Labo 5 – Passage à une Architecture Microservices avec API Gateway et Observabilité

Cours : **Architecture Logicielle (LOG430)**

Session : **Été 2025**

Date du laboratoire : **Semaine du 16 Juin 2025**

Le génie pour l'industrie

Contexte et objectifs du laboratoire

Ce laboratoire est une **extension directe du Labo 3**, avec pour objectif de faire évoluer votre système multi-magasins vers une **architecture orientée microservices**, adaptée à un **contexte e-commerce**.

L'idée n'est pas d'ajouter une multitude de nouvelles fonctionnalités, mais de **réorganiser les services existants**, en y ajoutant **quelques services propres au commerce en ligne**.

Objectifs pédagogiques

- Comprendre les fondements de l'architecture basée sur services (SBA) et microservices.
- Découper un système monolithique en services plus petits (sans réécrire toute la logique). Cloud Gateway...) ;
- la distinction entre les composantes du magasin physique (par exemple gestion des rayons, caisses, etc.) et celles d'un site e-commerce (compte client, panier, commande) ;
- Mettre en place une API Gateway (comme Kong, Spring Cloud Gateway, APISIX ou KrakenD).
- Configurer des routes vers les services internes.
- Protéger et documenter les points d'entrée via la Gateway.

Travail demandé

1. Découpage logique du système

- Identifiez 3 à 4 services potentiels dans votre système existant (par exemple : gestion des produits, gestion des ventes, reporting, gestion du stock).
- Séparez-les logiquement dans votre projet (packages distincts, modules, services Docker isolés).
- Chaque service doit être déployé indépendamment dans son propre conteneur.
- **Ajouter au moins 3 autres API :**
  - Une API **pour la création de comptes clients** ;
  - Une API **de gestion du panier d'achat** ;
  - Une API **de validation de commande (check-out)** ;

  Ces services doivent coexister avec les services du système magasin (ex : gestion de stock), certains pouvant être partagés. Par exemple, le stock est un service partagé entre le magasin physique et la boutique en ligne.

2. Mise en place de l'API Gateway

— Choisissez une API Gateway open-source (exemples : Kong, KrakenD, Spring Cloud Gateway...).

— Configurez la Gateway pour qu'elle gère l'accès aux différents services à partir d'un point d'entrée unique.

— Implémentez au moins deux fonctionnalités suivantes :

  — Routage dynamique

  — Ajout de clés API ou d'en-têtes

  — Logging centralisé ou authentification simplifiée

3. Configurer un scénario de load balancing via l'API Gateway

:

— Mettre en place une répartition de charge (round-robin ou autre) entre au moins deux instances d'un microservice (par exemple, service de gestion du panier ou du stock).

— Décrire ce comportement dans la documentation technique.

— Tester la distribution de charge à l'aide d'un outil de test (comme k6, JMeter ou curl en boucle).

— Bonus : illustrer la répartition via des métriques d'observabilité (Grafana, Prometheus ou équivalent).

4. Sécurité et gestion des accès

— Configurez des règles de CORS dans la Gateway.

— Facultatif : journaux d'accès, throttling ou quota si supporté par votre Gateway.

5. Documentation et tests

— Fournissez un fichier Swagger/OpenAPI mis à jour si l'interface a évolué.

— Mettez à jour vos requêtes Postman

— Vérifiez que l'exposition via la Gateway fonctionne correctement.

5. Observabilité et comparaison

— Réutilisez les outils d'observabilité du Labo 4 (Prometheus, Grafana).

— Comparez les résultats obtenus avec ceux de l'architecture précédente :

  — Latence

  — Disponibilité des services

  — Traçabilité des appels

— Ajoutez un tableau comparatif dans votre rapport.

— **Comparez les architectures** :
  — Réalisez un test de charge sur 2 scénarios :
    — Appels directs a l'API (ancienne architecture)
    — Appels via API Gateway (nouvelle architecture)
  — Analysez et commentez les différences observées :
    — Taux d'erreur, temps de réponse.
    — Visibilité améliorée ou complexifiée (grâce à la Gateway ou à la décomposition)
  — **Présentez vos résultats dans un dashboard Grafana** et intégrez des captures d'écran dans votre livrable.

Livrables attendus

Votre remise devra contenir les éléments suivants :

— **Code source complet et fonctionnel**, dans un dépôt GitHub ou GitLab. dossier.
— Un rapport structué suivant le format Arc42. Ce rapport présentera votre propre proposition d'architecture en visant la solution la plus simple possible répondant aux besoins exprimés. Il devra inclure la justification de vos d´ecisions a travers au moins deux Architectural Decision Records (ADR)
— **Un fichier** `README.md` clair et structuré contenant :
  — Instructions de déploiement
  — Captures d'écran ou liens vers les dashboards d'observabilité (Grafana, etc.).

Conseils

— Ce laboratoire est l'occasion de mieux comprendre les principes de découplage.
— Ne surchargez pas le projet avec des fonctionnalités supplémentaires. Restez simple et fonctionnel.
— Favorisez la clarté dans la documentation et la cohérence des noms de