

# 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.



```
Terminal

=> [frontend internal] load .dockerignore
=> => transferring context: 2B
=> [frontend 1/4] FROM docker.io/library/nginx:alpine@sha256:41523187cf7d7a2f2677a8969d9caa14388bf5c1fba9c418ba3de662aaaab4
=> [frontend internal] load build context
=> => transferring context: 1.51kB
=> CACHED [frontend 2/4] WORKDIR /usr/share/nginx/html
=> CACHED [frontend 3/4] RUN rm -rf ./#
=> [frontend 4/4] COPY src/ .
=> [frontend] exporting to image
=> => exporting layers
[+] Building 2/2stage:sha256:4215c86df7fd11302a2fa9945c9b2058ffbb8f9b78a52cc4c0ee2be9150e72
✓ Service backend built
✓ Service frontend built
```

```
Terminal

frontend-1 | 2024/12/16 16:31:34 [notice] 1#1: using the "epoll" event method
frontend-1 | 2024/12/16 16:31:34 [notice] 1#1: nginx/1.27.3
frontend-1 | 2024/12/16 16:31:34 [notice] 1#1: built by gcc 13.2.1 20240309 (Alpine 13.2.1-glt20240309)
frontend-1 | 2024/12/16 16:31:34 [notice] 1#1: OS: Linux 5.15.167.4-microsoft-standard-WSL2
frontend-1 | 2024/12/16 16:31:34 [notice] 1#1: getrlimit(RLIMIT_NOFILE): 1048576:1048576
frontend-1 | 2024/12/16 16:31:34 [notice] 1#1: start worker processes
frontend-1 | 2024/12/16 16:31:34 [notice] 1#1: start worker process 30
frontend-1 | 2024/12/16 16:31:34 [notice] 1#1: start worker process 33
frontend-1 | 2024/12/16 16:31:34 [notice] 1#1: start worker process 34
frontend-1 | 2024/12/16 16:31:34 [notice] 1#1: start worker process 35
frontend-1 | 2024/12/16 16:31:34 [notice] 1#1: start worker process 36
frontend-1 | 2024/12/16 16:31:34 [notice] 1#1: start worker process 37
frontend-1 | 2024/12/16 16:31:34 [notice] 1#1: start worker process 38
frontend-1 | 2024/12/16 16:31:34 [notice] 1#1: start worker process 39
frontend-1 | 2024/12/16 16:31:34 [notice] 1#1: start worker process 40
frontend-1 | 2024/12/16 16:31:34 [notice] 1#1: start worker process 41

frontend-1 | 172.20.0.1 - - [16/Dec/2024:16:31:54 +0000] "GET / HTTP/1.1" 304 0 "-" "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/128.0.0.0 Safari/537.36 OPR/14.0.0.0" "-"
frontend-1 | 172.20.0.1 - - [16/Dec/2024:16:31:54 +0000] "GET /style.css HTTP/1.1" 404 555 "http://localhost:3000/" "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/128.0.0.0 Safari/537.36 OPR/14.0.0.0" "-"
frontend-1 | 2024/12/16 16:31:54 [error] 300#0: *1 open() "/usr/share/nginx/html/style.css" failed (2: No such file or directory), client: 172.20.0.1, server: localhost, request: "GET /style.css HTTP/1.1", host: "localhost:3000", referer: "http://localhost:3000/"
frontend-1 | 172.20.0.1 - - [16/Dec/2024:16:31:54 +0000] "GET /app.js HTTP/1.1" 200 2521 "http://localhost:3000/" "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/128.0.0.0 Safari/537.36 OPR/14.0.0.0" "-"

View in Docker Desktop View Config Enable Watch

RAM 1.02 GB CPU 0.00% Disk 4.68 GB used (limit 1006.85 GB)
```

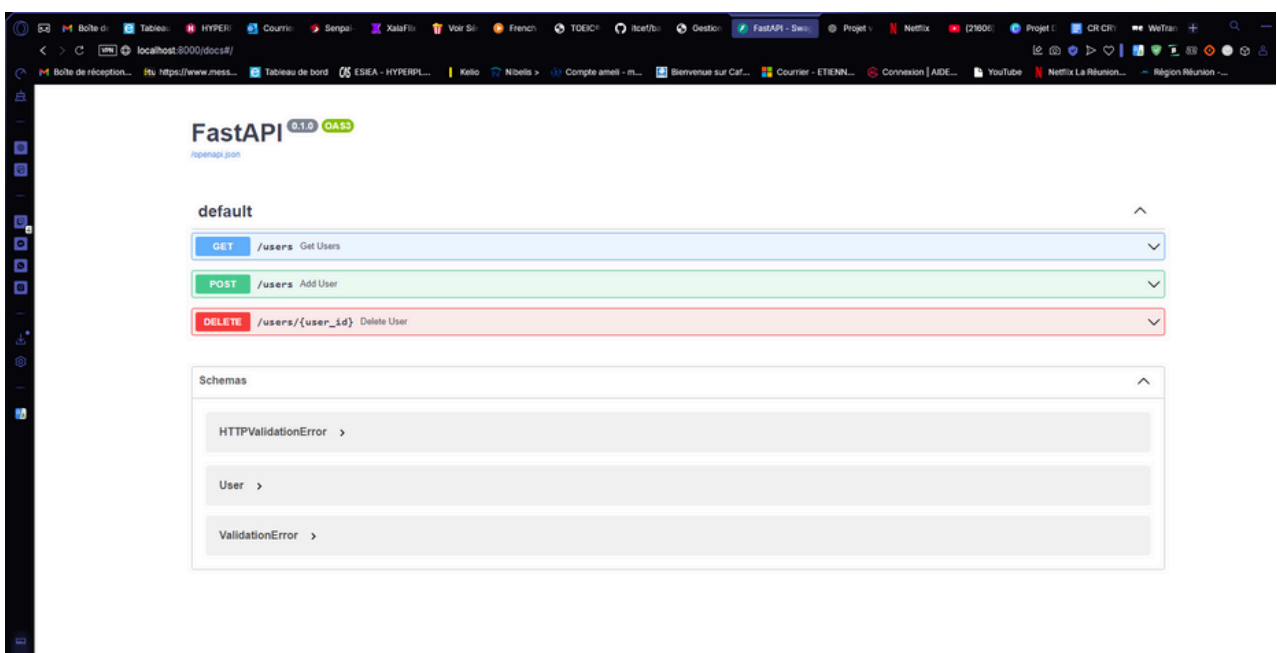
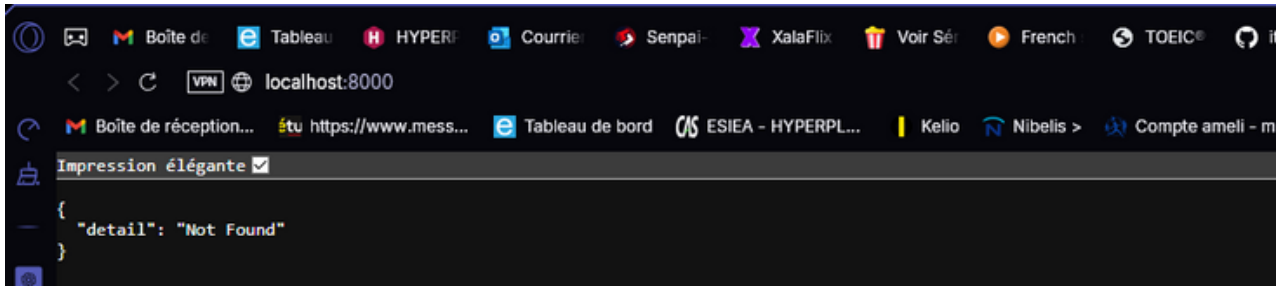
Le projet va se compiler et se construire dans le terminal du Docker. On tape la commande suivante pour démarrer les services: **docker-compose up**

```
Terminal

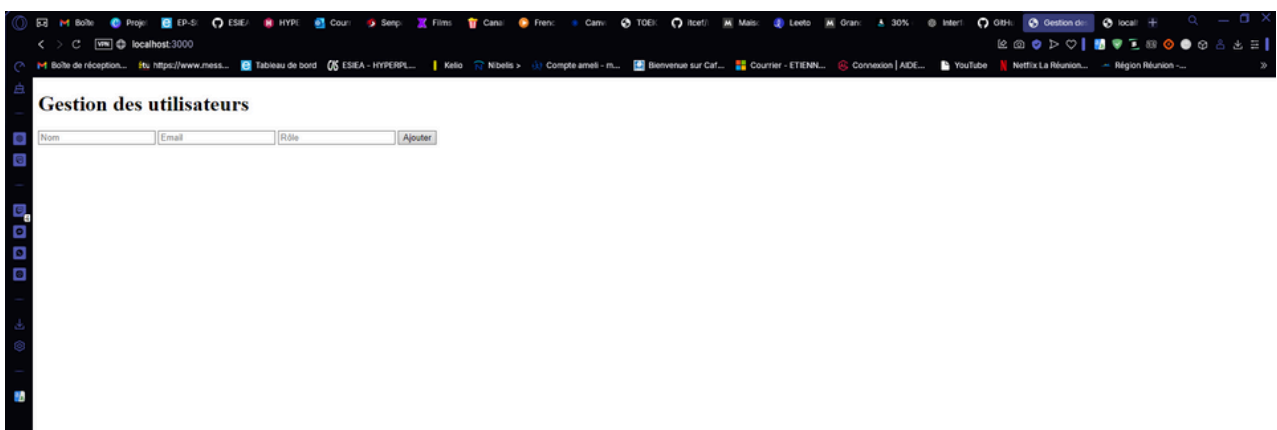
PS C:\Users\tantot\OneDrive - Groupe ESIEA\Documents\DOCUMENTS\ESIEA 2024-2025\Virtualisation-Contenarisation\Projet DOCKER\FINAL_PROJECT> docker-compose up
time="2024-12-16T17:31:32+01:00" level=warning msg="C:\Users\tantot\OneDrive - Groupe ESIEA\Documents\DOCUMENTS\ESIEA 2024-2025\Virtualisation-Contenarisation\Projet DOCKER\FINAL_PROJECT\docker-compose.yml: the attribute 'version' is obsolete, it will be ignored, please remove it to avoid potential confusion"
[+] Running 3/0
✓ Container final_project-database-1 Created
✓ Container final_project-frontend-1 Recreated
✓ Container final_project-backend-1 Recreated
Attaching to backend-1, database-1, frontend-1
database-1 | PostgreSQL Database directory appears to contain a database; Skipping initialization
database-1 |
database-1 | 2024-12-16 16:31:33.697 UTC [1] LOG: starting PostgreSQL 15.10 (Debian 15.10-1.pgdg12041) on x86_64-pc-linux-gnu, compiled by gcc (Debian 12.2.0-14) 12.2.0, 64-bit
database-1 | 2024-12-16 16:31:33.698 UTC [1] LOG: listening on IPv4 address "0.0.0.0", port 5432
database-1 | 2024-12-16 16:31:33.698 UTC [1] LOG: listening on IPv6 address ":::", port 5432
database-1 | 2024-12-16 16:31:33.719 UTC [1] LOG: listening on Unix socket "/var/run/postgresql/.s.PGSQL.5432"
database-1 | 2024-12-16 16:31:33.752 UTC [30] LOG: database system was shut down at 2024-12-16 12:13:23 UTC
database-1 | 2024-12-16 16:31:33.771 UTC [1] LOG: database system is ready to accept connections
backend-1 | ERROR: Error loading ASGI app. Could not import module "main".
frontend-1 | /docker-entrypoint.sh: /docker-entrypoint.d/ is not empty, will attempt to perform configuration
frontend-1 | /docker-entrypoint.sh: Looking for shell scripts in /docker-entrypoint.d/
frontend-1 | /docker-entrypoint.sh: Launching /docker-entrypoint.d/10-listen-on-ipv6-by-default.sh
frontend-1 | 10-listen-on-ipv6-by-default.sh: info: Getting the checksum of /etc/nginx/conf.d/default.conf
frontend-1 | 10-listen-on-ipv6-by-default.sh: info: Enabled listen on IPv6 in /etc/nginx/conf.d/default.conf
frontend-1 | /docker-entrypoint.sh: Sourcing /docker-entrypoint.d/15-local-resolvers.envsh
frontend-1 | /docker-entrypoint.sh: Launching /docker-entrypoint.d/20-envsubst-on-templates.sh
frontend-1 | /docker-entrypoint.sh: Launching /docker-entrypoint.d/30-tune-worker-processes.sh
frontend-1 | /docker-entrypoint.sh: Configuration complete; ready for start up
backend-1 exited with code 1
frontend-1 | 2024/12/16 16:31:34 [notice] 1#1: using the "epoll" event method

RAM 1.01 GB CPU 0.00% Disk 4.68 GB used (limit 1006.85 GB)
```

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

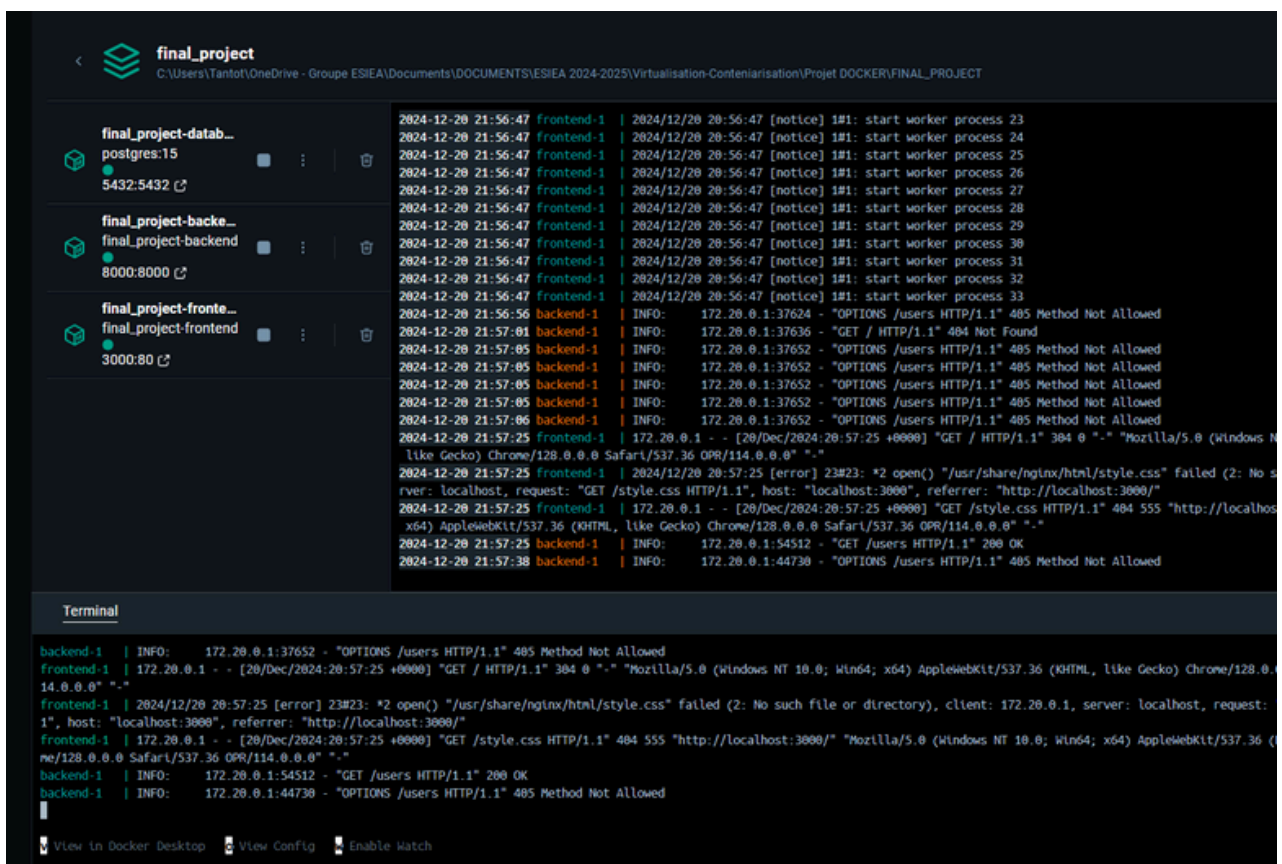


On peut accéder au backend avec le lien: **<http://localhost:3000>** auquel on peut ajouter, un nom, un email et un rôle.





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](https://github.com/Etienne97460/FINAL_PROJECT_DOCKER.git)