Présentation du projet architecture logiciel

Date	16 décembre 2024
Participants	Nina Guiguet Catherine Jules Angie Pons Julien Vaglia Ludovic Andreotti

Conception d’un système de gestion de boutique en ligne

Sommaire

1. Référencement des caractéristiques architecturales

- **<u>Agilité :</u> La boutique doit évoluer de 5 000 à plusieurs millions de produits, en gérant un grand volume d'utilisateurs**.
  - *Volumétrie actuelle : 5 000 produits.*
  - *Objectif : Gestion future de plusieurs millions de produits en 2 ans.*

- **<u>Modularité :</u> Les produits seront fréquemment mis à jour, avec des modifications régulières de paramètres, comme les prix**.
  - Mise à jour : Changements fréquents (prix, descriptions, disponibilités).

- **<u>Performance :</u> Réactivité essentielle pour garantir une expérience fluide, notamment lors de l'ajout de produits au panier**.
  - Réactivité
  - Fluidité

- **<u>Sécurité :</u> Gestion des comptes utilisateurs et sécurité des transactions sont des priorités absolues**
  - Gestion des comptes utilisateurs
  - Paiements

- **<u>Disponibilité :</u> Boutique doit être accessible 24/7, avec une tolérance minimale d'indisponibilité.**
  - Objectif de disponibilité

- **<u>Extensibilité :</u> Synchronisation prévue avec des stocks physiques à l'avenir.**
  - Perspectives d'évolution

- **<u>Robustesse :</u> Capacité à absorber les pics de trafic, notamment lors des soldes.**
  - Trafic estimé

2. Création de décisions d’architecture

1 - Quel type de base de données :

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 :** Pour la gestion des produits et utilisateur, car elle garantit scalabilité et rapidité.
  - + Rapidité, scalabilité
  - - Moins de sécurité intégrée, Moins adapté pour les transactions complexes

- **SQL :** Pour les transactions et le panier, avec un focus sur la sécurité
  - + Sécurisation des données utilisateur et transaction
  - - Latences et performance

2 - Choix des technologies :

- **Frontend :**
  - React.js, choisi pour sa communauté active et sa facilité de mise en œuvre.
- **Backend :**
  - Java Spring pour gérer les produits en raison de sa robustesse.
  - Nest.js pour la gestion des paniers et des utilisateurs, aligné sur la logique TypeScript.

3 - Choix de l’architecture logicielle

- Micro-services :
  - Permet un déploiement indépendant des services pour limiter les impacts des mises à jour

4 - Stratégie d’infrastructure :

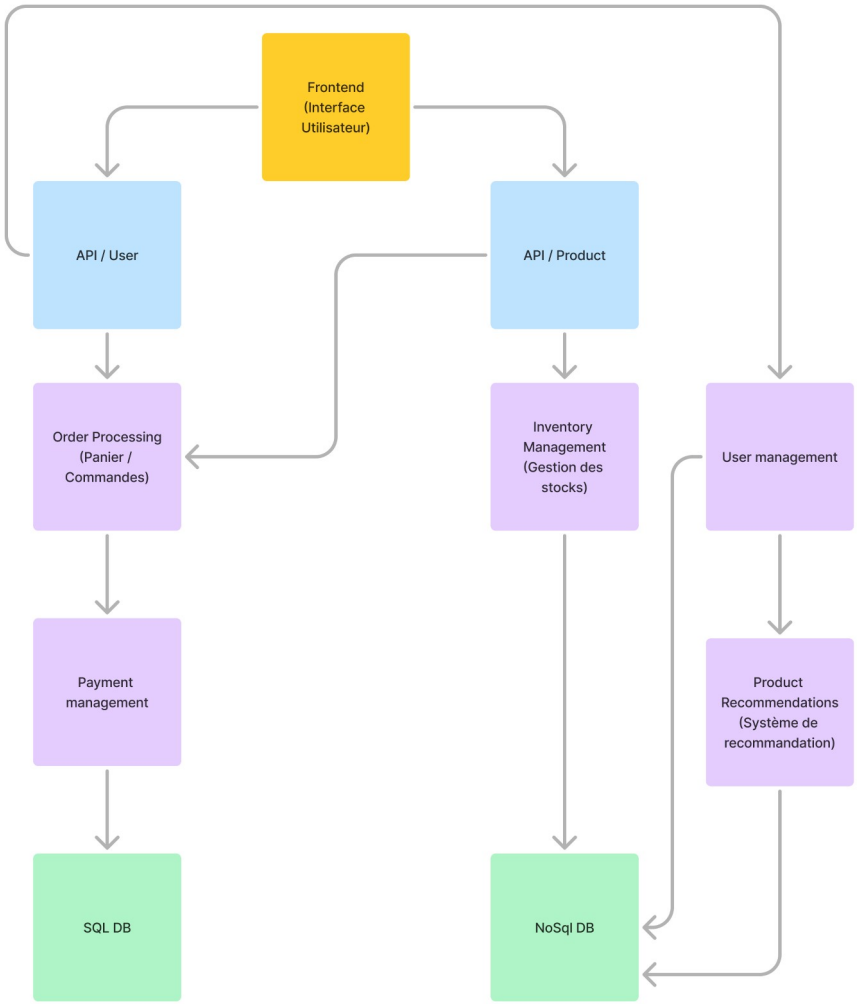
- Infrastructure interne :

- Permet à l'entreprise de gérer ses propres serveurs et données, assurant sécurité et contrôle total. Idéale pour les applications avec des données sensibles.

5 - Gestion de l'authentification :

- JWT :
  - Permet de faciliter l'authentification et les autorisations entre nos différents microservices, assurant ainsi une communication fluide et sécurisée.

3. Schéma des composants logiques



4. Choix du style architectural et justification

Choix du style architectural : Microservices

Pourquoi : Scalabilité, disponibilité

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.