TP3 - Full stack
Gestion de boutiques, produits et catégories associées

Réalisé par:

- AMARA Lounas
- MOKRANI Yahia

Encadré par:

-M. Lina F. Soualmia

Année universitaire: M2 GIL 2024/2025

Introduction

Dans le cadre de notre projet, qui repose sur l'utilisation de **Spring Boot** pour la partie back-end et **React** pour la partie front-end, on a collaboré afin de développer un **système de gestion de boutiques**. Ce système inclut la gestion des **produits** et des **catégories** associées.

Ce document a pour objectif de :

- Décrire en détail les travaux qu'on a réalisés,
- Expliquer les problèmes rencontrés et les solutions mises en place,
- Présenter les tâches non finalisées ainsi que les pistes d'amélioration pour la suite du projet.

Contexte du Projet

Le projet a pour objectif de créer une **plateforme de gestion de boutiques**, permettant de :

1. **Gérer des produits** : nom, prix, description, etc.
2. **Assigner des catégories** aux différents produits.
3. **Gérer des boutiques** : horaires,produits,congés, etc.
4. **Fournir une interface ergonomique** permettant de naviguer et rechercher des produits et boutiques facilement.
5. **Correction des bugs:** date, configurations, etc.

Pour ce faire, nous avons intégré plusieurs fonctionnalités clés :

- **Front-end React** : affichage dynamique, réactivité, gestion des états.

- **Back-end Spring Boot** : logique métier, accès à la base de données, services RESTful.
- **Recherche avancée (Elasticsearch)** pour une recherche rapide de boutiques ou produits.

Dans le cadre de ce projet, notre équipe a été chargée de plusieurs tâches, notamment :

- **Améliorer l'ergonomie** de l'application,
- **Corriger des anomalies** (gestion des horaires des boutiques, affichage des prix, etc.),
- **Renforcer la robustesse** du système dans son ensemble.

Fonctionnalités Développées et Difficultés Rencontrées

1 Lancement du projet et premières difficultés

Le lancement du projet a été marqué par plusieurs difficultés techniques, notamment :

- **Dockerfile du back-end** : absence de build, problèmes de classpath sur IntelliJ, et dépendances avec le conteneur Elasticsearch.

Nous avons commencé par la mise à jour des versions, notamment la mise à jour du back-end et de React 18 avec Node 22, ainsi que l'upgrade des différents packages Node, sans rencontrer de problèmes majeurs.

2 Mise à jour des versions

La mise à jour des versions du back-end a été plus complexe :

- Passage à **Spring 3.4.2** et **Java 21**,

- Mise à jour des différentes dépendances.

Les principales difficultés sur ce point ont été :

- La nécessité de **changer manuellement le code** pour plusieurs parties du projet (par exemple, migration de `javax` vers `jakarta`),
- Mise à jour de **Springfox** pour la migration vers SpringDoc.

3 Correction du Dockerfile et Docker Compose

Des ajustements ont été nécessaires pour le Dockerfile et Docker Compose :

- Mise à jour des versions des conteneurs (notamment Elasticsearch),
- Limitation de la mémoire pour Elasticsearch avec `ES_JAVA_OPTS : "-Xms1g -Xmx1g"`,
- Ajout d'un **healthcheck** pour garantir que le conteneur soit correctement démarré, car le conteneur prenait beaucoup de temps pour se lancer.

4 Intégration des fonctionnalités

Une fois les mises à jour effectuées, nous avons intégré plusieurs fonctionnalités importantes :

1. **Nombre de catégories par boutique** (modifications front-end et back-end) : Aucune difficulté particulière.
2. **Modification du fichier d'insertion** avec ajout d'index pour la base de données.
3. **Changement de la représentation des prix en centimes** : Affects à la fois le front-end et le back-end.

5 Responsivité et gestion d'erreur

Nous avons ensuite ajouté un minimum de **responsivité** et amélioré la **gestion des erreurs**. Cette partie n'a pas posé de difficultés majeures, mais des ajustements ont été nécessaires pour garantir une bonne expérience utilisateur.

6 Correction du bug des horaires d'ouverture

Un bug a été identifié et corrigé du côté back-end concernant la gestion des horaires d'ouverture des boutiques.

7 Partie Elasticsearch (EL)

La partie Elasticsearch a été particulièrement complexe :

- **Migration des données** avec Liquibase,
- Configuration des fichiers nécessaires (changelog et modification des `application.properties`),
- Prise en compte de la **taille des batches** pour éviter une utilisation excessive de la mémoire (Elasticsearch tend à consommer beaucoup de RAM),
- Intégration avec **Hibernate Search** pour la recherche dans Elasticsearch.

Cette partie a été particulièrement difficile à comprendre et à organiser, ce qui nous a pris beaucoup de temps pour effectuer les tests nécessaires.

Ce qui aurait pu être amélioré

Plusieurs aspects du projet auraient pu être améliorés :

1. **Qualité et propreté du code** : Bien que nous ayons essayé de maintenir un code de qualité, cela a parfois affecté la propreté et la lisibilité du code.

2. **Gestion front-end** : La gestion et le formatage des éléments front-end n’étaient pas idéaux.
3. **Gestion des erreurs** : La gestion des erreurs n’était pas suffisamment robuste. Par exemple, il y a des erreurs où nous ne pouvons pas ajouter une boutique pendant un certain temps, probablement à cause d'une mauvaise configuration.

Méthodologie de Travail

Organisation du travail

Tout au long du projet, nous avons souvent travaillé **ensemble** sur différents sujets en raison de la **complexité des configurations**. La collaboration étroite a été essentielle pour surmonter les défis techniques et les problématiques de mise en place des différents composants du projet.

Nous avons utilisé **Git** pour organiser et gérer le code source, ce qui a facilité la gestion des versions et la coordination entre nous.

Acquis

Ce projet nous a permis de découvrir plusieurs **nouvelles technologies** et d’améliorer notre **technique de résolution de problèmes et de bugs**. La **collaboration** a été un aspect central du travail, ce qui nous a permis d’échanger nos idées et de trouver des solutions efficaces.

Bien que le travail ait parfois été **stressant**, il a constitué un excellent **défi à relever**, nous poussant à sortir de notre zone de confort.

Lancer le projet

Pour que l'application fonctionne correctement, il est nécessaire que les conteneurs de **base de données** et **Elasticsearch** soient en cours d'exécution.

Voici les étapes pour lancer le projet :

- Utiliser la commande suivante pour démarrer les conteneurs avec Docker Compose :

```
docker-compose up --build
```

(N'oubliez pas de supprimer les anciens volumes avant de relancer).

- Vous pouvez également démarrer le front-end et le back-end sans conteneurisation en utilisant :
  - **Front-end** : `npm start`
  - **Back-end** : `mvn spring-boot:run`