TP – Création et conteneurisation d’une mini-application CRUD

Objectifs

- Concevoir une **application CRUD full-stack**
- Développer :
  - un **frontend**
  - une **API REST**
  - une **base de données MongoDB**
- Utiliser **MongoDB Compass** pour visualiser et manipuler les données
- **Dockeriser** chaque composant
- Mettre en place un **docker-compose** avec **bind mounts**
- Comprendre l’architecture d’une application moderne conteneurisée

Contexte de l’application

Vous devez réaliser une **mini-application CRUD** au choix :

- Gestion de **clients**
- Gestion de **produits**
- Gestion de **contacts**
- Gestion de **tâches**

L’application doit permettre **au minimum** :

- Ajouter un élément
- Lister les éléments
- Modifier un élément (uniquement back)
- Supprimer un élément

Architecture attendue

```
mini-app/
│
├── frontend/        Application Angular ou autre
├── backend/         API Node.js / Express ou autre
├── docker/
│    ├── frontend/
│    └── backend/
│
├── docker-compose.yml
└── README.md
```

Partie 1 – Frontend

Exigences

- Création d'un projet front
- Affichage de la liste des éléments
- Communication avec l'API via HTTP

Partie 2 – Backend

Exigences

- Création d'un serveur
- Mise en place d'une API REST :
  - GET
  - POST
  - PUT
  - DELETE
- Connexion à MongoDB
- Utilisation de variables d'environnement

Partie 3 – Base de données MongoDB

Exigences

- Base MongoDB exécutée dans un conteneur Docker
- Visualisation et manipulation via **MongoDB Compass**
- Création d'une collection liée au thème choisi
- Données persistantes via **volume**

Partie 4 – Dockerisation

Backend

- Création d'un Dockerfile
- Installation des dépendances
- Lancement du serveur
- Utilisation d'un **bind mount** pour le code

Frontend

- Création d'un Dockerfile
- Lancement de l'application front
- Accès depuis le navigateur
- Utilisation d'un **bind mount** pour le code

Partie 5 – Docker Compose

Exigences

- Un seul fichier docker-compose.yml
- Services :
    - frontend
    - backend
    - mongodb
    - compass

- Réseau commun
- Variables d'environnement
- **Bind mounts obligatoires**
- Redémarrage automatique des services

Partie 6 – Tests & Validation

Les étudiants devront vérifier :

- L'application fonctionne via le navigateur
- Les opérations CRUD sont fonctionnelles
- Les données sont visibles dans MongoDB Compass
- L'arrêt / redémarrage des conteneurs ne supprime pas les données
- Le code est modifiable sans rebuild grâce aux bind mounts

Partie 7 – Documentation (README)

Le projet devra contenir un README.md expliquant :

- Le thème de l'application
- La classe et les personnes presents dans le groupe
- L'architecture
- Les commandes pour :
  - lancer le projet
  - arrêter le projet
- Les ports utilisés

Livraison

Les étudiants devront remettre lle lien vers le GitHub avec le projet
Le travail peut être réalisé **individuellement ou en groupe de 3 à 4 étudiants maximum**.

Les fichiers devront être envoyés par e-mail à **ehouri@formateur.ief2i.fr avant 17h00**.