Conception d’un système de gestion de boutique en ligne

- Les caractéristiques architecturales identifiées
- Les décisions architecturales prises et leur justification
- Le schéma des composants logiques
- Le style architectural choisi et les raisons de ce choix

1. Référencement des caractéristiques architecturales : (7 maximum)

- Agilité (nombre de produits qui va évoluer de 5000 à plusieurs millions, gestions de produit, masse d’utilisateurs)
    - Volumétrie actuelle
    - Objectif de croissance

- Modularité (mise à jour régulières des produits et évolution des paramètres)
    - Personnalisation
    - Mise à jour fréquente des produits

- Performance (liée à la réactivité, temps de réponse rapide, gestion du panier)
    - Réactivité

- Sécurité (Gestion de comptes utilisateurs et sécurité des transactions)
    - Gestion des comptes utilisateurs
    - Paiements

- Disponibilité (Boutique accessible H24)
    - Objectif de disponibilité

- Extensibilité (Stock numérique qui va être liée aux stocks magasin physique)
    - Perspectives d’évolution

- Robustesse (pic d’activités)

- Trafic estimé
- 2. Création de décisions d'architecture

<u>5 décisions architecturales</u>

- Quel type de base de données : Relationnelle ET NoSQL
  - Adopter une approche hybride permet de tirer le meilleur parti des forces des deux types de bases de données en fonction des besoins spécifiques de la cible.

  - NoSQL : gestion des produits
    - + Rapidité, scalabilité

  - SQL : Gestion de la transaction et de l'utilisateur et stock
    - + Sécurisation des données utilisateur et transaction
    - - Latences

- Choix des technologies de front-end et de back-end
  - Front : TypeScript - React.js beaucoup de ressources disponible et facilité de mise en œuvre

  - Back - gestion des produits : Java Spring, connu pour sa robustesse et peut gérer les requêtes intensives de façon performante.

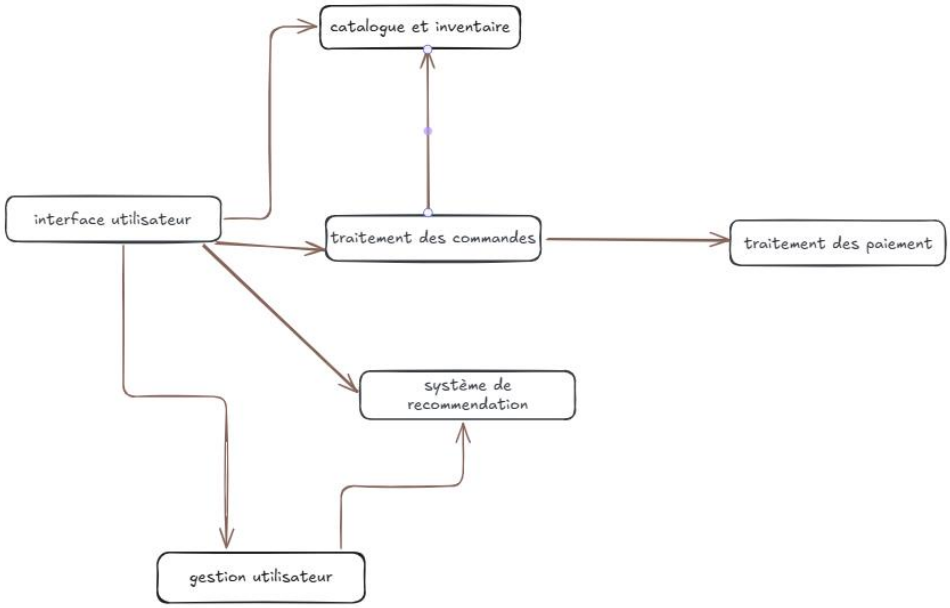
  - Back - Gestion du panier et de l'utilisateur : FrameWork tel que TypeScript- Nest.js (similaire à java dans sa logique d'utilisation)

- Choix de l'architecture logicielle
  - Microservices : Besoin scalabilité, fléxibilité et résilience

- Stratégie d'infrastructure :
  - Infrastructure interne : Pour la gestion des utilisateurs et des stocks et des commandes

- Gestion de l'authentification :
  - JWT (facilement propagable entre microservices)
  -

3. Schéma des composants logiques



4. Choix du style architectural et justification

Choix du style architectural : microservices

Pourquoi : Scalabilité, disponibilité

Avantages et inconvénients :

Avantages :

- **Scalabilité** : Les microservices permettent une scalabilité individuelle, optimisant l'utilisation des ressources.
- **Déploiement indépendant** : Chaque service peut être déployé et mis à jour sans impacter l'ensemble.
- **Flexibilité technologique** : Possibilité d'adopter différents langages ou frameworks pour chaque service (ex. Java Spring pour la gestion des produits, TypeScript/Nest.js pour le panier).

- **Inconvénients** :
- **Complexité de gestion** : La gestion des communications entre services (REST ou message broker) peut augmenter la complexité.
- **Surveillance et Sécurité** : Nécessite des outils spécifiques pour la surveillance (monitoring) et la sécurisation de chaque service (authentification et autorisation).
- **Coûts en infrastructure** : Peut entraîner des coûts d'infrastructure plus élevés en raison des ressources nécessaires pour orchestrer les services et pour le monitoring.