

Portfolio Dev

Mateen Khalil, Julien Mares

Mateen Khalil, Julien Mares
05/12/2024

Table des matières

Introduction	2
Technologies utilisées.....	2
Analyse et planification	2
Compte Rendu	4
Procès verbale (28.11.2024 – 15h)	4
PV (28.11.2024 – 17h10).....	4
Configuration environnement de développement.....	4
Journal des commits datés	5
Développement et tests	6
Système de notation	8
Filtrage par catégorie	11
Résultats finaux et conclusions	13

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.

Parmi ces ajouts, nous comptons intégrer une barre de recherche, un système de notation, un filtrage par catégorie, ainsi que des tests unitaires pour assurer la fiabilité du projet.

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

Barre de recherche :

Permettre aux utilisateurs de trouver rapidement un livre parmi ceux disponibles, facilitant l'accès aux informations pertinentes.

Barre de recherche En tant qu'utilisateur

Je veux une barre de recherche Pour rechercher facilement et simplement un livre
Affichage de la barre de recherche L'utilisateur est sur la page de la liste des livres. La page est chargée. La barre de recherche doit être visible.
Recherche de livre avec des résultats L'utilisateur est sur la page de la liste des livres et il y a des livres disponibles. L'utilisateur entre un mot-clé correspondant à un livre dans la barre de recherche. Les livres correspondant au mot-clé doivent être affichés en temps réel.
Recherche de livre sans résultats L'utilisateur est sur la page de la liste des livres et il y a des livres disponibles. L'utilisateur entre un mot-clé ne correspondant à aucun livre dans la barre de recherche. Un message indiquant qu'aucun livre n'a été trouvé doit être affiché.

Système de notation (1 à 5) :

Offrir aux utilisateurs la possibilité d'évaluer les livres, afin de partager leur avis et de faire des choix éclairés.

Système de notation En tant qu'utilisateur du site web, Je veux pouvoir noter un livre, Afin de partager mon avis sur le livre en utilisant une note de 0 à 5.
Affichage de la liste déroulante de notation L'utilisateur est sur la page de détails d'un livre. La page est chargée. La liste déroulante de notation doit être visible.
Soumission d'une note L'utilisateur est sur la page de détails d'un livre et a sélectionné une note. L'utilisateur soumet la note. La note doit être enregistrée et la note moyenne mise à jour doit être affichée.

Filtrage par catégorie

Liste déroulante pour filtrer catégories En tant qu'utilisateur Je veux une liste déroulante des catégories de livres Afin de pouvoir filtrer les livres par leurs catégories
Affichage liste catégories Etant donné que je suis sur la page « livre » Lorsque que j'appuie sur le "liste déroulante" nommé "Trier par catégories" Cela m'affiche la liste des catégories
Sélection catégorie Etant donné que j'ai cliqué sur le menu et que je vois toutes les catégories Lorsque je clique sur une catégorie La page n'affiche plus que les livres appartenant à la catégorie sélectionnée
Catégorie sélectionnée

Etant donné que j'ai sélectionné une catégorie
Lorsque je regarde la liste déroulante
Je vois qu'il est marqué "Filtré par" la catégorie choisie

Tests unitaires

Garantir la robustesse du code et la stabilité de l'application grâce à l'implémentation de tests automatisés.

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` (*Multiple 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 utilisée 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 appelle 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 entré. Ensuite la page « *SearchResult* » récupère tous les livres de la DB et ajoute dans une liste tous les livres qui comportent dans leur nom le paramètre donné dans la barre de recherche. Pour finalement afficher tous les livres qui se trouvent dans cette liste grâce à une liste.

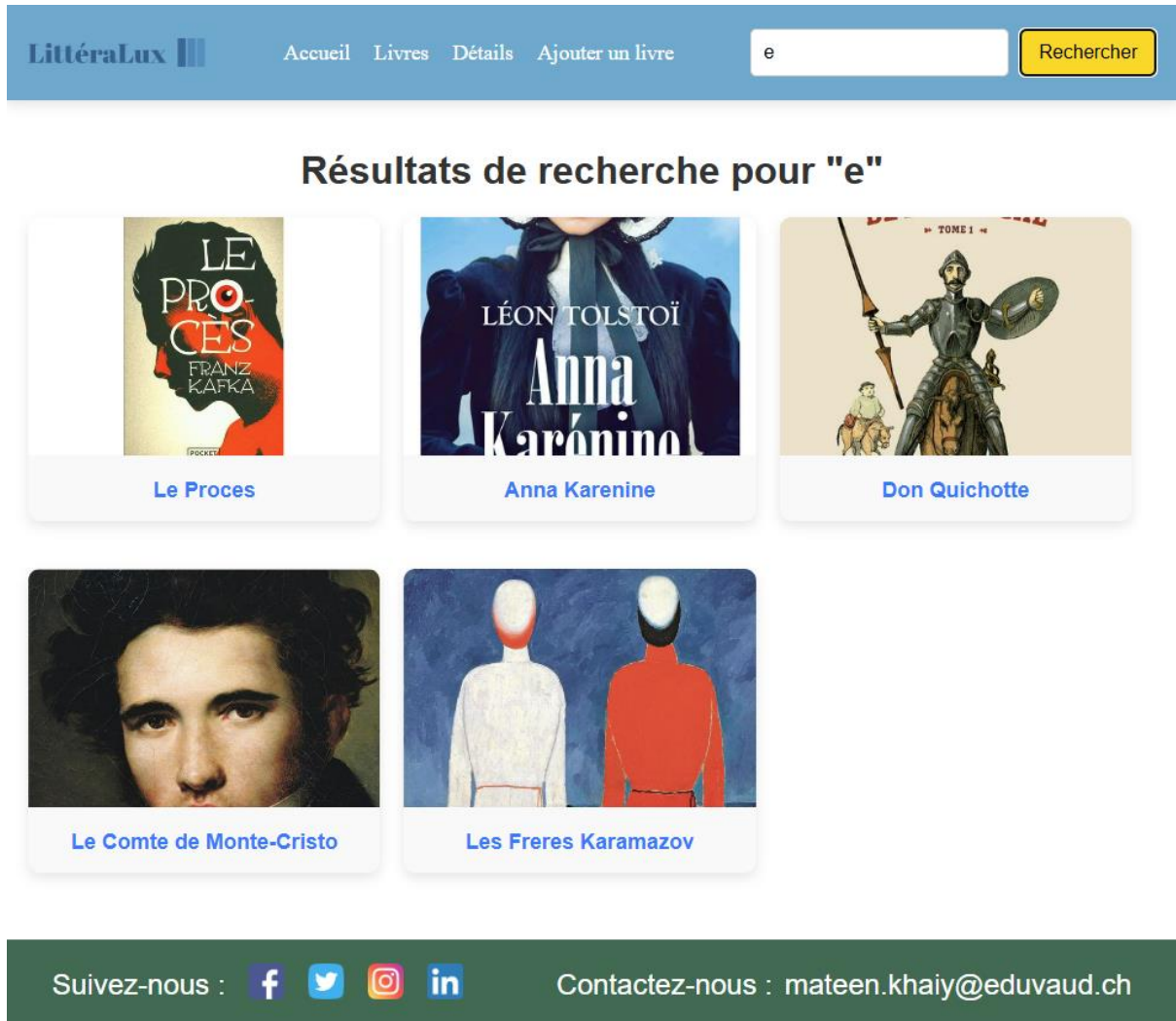


Figure 1 exemple résultat recherche

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 »

2. 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 remplit 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 fait 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 conclure ce test.

```
// Manière alternative de vérifier que le livre a bien été ajouté (non fonctionnelle)
// cy.request('GET', 'http://localhost:3000/api/livres').then((response) => {
//   const books = response.body
//   const newBook = books.find((book) => book.titre === 'test-titre')
//   expect(newBook).to.not.be.undefined
//   expect(newBook).to.have.property('resume', 'test-resume')
//   expect(newBook).to.have.property('extrait', 'test-extrait')
//   expect(newBook).to.have.property('nbPage', 360)
//   expect(newBook).to.have.property('anneeEdition', 2023)
//   expect(newBook).to.have.property('imageCouverture', 'http://example.com/image.jpg')
// })
})
})
```

Figure 2 manière non fonctionnel de vérifier ajout livre dans db

```
Running: add-books.cy.js (1 of 2)

Add Books Functionality
  1) should add a new book to the database

0 passing (3s)
1 failing

1) Add Books Functionality
   should add a new book to the database:
     AxiosError: The following error originated from your application code, not from Cypress. It was caused by an unhandled promise rejection.

> Request failed with status code 500

When Cypress detects uncaught errors originating from your application it will automatically fail the current test.

This behavior is configurable, and you can choose to turn this off by listening to the 'uncaught:exception' event.
https://on.cypress.io/uncaught-exception-from-application
at settle (http://localhost:5173/node_modules/.vite/deps/axios.js?v=28f0ba03:1230:12)
at XMLHttpRequest.onloadend (http://localhost:5173/node_modules/.vite/deps/axios.js?v=28f0ba03:1593:7)
at Axios.request (http://localhost:5173/node_modules/.vite/deps/axios.js?v=28f0ba03:2145:41)
at async Proxy.ajouter (http://localhost:5173/src/views/Partial/AddOuvrage.vue?t=1733405317172:24:30)

(Results)













Tests:      1
Passing:    0
Failing:    1
Pending:    0
Skipped:    0
Screenshots: 1
Video:      false
Duration:   2 seconds
Spec Ran:   add-books.cy.js
```

Figure 3 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)

		idCommentaire	contenu	appreciation	idLivre
<input type="checkbox"/>	 Éditer  Copier  Supprimer	1	J'ai adoré ce livre, il est vraiment passionnant!	5	2
<input type="checkbox"/>	 Éditer  Copier  Supprimer	2	Je n'ai pas du tout aimé ce livre, il est vraiment...	1	2
<input type="checkbox"/>	 Éditer  Copier  Supprimer	3	Ce livre est vraiment moyen, je ne sais pas trop q...	3	3
<input type="checkbox"/>	 Éditer  Copier  Supprimer	4	J'ai bien aimé ce livre, il est vraiment intéressa...	4	4

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

```
<script>
import axios from 'axios'

export default {
  name: 'FormAppreciation',
  data() {
    return {
      commentaireValue: null,
      appreciationValue: null
    }
  },
  methods: {
    async envoyer(event) {
      event.preventDefault() // Empêche le formulaire de se soumettre normalement

      axios.post(`http://localhost:3000/api/commentaires`, {
        contenu: this.contenu,
        appreciation: this.appreciation,
        idLivre: this.livreId
      })
    }
  },
  props: {
    livreId: {
      type: Boolean,
      required: true,
    }
  }
}
</script>
```

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

```
<FormAppreciation :livreId="livres.data.idLivre"></FormAppreciation>
```

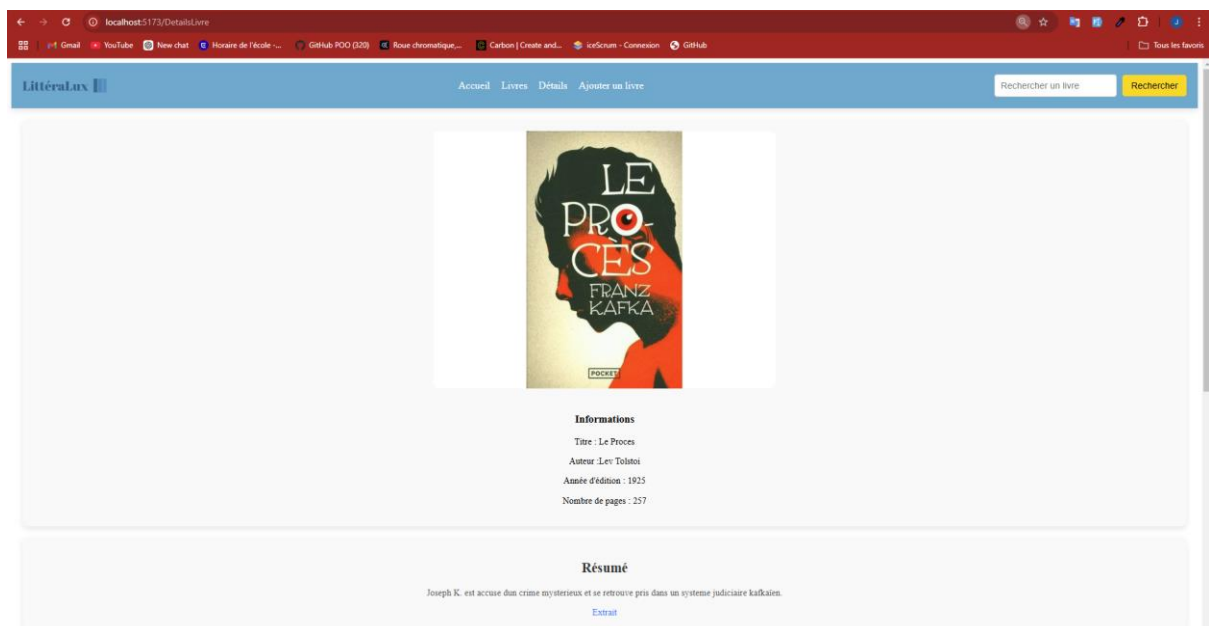
Création d'une nouvelle route spécialement faites pour retourner la moyenne de toutes notes(appréciations)

```
commentairesRouter.get("/livre/:livreId/moyenneA", (req, res) => { // Récupérer tous les commentaires d'un livre spécifique
  Commentaire.findAll({ where: { idLivre: req.params.livreId } })
    .then((commentaires) => {
      if (commentaires.length === 0) {
        const message =
          "Il n'y a pas de commentaires pour ce livre. Merci de réessayer avec un autre identifiant de livre.";
        return res.status(404).json({ message });
      }
      // Calcul de la moyenne des appréciations
      const totalAppreciation = commentaires.reduce((acc, cur) => acc + cur.appreciation, 0);
      const moyenneAppreciation = totalAppreciation / commentaires.length;
      const moyenneFormatted = moyenneAppreciation.toFixed(1);

      const message = `La moyenne des appréciations pour le livre dont l'id vaut ${req.params.livreId} est ${moyenneFormatted}`;
      res.json({ message, moyenneAppreciation: parseFloat(moyenneFormatted), nombreAppreciations: commentaires.length });
    })
    .catch((error) => {
      const message =
        "Les commentaires pour ce livre n'ont pas pu être récupérés. Merci de réessayer dans quelques instants.";
      res.status(500).json({ message, data: error });
    });
});
```

Afficher la moyenne des notes sur le site

```
<div v-if="commentaires" class="partie3" v-for="commentaire in commentaires" :key="commentaire.idCommentaire">
  <h2>Appréciations des lecteurs</h2>
  <p v-if="moyenneA.moyenneAppreciation != null"> Note moyenne : {{ moyenneA.moyenneAppreciation }} avec {{
    moyenneA.nombreAppreciations }} notes</p>
  <p v-if="moyenneA.moyenneAppreciation == null"> Ce livre n'a pas de commentaires</p>
```



Résumé

Joseph K. est accusé d'un crime mystérieux et se retrouve pris dans un système judiciaire kafkaïen.

[Extrait](#)

Appréciations des lecteurs

Note moyenne : 3 avec 2 notes

Formulaire d'appréciation

Qu'avez-vous pensé de ce livre ?

Je trouve ce livre

Entrez un nombre entre 0 et 5

Envoyer

Moyenne d'appréciation : 3 (2 appréciations)

Suivez-nous : [f](#) [t](#) [i](#) [in](#)

Contactez-nous : mateen.khaiy@eduvaud.ch

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 »

```
mounted() {  
  // Effectuer une requête GET pour récupérer la liste des livres depuis t  
  axios  
    .get('http://localhost:3000/api/livres')  
    .then((response) => {  
      this.books = response.data.data  
    })  
    .catch((error) => {  
      console.error('Erreur lors de la récupération des livres :', error)  
    })  
}
```

Implementation de l'outil de sélection de catégorie

```
<form @submit.prevent="getCatIdSelected">
  <label for="categorySelect">Filtrer par catégorie :</label>
  <select id="categorySelect" class="aa" v-model="categorySelected">
    <option v-for="category in categories" :key="category.idCategorie" :value="category.idCategorie">
      {{ category.nom }}
    </option>
  </select>
  <input type="submit" value="Rechercher">
</form>
```

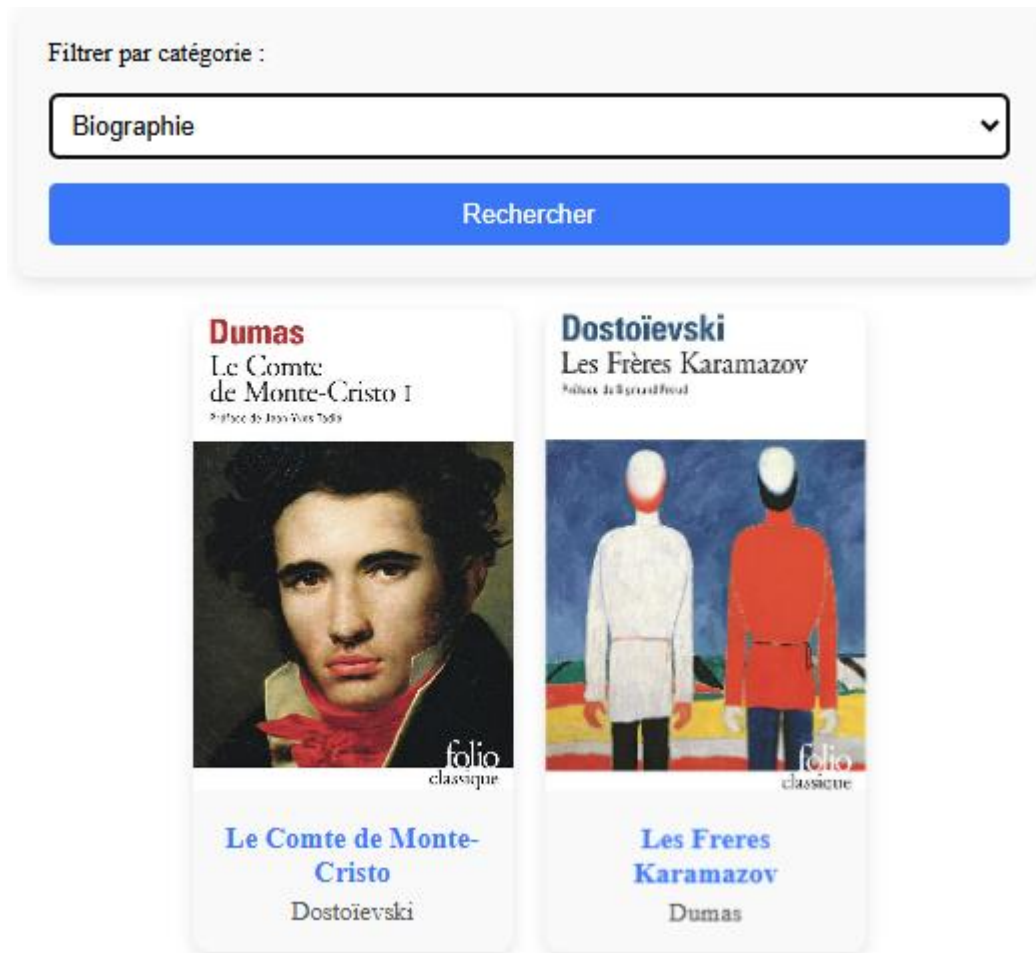
```
getCatIdSelected(event){
  if(this.categorySelected){
    this.catIdSelected = (this.categorySelected)
  }else{
    this.catIdSelected =2
  }
  console.log("cat id : "+this.catIdSelected)
},
```

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

```
filteredBooks() {
  if (this.catIdSelected) {
    return this.books.filter(book => book.id_categorie === this.catIdSelected);
  }
}
```

Affichage des informations des livres de la catégorie

```
<div v-if="catIdSelected">
  <ul class="book-list">
    <li v-for="book in filteredBooks">
      
      <div class="book-info">
        <h3 v-if="book.id_categorie == catIdSelected ">{{ book.titre }}</h3>
        <p>{{ findAuthor(book.idAuteur).nom }}</p>
      </div>
    </li>
  </ul>
</div>
```



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 réussi à implémenter les fonctionnalités voulues.

Ce projet nous a fait 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 avons codé à l'époque pouvait des fois être peu lisible.