TANTOT Etienne

2024-2025

PROJET FINAL

VIRTUALISATION-CONTENEURISATION

1 INTRODUCTION

Le projet vise à concevoir une application web basée sur une architecture microservices. À travers cette réalisation, nous avons exploré les concepts de conteneurisation avec Docker, d'orchestration avec Docker Compose, et d'interaction entre plusieurs services conteneurisés (backend, frontend et base de données).

Objectifs du Projet:

- Développer une application composée de plusieurs services collaboratifs.
- Conteneuriser ces services avec Docker.
- Orchestrer les conteneurs pour garantir leur fonctionnement ensemble.
- Créer une documentation claire pour faciliter l'installation et l'utilisation.

Structure du Projet

```
FINAL_PROJECT_DOCKER/
├──── backend/
│   ├──── app/
│   ├──── main.py
│   ├──── Dockerfile
│   └──── requirements.txt
├──── frontend/
│   ├──── src/
│   ├──── app.js
│   ├──── index.html
│   └──── styles.css
├──── Dockerfile
├──── database/
│   └────Dockerfile
├──── docker-compose.yml
├──── FINAL_PROJECT.md
└──── README.md      # Documentation du projet
```

- Backend : Développement d'une API REST pour la gestion des utilisateurs avec Python (FastAPI).
- Frontend : Création d'une interface utilisateur en HTML, CSS, et JavaScript.
- Database : Mise en place d'une base de données PostgreSQL pour le stockage des informations.
- Orchestration : Gestion de l'interaction entre les conteneurs avec Docker Compose.

Développement des Services

Backend

- Technologie : Python avec FastAPI.
- Fonctionnalités :
  - POST /users : Ajouter un utilisateur.
  - GET /users : Récupérer la liste des utilisateurs.
  - DELETE /users/{id} : Supprimer un utilisateur.
- Connexion avec PostgreSQL via psycopg2.

Frontend

- Technologie : HTML/CSS/JavaScript.
- Fonctionnalités :
  - Formulaire pour ajouter un utilisateur.
  - Liste dynamique affichant les utilisateurs.
  - Boutons pour supprimer des utilisateurs.
- Interaction avec l'API via Fetch API.

Base de Données

- Technologie : PostgreSQL.
- Conteneur PostgreSQL configuré avec un utilisateur, mot de passe et base de données.
- Option de script SQL d'initialisation pour préremplir des données.

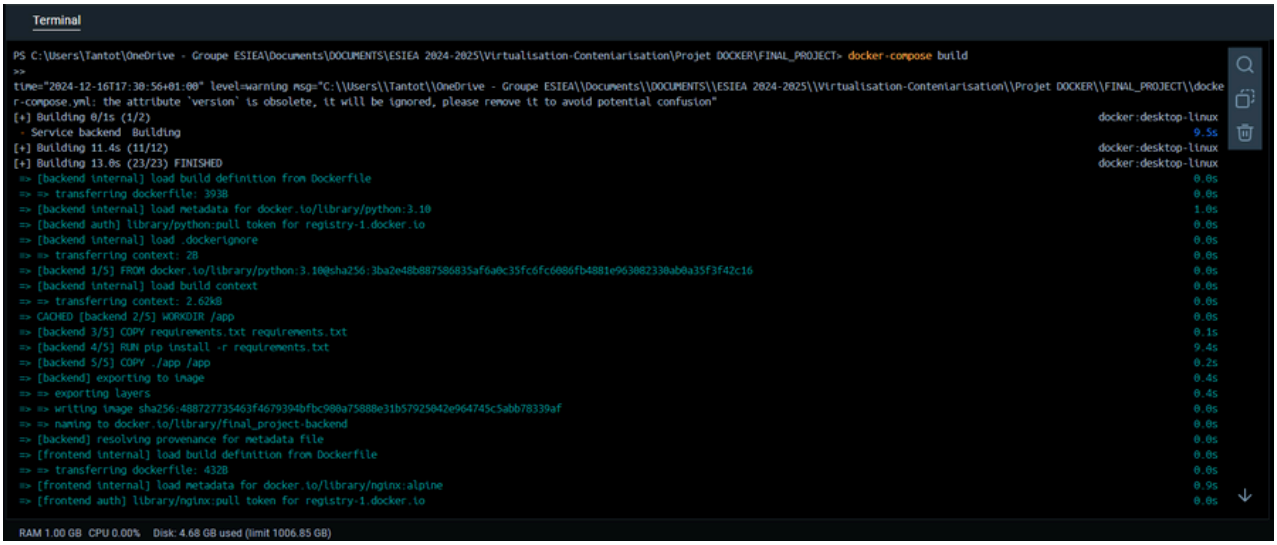
Compilation et Exécution

Noté qu'il est important de bien installer toutes les librairies et bibliothèques nécessaires au projet. Il suffit d'installer toutes les librairies présentes dans le fichier" requirements.txt" dans le dossier "backend".

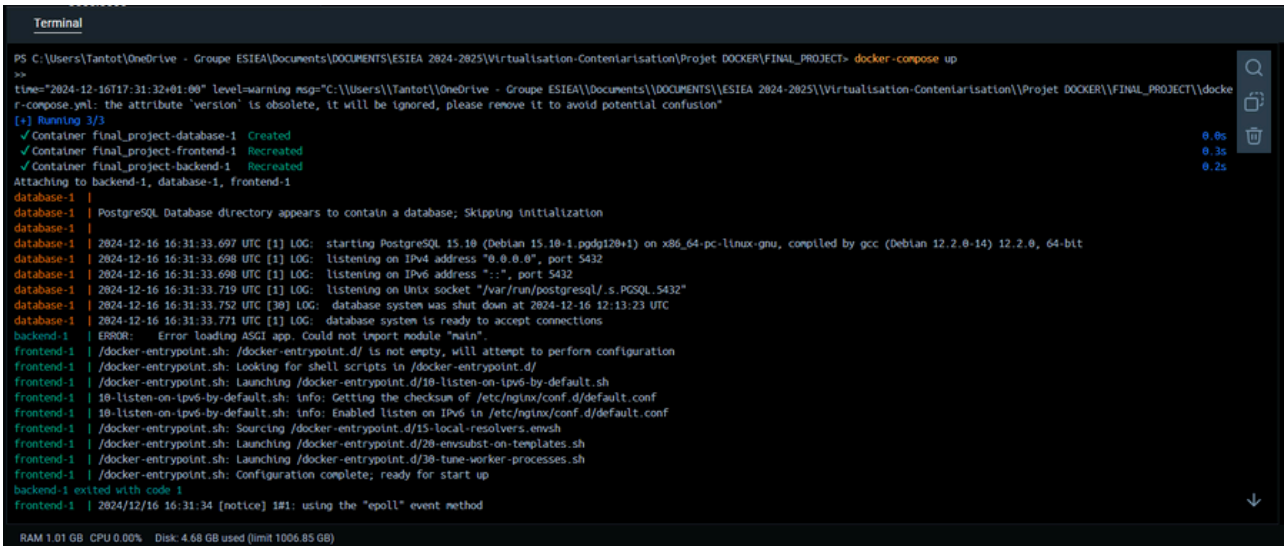
Vous pouvez installer les librairies avec la commande suivante: **pip install -r requirements.txt**

```
# FastAPI pour créer et gérer l'API
fastapi==0.95.2

# Uvicorn comme serveur ASGI pour exécuter FastAPI
uvicorn[standard]==0.22.0

# Pydantic pour la validation des données utilisateur
pydantic==1.10.2

# psycopg2 pour interagir avec PostgreSQL
psycopg2-binary==2.9.6

# python-dotenv pour la gestion des variables d'environnement (fichier .env)
python-dotenv==1.0.0
```

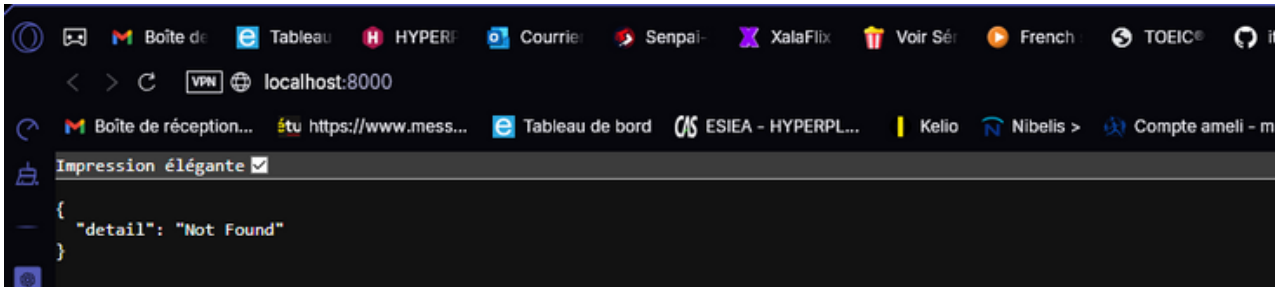
A partir de la depuis le chemin du projet dans le terminal, on peut taper la commande suivante afin de construire les services Docker: **docker-compose build**

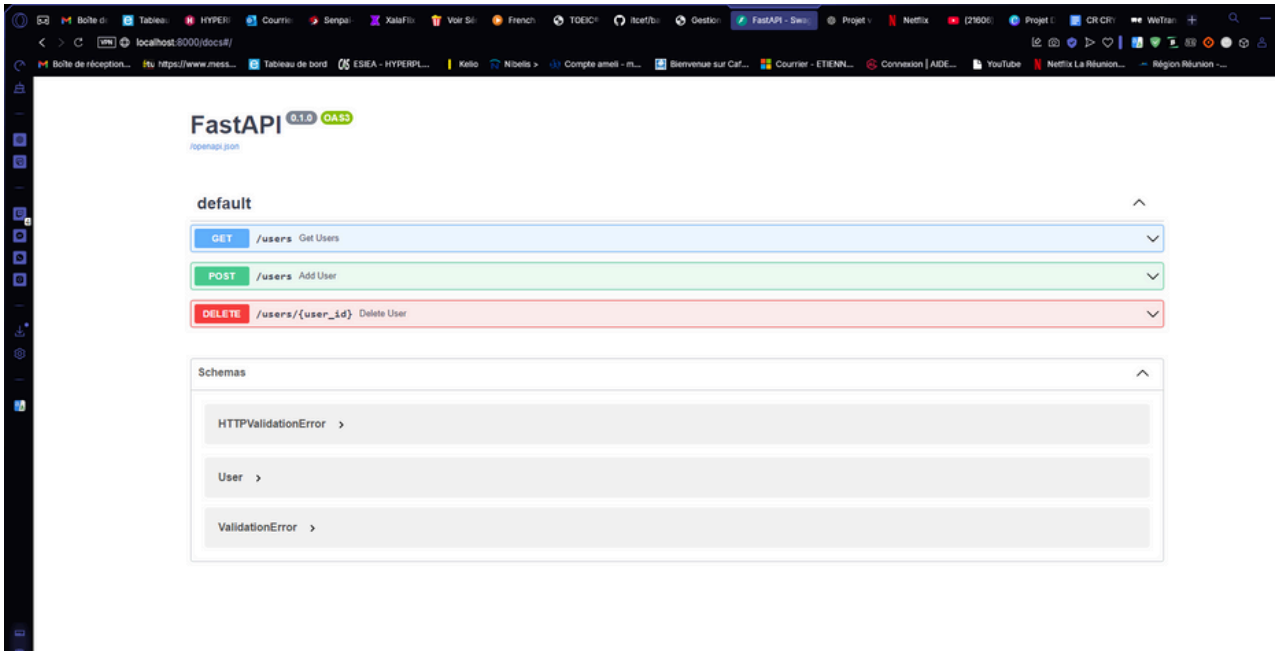


Le projet va se compiler et se construire dans le terminal du Docker. On tape la commande suivante pour démarrer les services: **docker-compose up**

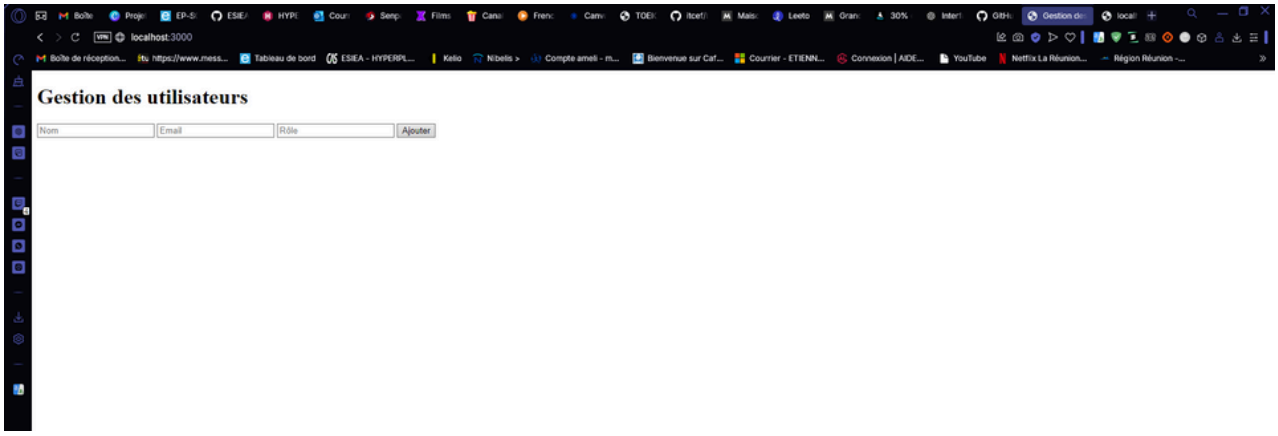


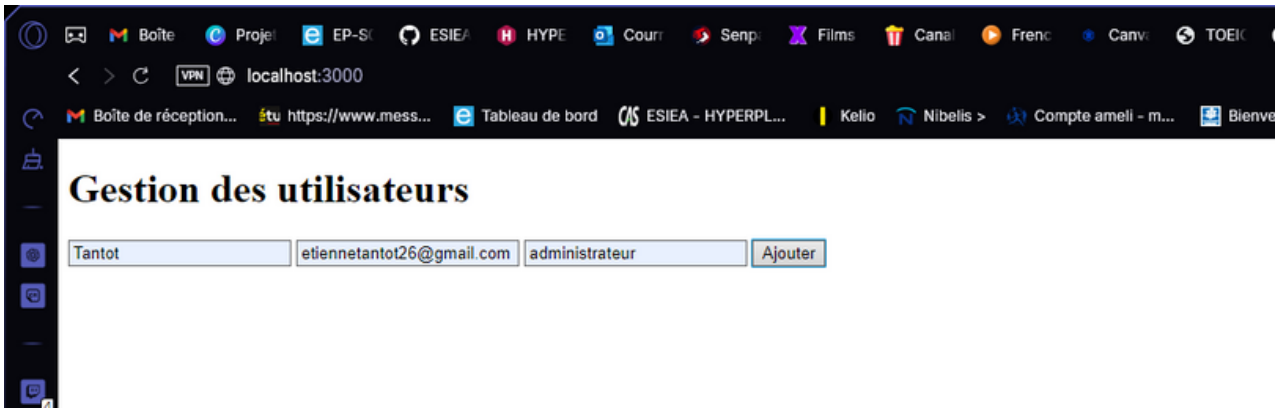
On peut accéder au backend avec le lien: **http://localhost:8000.** On peut même accéder au backend: **http://localhost:8000/docs**.



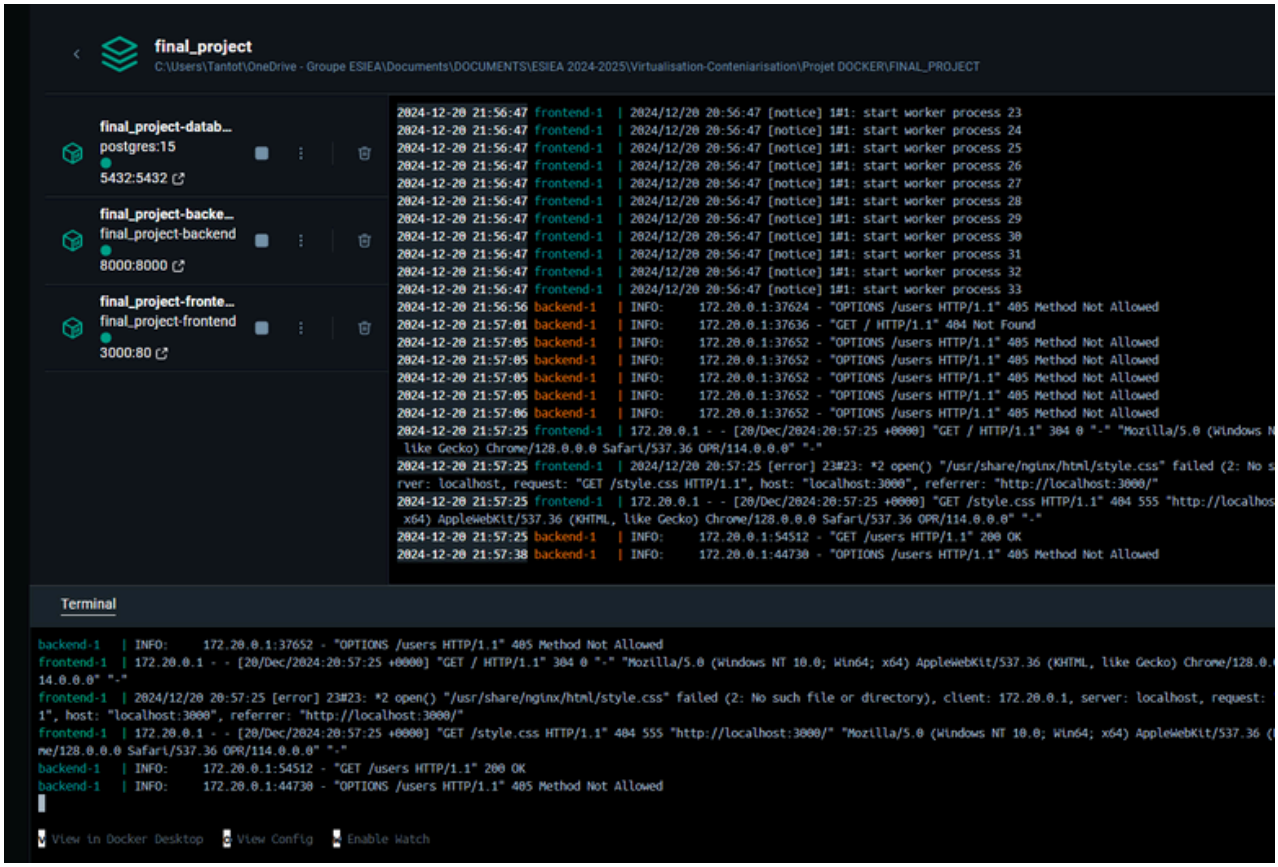


On peut accéder au backend avec le lien: **http://localhost:3000** auquel on peut ajouter, un nom, un email et un rôle.





On peut voir tout les informations et les données dans le logiciel Docker.



Voici le lien du projet finale Docker (Virtualisation): https://github.com/Etienne97460/FINAL_PROJECT_DOCKER.git