**1. Planification**

* **Choix du thème** : Définis une application pratique comme la gestion d'utilisateurs ou un mini e-commerce.
* **Spécifications** :
  + Fonctionnalités minimales : POST, GET, DELETE.
  + Interaction frontend-backend.

**2. Mise en place du backend**

* **Technologie** : Choisis **Flask** ou **FastAPI** si tu préfères Python.
* **Étapes** :
  1. Crée les endpoints requis.
  2. Implémente la connexion à la base de données (MySQL, PostgreSQL ou MongoDB).
  3. Structure les fichiers (app.py, modèles, contrôleurs).
* **Dockerisation** :
  1. Écris un Dockerfile pour conteneuriser l'API (inclure les dépendances depuis requirements.txt).

**3. Développement de la base de données**

* Choisis le type de base (relationnelle ou NoSQL).
* Crée un Dockerfile spécifique.
* Gère les migrations pour MySQL/PostgreSQL ou les schémas pour MongoDB.

**4. Création du frontend**

* **Technologie** : Utilise HTML/CSS/JS ou un framework (React/Vue.js).
* **Fonctionnalités** :
  1. Un formulaire pour ajouter des données.
  2. Une liste pour afficher les données.
* **Dockerisation** :
  1. Écris un Dockerfile pour conteneuriser l'interface.

**5. Orchestration avec Docker Compose**

* **Fichier docker-compose.yml** :
  1. Définis les services (backend, frontend, database).
  2. Configure les volumes et les réseaux pour la communication.

**6. Tests**

* **Backend** : Utilise Postman ou curl pour tester les endpoints.
* **Frontend** : Vérifie que l'application consomme correctement l'API.
* **Global** : Teste l'interaction entre tous les services avec Docker Compose.

**7. Documentation**

* Inclure dans le fichier README.md :
  + Prérequis (Docker, Docker Compose).
  + Instructions pour cloner et exécuter le projet.
  + Liste des endpoints et leur description.
  + Fonctionnalités frontend.
  + Captures d’écran.

**8. Suivi et gestion**

* **Git** : Crée un dépôt et pousse tes changements régulièrement.
* **Variables d'environnement** : Place les données sensibles dans un fichier .env.