TP – Containerisation de l'application Digicheese

Vous allez **containeriser l'application Digicheese**, un projet de gestion de fidélisation client pour une fromagerie artisanale.

Cette application utilise :

- **FastAPI** pour l'API web
- **SQLAlchemy** comme ORM
- **MariaDB** pour la base de données
- **Swagger UI** pour la documentation et les tests interactifs

Le code source est déjà fourni dans le dossier `digicheese/`.

🎯 Objectif

Mettre en place un environnement **Docker complet, isolé et reproductible** pour exécuter l'application.

Vous devrez :

- Créer un `Dockerfile` pour l'application FastAPI
- Configurer un `docker-compose.yml` avec plusieurs services (API, DB, interface Adminer)
- Utiliser un fichier `.env` pour centraliser la configuration
- Monter les bons volumes (code, tests, base de données)
- Ajouter un réseau Docker pour permettre la communication entre conteneurs

🧱 Architecture attendue

L'environnement comportera les services suivants :

Service	Nom du container	Rôle
`fastapi`	api-digicheese	Application FastAPI avec Uvicorn
`db`	db-digicheese	Base de données MariaDB 10.6
`adminer`	adminer-digicheese	Interface web pour gérer la base de données

Volumes :

Source	Cible
`./api/`	`/code/api`
`./tests/`	`/code/tests`
Volume `digicheese_data`	`/var/lib/mysql`

Arborescence du projet

Voici l'arborescence attendue pour le projet Digicheese :

```
digicheese/
├── api/
│   ├── main.py
│   ├── models/
│   ├── routers/
│   ├── schemas/
│   └── services/
├── tests/
│   ├── conftest.py
│   └── test_client.py
├── .env
├── requirements.txt
├── Dockerfile
└── docker-compose.yml
```

Dossier `api/` et `tests/`

Le dossier `api/` contient le code source de l'application FastAPI. Le dossier `tests/` contient les tests unitaires. Ces deux dossiers doivent être montés en volume dans le conteneur.

Vous n'avez pas besoin de modifier le code source, mais vous devez vous assurer que les **chemins dans le `Dockerfile` et le `docker-compose.yml`** sont corrects.

Pour exécuter FastAPI sans Docker, on utilise Uvicorn :

```
uvicorn api.main:app --port 8000 --reload
```

> Cette commande sera adaptée dans le `Dockerfile` pour que le serveur démarre automatiquement dans le conteneur.

Fichier `requirements.txt`

Le fichier `requirements.txt` contient les dépendances nécessaires à l'application. Il devra être utilisé dans le `Dockerfile` pour les installer dans l'image.

```
pip install -r requirements.txt
```

> Cette commande est à insérer dans le `Dockerfile`.

Fichier `.env`

Le fichier `.env` contient la configuration partagée entre les services.

```
MYSQL_ROOT_PASSWORD=securepassword
USER=admin
PASSWORD=Admin123!
DATABASE=digicheese
```

> Ce fichier sera utilisé dans `docker-compose.yml` à la fois pour la base de données et l'application FastAPI.
>
> Exemple : la variable `USER` dans `.env` sera mappée à `DB_USER` dans le service FastAPI et à `MYSQL_USER` dans le service MariaDB.

Étapes pour le Dockerfile

1. Utiliser une image de base Python (`python:3.11`)
2. Définir le répertoire de travail dans le conteneur (`/code`)
3. Copier le fichier `requirements.txt` et installer les dépendances avec `pip`
4. Exposer le port `80` pour l'application FastAPI
5. Lancer le serveur FastAPI avec Uvicorn :

```
CMD ["uvicorn", "api.main:app", "--host", "0.0.0.0", "--port", "80", "--reload",
"--reload-dir", "api"]
```

> Le `--reload-dir` permet de recharger automatiquement les fichiers montés.

Étapes pour le `docker-compose.yml`

1. **Service `fastapi`**
    - Nom du conteneur : `api-digicheese`
    - Construit à partir du `Dockerfile`
    - Port exposé : `8000` → `80` dans le conteneur
    - Monte les volumes `./api` et `./tests`
    - Dépend du service `db`
    - Définir les variables d'environnement depuis `.env` ou inséré en brut:
        - `DB_USER` (.env)
        - `DB_PASSWORD` (.env)
        - `DB_HOST` (brut)

        - `DB_PORT` (brut)
        - `DB_NAME` (.env)
    - Réseau : `digicheese-net`

2. **Service `db`**

    - Image : `mariadb:10.6`
    - Nom du conteneur : `db-digicheese`
    - Volume nommé `digicheese_data` → `/var/lib/mysql`
    - Grâce au fichier `.env`, renseignez les variables :
        - `MYSQL_ROOT_PASSWORD`,
        - `MYSQL_DATABASE`,
        - `MYSQL_USER`,
        - `MYSQL_PASSWORD`
    - Réseau : `digicheese-net`

3. **Service `adminer`**

    - Image : `adminer`
    - Nom du container : `adminer-digicheese`
    - Port exposé : `8070` → `8080` dans le conteneur

> N'oubliez pas de créer un réseau Docker nommé `digicheese-net` pour permettre la communication entre les conteneurs et de definir le volume nommé `digicheese_data` pour la persistance de la base de données.

🔧 Commandes Docker utiles

```
# Lancer tous les conteneurs en arrière-plan
docker-compose up -d

# Reconstruire uniquement le conteneur de l'application
docker-compose build api-digicheese

# Afficher les logs de l'application FastAPI
docker-compose logs -f api-digicheese

# Ouvrir un terminal dans le conteneur de l'application
docker-compose exec api-digicheese bash

# Se connecter à la base MariaDB via CLI
docker-compose exec db-digicheese mysql -u admin -p digicheese
```

☑ Livrables attendus

- Un `Dockerfile` fonctionnel
- Un `docker-compose.yml` bien structuré
- Une application accessible à `http://localhost:8000/docs`
- Une base de données persistante et connectée
- Un dépôt GitHub contenant le dossier `digicheese/` avec tous les fichiers nécessaires

📝 **À remettre :** Lien GitHub vers le dossier `digicheese/` complet.