NoSQL - Projet final - Backend multi-bases

Objectif du projet

L'objectif de ce projet est de concevoir et implémenter un **mini-système backend** qui exploite à la fois **une base relationnelle** et **au moins deux bases NoSQL** pour couvrir des besoins spécifiques que la base relationnelle ne gère pas efficacement.

Il s'agit de démontrer votre capacité à :

- Modéliser des données relationnelles classiques.
- Identifier les cas où une base NoSQL est plus adaptée et l'utiliser à bon escient.
- Concevoir une API simple pour interagir avec les différentes sources de données.

Conditions à respecter

1. **Bases de données utilisées**
    - **1 base relationnelle obligatoire** (PostgreSQL, MariaDB ou MySQL)

      → Stockage des données structurées principales (ex. utilisateurs, contenus, produits, tâches...).
    - **2 bases NoSQL au minimum** parmi :
        - **Redis** : caching, sessions, rate limiting...
        - **MongoDB** : données semi-structurées ou variables
        - **ChromaDB** : recherche vectorielle / recommandations par similarité
        - **Neo4j** : graphes pour relations / recommandations

2. **API minimale à fournir**
    - CRUD pour la ressource principale (`GET /items`, `GET /items/:id`, `POST /items`)
    - Au moins une route démontrant l'usage de chaque base NoSQL (cache, recherche, graphe, recommandation...)
3. **Architecture et justification**
    - Schéma SQL clair
    - Modèles NoSQL adaptés au cas d'usage
    - Explication courte du rôle de chaque base

Livrables attendus via un dépôt GitHub

1. **Code backend fonctionnel**
    - Langage libre (Node.js, Go, Python...)
    - Minimaliste : quelques routes suffisent
2. **Docker Compose** ou équivalent pour lancer toutes les bases
3. **Scripts de seed** pour remplir les bases avec des données exemples
4. **Document d'architecture** (1–2 pages)
    - Schéma relationnel
    - Modèles NoSQL
    - Justification des choix
5. **Démo rapide** : montrer que les endpoints fonctionnent et que les NoSQL sont exploitées correctement

---

Trame de soutenance (30 min)

1. **Introduction**

- Présentation du projet : objectifs et fonctionnalités clés
- Contexte simulé d'utilisation (types d'utilisateurs, besoins)
- Vue d'ensemble de l'architecture technique :
    - Langages / frameworks utilisés

- Bases de données SQL / NoSQL choisies
- Raisons générales du choix multi-NoSQL

---

2. **Présentation des bases utilisées**

Pour chaque base utilisée :

- Nom de la base (ex: MongoDB, Redis, Firebase, etc.)
- Type de base (document, clé-valeur, graphe, etc.)
- Cas d'usage couvert dans le projet
- Avantages spécifiques dans ce contexte

---

3. **Modélisation des données**

3.1. Vue d'ensemble

- Diagramme d'architecture de la gestion des données (schémas, flux entre bases)
- Stratégie de découplage ou de synchronisation entre les bases si nécessaire

3.2. Par base :

- Schéma des collections/documents/structures principales
- Échantillons de documents (JSON)
- Justification des choix de structure et des relations (imbriquées, référencées...)

---

4. **Fonctionnalités métier et utilisation des bases**

Pour chaque fonctionnalité importante de l'outil (ex. création de transaction, vue du budget, analyse des dépenses) :

- Description du besoin
- Quelles bases sont mobilisées et comment ?
- Exemple(s) de requêtes (NoSQL ou commandes spécifiques)
- Raisonnement technique derrière l'utilisation de chaque base

---

5. **Synchronisation et cohérence**

- Gestion de la cohérence entre les différentes bases (si nécessaire)

- Mécanismes de mise à jour croisée ou d’invalidation de cache
- Outils ou patterns utilisés

---

6. **Problèmes rencontrés et solutions**

- Difficultés techniques spécifiques à l’usage combiné de plusieurs bases
- Problèmes de performances, latence, incohérence potentielle
- Solutions mises en place, arbitrages techniques

---

7. **Perspectives et scalabilité**

- Évolutivité de l’architecture multi-NoSQL
- Comment cette architecture pourrait répondre à une montée en charge
- Quels services ou bases supplémentaires pourraient être intégrés ?

---

8. **Conclusion**

- Retour sur les apports du projet en termes d’apprentissage NoSQL
- Pertinence d’une architecture multi-NoSQL dans le monde réel
- Limites du projet et améliorations futures