Documentation de Conception de la Base de Données

Projet Marketplace Microservices – Architecture SQL & MongoDB

1. Contexte du projet

L’application est une marketplace e-commerce basée sur une architecture microservices. Chaque domaine fonctionnel possède son propre service et sa propre base de données afin d’assurer scalabilité, isolation et performance.

2. Architecture globale

Service	Port	Type de base	Rôle
Users	3001	SQL (MySQL)	Authentification / comptes
Orders	3003	SQL (MySQL)	Commandes ACID
Payments	3005	SQL (MySQL)	Paiements sécurisés
Addresses	3006	SQL (MySQL)	Adresses clients
Catalog	3002	MongoDB	Produits & catégories
Cart	3004	MongoDB	Panier éphémère

3. Conception SQL

3.1 Rôle du SQL

SQL est utilisé pour les données critiques nécessitant des transactions ACID, des contraintes d’intégrité et des relations fortes.

3.2 MCD SQL (relations)

```
ROLE 1 ----- N USER
USER 1 ----- N ADDRESS
USER 1 ----- N ORDER
ORDER 1 ----- N ORDER_ITEM
ORDER 1 ----- 1 PAYMENT
```

3.3 Tables SQL principales

roles, users, addresses, orders, order_items, payments

3.4 Justification technique

Les commandes et paiements impliquent des montants financiers. Une perte de cohérence serait critique. SQL garantit atomicité, cohérence et intégrité référentielle grâce aux clés étrangères et transactions.

4. Conception MongoDB

4.1 Rôle de MongoDB

MongoDB est utilisé pour les données volumineuses, fortement lues et flexibles : catalogue et panier.

4.2 Collections

products, categories, carts

4.3 Structure produit (exemple)

```
{
  _id: "uuid",
  name: "iPhone 15",
  price: 1200,
  stock: 25,
  attributes: { ram: "8GB" }
}
```

4.4 Pourquoi le panier n’est pas en SQL ?

Le panier est temporaire, très souvent modifié et non critique. MongoDB permet un document unique par utilisateur, des mises à jour rapides et une suppression automatique via TTL, ce qui serait coûteux en SQL.

5. Comparaison SQL vs MongoDB

Critère	SQL	MongoDB
Transactions financières	Oui	Non
Relations fortes	Oui	Non
Recherche texte	Limité	Très performant
Scalabilité catalogue	Moyenne	Élevée
Panier éphémère	Non adapté	Idéal

6. Conclusion

L’architecture hybride SQL/MongoDB permet d’utiliser la bonne technologie pour le bon besoin. SQL garantit l’intégrité des transactions critiques tandis que MongoDB assure performance et scalabilité du catalogue et du panier. Cette approche respecte totalement le contrat d’API et les bonnes pratiques modernes des microservices.