# Plateforme d'e-Commerce

*BINV3140 – Spring : Projet*

## Contexte

Vous êtes assignés à la création d'une application web de vente en ligne. Afin de rendre la plateforme la plus scalable possible, on vous demande d'utiliser l'architecture micro-service. Vous êtes libres de choisir le thème et le nom de la boîte fictive pour laquelle vous développez, ainsi que les produits vendus (attention, pas (uniquement) des Legos !)

Vous allez donc utiliser Spring pour développer l'application dans son ensemble, en utilisant les techniques et concepts vus aux cours.

## Organisation

1. **Pour le dimanche 17/10 : remise de la répartition des tâches**
   Choisissez quel membre du groupe sera responsable de quels micro-services de base. Présentez brièvement le thème choisi et le genre de produits vendus. Dans le même document, fournissez les spécifications des services REST, sous forme de tableau présentant le résultat pour chaque paire URI + méthode disponible. Cette soumission n'est pas évaluée, mais nous permettra de vous aiguillez dans la bonne direction si vous faites fausse route.
2. **Pour le vendredi 26/11 : remise du projet et du rapport**
   Remettez vos projet IntelliJ complets sous forme d'archive et votre rapport sur Moodle.
3. **Pour le samedi 4/12 : remise de l'extension de projet**
   Petite extension du projet à faire en dernière semaine, les consignes seront fournies à ce moment-là.

La répartition des points se fera comme suit :
- 8/20 pour les micro-services de base ;
- 6/20 pour les edge micro-services ;
- 4/20 pour les tests ;
- 2/20 pour le rapport.

**Attention :** ne pas rendre une des quatre parties ou la bâcler complètement bloquera la note de votre projet à 10 au maximum : ne négligez donc pas les parties qui valent peu de points.

## I. Micro-services de base

Vous aurez 8 micro-services à implémenter pour le projet, 2 par membre du groupe. Chaque membre doit être responsable d'un service REST et d'un front-end. Les étoiles (*) fournissent une aide pour la répartition des tâches : les services représentant des charges de travail différentes, il est *recommandé* que chaque membre du groupe choisisse ses services de manière à avoir 4 étoiles.

1. Service REST des données des produits (***)
2. Service REST des données des utilisateurs (*)
3. Service REST des paniers des utilisateurs (**)
4. Service REST des commentaires des clients (***)
5. Un front-end qui affiche la liste des produits (*)
6. Un front-end qui affiche les détails d'un produit (***)
7. Un front-end qui affiche le panier d'un utilisateur (**)
8. Un front-end de connexion/inscription d'utilisateur (*)

Les services REST doivent respecter les conventions REST et être organisés selon ce qui a été vu en cours. Vous devez utiliser H2 comme moteur de base de données.

Les front-ends doivent être mis en page correctement. Toutefois, ne perdez pas trop de temps avec la beauté du projet : cherchez plutôt quelque chose lisible. Vous pouvez bien entendu utiliser le squelette html du cours.

### *1. Données de produits*

Ce service contient une base de données contenant les produits du site. Un produit possède un **id**, un **nom**, une **description courte**, une **description détaillée**, un **prix** et une **catégorie**. Il doit être possible, via ce service, d'effectuer les opérations CRUD (Create, Read, Delete, Update) de base.

Il doit être possible de récupérer les produits triés :

- Par prix croissant
- Par prix décroissant

Il doit être possible de filtrer les produits :

- D'une catégorie donnée
- D'un prix supérieur ou égal et inférieur ou égal à un prix donné (i.e. $min \leq prix \leq max$)

Enfin, il doit être possible de récupérer tous les produits ayants les ids données, sans tri ni filtre.

**Attention :** les filtres et les tris doivent pouvoirs être combinés (donc il faudra, dans la couche repository, une version de chaque tri pour chaque filtre et une version sans filtre).

**Utilisation de GitLab :** nous vous demandons de collaborer grâce à GitLab. Vous devez créer un projet sur GitLab pour le groupe, sur lequel se trouveront tous les projets IntelliJ (un pour chaque service). Invitez-nous comme maintainers dans ce projet en choisissant comme expiration date le 21 février. Nous utiliserons l'historique GitLab en cas de doute sur la participation d'un membre du groupe : il est donc dans votre intérêt d'utiliser GitLab correctement et régulièrement.

### *2. Données des utilisateurs*

Ce service contient une base de données contenant les utilisateurs du site. Un utilisateur contient un **id**, un **nom**, un **prénom**, un **pseudo**, une **date de naissance**, une **adresse**, une **adresse mail unique** et un **mot de passe**, et est soit un **client**, soit un **administrateur**. Il doit être possible, via ce service, d'effectuer les opérations CRUD (Create, Read, Delete, Update) de base.

Il faut, de plus, pouvoir récupérer un utilisateur par son adresse mail.

### *3. Paniers des utilisateurs*

Ce service contient une base de données contenant les paniers des utilisateurs. Les paniers sont en fait une table qui comprend un **id**, un **id d'utilisateur** un **id de produit**, une **quantité**. Il doit être possible d'insérer, supprimer et lire des données dans cette table. Il doit être également possible de récupérer tous les éléments d'une id d'utilisateur donnée.

La modification d'une entrée de la table ne permet que de modifier la quantité.

### *4. Commentaires des utilisateurs*

Ce service contient une base de données contenant les paniers des utilisateurs. Un commentaire possède un **id**, un **texte**, une **évaluation** sur 5, une **date de création**, un **id d'utilisateur**, un **id de produit** et est soit **supprimé**, soit **valide**. Il doit être possible, via ce service, d'effectuer les opérations CRUD (Create, Read, Delete, Update) de base.

En ce qui concerne la lecture des commentaires, elle doit toujours être ordonnée par date de création.

Il doit être possible récupérer uniquement les commentaires d'un id utilisateur et id produit donnés, ainsi que tous les commentaires d'un id produit donné sauf ceux de l'id utilisateur donné.

Il doit être possible d'obtenir le nombre de commentaire pour un id produit donné, ainsi que la moyenne des évaluations pour un id produit donné.

Seuls les champs texte et la validité (supprimé/valide) peuvent être mis à jour.

**NOTES pour les services REST :**
**a) Vos services devront être sécurisés de manière à ce que seuls les services de l'application puissent les contacter pour récupérer des données.**
**b) Les paniers des utilisateurs et les données des utilisateurs ne pourront être accessible/modifiées que par l'utilisateur concerné, connecté.**
**c) La suppression de commentaire ne peut se faire que par un le propriétaire du commentaire ou par un administrateur. De même, la suppression d'un utilisateur ne peut se faire que par un administrateur ou l'utilisateur lui-même.**
**d) Seul un administrateur d'ajouter, supprimer, modifier les données des produits.**

### *5. Front-end liste produits*

Ce service affiche une (des) page(s) web. Cette page permet de visualiser une liste de tous les produits en vente.

Pour chaque produit affiché sur cette page, on doit voir son nom, son prix et sa description courte.

Il doit être possible de filtrer les produits par catégorie, et de sélectionner un prix maximum et un prix minimum entre lesquels on désire voir les produits.

Il doit être possible de rediriger l'utilisateur vers le front-end des détails des produits de la liste.

Un administrateur doit avoir le droit de modifier, supprimer ou créer un produit grâce à ce service.

### *6. Front-end détail produit*

Ce service affiche une page web. Cette page permet de voir les détails d'un produit. Doivent s'y trouver son nom, sa description détaillée, son prix, sa catégorie. Via cette page, on doit pouvoir ajouter le produit au panier si l'utilisateur est connecté.

De plus, la moyenne des notes (sur 5) des utilisateurs pour ce produit doit être visible sur cette page, de même que la liste des commentaires associés au produits et leur évaluation par ordre chronologique. Un administrateur doit pouvoir supprimer un commentaire de la liste.

Si l'utilisateur est connecté, il voit en premier lieu ses propres commentaires et évaluations concernant le produit, et reçoit la possibilité de les supprimer ou de les modifier.

Via cette page, un administrateur doit pouvoir modifier ou supprimer le produit.

### *7. Front-end panier utilisateur*

Ce service affiche une page web. Cette page permet de visualiser le panier de l'utilisateur connecté, sous forme de liste de produits. Pour chaque produit, on trouve son nom et son prix unitaire, ainsi que le prix total (si la quantité est supérieure à 1). En fin de liste, on doit pouvoir voir le total des prix.

Il doit être possible de supprimer un produit de cette liste, d'augmenter ou diminuer la quantité d'un produit déjà présent, et de valider le panier en procédant au paiement. Bien sûr, ici on simulera simplement cela en redirigeant l'utilisateur vers une page qui affirme simplement que la commande a été effectuée, en précisant qu'il arrivera à l'adresse postale de l'utilisateur bientôt. Cette nouvelle page permet offre la possibilité d'aller vers la liste des produits.

#### *8. Front-end connexion/inscription*

Ce service affiche une page web. Cette page web permet de se connecter ou de s'inscrire sur le site. Ce service devra établir une session et permettre l'authentification au niveau du reste de l'application.

**NOTES pour les services front-end :**
**a) Toutes pages du site doivent permettre d'être redirigé vers la liste des produits, vers la connexion/inscription (si pas encore connecté) et vers le panier (si connecté).**
**b) Étant donné que vous n'avez pas encore vu la sécurité ni la gestion de session, vous devrez en attendant passer un utilisateur ou son id dans les requêtes qui concernent les utilisateurs (paniers, etc), afin de pouvoir tester votre travail. Vous pourrez modifier le projet avec les sessions en temps voulu.**

## II. Edge micro-services

Vous devrez ajouter, pour optimiser votre architecture micro-services, les edge micro-services. Attendez d'avoir vu la matière en cours pour les implémenter.

Vous devrez externaliser les configurations de vos services en les plaçant sur un service de configuration.

Vous devrez permettre la découvrabilité des nouvelles instances de vos services grâce à un service dédié.

Vous devrez effectuer un partage des requêtes entre les différentes instances d'un même service grâce à un service de load-balancing.

Vous devrez faciliter les points de contact avec vos services REST grâce à un service d'API gateway.

## III. Tests

Vous allez également devoir tester votre application. Vous allez donc devoir déployer un jeu de tests pour chacun des huit services de base. Vous pourrez vous référer au cours pour juger de la complétude de vos tests.

## IV. Rapport écrit

Enfin, il vous est demandé d'écrire un rapport décrivant votre travail lors de ce projet. Nous y attendons :

a. Une brève introduction (nom du site, choix du thème, etc.) de quelques lignes.
b. Une courte description de chaque service : quelles sont ses responsabilités au sein de l'application ?
c. Une section personnelle pour chaque étudiant·e du groupe : précisez à quelles parties du projet vous avez pris part et quelles ont été vos difficultés et leurs solutions.
d. Une conclusion.

Votre rapport doit contenir environ 900 à 2000 mots. Afin de ne pas devoir courir en fin de quadrimestre et d'être stressé par cette rédaction, nous vous recommandons de le remplir chaque partie le plus tôt possible : il vous suffira de rassembler ce qui a été écrit en fin de parcours.

Veillez à l'orthographe et à la qualité de la rédaction, car ces deux points seront pris en compte dans la note. Faites également attention à avoir une mise en page cohérente et agréable à lire.