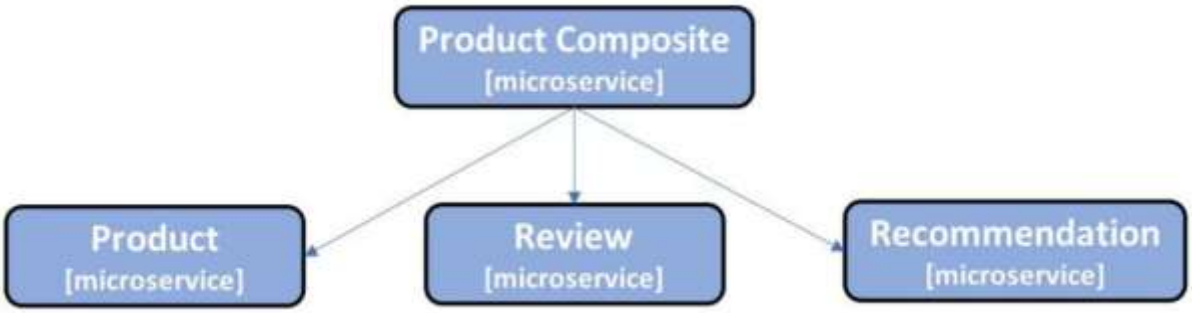
TP : Mise en place d'une application suivant l'architecture microservice

(Durée : 4h)

Le système des microservices se compose de :



Il se compose de trois microservices principaux, les services **Product**, **Review** et **Recommendation**, qui traitent tous un type de ressource, et d'un microservice composite appelé service **Product Composite**, qui regroupe les informations des trois services principaux.

Le service Produit gère les informations du produit et le décrit avec les attributs suivants :

- Product ID
- Name
- Weight

NB : La valeur de **weight** ne peut être négative et ne doit pas dépasser 100kg. Utiliser le mécanisme de validation pour cette entrée.

Le service Review gère les avis sur les produits et stocke les informations suivantes sur chaque avis :

- Product ID
- Review ID
- Author
- Subject
- Content

Le service Recommendation gère les recommandations de produits et stocke les informations suivantes sur chaque recommandation :

- Product ID
- Recommendation ID
- Author
- Rate
- Content

NB : La champ Rate est un pourcentage. Valider cette entrée aussi.

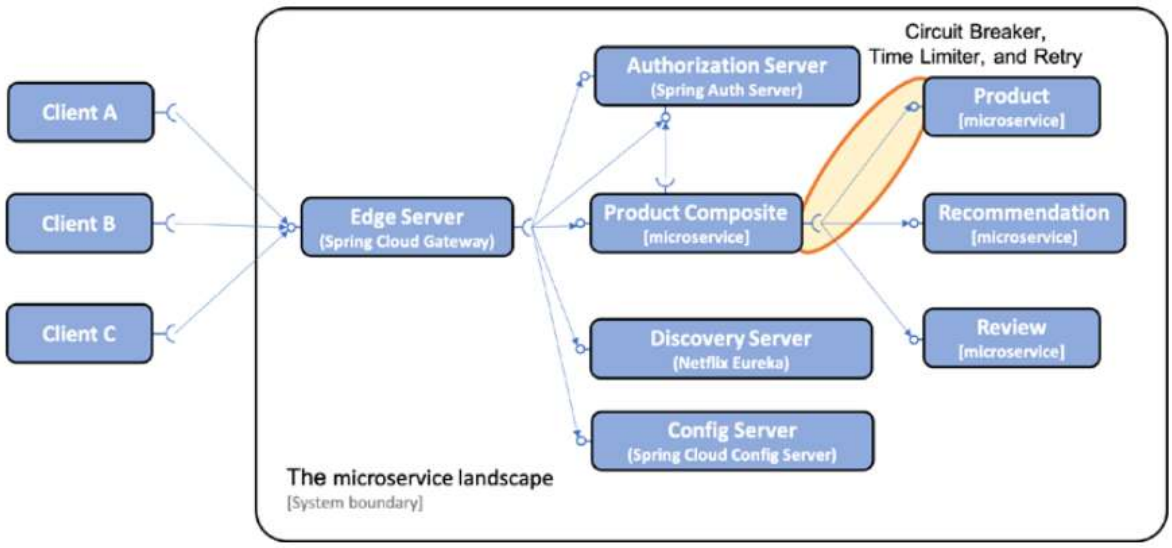
Le service **Product Composite** regroupe les informations des trois services principaux et présente les informations sur un produit comme suit :

- Informations sur le produit, telles que décrites dans le service produit.
- Une liste des avis sur le produit spécifié, comme décrit dans le service des avis.
- Une liste de recommandations pour le produit spécifié, comme décrit dans le service de recommandation.

On demande de réaliser une application qui suit le diagramme suivant :

Ici : Edge Server désigne votre passerelle

N'oublier pas d'intégrer **Zipkin** pour la traçabilité des requêtes et d'utiliser comme base de données pour l'environnement de développement (par défaut), la base de données **H2**.



1. Pour implémenter la sécurité, nous allons opter pour un microservice **autorisation-service** qui centralise les informations de sécurité. Par exemple nous allons les stocker dans ce service dans son fichier de configuration application.properties :
   Exemple :
   security.productcomposite. **admin**.username =
   security.productcomposite. **admin**.password =
   security.productcomposite. **user**.username
   security.productcomposite. **user**.password=
   Ici **admin** fait référence au rôle **ADMIN** qui a un accès en **lecture/écriture** et **user** fait référence au rôle **USER** qui a un accès limité en **lecture** seule.
   La passerelle devra intercepter toutes sortes de requêtes vers le service **product-composite-service** et la redirigé en premier au service **autorisation-service** qui vérifie les informations d'accès et d'autorisation (écrites dans son fichier de configuration **application**.**properties**) avant de rediriger la requête vers la passerelle qui à son tour redirige la requête vers la destination
   ➔ **Indication** : utiliser les filtres au niveau de la passerelle et un client openFeign au niveau du service **autorisation-service.**

L'utilisateur pour se connecter devra saisir l'adresse de l'API sollicité ainsi que trois champs dans l'entête http : le champ **username** ; le champ **password** et le champ **rôle**.

2. Créer plusieurs instances du service composite.
3. Mettez en place un disjoncteur au niveau du service composite et tester la résilience de ce service avec une méthode **fallback** appropriée.
4. Mettez en place un **Load Balancer** au niveau du service composite et créer deux instances de chaque microservice appelé via ce service (product ; recommendation ; review). Tester la distribution du trafic.
5. Centraliser toutes les configuration via le serveur de configuration de Spring-Cloud.
6. Créer un profil de production qui utilise un système de gestion de base de données **Mysql** et des ports différents pour l'ensemble des microservices mis en place. Activer ce profil et observer la santé de votre application.
7. Création de métrique :
    a. Créer une métrique qui collecte le nombre de requêtes de type POST et PUT qui ont été reçues par le service composite.
    b. Créer une autre métrique qui collecte le nombre de requêtes de type GET qui ont été reçues par le service composite.
    c. Observer ces deux métriques via spring actuator.
8. Créer un client REST (un projet SpringBoot utilisant RestTemplate) qui consomme votre API.