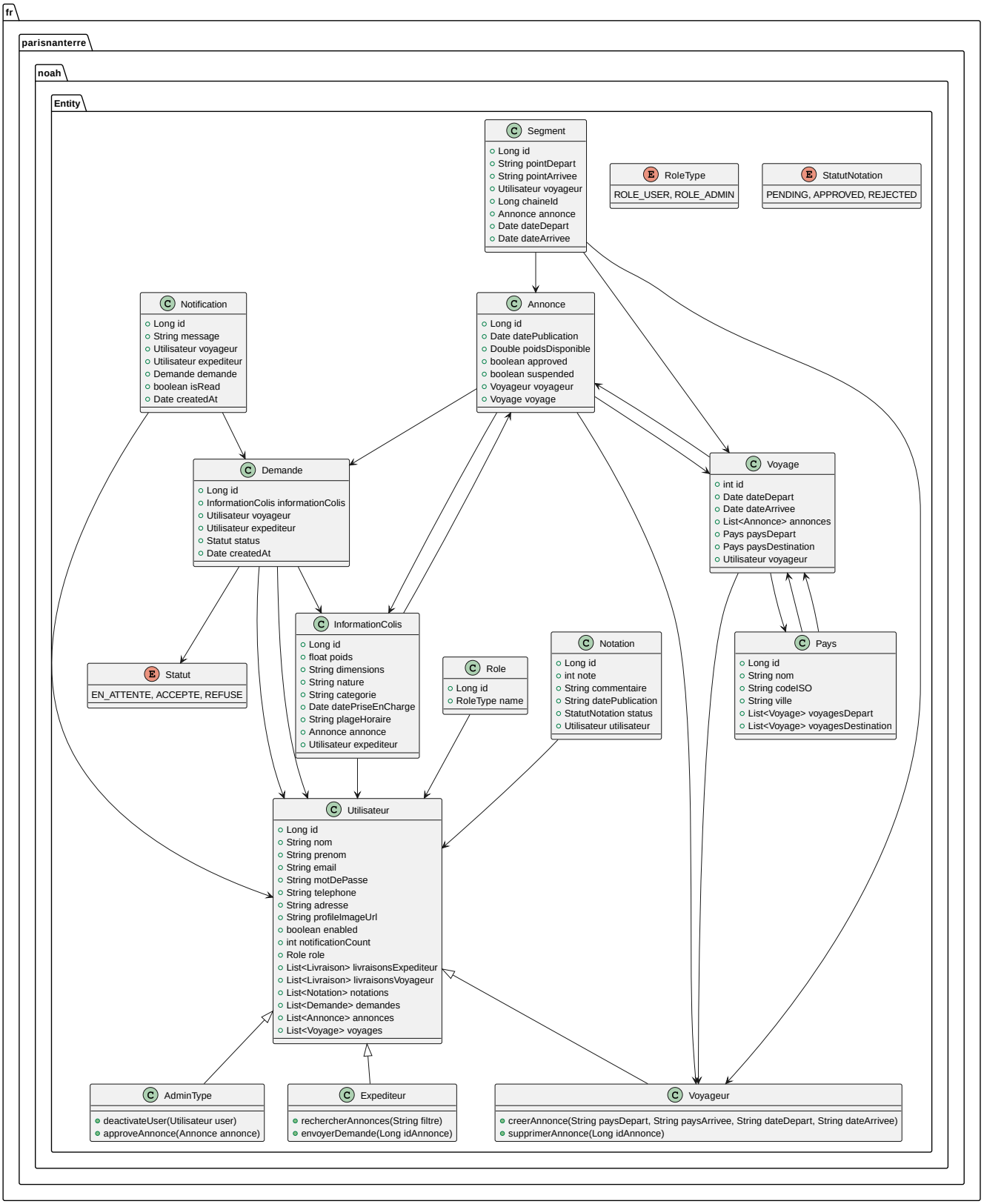
- Nabintou FOFANA
- Hawa DIALLO
- Amare NIGATU
- Oumar MAHAMAT

Application goal

TravelCarry est une plateforme de transport collaboratif de colis qui met en relation des voyageurs ayant de l'espace disponible dans leurs bagages avec des expéditeurs souhaitant envoyer des objets à moindre coût. Grâce à TravelCarry, l'envoi d'objets d'un pays à un autre devient plus économique, rapide et sécurisé, tout en optimisant les déplacements des voyageurs.

Architecture

Diagramme de classes :



Technologies Utilisées

- **Spring Boot** : Framework principal pour le backend.
- **Spring Data JPA** : Gestion de l'accès aux données via des entités Java.
- **Spring Security** : Gestion de l'authentification et de l'autorisation.
- **JWT (JSON Web Tokens)** : Pour sécuriser les API via un système de jetons.

- **Neon** : Base de données relationnelle PostgreSQL hébergée en ligne.
- **React.js** : Framework utilisé pour le frontend, permettant une interface dynamique et réactive.
- **Gradle** : Outil de gestion des dépendances et d'automatisation des builds.
- **Lombok** : Réduit la verbosité du code Java, notamment pour les getters/setters.
- **Swagger** : Documentation interactive de l'API pour les développeurs et utilisateurs.

Installation et Configuration

Prérequis

- **Java 21** : Assurez-vous que Java 21 est installé sur votre machine.
- **Gradle** : Outil de gestion de dépendances et d'exécution de builds.
- **Neon** : Base de données PostgreSQL hébergée en ligne.
- **React.js** : Utilisé pour la construction du frontend de l'application.

Installation des Dépendances

Clonez le projet backend et installez les dépendances en exécutant les commandes suivantes :

```
git clone https://github.com/Devops-noah/Backend.git
cd Backend
```

Clonez le projet frontend et installez les dépendances avec npm

```
git clone https://github.com/Devops-noah/Frontend.git
cd frontend
npm install
```

Exécution de l'Application

```
./gradlew clean
./gradlew build
./gradlew run
```

L'API sera alors disponible à l'adresse suivante : http://localhost:8080

Pour démarrer le frontend React.js :

```
npm start
```

Sécurité

1. **Authentification JWT** : L'application utilise JWT pour sécuriser ses endpoints. Lors de l'authentification (/api/auth/login), un jeton JWT est généré et doit être inclus dans l'en-tête Authorization pour toutes les requêtes nécessitant une authentification.

Exemples d'Utilisation :

Connexion :

Pour se connecter, envoyez une requête POST à /api/auth/login avec les identifiants de l'utilisateur. Exemple de requête : POST /api/auth/login Content-Type: application/json

```
{
"email": "expediteur@example.com",
"password": "yourpassword"
}
```

Accès aux API sécurisées :

Exemple d'en-tête de requête avec un token JWT :

```
Authorization: Bearer <YOUR_JWT_TOKEN>
```

Documentation des API

La documentation des API est générée et disponible via Swagger. Pour y accéder, ouvrez le navigateur à l'adresse suivante après avoir démarré l'application :

http://localhost:8080/swagger-ui.html

Swagger fournit une interface interactive permettant de tester toutes les API disponibles de manière conviviale.

Tests

Les tests unitaires et d'intégration sont exécutés avec Gradle. Pour lancer les tests, utilisez la commande suivante :

```
./gradlew test
```

Cela exécutera tous les tests du projet et génèrera un rapport détaillé dans le dossier

`build/reports/tests`.

Fonctionnalités Principales

TravelCarry propose les fonctionnalités suivantes :

- **Gestion des utilisateurs** : Création et gestion des profils des utilisateurs (expéditeurs, voyageurs).
- **Gestion des colis** : Création, soumission et validation des demandes de transport de colis.
- **Notifications** : Notifications en temps réel pour les voyageurs.
- **Notations** : Noter le site.
- **Gestion des voyages et annonces** : Création et gestion des voyages et des annonces associées aux voyages.
- **Transfert en chaine** : Permet d'achéminer un colis lorsque aucune ne couvre l'intégralité de trajet souhaité.

Outils de Développement

Voici les outils recommandés pour le développement et le test de l'application :

- **IDE** : IntelliJ IDEA.
- **Base de Données** : **Neon** (hébergé en ligne).
- **Frontend** : **React.js** pour l'interface utilisateur.
- **Neon** pour la base de données en ligne : [Console Neon](https://console.neon.tech/app/projects/misty-paper-89322152/branches/br-little-shape-a2dhwb4o/tables?database=travel_carry_db)
- **Postman** : Outil pour tester les API REST de manière simple et interactive.

Conclusion

Ce projet vise à faciliter le transport de colis de manière sécurisée et efficace. Grâce à l'architecture modulaire et sécurisée, **TravelCarry** offre une solution robuste et extensible pour les besoins de gestion de transport de colis entre expéditeurs et voyageurs.