BELAAROUS
Yassine

# Rapport individuel SAE 3.01c

## *Synthèse :*

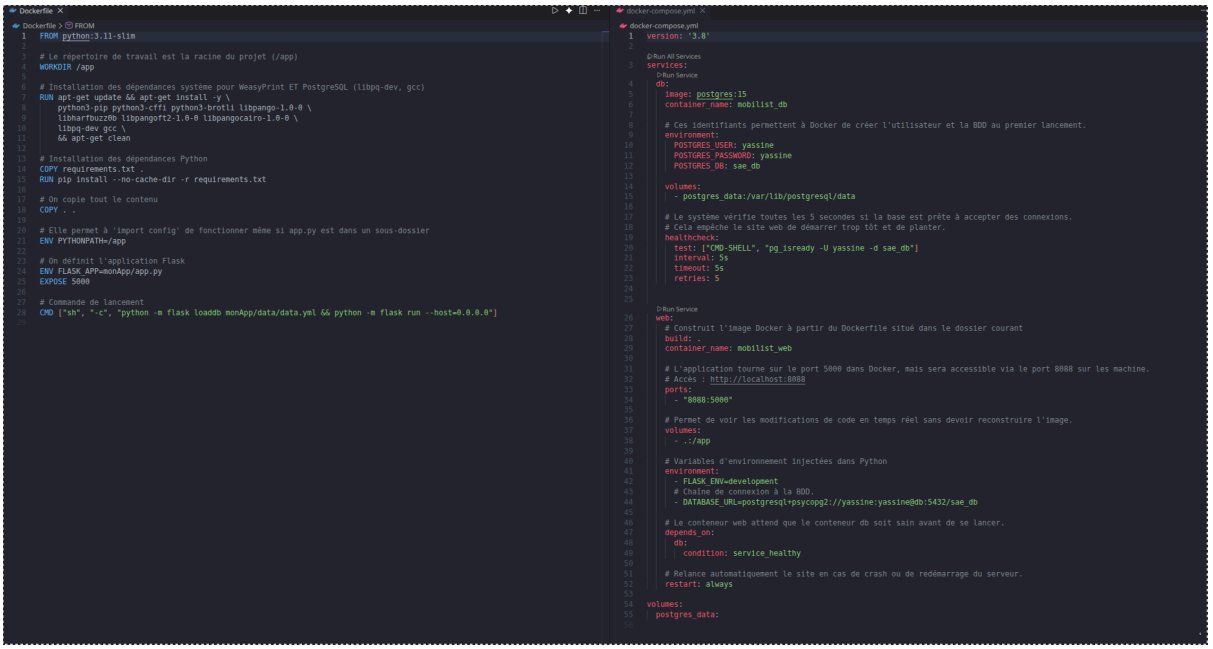
Le but de notre projet est de faire une application d'assurance permettant aux assurés d'ajouter des logements, des pièces et des biens, déclarer des sinistres…. et aux assureurs de gérer chaque assuré ainsi que de gérer les sinistres.

Durant cette semaine, nous avons continué l'implémentation de la partie assurée et commencé celle d'assureur.

## *Analyse :*

Personnellement, je me suis occupé de faire quelques modification demandé par le professeur tel que l'ajout d'une liste de sinistres pour un assuré, l'ajout de la visibilités des biens par sinistre dans la page detail_sinistre, ainsi que l'ajout du fait qu'un assureur puisse créer des comptes assureur et assuré ainsi que créer des comptes assurés.

De plus, j'ai mis en place le docker ainsi que la base de données en postgreSQL, pour faire ceci, j'ai utilisé comme ressource internet ainsi que le tp de virtualisation auquel on a accès.



```
Dockerfile
FROM python:3.11-slim

# Le répertoire de travail est la racine du projet (/app)
WORKDIR /app

# Installation des dépendances système pour WeasyPrint ET PostgreSQL (libpq-dev, gcc)
RUN apt-get update && apt-get install -y \
    python3-pip python3-cffi python3-brotli libpango-1.0-0 \
    libharfbuzz0b libpangoft2-1.0-0 libpangocairo-1.0-0 \
    libpq-dev gcc \
    && apt-get clean

# Installation des dépendances Python
COPY requirements.txt .
RUN pip install --no-cache-dir -r requirements.txt

# On copie tout le contenu
COPY . .

# Elle permet à 'import config' de fonctionner même si app.py est dans un sous-dossier
ENV PYTHONPATH=/app

# On définit l'application Flask
ENV FLASK_APP=monApp/app.py
EXPOSE 5000

# Commande de lancement
CMD ["sh", "-c", "python -m flask loaddb monApp/data/data.yml && python -m flask run --host=0.0.0.0"]
```

```
docker-compose.yml
version: '3.8'

Run All Services
services:
Run Service
  db:
    image: postgres:15
    container_name: mobilist_db

    # Ces identifiants permettent à Docker de créer l'utilisateur et la BDD au premier lancement.
    environment:
      POSTGRES_USER: yassine
      POSTGRES_PASSWORD: yassine
      POSTGRES_DB: sae_db

    volumes:
      - postgres_data:/var/lib/postgresql/data

    # Le système vérifie toutes les 5 secondes si la base est prête à accepter des connexions.
    # Cela empêche le site web de démarrer trop tôt et de planter.
    healthcheck:
      test: ["CMD-SHELL", "pg_isready -U yassine -d sae_db"]
      interval: 5s
      timeout: 5s
      retries: 5


Run Service
  web:
    # Construit l'image Docker à partir du Dockerfile situé dans le dossier courant
    build: .
    container_name: mobilist_web

    # L'application tourne sur le port 5000 dans Docker, mais sera accessible via le port 8088 sur les machine.
    # Accès : http://localhost:8088
    ports:
      - "8088:5000"

    # Permet de voir les modifications de code en temps réel sans devoir reconstruire l'image.
    volumes:
      - .:/app

    # Variables d'environnement injectées dans Python
    environment:
      - FLASK_ENV=development
      # Chaîne de connexion à la BDD.
      - DATABASE_URL=postgresql+psycopg2://yassine:yassine@db:5432/sae_db

    # Le conteneur web attend que le conteneur db soit sain avant de se lancer.
    depends_on:
      db:
        condition: service_healthy

    # Relance automatiquement le site en cas de crash ou de redémarrage du serveur.
    restart: always

volumes:
  postgres_data:
```

## *Démonstration de compétence:*

- **AC21.01 | Élaborer et implémenter les spécifications :** Intégration des retours du client (professeur) : liste des sinistres et création de comptes assureurs.

- **AC21.02 | Appliquer des principes d'ergonomie :** Amélioration de la vue détail sinistre pour visualiser directement les biens impactés.

- **AC21.03 | Bonnes pratiques de conception :** Architecture modulaire maintenue malgré la migration de la base de données.

- **AC22.01 | Structures de données complexes :** Passage à PostgreSQL pour gérer les relations robustes entre Utilisateurs, Sinistres et Biens.

- **AC23.01 | Applications communicantes :** Connexion du backend Flask à une base de données PostgreSQL externe au code.

- **AC23.02 | Serveurs et services virtualisés :** Mise en place d'un environnement Docker (conteneurs App et BDD) pour fiabiliser le déploiement.

- **AC24.03 | Restitution de données :** Agrégation des données (sinistre + biens associés) pour l'affichage tableau de bord.

- **AC24.04 | Données hétérogènes :** Manipulation et stockage de types variés (dates, statuts, textes) via l'ORM.

- **AC25.02 | Formaliser les besoins :** Traduction technique des demandes d'évolution fonctionnelle (gestion des comptes).

- **AC25.03 | Critères de faisabilité :** Validation et mise en œuvre technique de la stack Docker/PostgreSQL.

- **AC26.02 | Intégration dans une organisation :** Standardisation de l'environnement de dev via Docker pour faciliter le travail du groupe.

- **AC26.03 | Compétences interpersonnelles :** Coordination avec l'équipe pour l'implémentation parallèle des fonctionnalités Assureur/Assuré.