Docker Compose PROJET

Consignes générales

- Tout le projet doit être orchestré avec **Docker Compose**
- Tous les services doivent être lancés via un **Makefile**
- Toute erreur de configuration entraînant un service non fonctionnel invalide l'exercice concerné
- un dossier par exo, mais tout les exos sont sur le même repo github (4 dossiers au total | 1 seul repo)

Exercice 1

Objectif

Mettre en place une architecture simple composée de deux services :

- un **backend**
- un **frontend**

Backend

- Expose une API HTTP sur le port **5000**
- Fournit une route accessible via :

```
GET /api/hello
```

- Cette route retourne un message **Hello World**

Frontend

- Exposé sur le port **3000**
- Récupère le message fourni par le backend
- Affiche ce message dans l'interface utilisateur

Makefile

Le Makefile doit contenir les règles suivantes :

- `all` : lance l'application via Docker Compose
- `clean` : arrête et supprime les conteneurs
- `fclean` : supprime les conteneurs **et** les images
- `re` : effectue un nettoyage complet puis relance l'application

---

Exercice 2 Base de données SQLite et CRUD

Base de données

- Utiliser **SQLite**
- La base doit être stockée dans un **volume**
- Toute modification effectuée :
  - depuis le conteneur
  - ou depuis l'hôte

    doit être répercutée dans les deux environnements

Backend

- Gère une ressource `user`
- Un utilisateur est défini par :
  - `username`
  - `password`
- Aucune validation des données n'est requise

API attendue

- Création d'un utilisateur
- Lecture d'un ou plusieurs utilisateurs
- Mise à jour d'un utilisateur
- Suppression d'un utilisateur

Frontend

- Permet d'interagir avec toutes les opérations CRUD

Makefile (nouvelle règle)

Une règle supplémentaire doit être ajoutée :

- `purge` :
  - supprime le contenu du volume
  - réinitialise complètement la base de données

Exercice 3 API externe et réseau Tor

Objectif

Faire transiter des requêtes HTTP via le réseau Tor.

API externe

- URL :

  ```
  https://randomuser.me/
  ```

Backend

- Récupère des utilisateurs depuis l'API externe
- Les requêtes doivent obligatoirement passer par **Tor**
- La communication avec Tor se fait via un **proxy SOCKS5**

Tor

- Exposé sur le port **9050**
- Accessible uniquement par le backend
- Service Snippet :

```yaml
tor:
  image: dperson/torproxy
  container_name: tor
  restart: unless-stopped
```

```
    # # ExitNodes {fr} tells Tor to prefer servers in France.
    # # StrictNodes 1 forces Tor to only use France.
    # # (If you set this to 0, Tor will try France first
    # # but use another country if the French servers are too slow).
    # environment:
    #   - LOCATION=FR      # Equivalent to ExitNodes {fr}
    #   - STRICT=1         # Equivalent to StrictNodes 1
    #   # This is the magic line for IP rotation
    #   - TOR_MaxCircuitDirtiness=600 # Rotate IP every 10 minutes (600 se
conds)
    ports:
      - "9050:9050"   # SOCKS5
      - "9051:9051"   # Control port
```

Frontend

- Affiche au minimum pour chaque utilisateur :
  - le **nom**
  - la **photo**

Exercice 4 Stack finale PostgreSQL

Makefile

Règles

- `purge bdd`
  - Supprime uniquement les volumes de bases de données
- `purge all`
  - Nettoie entièrement le projet :
    - conteneurs
    - images
    - volumes
    - réseaux

        - dossiers générés (ex : `node_modules`, fichiers temporaires, etc.)
    - Relance ensuite l'ensemble du stack Docker Compose

Configuration des fichiers `.env`

Les services doivent être configurés **exclusivement** via des fichiers d'environnement.

Aucune variable sensible ne doit être écrite en dur dans les fichiers Docker ou Docker Compose.

Backend (`backend.env`)

Le fichier doit contenir les informations suivantes :

- Nom de la base de données
- Utilisateur de la base de données
- Hôte de la base de données
- Port de la base de données
- Port d'exposition du backend
- URL du frontend
- Port du frontend

Frontend (`frontend.env`)

Le fichier doit contenir les informations suivantes :

- Port d'exposition du frontend
- URL du backend
- Port du backend

PgAdmin (`pgadmin.env`)

Le fichier doit contenir les informations suivantes :

- Adresse email par défaut pour l'accès à PgAdmin
- Mot de passe par défaut pour l'accès à PgAdmin

PostgreSQL (`postgres.env`)

Le fichier doit contenir les informations suivantes :

- Nom de la base de données
- Utilisateur PostgreSQL
- Mot de passe PostgreSQL

Fichier `pgadmin_servers.json`

Un fichier `pgadmin_servers.json` doit être fourni.

Ce fichier doit :

- Définir les serveurs PostgreSQL
- Permettre leur création **automatique** dans PgAdmin au démarrage via Docker Compose
- Assurer une connexion fonctionnelle entre PgAdmin et le service PostgreSQL

Arborescence minimale requise

Le projet doit respecter **strictement** l'arborescence suivante :

```
.
├── backend
│   └── src
│       └── etc
│
├── frontend
│   └── src
│       └── etc
│
├── docker
│   ├── backend.dockerfile
│   ├── frontend.dockerfile
│   ├── backend.env
│   ├── frontend.env
│   ├── pgadmin.env
```

```
│   ├── postgres.env
│   └── pgadmin_servers.json
│
├── makefile
│
└── docker-compose.yaml
```
