



**ECOLE MAROCAINE DES
SCIENCES DE L'INGENIEUR**

Membre de

HONORIS UNITED UNIVERSITIES

Thème

*Application web pour la gestion
d'un book store en ligne*

Option : Ingénierie informatique et réseaux

Réalisé par :

Anas ANASRI

2022-2023

Table des matières

Introduction	1
1 Outils de développement.....	2
1.1 Github	2
1.2 Javascript.....	2
1.3 NodeJs	3
1.4 React.....	3
1.5 Html	4
1.6 Css.....	4
1.7 Express.....	5
1.8 MongoDB.....	5
1.9 Mongoose.....	6
1.10 Axios	6
2 Explication de code.....	7
2.1 Backend	7
2.1.1 Partie de book Controller	7
2.1.2 book Services	8
2.2 Frontend	9
2.2.1 Book liste component.....	9
2.2.2 Book services	10
2.2.3 Add Book component	11
3 Présentation des interfaces de l'application.....	12
3.1 Interface d'accueil	12
3.2 Page la liste des livres.....	12
3.3 Page détail d'un livre	13
3.4 Interface d'authentification.....	14
3.5 Filtrer les livres par mot clé	14
3.6 Filtrer les livres par catégorie	15
3.7 Interface de Responsable modifier ou supprimer un livre.....	15
3.8 modifier un livre	16
3.9 Interface de Responsable ajouter un livre.....	17
Conclusion	18

Introduction

L'objectif principal consiste à concevoir et réaliser une application web qui permet de gérer un book store. Le but principal de l'application est de simplifier les différentes tâches de gestion des livres avec leurs catégories dans le store et de donner la possibilité de les visualiser aux visiteurs. L'application sera développée en deux parties, backend et frontend. Les technologies utilisées sont : NodeJS avec le framework Express, TypeScript, JWT, React et Tailwind.

La réalisation d'une telle application nous permet d'atteindre les objectifs suivants :

- Avoir une vision globale de la situation des livres de store.
- Alléger les traitements manuels des responsables au niveau de l'organisation physique des livres.
- Faciliter le contrôle de store pour les responsables.

1 Outils de développement

1.1 Github



Figure 1.1 Github

Github est une entreprise de développement et services logiciels sise aux États-Unis. Github développe notamment la plateforme Github, l'éditeur de texte Atom ou encore la structure Electron . Le 4 juin 2018, Microsoft annonce l'acquisition de l'entreprise pour la somme de 7,5 milliards de dollars américains GitHub est un site web et un service de cloud qui aide les développeurs à stocker et à gérer leur code, ainsi qu'à suivre et contrôler les modifications qui lui sont apportées

1.2 Javascript



Figure 1.2 Javascript

JavaScript est un langage de programmation de scripts principalement employé dans les pages web interactives et à ce titre est une partie essentielle des applications web. Avec les langages HTML et CSS, JavaScript est au cœur des langages utilisés par les développeurs web.

1.3 NodeJs



Figure 1.3 NodeJs

Node.js est une plateforme logicielle libre en JavaScript, orientée vers les applications réseau évènementielles hautement concurrentes qui doivent pouvoir monter en charge. Elle utilise la machine virtuelle V8, la librairie libuv pour sa boucle d'évènements, et implémente sous licence MIT les spécifications CommonJS

1.4 React

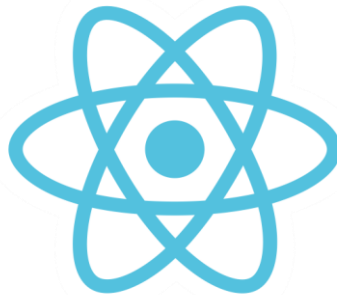


Figure 1.4 React

React est un cadre JavaScript développé par Facebook pour la création d'interfaces utilisateur dynamiques et riches. Il permet aux développeurs de construire des applications web en utilisant des composants réutilisables qui peuvent être facilement mis à jour en fonction des données en temps réel. React est très populaire pour la création de sites web et d'applications mobiles en raison de sa vitesse, de sa flexibilité et de sa facilité d'utilisation. Il travaille bien avec d'autres bibliothèques et outils pour fournir une expérience utilisateur fluide et réactive.

1.5 Html



Figure 1.5 Html

Le HyperText Markup Language, généralement abrégé HTML ou, dans sa dernière version, HTML5, est le langage de balisage conçu pour représenter les pages web. Ce langage permet : d'écrire de l'hypertexte, d'où son nom, de structurer sémantiquement la page, de mettre en forme le contenu,

1.6 Css



Figure 1.6 Css

Les feuilles de style en cascade, généralement appelées CSS de l'anglais Cascading Style Sheets, forment un langage informatique qui décrit la présentation des documents HTML et XML. Les standards définissant CSS sont publiés par le World Wide Web Consortium.

1.7 Express



Figure 1.7 Express

Express est un framework Node.js pour les applications web et les API. Il offre une structure pour les applications web et permet de gérer les requêtes HTTP et les réponses HTTP. Il est souvent utilisé avec d'autres bibliothèques pour offrir une gamme de fonctionnalités supplémentaires.

1.8 MongoDB



Figure 1.8 MongoDB

MongoDB est un système de gestion de base de données NoSQL qui permet de stocker et de gérer des données de manière flexible et scalable. Il utilise le modèle de données BSON (Binary JSON) pour stocker les données et offre une gamme de fonctionnalités pour la recherche et la manipulation des données.

1.9 Mongoose



Figure 1.9 Mongoose

Mongoose est un ORM (Object-Relational Mapping) pour MongoDB. Il permet aux développeurs de travailler avec les données de MongoDB en utilisant une syntaxe similaire à celle de JavaScript. Il offre une validation de schéma, une gestion des relations entre les données et d'autres fonctionnalités pour faciliter la gestion des données.

1.10 Axios



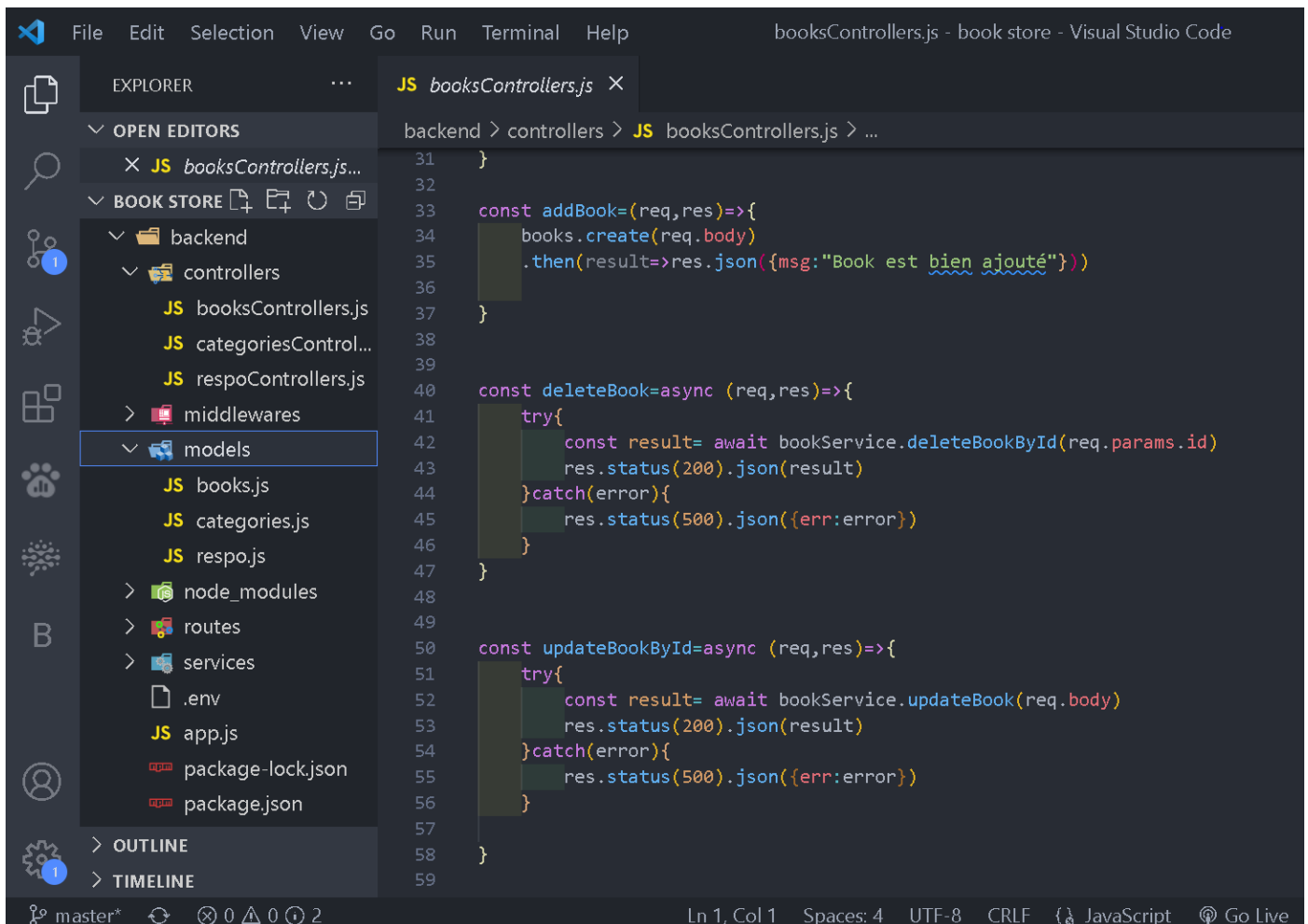
Figure 1.10 axios

Axios est une bibliothèque JavaScript pour les requêtes HTTP. Il permet de faire des requêtes HTTP à partir du client ou du serveur et prend en charge les fonctionnalités telles que les requêtes asynchrones, les intercepteurs de requêtes et les réponses, ainsi que la gestion des erreurs. Axios est facile à utiliser et est très populaire auprès des développeurs JavaScript.

2 Explication de code

2.1 Backend

2.1.1 Partie de book Controller



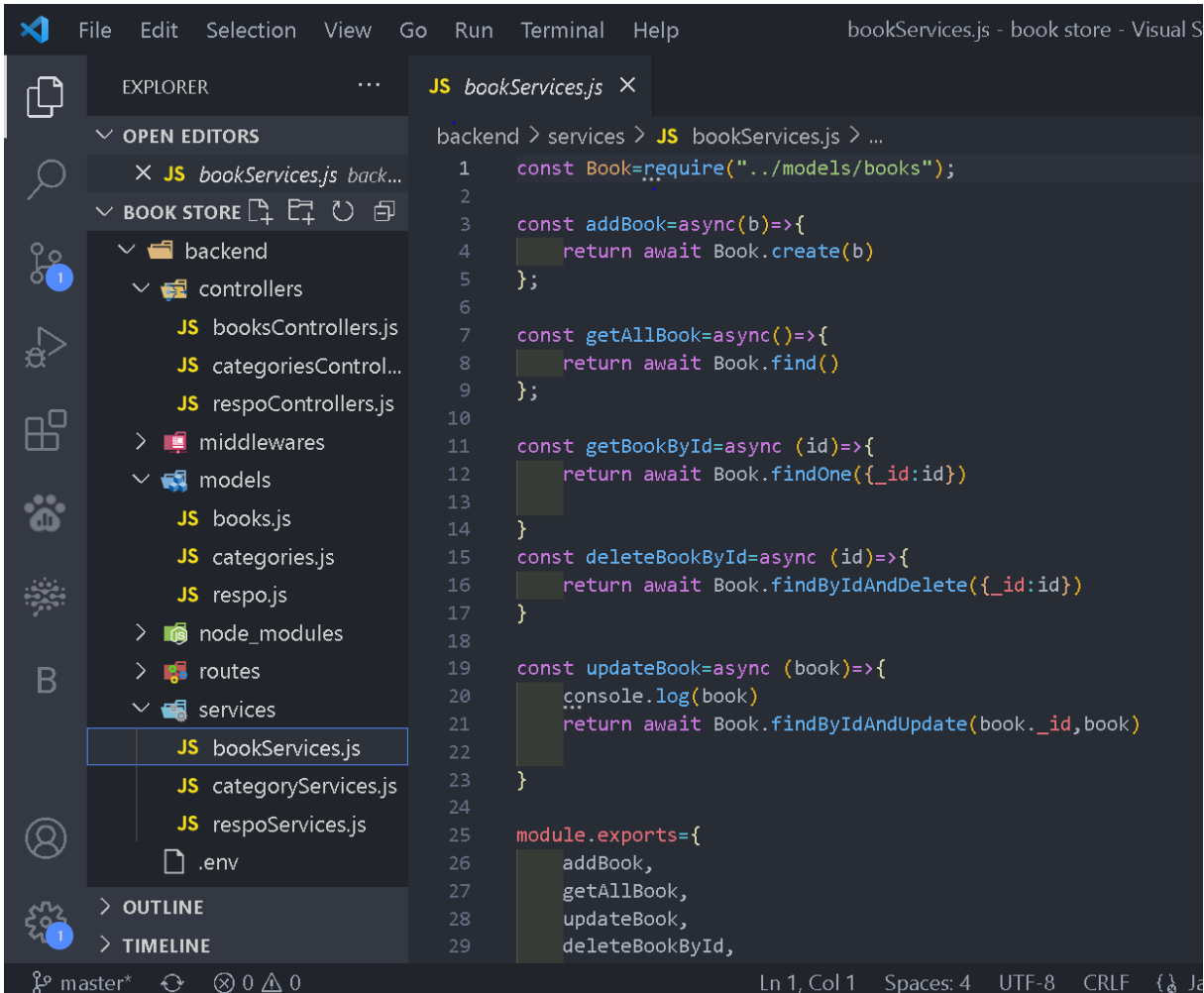
```
31 }
32
33 const addBook=(req,res)=>{
34   books.create(req.body)
35   .then(result=>res.json({msg:"Book est bien ajouté"}))
36 }
37
38
39
40 const deleteBook=async (req,res)=>{
41   try{
42     const result= await bookService.deleteBookById(req.params.id)
43     res.status(200).json(result)
44   }catch(error){
45     res.status(500).json({err:error})
46   }
47 }
48
49
50 const updateBookById=async (req,res)=>{
51   try{
52     const result= await bookService.updateBook(req.body)
53     res.status(200).json(result)
54   }catch(error){
55     res.status(500).json({err:error})
56   }
57 }
58
59 }
```

La fonction "addBook" crée un nouveau livre en utilisant les données fournies dans la requête, puis envoie une réponse JSON indiquant que le livre a été ajouté avec succès.

La fonction "deleteBook" supprime un livre de la base de données en utilisant l'identifiant fourni dans la requête, puis envoie une réponse JSON contenant les informations du livre supprimé.

La fonction "updateBookById" met à jour les informations d'un livre dans la base de données en utilisant les données fournies dans la requête, puis envoie une réponse JSON contenant les informations du livre mis à jour. Si une erreur se produit pendant la mise à jour, la fonction renvoie une réponse JSON contenant un message d'erreur.

2.1.2 book Services



The screenshot shows the Visual Studio Code interface. On the left, the Explorer sidebar displays the project structure for 'book store'. The 'services' folder is expanded, showing 'bookServices.js' selected. The main editor window displays the code for 'bookServices.js'. The code defines several asynchronous methods for interacting with a database: 'addBook', 'getAllBook', 'getBookById', 'deleteBookById', and 'updateBook'. Each method uses the 'Book' model and returns a promise. The 'module.exports' object lists the exported functions.

```
backend > services > JS bookServices.js > ...
1  const Book=require("../models/books");
2
3  const addBook=async(b)=>{
4    return await Book.create(b)
5  };
6
7  const getAllBook=async()=>{
8    return await Book.find()
9  };
10
11 const getBookById=async (id)=>{
12   return await Book.findOne({_id:id})
13 }
14
15 const deleteBookById=async (id)=>{
16   return await Book.findByIdAndDelete({_id:id})
17 }
18
19 const updateBook=async (book)=>{
20   console.log(book)
21   return await Book.findByIdAndUpdate(book._id,book)
22 }
23
24
25 module.exports={
26   addBook,
27   getAllBook,
28   updateBook,
29   deleteBookById,
```

La méthode "addBook" ajoute un livre à la base de données en utilisant les données passées en paramètre et renvoie une promesse qui résout avec le résultat de la création du livre.

La méthode "getAllBook" récupère tous les livres de la base de données et renvoie une promesse qui résout avec une liste de livres.

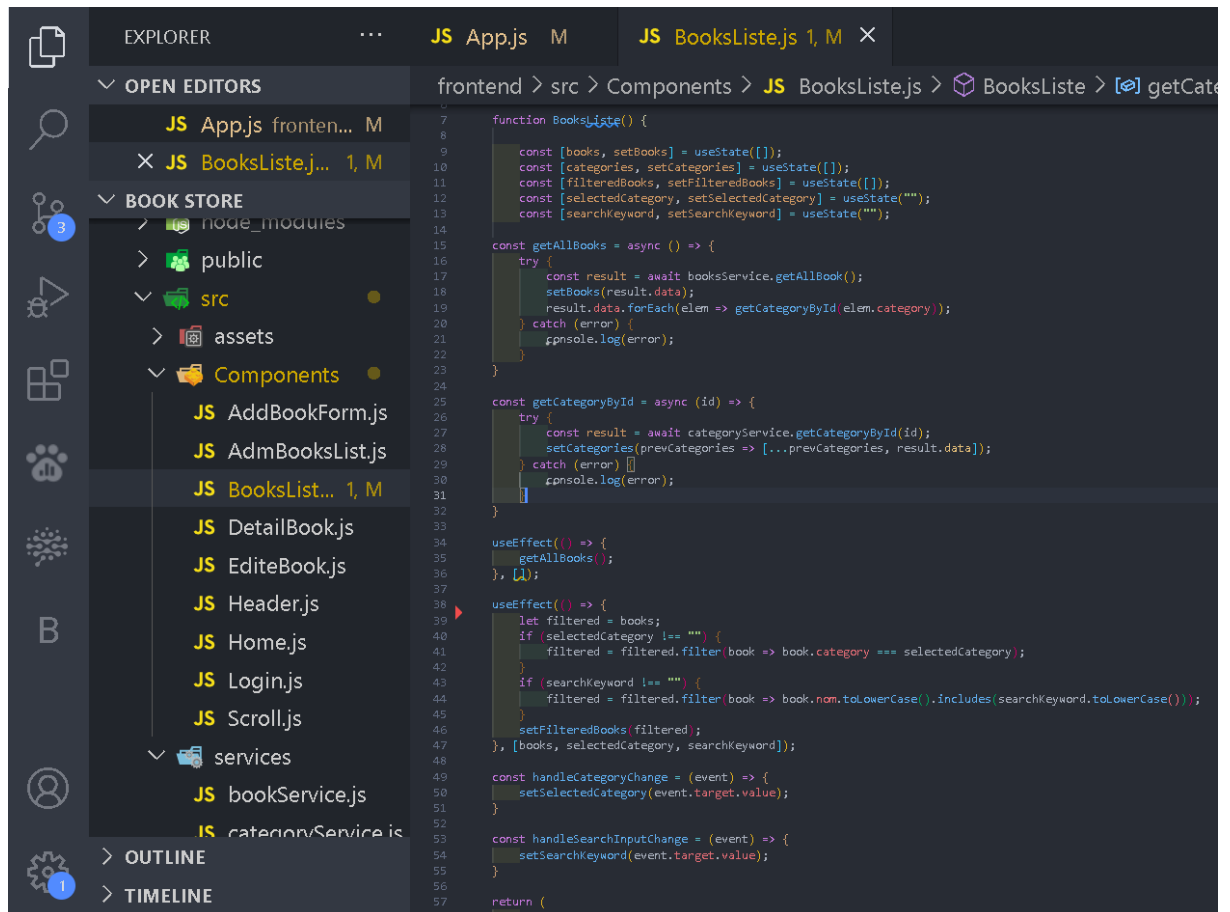
La méthode "getBookById" récupère un livre de la base de données en utilisant l'identifiant fourni en paramètre et renvoie une promesse qui résout avec le livre correspondant.

La méthode "deleteBookById" supprime un livre de la base de données en utilisant l'identifiant fourni en paramètre et renvoie une promesse qui résout avec le résultat de la suppression du livre.

La méthode "updateBook" met à jour les informations d'un livre dans la base de données en utilisant les données passées en paramètre et renvoie une promesse qui résout avec le résultat de la mise à jour. Cette méthode utilise l'identifiant du livre pour trouver le livre à mettre à jour dans la base de données.

2.2 Frontend

2.2.1 Book liste component



La fonction "*getAllBooks*" récupère tous les livres de la base de données en utilisant le service "*booksService*" et met à jour l'état des livres ainsi que les catégories correspondantes en utilisant la fonction "*getCategoryById*".

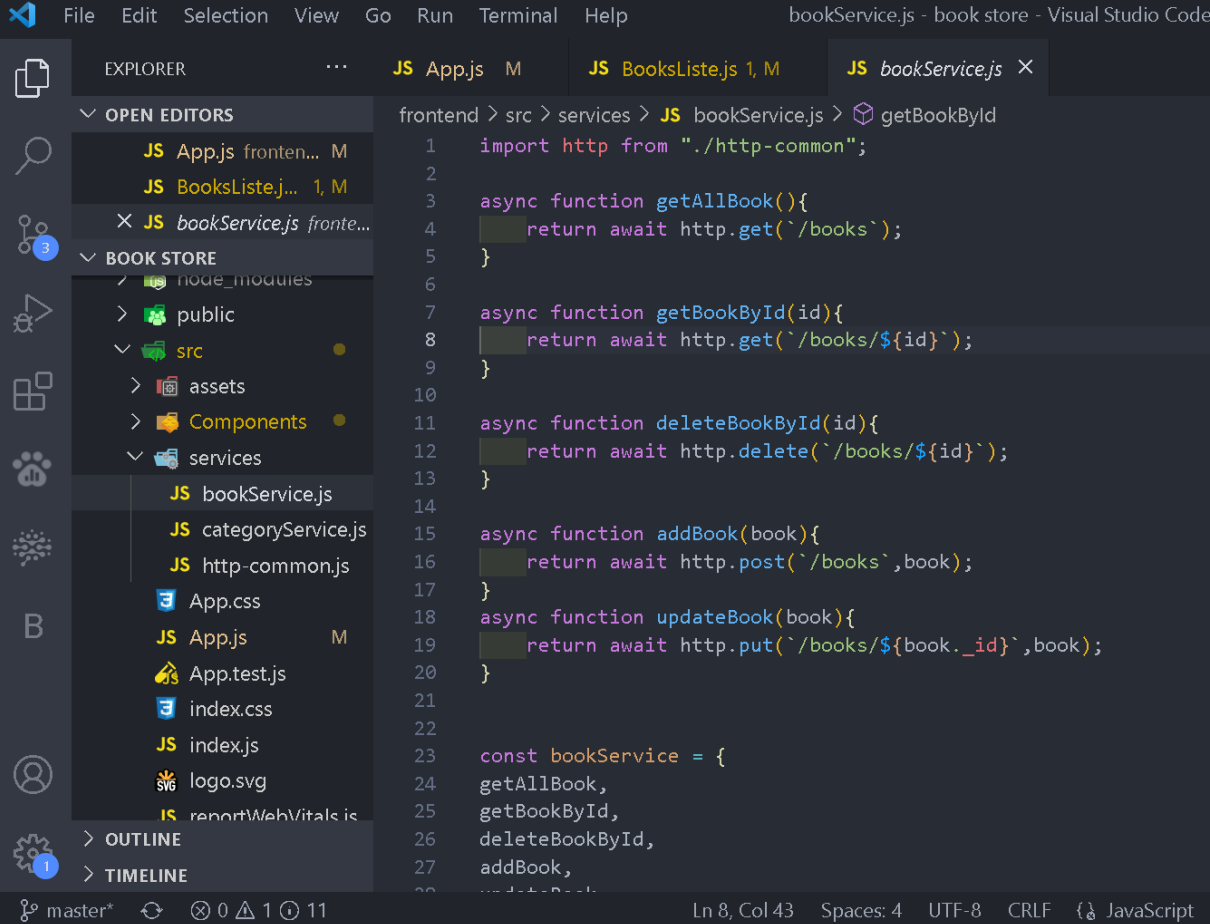
La fonction "*getCategoryById*" récupère les informations d'une catégorie spécifique en utilisant l'identifiant fourni en paramètre et met à jour l'état des catégories en ajoutant la catégorie récupérée.

La fonction "*handleCategoryChange*" est appelée chaque fois que l'utilisateur sélectionne une catégorie dans une liste déroulante, ce qui met à jour l'état de la catégorie sélectionnée.

La fonction "*handleSearchInputChange*" est appelée chaque fois que l'utilisateur saisit un mot-clé de recherche dans un champ de recherche, ce qui met à jour l'état du mot-clé de recherche.

La liste de livres affichée est filtrée en fonction de la catégorie sélectionnée et du mot-clé de recherche en utilisant la fonction "*filter*".

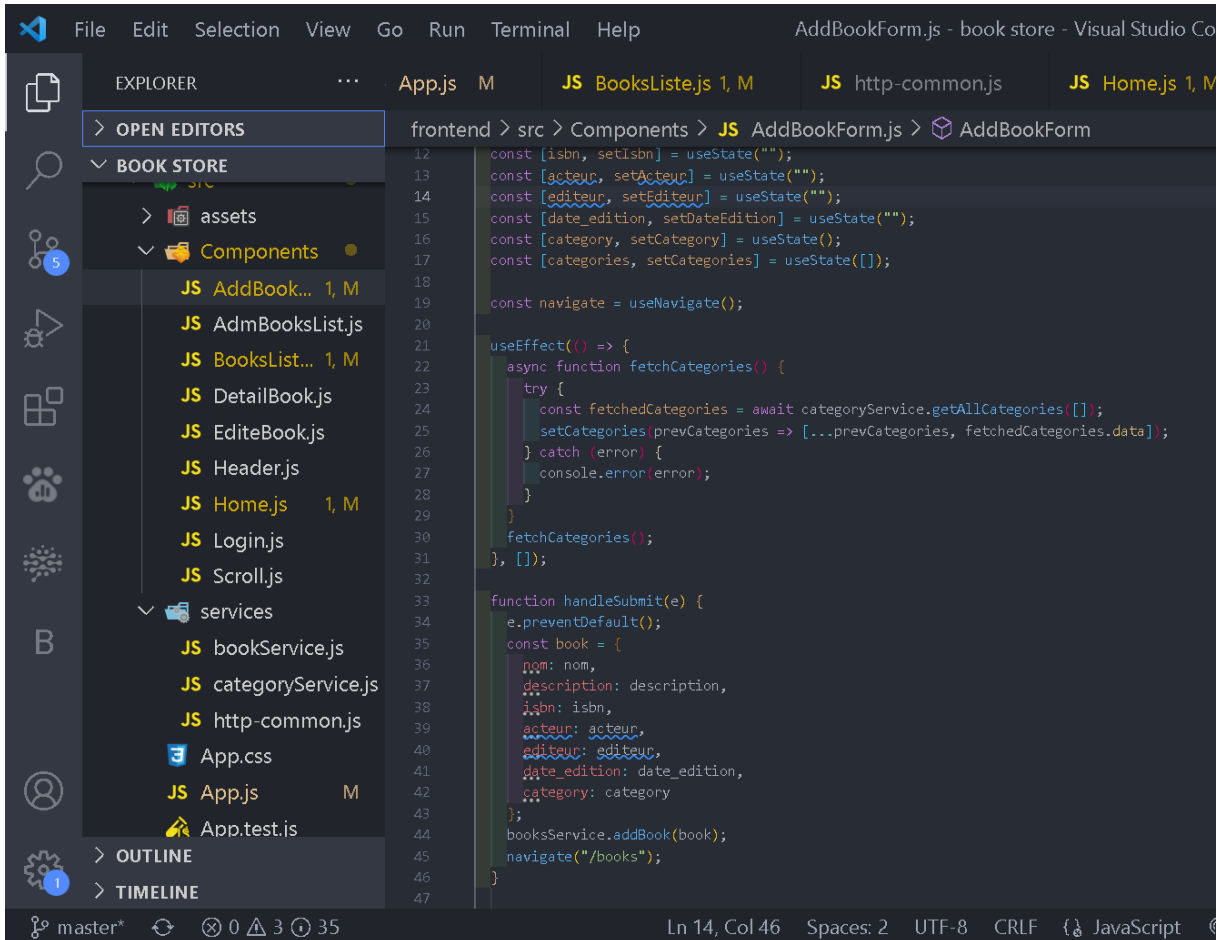
2.2.2 Book services



```
1 import http from "../http-common";
2
3 async function getAllBook(){
4   return await http.get(`/books`);
5 }
6
7 async function getBookById(id){
8   return await http.get(`/books/${id}`);
9 }
10
11 async function deleteBookById(id){
12   return await http.delete(`/books/${id}`);
13 }
14
15 async function addBook(book){
16   return await http.post(`/books`,book);
17 }
18 async function updateBook(book){
19   return await http.put(`/books/${book._id}`,book);
20 }
21
22
23 const bookService = {
24   getAllBook,
25   getBookById,
26   deleteBookById,
27   addBook,
28   updateBook
29 }
```

Ce code définit un service pour interagir avec une API de gestion de livres. Il utilise le module **axios** pour envoyer des requêtes HTTP à l'API via un objet **http** qui est créé à partir du module **http-common**. Les fonctions **getAllBook()**, **getBookById(id)**, **deleteBookById(id)**, **addBook(book)** et **updateBook(book)** sont des méthodes qui permettent d'effectuer des opérations de lecture, écriture, mise à jour et suppression de données de livres via l'API en utilisant les verbes HTTP GET, POST, PUT et DELETE. Ces méthodes sont regroupées dans un objet **bookService** qui est exporté pour être utilisé dans d'autres parties de l'application.

2.2.3 Add Book component



```
12 const [isbn, setIsbn] = useState("");
13 const [acteur, setActeur] = useState("");
14 const [editeur, setEditeur] = useState("");
15 const [date_edition, setDateEdition] = useState("");
16 const [category, setCategory] = useState("");
17 const [categories, setCategories] = useState([]);
18
19 const navigate = useNavigate();
20
21 useEffect(() => {
22   async function fetchCategories() {
23     try {
24       const fetchedCategories = await categoryService.getAllCategories([]);
25       setCategories(prevCategories => [...prevCategories, fetchedCategories.data]);
26     } catch (error) {
27       console.error(error);
28     }
29   }
30   fetchCategories();
31 }, []);
32
33 function handleSubmit(e) {
34   e.preventDefault();
35   const book = {
36     nom: nom,
37     description: description,
38     isbn: isbn,
39     acteur: acteur,
40     editeur: editeur,
41     date_edition: date_edition,
42     category: category
43   };
44   booksService.addBook(book);
45   navigate("/books");
46 }
47
```

"AddBookForm", affiche un formulaire pour ajouter un livre. La fonction utilise des hooks de l'état (useState) pour gérer les différentes entrées du formulaire, et utilise également le hook useEffect pour récupérer les catégories à partir d'un service, lors du chargement initial de la page.

Lorsque l'utilisateur soumet le formulaire, les détails du livre sont envoyés à un service "booksService" pour être ajoutés à la liste des livres, et l'utilisateur est redirigé vers la liste des livres.

3 Présentation des interfaces de l'application

Dans ce qui suit, nous allons présenter les interfaces de notre application.

3.1 Interface d'accueil

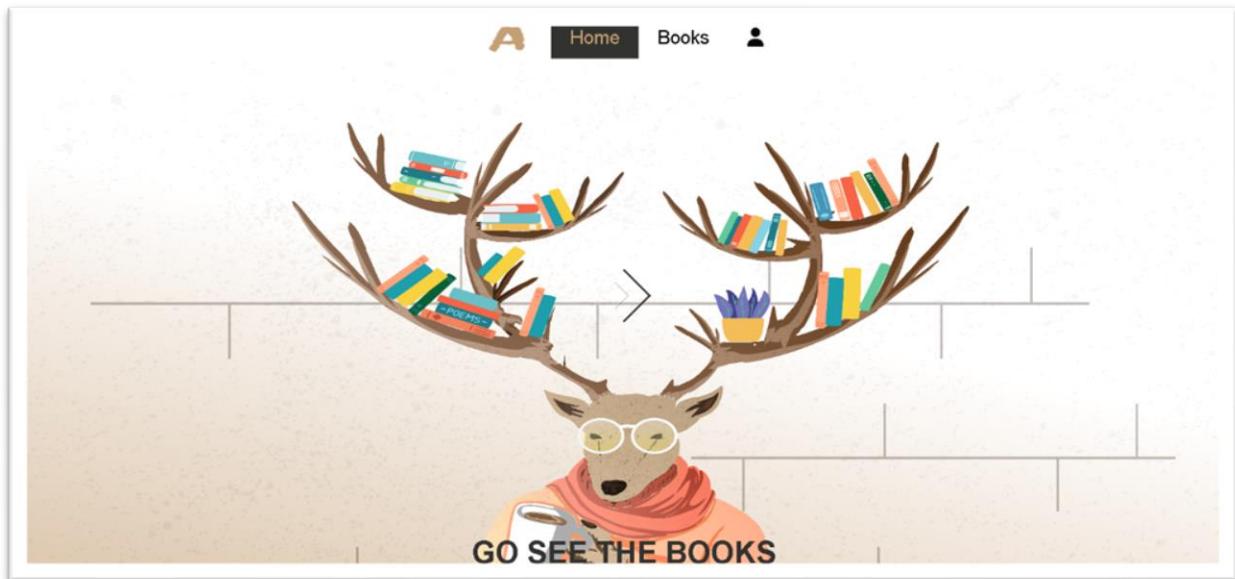


Figure 3.1 interface d'accueil

3.2 Page la liste des livres

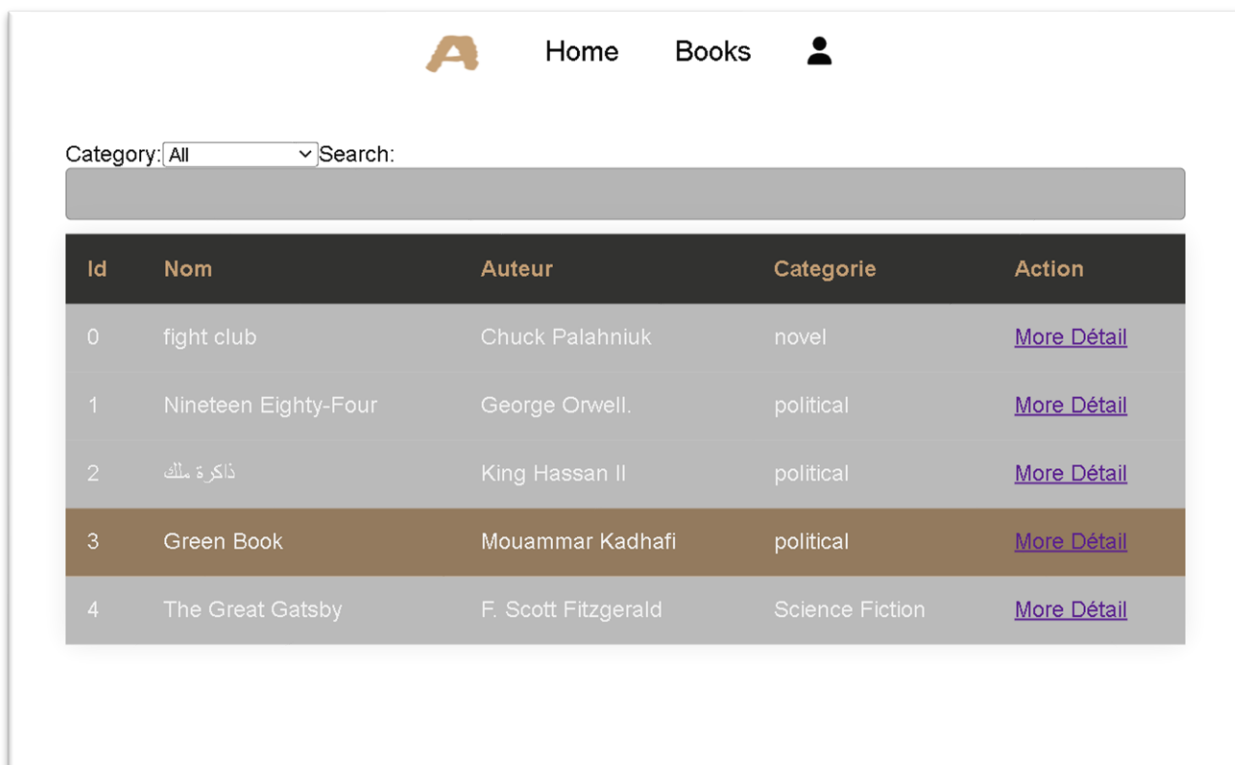


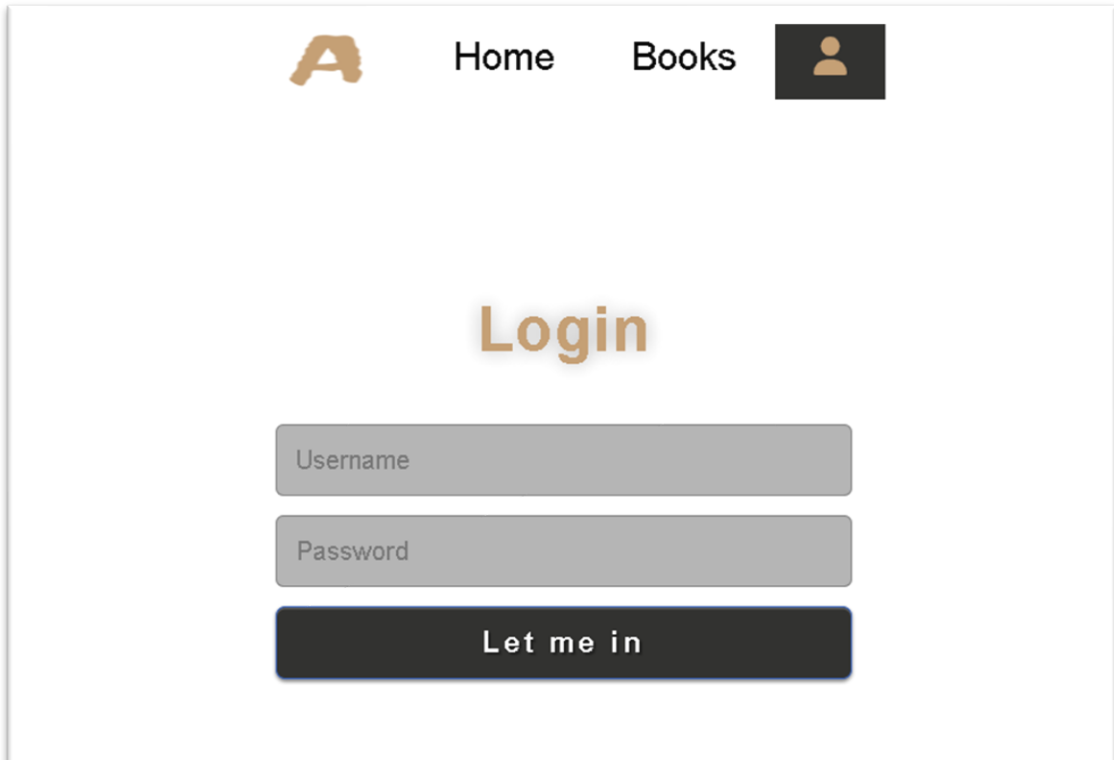
Figure 3.2 liste des livres

3.3 Page détail d'un livre



Figure 3.3 détail d'un livre

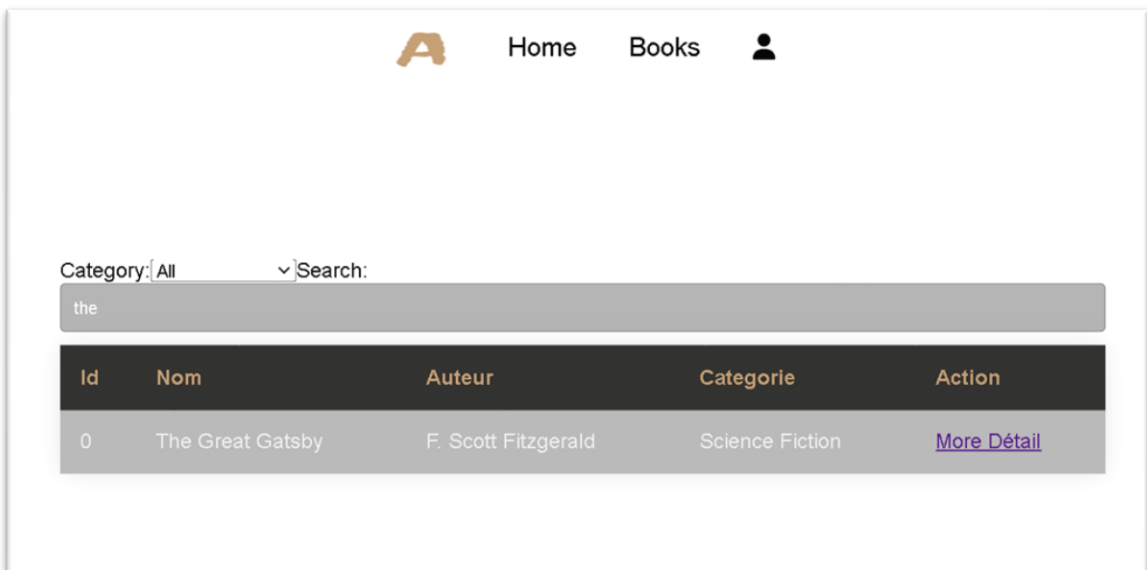
3.4 Interface d'authentification



The login interface features a top navigation bar with a stylized 'A' logo, 'Home', 'Books', and a user icon. The main heading is 'Login'. Below it are two input fields: 'Username' and 'Password'. A dark button labeled 'Let me in' is positioned at the bottom.

Figure 3.4 interface d'authentification

3.5 Filtrer les livres par mot clé





The search interface includes a top navigation bar with a stylized 'A' logo, 'Home', 'Books', and a user icon. Below the navigation bar, there is a 'Category:' dropdown menu set to 'All' and a 'Search:' input field containing the text 'the'. Below the search field is a table displaying search results.

Id	Nom	Auteur	Categorie	Action
0	The Great Gatsby	F. Scott Fitzgerald	Science Fiction	More Détail

Figure 3.5 recherche par mot clé

3.6 Filtrer les livres par catégorie



 Home Books 

Category: Search:

Id	Nom	Auteur	Categorie	Action
0	fight club	Chuck Palahniuk	novel	More Détail

Figure 3.6 recherche par catégorie

3.7 Interface de Responsable modifier ou supprimer un livre



 Home Books 

Category: Search:

Id	Nom	Auteur	Categorie	Action
0	fight club	Chuck Palahniuk	novel	Supp Edit
1	Nineteen Eighty-Four	George Orwell.	political	Supp Edit
2	ذاكرة ملك	King Hassan II	political	Supp Edit
3	The Great Gatsby	F. Scott Fitzgerald	Science Fiction	Supp Edit
4	Green Book	Mouammar Kadhafi	political	Supp Edit

Figure 3.7 modifier ou supprimer un livre

3.8 modifier un livre

 Home Books 

Titre

modifier le titre

Description

modifier la description

ISBN

modifier ISBN


Acteur

modifier acteur nom


Editeur

modifier editeur nom

Date Edition

mm/dd/yyyy 



Category

selectioner category 

Edite

Figure 3.8 modifier un livre

3.9 Interface de Responsable ajouter un livre

 Home Books 

Titre

Enter le titre

Description

Enter la description

ISBN

Enter ISBN


Acteur

Enter acteur nom


Editeur

Enter editeur nom

Date Edition

mm/dd/yyyy 

Category

Select a category 

Add

Figure 3.9 ajouter un livre

Conclusion

En somme, l'application web de gestion de book store développée avec la pile MERN répond de manière efficace et conviviale aux besoins de gestion des responsables et de visualisation des visiteurs. Elle permet d'avoir une vue globale de la situation des livres, d'alléger les traitements manuels des responsables et de faciliter le contrôle de store. L'utilisation de technologies modernes a également permis de créer une application robuste et évolutive.