Nidhal ammar sahraoui

📄 Cahier des Charges - Projet React / Flask / PostgreSQL Template

🚀 1. Contexte du Projet

Ce projet a pour objectif de fournir un **template full-stack prêt à l'emploi** permettant de démarrer rapidement une application web avec :

- Un **frontend** React
- Un **backend** Flask (API REST)
- Une **base de données** PostgreSQL
- Une configuration prête pour **Docker**

🎯 2. Objectifs du Projet

- Offrir une architecture claire et modulaire pour séparer les responsabilités frontend/backend.
- Permettre le développement rapide de projets CRUD complets.
- Fournir une base robuste pour les déploiements en local et sur des serveurs de production via Docker.
- Implémenter un système d'authentification simple (JWT).
- Assurer la communication sécurisée entre le frontend et le backend.

📦 3. Portée Fonctionnelle

✅ Fonctionnalités Frontend (React)

- Structure de projet React avec:
    - Routing via react-router-dom
    - Authentification utilisateur (connexion, déconnexion)
    - Gestion d'état via useContext ou un équivalent
    - Appels API sécurisés (token JWT)
- Interface utilisateur responsive
- Exemple de composant pour CRUD (ex: gestion des utilisateurs)

✅ Fonctionnalités Backend (Flask)

- API REST structurée avec :
    - Endpoints CRUD (ex: /api/users, /api/items)
    - Authentification via JWT (login, register)
    - Middleware de vérification de token

- ORM via SQLAlchemy
- Migrations de base de données (via Flask-Migrate)

✅ **Fonctionnalités Base de Données (PostgreSQL)**

- Connexion sécurisée à PostgreSQL
- Structure de tables pour :
    - Utilisateurs
    - Données génériques (ex: Items)
- Script de création et initialisation

✅ **Déploiement / Environnement**

- Environnement de développement avec Docker (multi-conteneur avec docker-compose)
- Scripts de build, migration, démarrage des services
- Fichiers .env pour configuration

🛠️ **4. Technologies Utilisées**

Composant	**Technologie**
Frontend	React, Vite
Backend	Python, Flask, Flask-JWT
Base de données	PostgreSQL
ORM	SQLAlchemy
DevOps	Docker, Docker Compose
Migrations	Flask-Migrate

🔐 **5. Sécurité**

- Authentification basée sur JWT
- Séparation claire des rôles frontend/backend
- Accès sécurisé à la base via variables d'environnement

📚 **6. Structure du Projet**

bash

CopierModifier

```
project-root/
|
├── client/           # Frontend React app
|   ├── public/
|   ├── src/
|   └── vite.config.js
|
├── server/           # Backend Flask API
|   ├── app/
|   ├── migrations/
|   ├── config.py
|   └── manage.py
|
├── docker-compose.yml   # Multi-service orchestration
├── .env              # Variables d'environnement
└── README.md
```

📈 7. Évolutions Possibles

- Intégration de gestion des rôles (admin, utilisateur)
- Authentification OAuth (Google, GitHub)
- Tests unitaires (Pytest, React Testing Library)
- CI/CD (GitHub Actions, DockerHub)

🗓️ 8. Planning Estimatif (optionnel)

Phase	Durée Estimée
Étude et personnalisation	2 jours
Configuration de l'environnement	1 jour
Développement des modèles	2 jours
Authentification JWT	1 jour

Phase	Durée Estimée
Intégration frontend/backend	2 jours
Dockerisation	1 jour
Tests & documentation	2 jours

👥 9. Équipe & Rôles

Rôle	Personne responsable
Développeur Frontend	...
Développeur Backend	...
DevOps / Déploiement	...

📌 10. Livrables Attendus

- Code source complet sur GitHub
- Documentation d’installation et d’utilisation
- Conteneurs Docker fonctionnels
- API documentée (ex: via Swagger