|  |
| --- |
|  |
| Portfolio Dev |
| Mateen Khalil, Julien Mares |

|  |
| --- |
| Mateen Khalil, Julien Mares  05/12/2024 |

Table des matières

[Introduction 1](#_Toc184311028)

[Technologies utilisées 1](#_Toc184311029)

[Analyse et planification 2](#_Toc184311030)

[Compte Rendu 3](#_Toc184311031)

[Procès verbale (28.11.2024 – 15h) 3](#_Toc184311032)

[PV (28.11.2024 – 17h10) 3](#_Toc184311033)

[Configuration environnement de développement 3](#_Toc184311034)

[Journal des commits datés 4](#_Toc184311035)

[Développement et tests 5](#_Toc184311046)

[Système de notation 5](#_Toc184311047)

[Filtrage par catégorie 8](#_Toc184311048)

[Résultats finaux et conclusions 10](#_Toc184311049)

# Introduction

Litteralux est un projet que nous reprenons. À l’origine, il servait à afficher des livres récupérés grâce à une API REST spécialement créée pour ce site, permettant également l’ajout de nouveaux ouvrages et la consultation d’informations détaillées. Cette API assure la communication entre le frontend et la base de données.

Cette partie du projet se concentre sur le fait d’appliquer les aspects du Ci/Cd sur Litteralux. Voici la principale feature que nous souhaitons instaurer : configuration de GitHub action pour déployer Litteralux sur Azure.

# Technologies utilisées

HTML/CSS

Langages utilisés pour structurer et styliser les pages web. Ils permettent de concevoir des interfaces utilisateurs et d’assurer un design responsif.

Nous avons utilisé de l’html/css pour notre site car c’est le plus simple, connu et universel.)

Vue js

Framework JavaScript utilisé pour gérer les interfaces dynamiques et réactives du projet.

Nous avons utilisé VueJS afin de pouvoir rendre notre site dynamique, vue js est une des options les plus simples pour cette fonction (par rapport à react js par exemple).

Base de données MySQL, en utilisant PhpMyAdmin comme gestionnaire de db

Système relationnel permettant le stockage et la gestion des livres (technologie non spécifiée, à compléter).

Docker

Afin de pouvoir utilisé mysql et phpmyadmin simplement sur n’importe quel PC nous avons utilisé des conteneurs docker contenant mysql et phpmyadmin.

# Analyse et planification

Analyse initiale :

L'objectif principal du projet est de mettre en place la méthodologie Ci/Cd afin de simplifier et automatiser le déploiement d'un site web sur environnement de production sur Azure. Ce processus doit permettre de réduire les risques d'erreurs humaines, d'accélérer le cycle de livraison et de garantir une qualité constante des livrables.

L’utilisateur soumet un pull request sur la branche main en incluant les mots-clés « major », « minor » ou « patch » dans le message d’accompagnement, selon l’importance des modifications proposées. Il est important de noter que si aucun de ces mots-clés n’est présent dans le message, aucun workflow ne sera déclenché. Cela garantit que seules les versions fonctionnelles et suffisamment avancées du projet reçoivent un tag versionné.

Une fois la pull request envoyée, le premier workflow exécute les tests unitaires avec Cypress et effectue une analyse lint du projet.

Si les tests ne passent pas tous avec succès, un deuxième workflow est déclenché pour déployer la version actuelle du site web dans l’environnement de staging sur Azure, permettant ainsi des vérifications supplémentaires.

En revanche, si tous les tests sont réussis, un troisième workflow est activé pour déployer directement la version actuelle du projet dans l’environnement de production sur Azure.

Analyse initiale :

# Compte Rendu

## Procès verbale (28.11.2024 – 15h)

**Mateen**

* **Rédaction UserStory Barre de recherche**
* **Rédaction UserStory Système de notation**

**Julien**

* **Rédaction UserStory Filtrage par catégorie**
* **Rédaction UserStory Tests Unitaires**

## PV (28.11.2024 – 17h10)

**Mateen**

* **Barre de recherche**
* **Tests unitaires**

**Julien**

* **Filtrage par catégorie**
* **Système de notation**

# **Configuration environnement de développement**

* Installer Node (v20.12.0) sur la machine
* Installer Docker Desktop sur la machine
* Installer Visual Studio Code
* Télécharger le repo suivant : <https://github.com/TheArabicMonster/PortofolioDev>
* Unzipper le dossier
* A la racine du projet, taper dans un cmd « docker-compose up -d »
* Dans un cmd depuis la racine du projet taper « npm install »
* Dans un cmd depuis la racine du projet se déplacer vers /Code/Frontend/ et taper « npm run dev »
* Dans un cmd depuis la racine du projet se déplacer vers /Code/Backend/ et taper « npm start »

# Journal des commits datés

## Commits du 5 décembre 2024

### Documentation

* **Doc** : Ajout en tête du portfolio (JulienETML-hub)
* **Doc** : Ajout des conclusions dans le portfolio (JulienETML-hub)
* **Doc** : Mise à jour du journal de travail (JulienETML-hub)
* **Doc** : Ajout du portfolio (rapport) (JulienETML-hub)

### Fonctionnalités et Refactorisations

* **Update** : Ajout des contacts dans le footer (JulienETML-hub)
* **Refactor** : Redimensionnement de la taille de l'image dans DetailsLivre (JulienETML-hub)
* **Feat** : Refactorisation du composant BookList pour un meilleur filtrage par catégories et amélioration de l'interface utilisateur (TheArabicMonster)
* **Feat** : Refactorisation du composant FormAppreciation pour inclure l'affichage de la note moyenne et améliorer la structure du formulaire (TheArabicMonster)
* **Feat** : Triage des livres par catégories dans BookList.vue (JulienETML-hub)
* **Refactor** : Renommage de variables (JulienETML-hub)

### Tests

* **Feat** : Mise à jour de la configuration de Cypress et ajout de tests End-to-End (E2E) pour l'ajout de livres (TheArabicMonster)

### Autres

* **Merge** : Fusion de la branche main depuis TheArabicMonster/PortofolioDev (Multiples commits)

## Commits du 4 décembre 2024

### Fonctionnalités

* **Feat** : Mise à jour de la configuration Cypress et ajout de tests pour la recherche (WIP) (TheArabicMonster)
* **Feat** : Ajout d'une fonctionnalité de recherche et d'une page de résultats pour les livres (TheArabicMonster)
* **Feat** : Ajout d'une option et d'un filtre pour trier les livres par catégorie (TheArabicMonster)

### Tests

* **Feat** : Ajout de Cypress pour réaliser des tests (TheArabicMonster)

## Commits du 28 novembre 2024

### Initialisation

* **Doc** : Journal de travail jour 1 (TheArabicMonster)
* **Ajout** : Projet Litteralux (JulienETML-hub)
* **Initial Commit** (TheArabicMonster)

# Développement et tests

## Barre de recherche

Nous avons ajouté une barre de recherche dans le header pour qu’elle puisse être utilisé à tout moment, elle est composée d’un champ de saisie et d’un bouton, voici quelque précision sur cette dernière :

Le bouton à un gestionnaire d’évènement qui appel la méthode « *search* », quand elle est appelée elle vérifie si la propriété a une valeur, si c’est le cas elle redirige l’utilisateur vers la vue « *SearchResult* » incluant dans l’url la valeur que l’utilisateur avait entrer. Ensuite la page « *SearchResult* » récupère tous les livres de la DB et ajoute dans une liste tout les livres qui comporte dans leur nom le paramètre donné dans la barre de recherche. Pour finalement afficher tous les livres qui se trouve dans ce liste grâce à une liste.



Figure exemple résultat recherche

## Réalisation de test

Nous avons utiliser cypress, pour ce faire nous avons ajouté un fichier de configuration pour définir l’url de base qui sera utilisé pour les test.

1. Test search

Ce fichier entre « comte » dans la barre de recherche et vérifie qu’il est ensuite rediriger vers une page qui comprends le titre « Résultats de recherche pour "comte »

1. Test add-book

Ce fichier test que quand un livre est ajouter depuis le site, il est ajouté dans la base de données, en premier lieu elle remplie le formulaire d’ajout avec des informations de test, pour ensuite submit le formulaire, et enfin, vérifie que dans la console il y a bien un message de confirmation vis-a-vis du fais que le livre est ajouté.

Au début nous voulions vérifier que le livre est ajouté en faisant une requête à l’api pour avoir tout les livres de la db et vérifier que le livre a bien été ajouté mais une erreur cypress nous empêchais de conlure ce test.

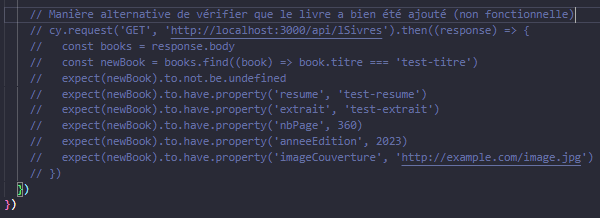


Figure maière non fonctionnel de vérifier ajout livre dans db

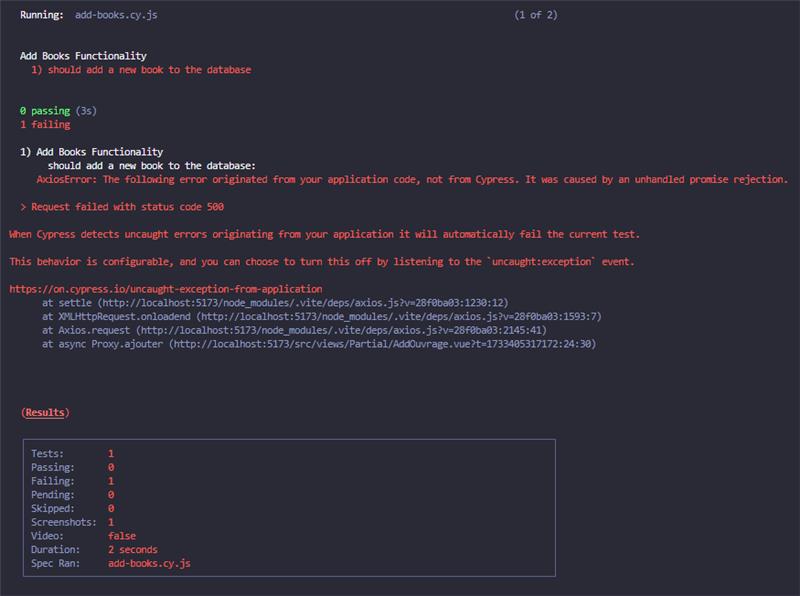
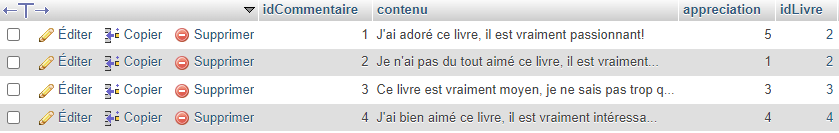


Figure message erreur cypress

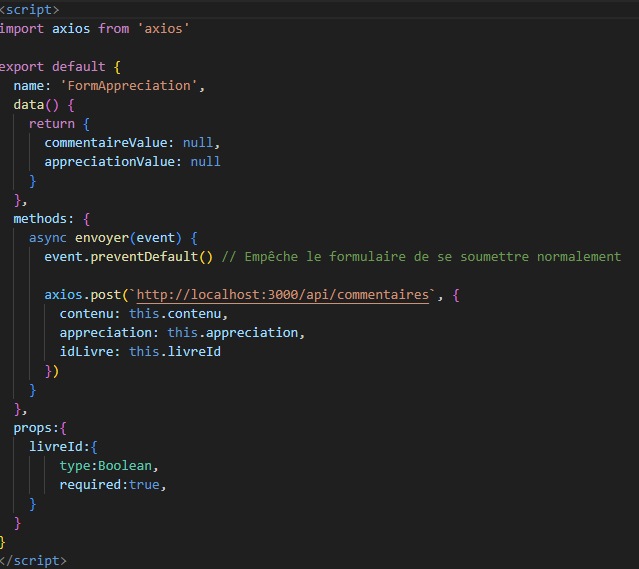
## Système de notation

Afin de pouvoir rajouter un système de notation, nous avons suivis ces étapes :

Ajout d’une table commentaire dans la db (adapter mockups & data models)



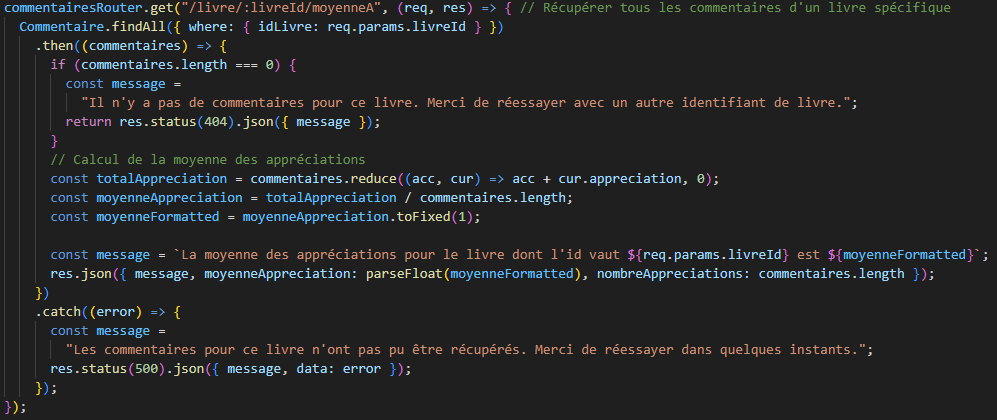
Création d’une vue servant de formulaire pour envoyer un commentaire ainsi qu’une note



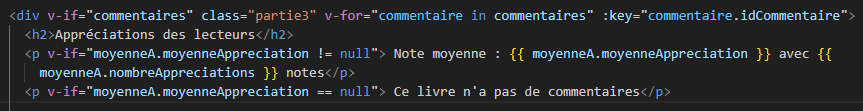
Afficher cette vue sur la page « DetailsLivre » (en fonction de livre sélectionné par l’utilisateur

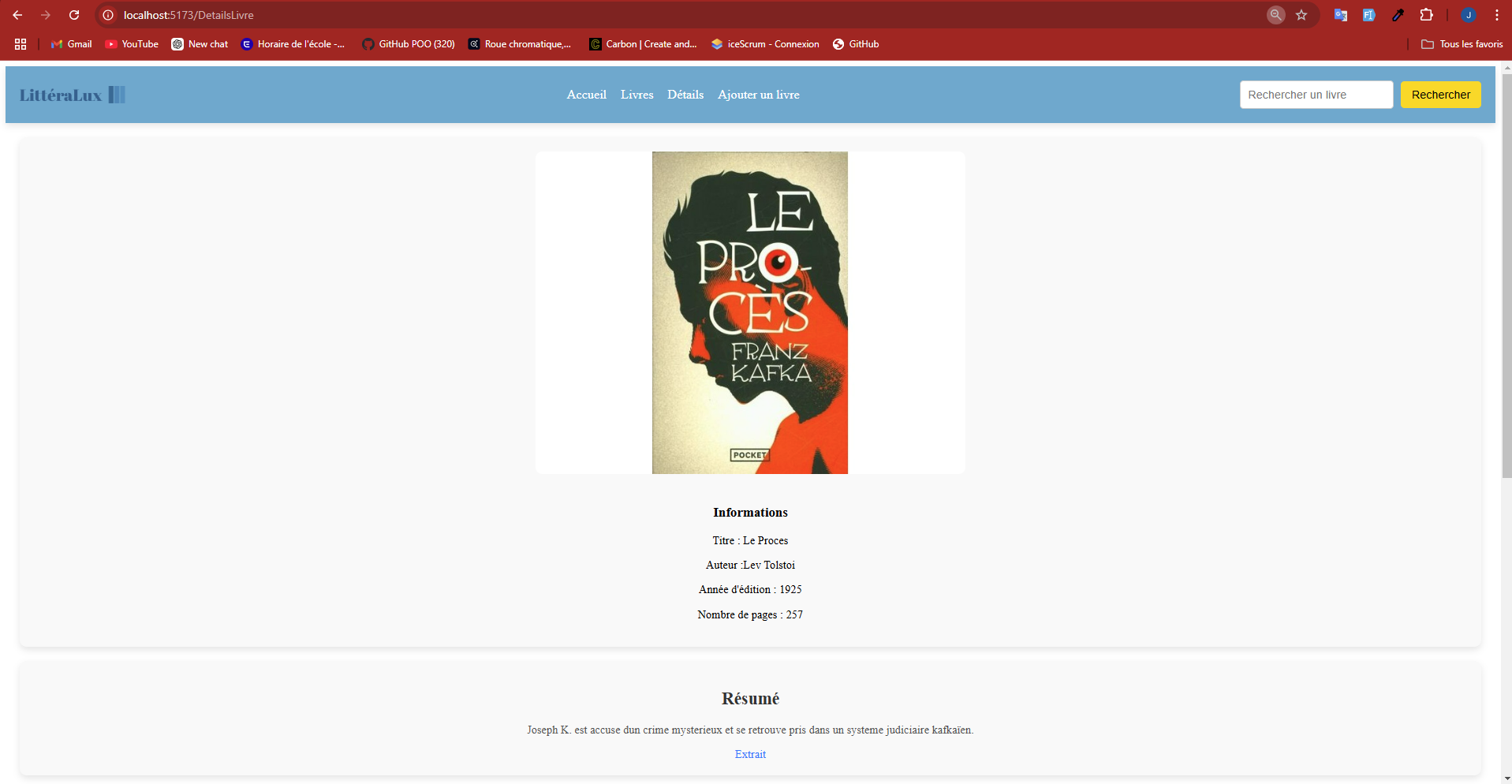


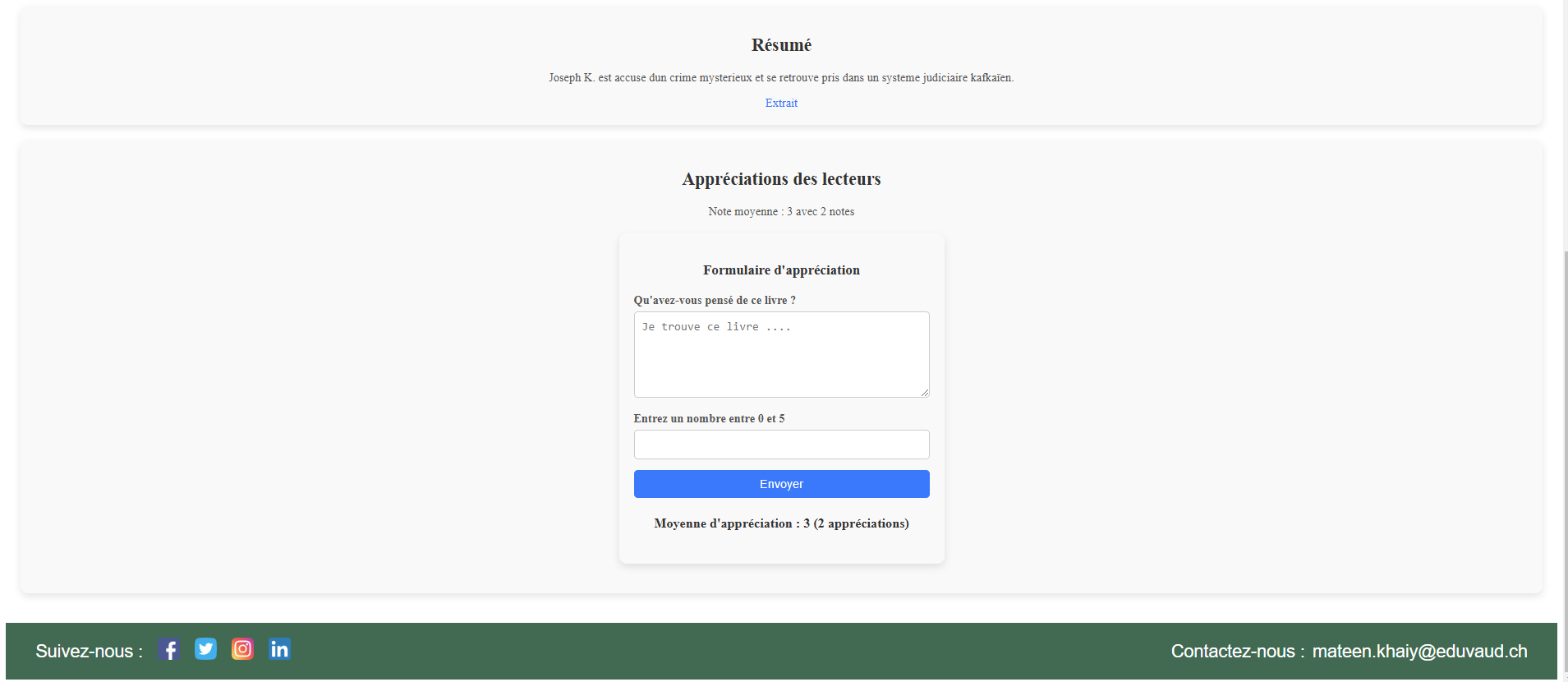
Création d’une nouvelle route spécialement faites pour retourner la moyenne de toutes notes(appréciations)



Afficher la moyenne des notes sur le site





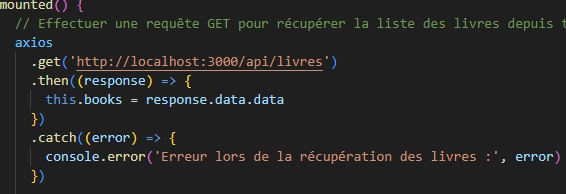


## Filtrage par catégorie

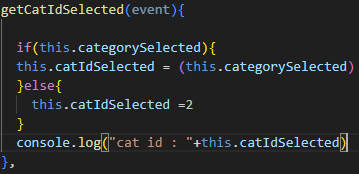
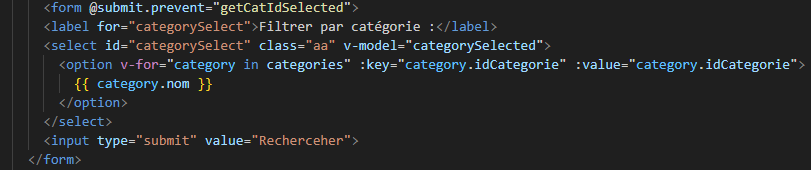
Afin d’implémenter un filtre par catégorie, nous avons suivis ces étapes :

Import des livres via l’API

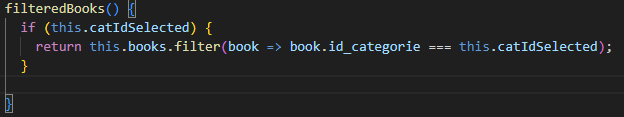
On stock l’ensemble des livres et leurs informations relatives dans « books »



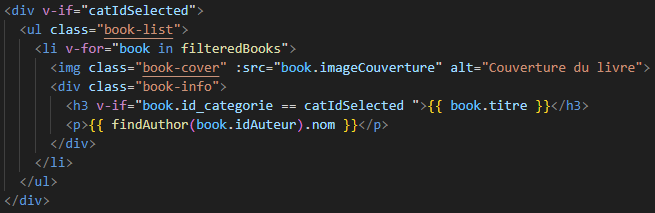
Implementation de l’outil de sélection de catégorie

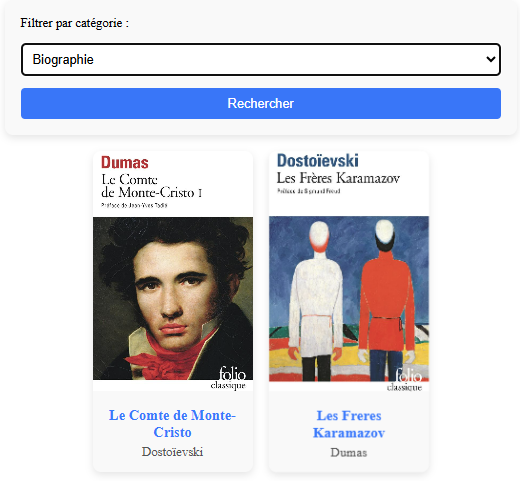


Création méthode de filtre qui renvoie seulement les livres correspondant à la catégorie sélectionnée



Affichage des informations des livres de la catégorie





# Résultats finaux et conclusions

**Ce projet nous a permis, à travers un ancien projet connu, de pratiquer le développement d'une application web complète, depuis l'intégration d'outils modernes comme Vue.js et Docker jusqu'à la mise en place de fonctionnalités utilisateurs. Finalement nous avons réussis à implémenter les fonctionnalités voulues.**

**Ce projet nous a fais prendre conscience d’à quel point il est important d’écrire un code clair, car nous nous sommes rendu compte que la manière dont nous avions codé à l’époque pouvait des fois être peu lisible.**