Rapport de Projet - Architecture Microservices

MIAGE IF Apprentissage – 2024 / 2025

Daryl MARTIN-DIPP & Salim MESSAOUDI

Application "My-Trip" - Gestion de voyages

Présentation du projet

L'application "My-Trip" est une solution backend développée selon une architecture microservices pour la gestion de voyages et d'itinéraires.

L'application permet de gérer, entre autres, la création, la modification et la suppression de voyages, d'étapes, de villes, d'hébergements, d'activités et de points d'intérêts.

Architecture technique

Pour coder le backend, nous avons utilisé Java avec Spring Boot (Maven), sur l'IDE IntelliJ IDEA. La base de données est PostgreSQL hébergée sur Supabase. La documentation API, faisant dans notre projet office de frontend, est faite grâce à Swagger. La conteneurisation a été faite avec Docker. Enfin, le versioning a été fait sur GitHub.

Choix architecturaux

Nous avons utilisé PostgreSQL sur Supabase car c'est une solution cloud managed réduisant la complexité d'infrastructure, avec une haute disponibilité et sauvegardes automatiques, et surtout une interface d'administration intuitive

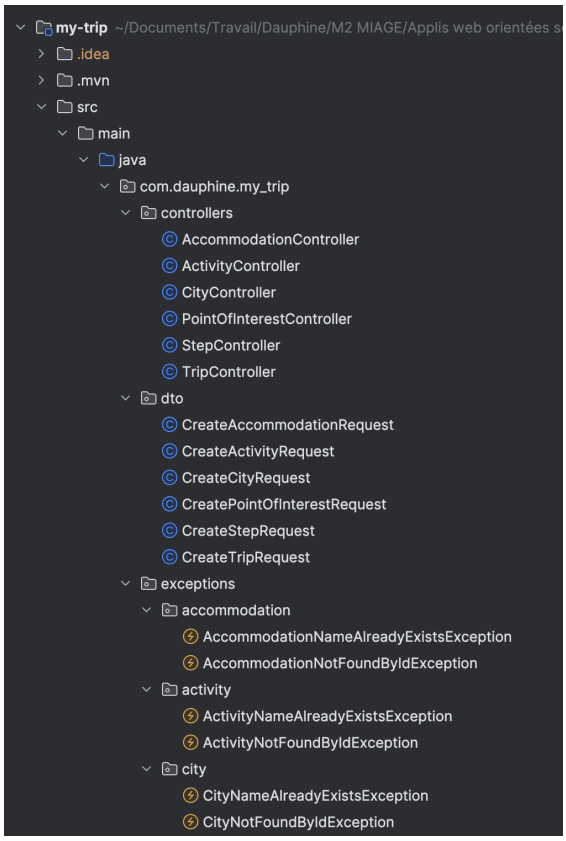
Nous avons également utilisé Swagger car Daryl le connaissait. Également, il est très facile d'utilisation, et permet facilement de tester les API en plus de les documenter.

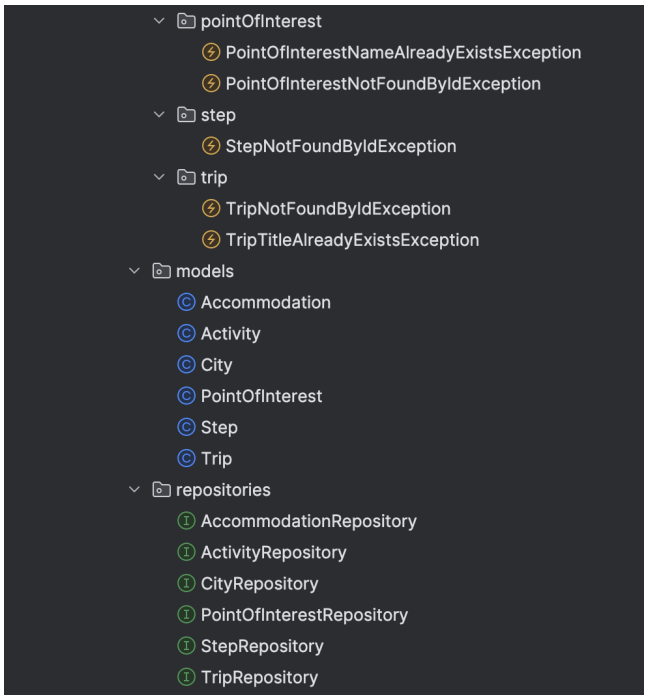
Organisation du code

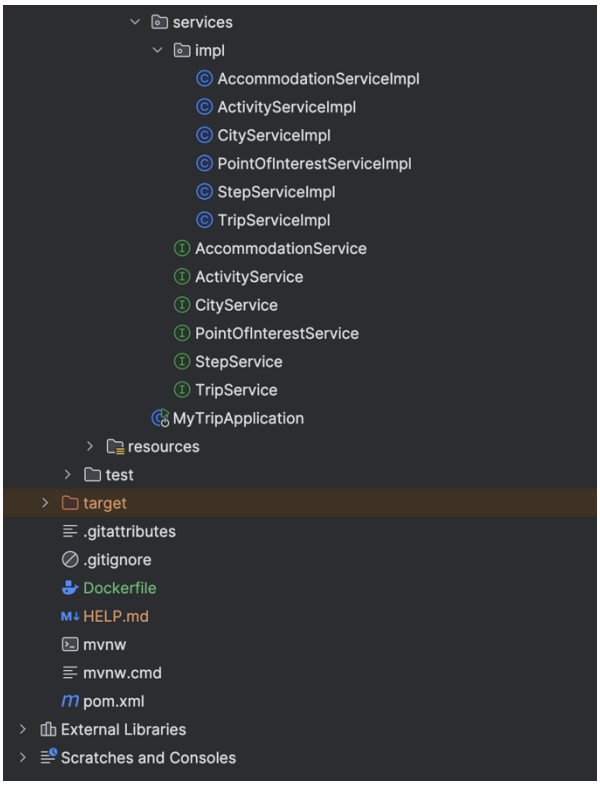
Structure du projet

Le projet suit une architecture en couches respectant les bonnes pratiques Spring Boot. Dans notre application, nous avons un répertoire « **controller** » (Couche présentation,

API REST), un répertoire « **service** » (couche métier), un répertoire « **repository** » (couche accès aux données), un répertoire « **model** » (entités métier), et un répertoire « **dto** » (pour les CreateRequest).



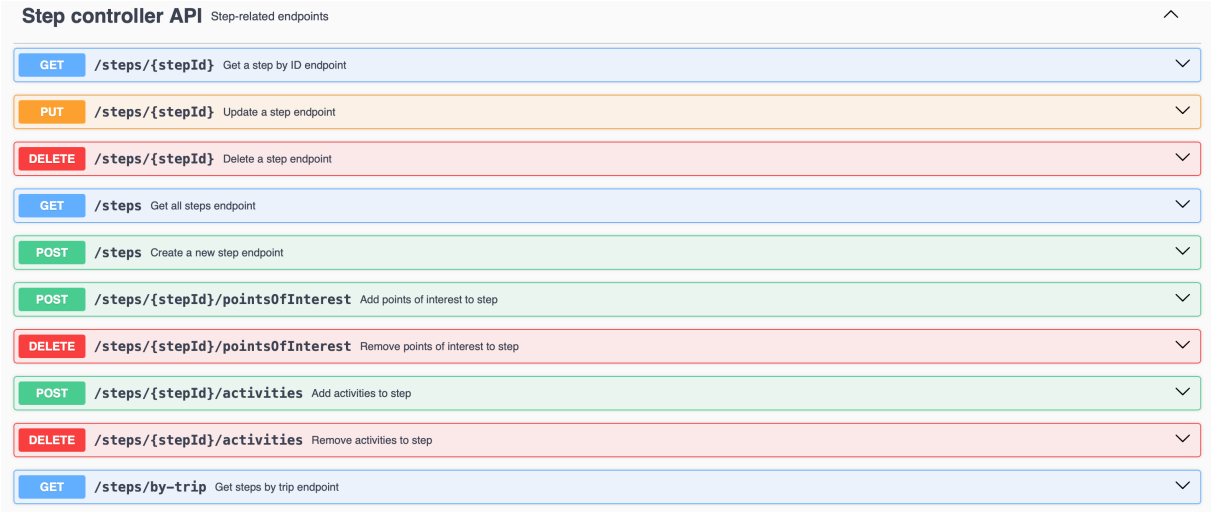




Endpoints API

Chaque contrôleur implémente les opérations CRUD standard (GET, POST, PUT, DELETE).

Exemple - StepController (Gestion des étapes) :



Cette approche garantit une API cohérente et prévisible pour tous les services.

Déploiement et conteneurisation

Pour utiliser notre application, vous pouvez accéder à notre repository GitHub, grâce au lien suivant : https://github.com/DarylMartinDipp/my-trip/

Clone du projet :

```
git clone git@github.com:DarylMartinDipp/my-trip.git
```

Docker

La conteneurisation avec Docker apporte plusieurs avantages :

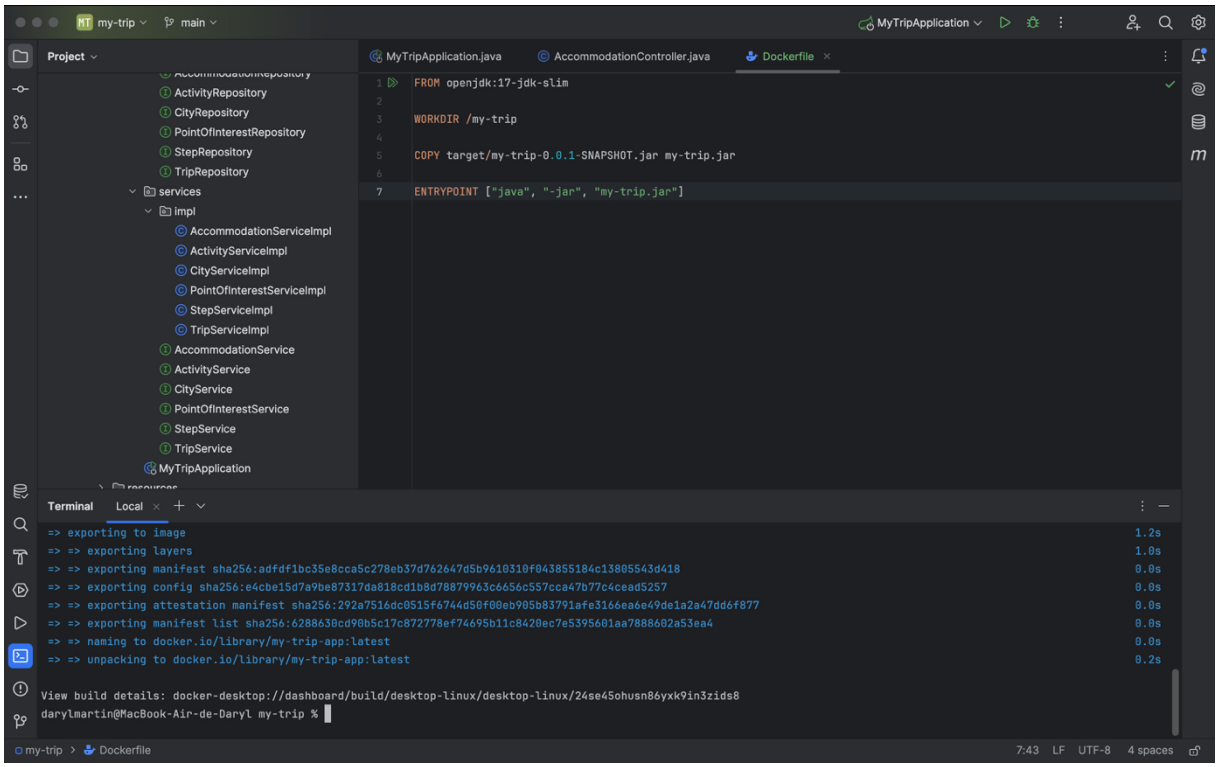
Dockerfile optimisé :

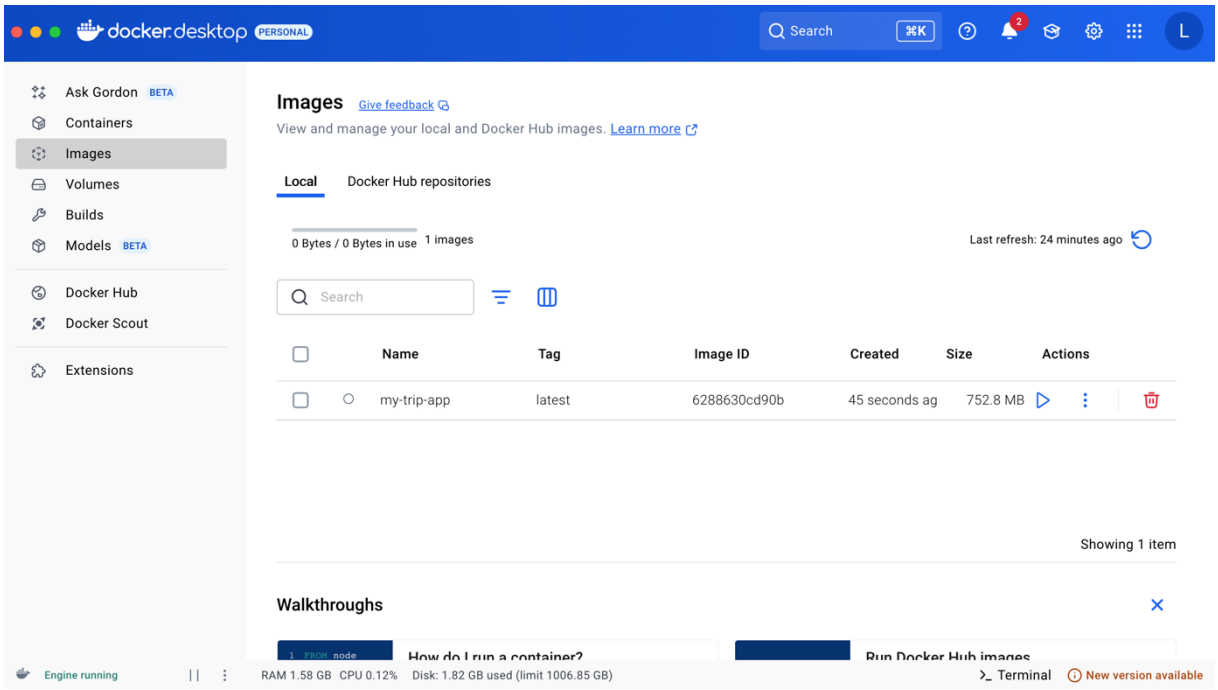
```
FROM openjdk:17-jdk-slim
COPY target/my-trip-app.jar app.jar
EXPOSE 8080
ENTRYPOINT ["java", "-jar", "/app.jar"]
```

Commandes de lancement :

```
# Construction de l'image
docker build -t my-trip-app .

# Lancement du conteneur
docker run -p 8080:8080 my-trip-app
```





Configuration de la base de données

Prérequis pour l'accès à Supabase :

En raison des limitations du plan gratuit de Supabase, la base de données se met en veille automatiquement après une période d'inactivité. Pour garantir le bon fonctionnement de l'application, il est nécessaire de réactiver la base de données avant chaque utilisation.

Procédure de réactivation :

1. Se connecter sur le tableau de bord Supabase avec les identifiants configurés dans `resources/application.properties`
2. Accéder au projet "my-trip"
3. Rafraîchir la page du dashboard pour réveiller la base de données
4. Vérifier que le statut de la base indique "Active" (indicateur vert)
5. Lancer l'application Spring Boot

Note importante : Cette étape est obligatoire à chaque redémarrage de session de travail. Sans cette réactivation préalable, les appels API retourneront des erreurs de connexion à la base de données.

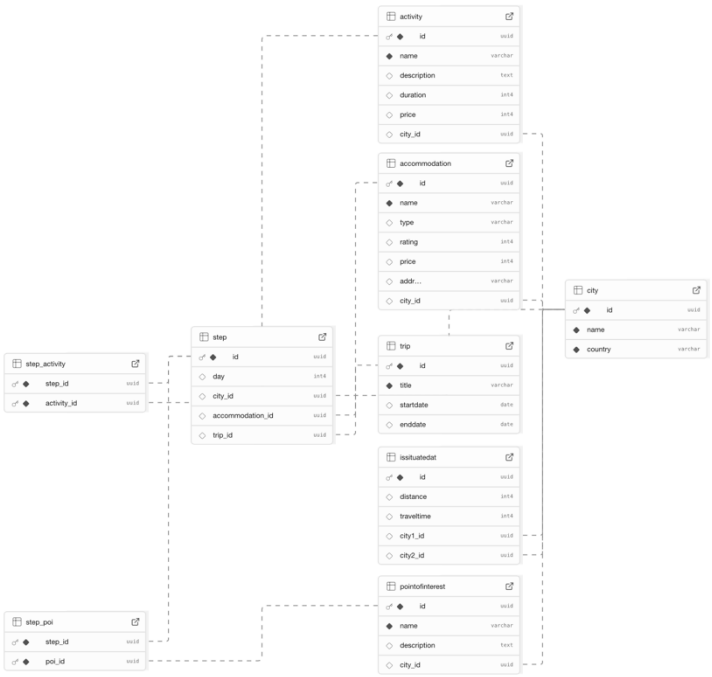
En cas de problème de connexion, vous pouvez contacter Daryl MARTIN-DIPP.

Modélisation métier

Diagramme de classes

Le modèle métier s'articule autour des entités principales :

- **Trip :** Représentation d'un voyage.
- **Step :** Représentation d'une étape, qui contient une **City**, un **Accommodation** et un **Trip**, une liste d'**Activity** et une liste de **PointOfInterest**.
- **City :** Représentation d'une ville.
- **Accommodation :** Représentation d'un hébergement, qui contient une **City**.
- **Activity :** Représentation d'une activité, qui contient une **City**.
- **PointOfInterest :** Représentation d'un point d'intérêt, qui contient une **City**.



Retour d'expérience

Points positifs

Ce projet nous a permis de développer une maîtrise de la séparation des responsabilités et d'appliquer de façon pratique les principes architecturaux des microservices étudiés en cours. L'utilisation d'outils modernes s'est révélée particulièrement enrichissante : Docker a facilité le déploiement et la portabilité, Swagger a amélioré significativement les tests et la documentation des APIs... Notre approche de développement backend-first, couplée aux facilités de test de Swagger, nous a permis de nous concentrer sur la logique métier tout en optimisant notre processus de développement et de validation.

Défis rencontrés

Nous avons rencontré quelques difficultés techniques qui se sont révélées formatrices. Nous avons eu une incompatibilité de types Date entre Java et PostgreSQL, mais ce problème a pu être plutôt rapidement réglé. De plus, un import incorrect de librairie nous a fait perdre du temps en débogage. Sur le plan pédagogique, nous avons ressenti un manque de séances de TP encadrées qui auraient pu nous aider dans notre apprentissage. Des échanges plus fréquents avec vous, en salle de classe, auraient pu été bénéfiques pour nous.