SAE — Développement & Déploiement d'une Application Web RESTful Conteneurisée

1. Objectif général

Dans cette SAE, vous êtes amené·e·s à **concevoir, développer, tester et déployer une application web** disposant d'un **backend exposant une API RESTful**. L'enjeu est de mettre en œuvre l'ensemble du cycle de vie logiciel, en appliquant les **bonnes pratiques du développement**, telles que la structuration modulaire du code, l'usage de contrôleurs/API, la documentation, les tests automatisés, et la gestion de versions, etc.

L'application devra être **conteneurisée** (via Docker) et déployée sur une **infrastructure simulée ou réelle** (par exemple en local, ou sur un cloud comme Heroku, Render). Le **sujet est libre**, à condition de respecter l'ensemble des **contraintes techniques** précisées ci-dessous (API REST, persistance des données, gestion des erreurs, etc.).

Cette SAE vous permettra de mobiliser des compétences en **architecture logicielle**, en **programmation orientée services**, ainsi qu'en **outillage DevOps** de base (conteneurisation, scripts de déploiement, documentation, etc.).

2. Contraintes techniques

- L'application doit être une **API REST** conforme aux standards HTTP (GET, POST, PUT/PATCH, DELETE).
- Le backend doit être développé dans le **langage/framework de votre choix** (Java/Spring Boot, Node/Express, Python/FastAPI, etc.).
- Les données doivent être persistées dans une **base de données relationnelle** (Oracle, PostgreSQL, MySQL, etc.).
- L'API doit être **conteneurisée** avec Docker, et la base de données aussi.
- Le tout doit être orchestré via **Docker Compose**.
- L'application doit obligatoirement utiliser un **framework de mapping objet-relationnel (ORM)** pour gérer la persistance des données.
- Vous devez modéliser au moins les trois types de relations suivantes :
  - *One-to-One*

- *One-to-Many / Many-to-One*
- *Many-to-Many*

Quelques exemple d'ORM recommandés

Langage	ORM possible
Java	Hibernate, JPA avec Spring Data
Python	SQLAlchemy, Peewee
Node.js	Sequelize, TypeORM, Prisma
PHP	Doctrine ORM, Eloquent (Laravel)

Les relations doivent être représentées à la fois dans le **code objet** (classes, entités) et dans le **schéma de la base relationnelle**.

3. Soutenance et évaluation

Chaque groupe devra présenter son projet lors d'une soutenance obligatoire, organisée en fin de SAE.

3.2. Format de la soutenance :

- **15 minutes** de présentation orale du projet :
  - Contexte et objectifs
  - Choix techniques
  - Démonstration de l'API (via Postman, Swagger, etc.)
  - Architecture Docker
- **5 minutes** de questions de l'enseignant et des pairs

3.2. Évaluation par les pairs :

Chaque étudiant devra également évaluer anonymement trois autres projets réalisés par ses camarades selon une grille d'évaluation fournie (qualité du code, clarté de l'API, respect des contraintes, pertinence de la modélisation…).
Les notes des pairs **viendront pondérer l'évaluation finale** du projet (dans la limite de 30%).
L'évaluation par les pairs est individuelle et obligatoire.

4. Dépôt GitHub et publication de l'image Docker

4.1. Dépôt du projet

Chaque groupe doit :

- Créer un dépôt GitHub public (ou privé).
- M’y inviter en tant que collaborateur (`@samiryoucef`) pour consultation et correction
- Organiser proprement le projet : structure des dossiers, nommage clair, commit messages explicites
  Le dépôt doit contenir au minimum :
- Le code source complet de l’application
- Un fichier `README.md` structuré (voir plus bas)
- Un `docker-compose.yml` fonctionnel
- Un fichier SQL ou script d’import pour un **jeu de données minimal de test**
- (Optionnel) Un volume Docker pour la persistance des données

4.2. Contenu attendu du `README.md`

Le `README.md` du dépôt doit expliquer de façon claire :

- Le contexte du projet (quel problème l’application résout)
- Comment **lancer l’application** avec Docker Compose
- L’**architecture du projet** (schéma ou description rapide)
- Exemples de **routes de l’API REST** (avec données d'exemple)
- (Facultatif) Données d'exemple / utilisateur de test

4.3. Publication de l’image Docker

Chaque groupe doit :

- Créer un compte sur Docker Hub
- Construire et **pousser une image Docker de leur API**
- Mentionner dans le README :
  - Le lien vers l’image sur Docker Hub
  - La commande pour la lancer en local
  - L’usage d’un volume (si utilisé)