

ÉCOLE MAROCAINE DES SCIENCES DE L'INGÉNIEUR

3ème année IIR

Annee universitaire 2025-2026

Systeme de Gestion Bibliothèque :

Rapport présenté par :

ilyasse elkaui - reda harchid - amine rahmouni

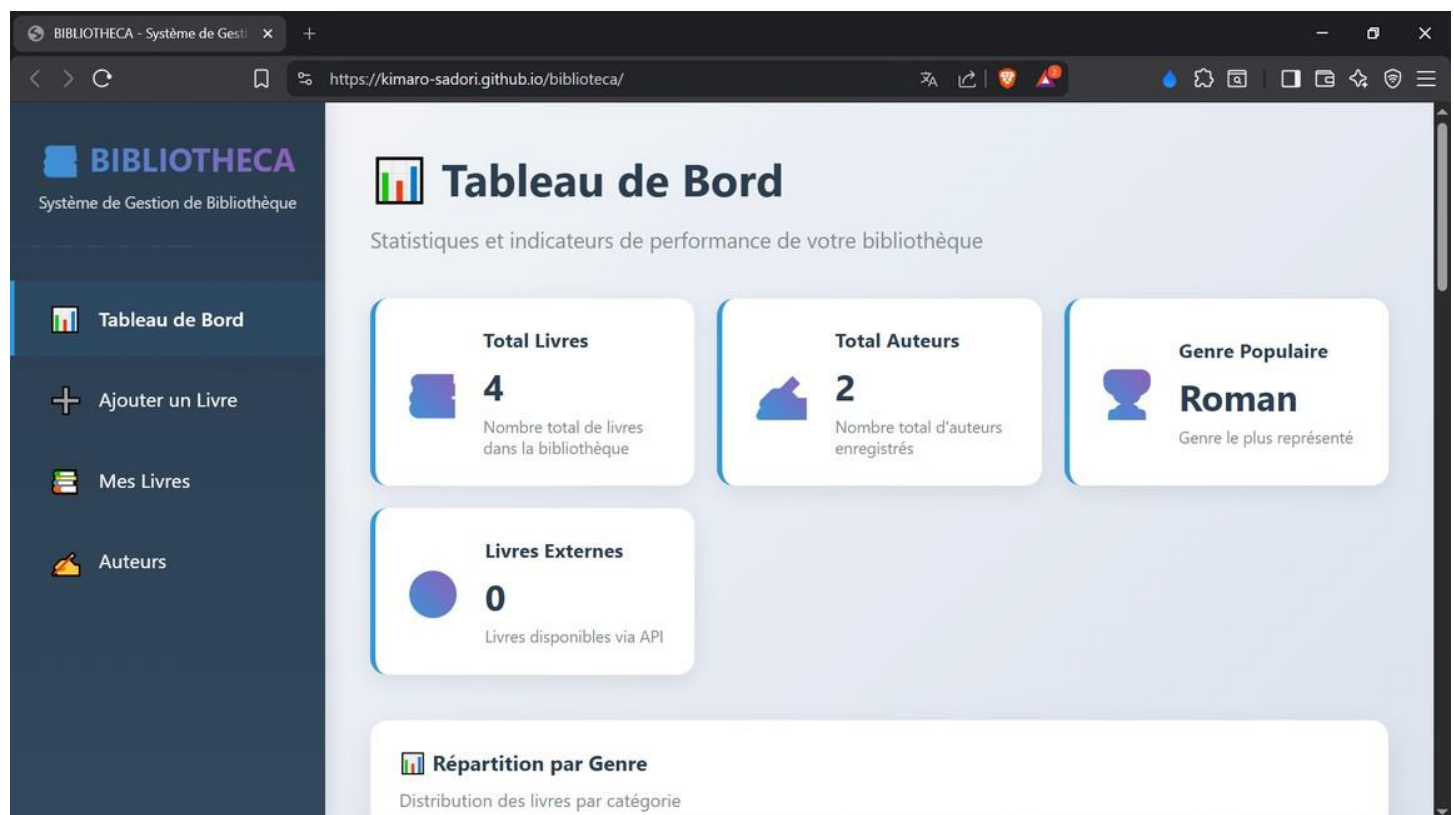
Sommaire :

1. Introduction générale
2. Objectifs du projet
3. Présentation générale de l'application
4. Technologies utilisées
5. Architecture générale du système
6. Structure de l'interface utilisateur
7. Tableau de bord et indicateurs de performance
8. Gestion des livres
9. Stockage des données
10. Recherche et tri des livres
11. Gestion des auteurs
12. Statistiques et visualisation des données
13. Intégration d'une API externe
14. Ajout de livres depuis l'API
15. Initialisation et fonctionnement global
16. Limites et contraintes du projet
17. Conclusion générale

1. Introduction générale

Le projet **BIBLIOTHECA** est une application web conçue pour gérer une bibliothèque de manière simple, moderne et efficace. Elle permet à un utilisateur de gérer une collection de livres, d'enregistrer des auteurs, de visualiser des statistiques, et d'enrichir sa bibliothèque grâce à une API externe.

Ce projet a été réalisé dans le cadre d'un examen, avec pour objectif principal de démontrer la maîtrise des technologies web fondamentales : HTML, CSS, JavaScript, ainsi que l'utilisation d'une API REST.



2. Objectifs du projet

Les objectifs principaux de BIBLIOTHECA sont :

- Créer une **application web dynamique** sans rechargement de page
- Implémenter un **CRUD complet** (Créer, Lire, Modifier, Supprimer)
- Gérer des données localement avec **LocalStorage**
- Afficher des **statistiques visuelles** à l'aide de graphiques
- Intégrer une **API externe** pour récupérer des livres

Offrir une interface claire et responsive

3. Technologies utilisées

3.1 HTML5

Utilisé pour structurer le contenu de l'application.

3.2 CSS3

Utilisé pour la mise en forme, la disposition (layout), et l'ergonomie.

3.3 JavaScript

Cœur logique du projet :

- gestion des événements
- manipulation du DOM
- stockage local
- communication avec une API

3.4 Chart.js

Bibliothèque JavaScript utilisée pour créer des graphiques interactifs.

3.5 OpenLibrary API

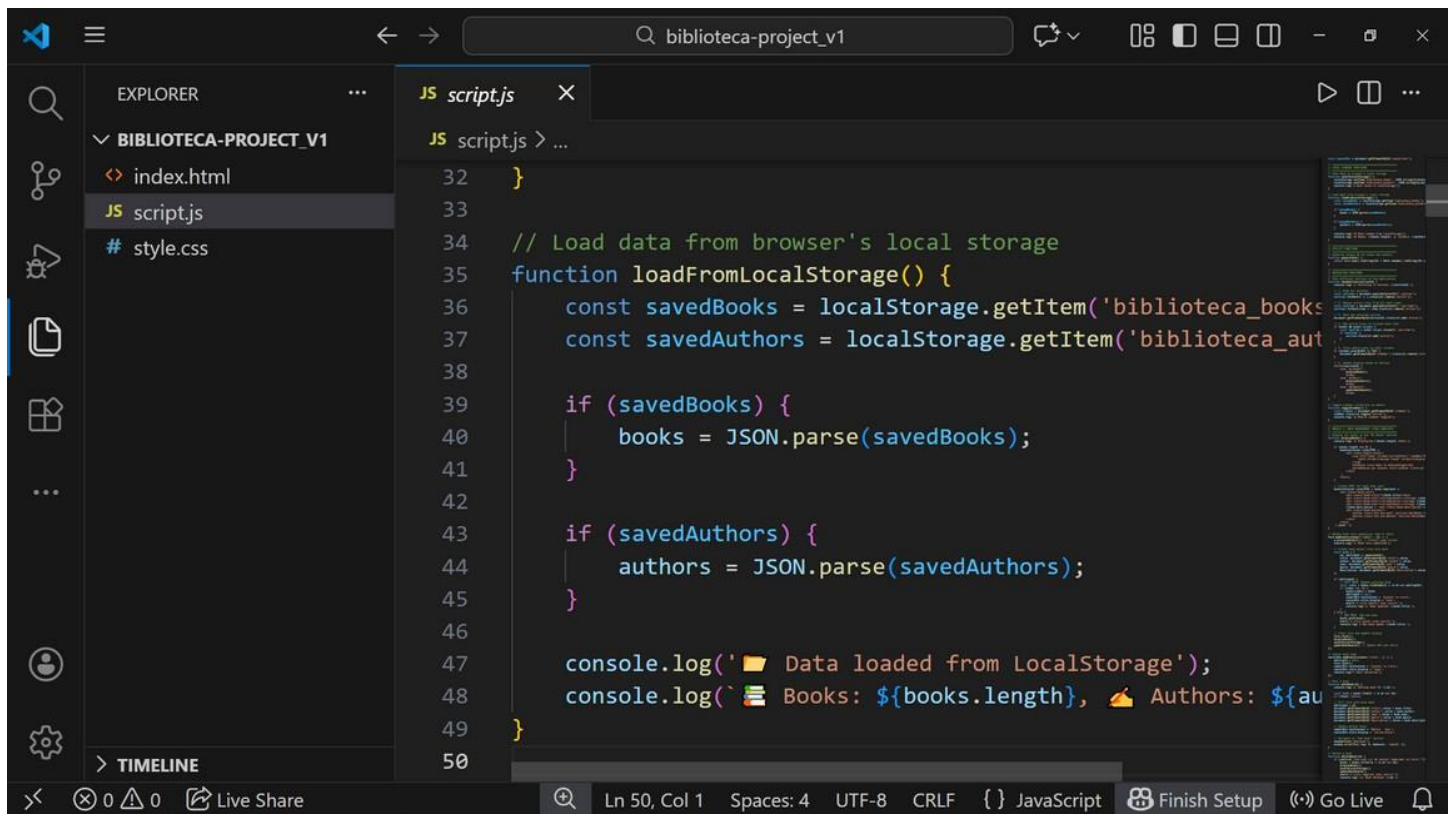
API publique permettant de rechercher des livres en ligne.

4. Architecture générale du projet

Le projet est composé de **trois fichiers principaux** :

- index.html → structure de l'interface
- style.css → apparence visuelle script.js →
- logique et fonctionnalités

Cette séparation respecte le principe de **séparation des responsabilités**.



The screenshot shows the Visual Studio Code interface with the 'BIBLIOTECA-PROJECT_V1' project open. The Explorer sidebar on the left lists 'index.html', 'script.js', and 'style.css'. The main editor window displays the 'script.js' file, which contains a function 'loadFromLocalStorage()' that retrieves data from browser's local storage and logs it to the console.

```
32 }
33
34 // Load data from browser's local storage
35 function loadFromLocalStorage() {
36     const savedBooks = localStorage.getItem('biblioteca_books');
37     const savedAuthors = localStorage.getItem('biblioteca_authors');
38
39     if (savedBooks) {
40         books = JSON.parse(savedBooks);
41     }
42
43     if (savedAuthors) {
44         authors = JSON.parse(savedAuthors);
45     }
46
47     console.log('📁 Data loaded from LocalStorage');
48     console.log(`📖 Books: ${books.length}, 👤 Authors: ${authors.length}`);
49 }
50
```

5. Structure de l'interface (HTML)

Le fichier index.html définit une **Single Page Application (SPA)**.

5.1 Barre latérale (Sidebar)

La barre latérale contient :

- Tableau de bord
- Ajouter un livre
- Mes livres
- Auteurs

Elle permet la navigation **sans recharger la page**.

5.2 Sections principales

Chaque partie de l'application est une <section> :

- dashboard add-
- book my-books
- authors
-

Une seule section est visible à la fois grâce à JavaScript.

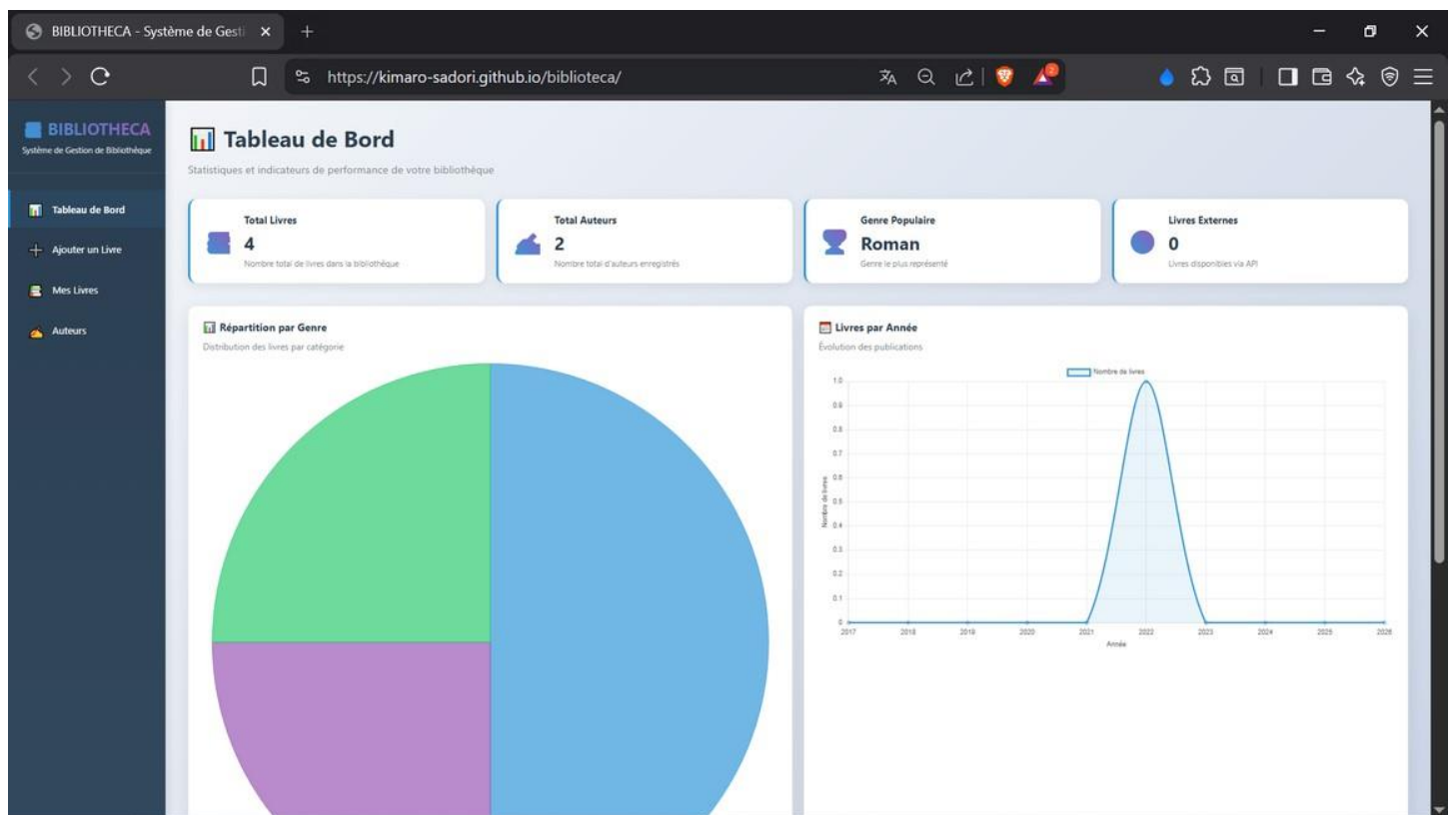
6. *Tableau de bord (Dashboard)*

Le tableau de bord affiche des **indicateurs clés (KPI)** :

- Nombre total de livres
- Nombre total d'auteurs Genre le
- plus populaire Nombre de livres
- trouvés via l'API

Il contient également deux graphiques :

- Répartition des livres par genre
- Évolution des livres par année



7. Gestion des livres (Module 1)

7.1 Ajout d'un livre

Un formulaire permet de saisir :

- Titre
- Auteur
- Année
- Genre
- Description

Lors de la soumission :

- les données sont validées un objet
- JavaScript est créé le livre est
- ajouté au tableau books

7.2 Modification d'un livre

Lorsqu'on clique sur **Modifier** : les données du livre sont

- chargées dans le formulaire l'utilisateur peut les
- mettre à jour

7.3 Suppression d'un livre

Un message de confirmation empêche toute suppression accidentelle.

8. *Stockage local (LocalStorage)*

8.1 Pourquoi LocalStorage ?

Le LocalStorage permet de :

- conserver les données même après fermeture du navigateur
- éviter l'utilisation d'une base de données serveur

8.2 Fonctionnement

Les données sont sauvegardées sous forme de **JSON** :

- biblioteca_books
- biblioteca_authors

Lors du chargement de la page, les données sont relues automatiquement.

9. Recherche et tri des livres

9.1 Recherche

La recherche fonctionne en temps réel :

- par titre
- par auteur

Elle utilise filter() et includes().

9.2 Tri

Le tri permet d'organiser les livres par :

- titre
- auteur
- année

Cela améliore l'expérience utilisateur.

10. Gestion des auteurs (Module 2)

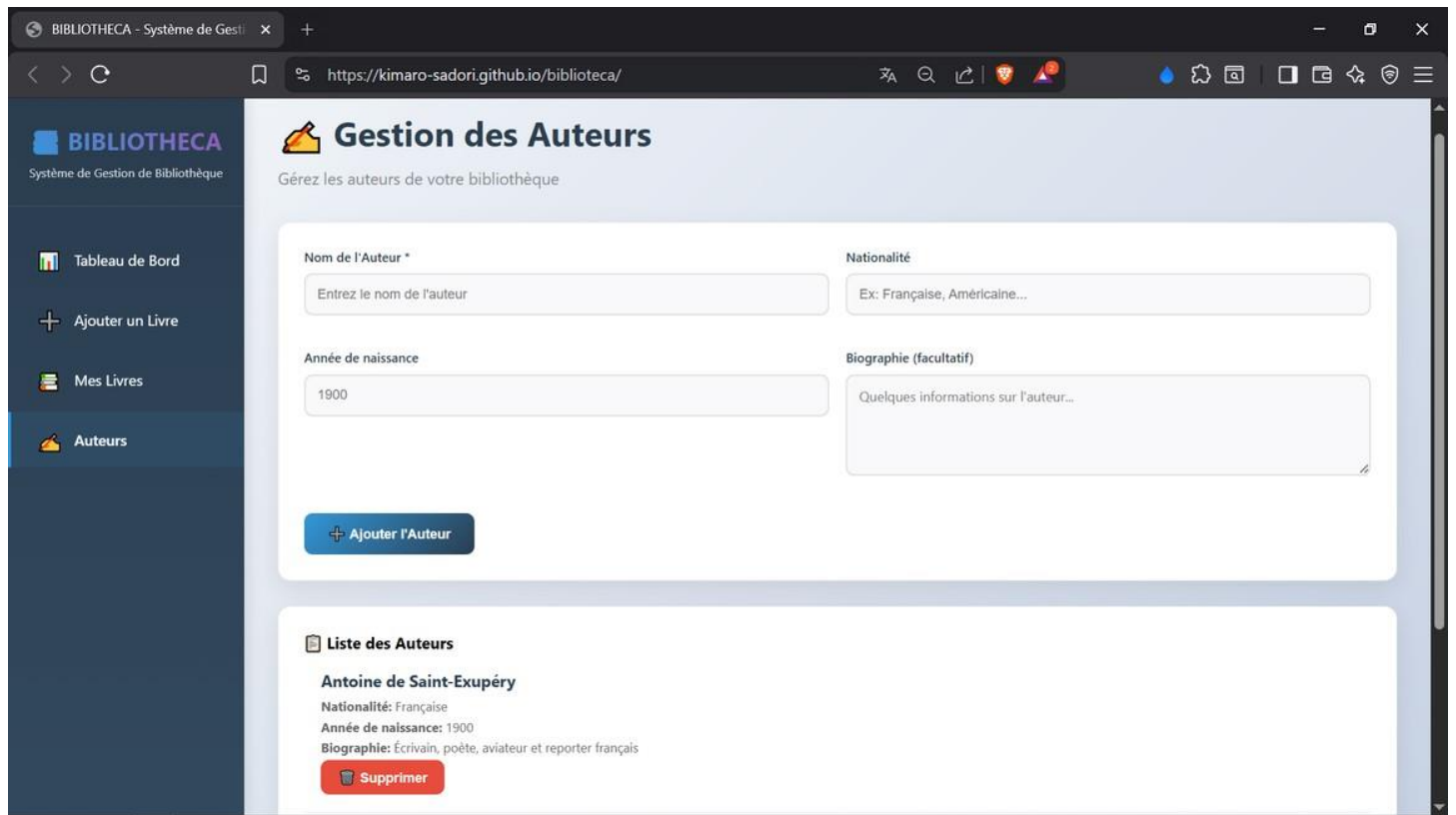
Chaque auteur possède :

- nom
- nationalité année
- de naissance
- biographie

Les auteurs peuvent être :

- ajoutés
- affichés
- supprimés

Le nombre total d'auteurs est automatiquement mis à jour dans le tableau de bord.



11. Graphiques et statistiques (Chart.js)

11.1 Graphique par genre

Un graphique circulaire (pie chart) montre :

- la répartition des livres par genre

11.2 Graphique par année

Un graphique linéaire montre :

- l'évolution du nombre de livres sur les 10 dernières années

Les graphiques sont **dynamiquement mis à jour** après chaque modification.

12. Intégration de l'API OpenLibrary

12.1 Qu'est-ce qu'une API ?

Une API (Application Programming Interface) permet à une application de communiquer avec une autre application via Internet.

12.2 Pourquoi OpenLibrary ?

OpenLibrary est une base de données publique contenant des millions de livres.

13. Fonctionnement de fetch (explication simple)

La fonction fetch permet d'envoyer une requête HTTP.

Étapes :

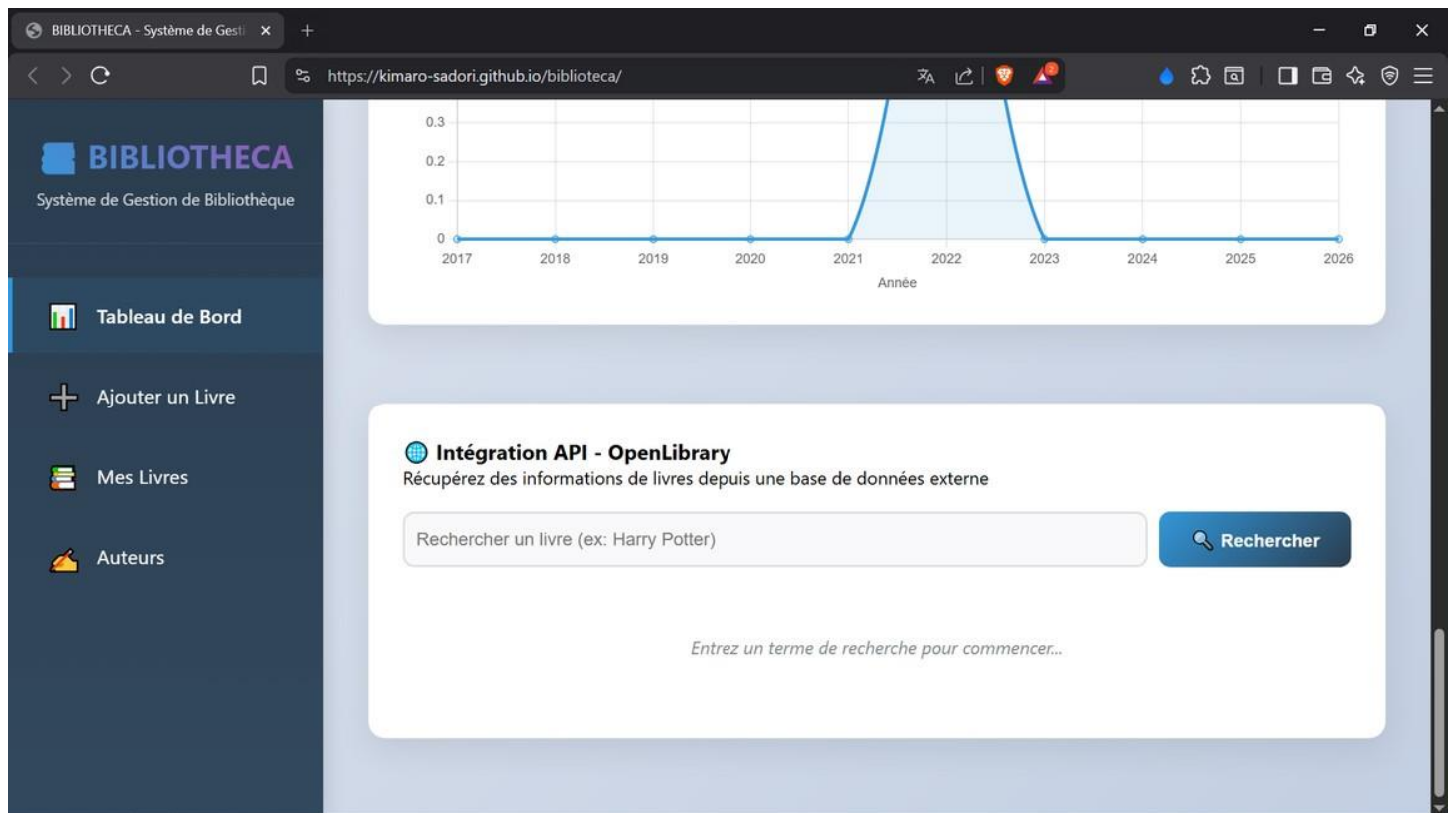
1. L'utilisateur saisit un mot-clé
2. fetch envoie une requête à OpenLibrary
3. Le serveur répond avec des données JSON
4. JavaScript traite ces données
5. Les résultats sont affichés à l'écran Cela se fait **sans recharger la page**.

14. Ajout d'un livre depuis l'API

Lorsqu'un utilisateur clique sur **Ajouter à ma bibliothèque** :

- un livre est créé à partir des données
- API il est ajouté au tableau books il est
- sauvegardé dans le LocalStorage le
- tableau de bord est mis à jour

Cela montre une **interaction complète entre API externe et application locale**.



BIBLIOTHECA - Système de Gestion de Bibliothèque

https://kimaro-sadori.github.io/biblioteca/

The figure shows the same web application interface as the first image, but with search results displayed. The search input field now contains the word 'happy'. Below the input field, three book results are listed, each with a title, author, year, and an 'Ajouter à ma bibliothèque' button. The first result is 'The Happy Prince and other tales' by Oscar Wilde, published in 1888. The second is 'Where the Wild Things Are' by Maurice Sendak, published in 1963. The third is 'A Wrinkle in Time' by Madeleine L'Engle. The browser's address bar remains the same.

Intégration API - OpenLibrary
Récupérez des informations de livres depuis une base de données externe

happy **Rechercher**

The Happy Prince and other tales
Auteur: Oscar Wilde
Année: 1888
Ajouter à ma bibliothèque

Where the Wild Things Are
Auteur: Maurice Sendak
Année: 1963
Ajouter à ma bibliothèque

A Wrinkle in Time
Auteur: Madeleine L'Engle

15. Initialisation de l'application

Au chargement de la page :

- les données sont chargées depuis le LocalStorage des
 - livres exemples sont ajoutés si la bibliothèque est vide •
- les graphiques et statistiques sont affichés

Cela garantit une application toujours fonctionnelle.

16. Sécurité et limitations

- Pas de base de données serveur
- Pas d'authentification utilisateur
- Dépendance à la connexion Internet pour l'API

Ces limites sont normales pour un projet académique.

17. Corrections et validation des bugs :

Dans cette section, nous présentons les corrections apportées au projet JavaScript suite aux remarques de l'enseignant.

Pour chaque bug corrigé, des preuves visuelles sont fournies afin de garantir la validité des modifications.

Bug n°1 — Problème d'affichage entre la barre de recherche et le sélecteur de tri :

Problème identifié

Les éléments #search-books (champ de recherche) et #sort-books (menu de tri) utilisaient tous les deux la propriété flex: 1.

Cela provoquait un partage égal de l'espace horizontal, ce qui rendait l'interface déséquilibrée et peu ergonomique.

Impact

- Le champ de recherche était trop petit
- Le menu de tri prenait trop d'espace inutilement
- Mauvaise expérience utilisateur (UI non optimisée)

Solution *appliquée*

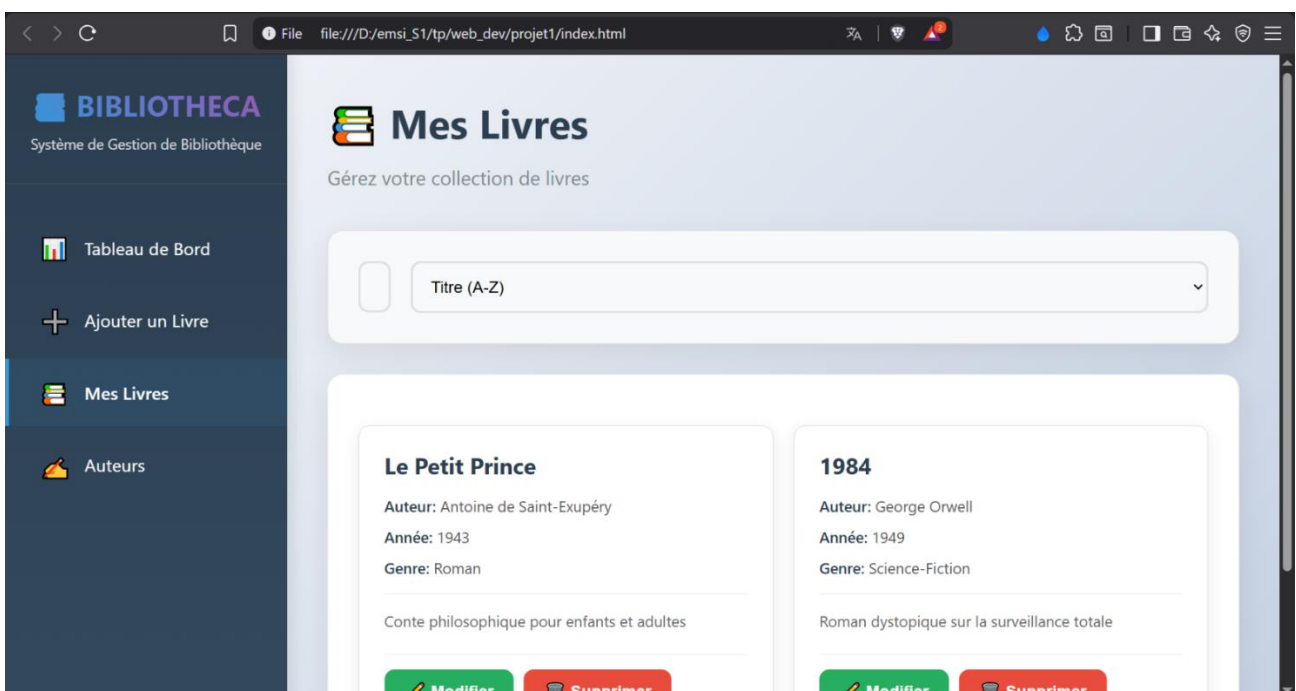
Suppression du partage égal de l'espace et définition de largeurs fixes proportionnelles :

```
#search-books {  
  
  width: 70%;  
  
}  
  
#sort-books {  
  
  width: 30%;  
  
}
```

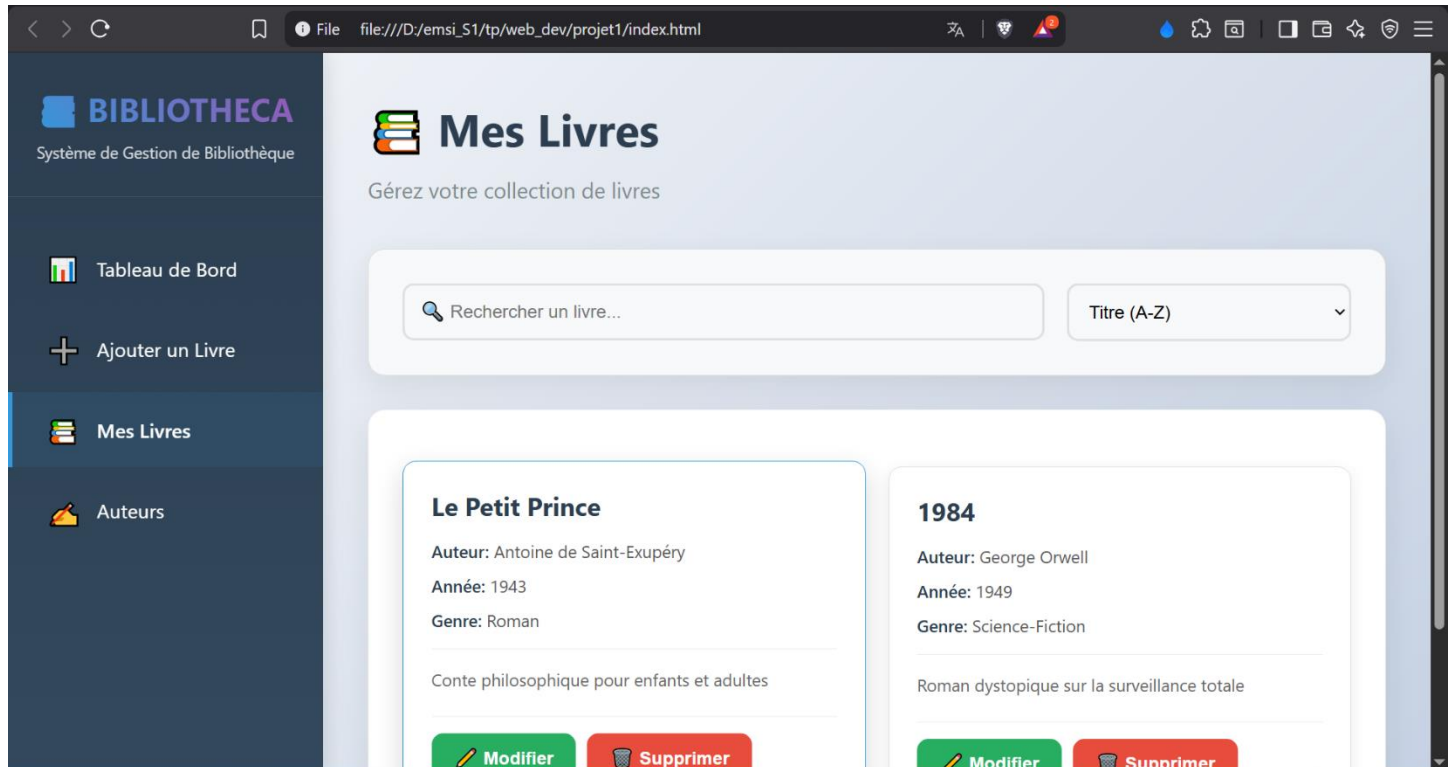
```
css  
Copy Download  
  
#search-books {  
  width: 70%;  
}  
  
#sort-books {  
  width: 30%;  
}
```

Validation visuelle

Avant correction :



Après correction :



Bug n°2 — Sélection des auteurs uniquement depuis la base de données :

Le système permettait à l'utilisateur d'ajouter manuellement un auteur lors de l'ajout d'un livre, ce qui pouvait provoquer :

- **des doublons d'auteurs**
- des incohérences dans la base de données
- une mauvaise normalisation des données
- une perte de cohérence entre livres et auteurs

⚠ Impact

- Données non structurées
- Difficulté de gestion de la base
- Erreurs logiques dans l'application
- Problèmes de fiabilité des statistiques et du dashboard

Solution appliquée

1) Modification HTML

Remplacement du champ input par une liste déroulante (select) alimentée automatiquement depuis la base de données :

```
159 <div class="form-group">
160   <label for="author">Auteur *</label>
161   <!-- Remplacement de l'input par un select -->
162   <select id="author" required>
163     <option value="">Sélectionnez un auteur</option>
164   </select>
165
166 </div>
```

2) Modifications JavaScript

Partie A — Fonction de chargement des auteurs

```
// =====
// FUNCTION TO POPULATE AUTHOR DROPDOWN
// =====
function populateAuthorDropdown() {
  const authorSelect = document.getElementById('author');
  authorSelect.innerHTML = '<option value="">Sélectionnez un auteur</option>';

  authors.forEach(author => {
    const option = document.createElement('option');
    option.value = author.name;
    option.textContent = author.name;
    authorSelect.appendChild(option);
  });
}
```

Partie B — Appel dans la navigation (showSection())

```
case 'add-book':
  populateAuthorDropdown(); // Ajout
  break;
```

Partie C — Mise à jour après ajout d'un auteur

```
updateDashboard();
populateAuthorDropdown(); // Ajout
alert('✅ Auteur ajouté avec succès!');
```


Avant correction :

 **BIBLIOTHECA**

Système de Gestion de Bibliothèque

 Tableau de Bord

 **Ajouter un Livre**

 Mes Livres

 Auteurs

 **Ajouter un Livre**

Remplissez le formulaire pour ajouter un nouveau livre à votre bibliothèque

Titre du Livre *

Entrez le titre du livre

Auteur *

Nom de l'auteur

Année de Publication *

2024


Genre *

Sélectionnez un genre


Description


Décrivez le livre... (facultatif)


Après correction :


 **BIBLIOTHECA**


Système de Gestion de Bibliothèque

 Tableau de Bord

 **Ajouter un Livre**

 Mes Livres

 Auteurs

 **Ajouter un Livre**

Remplissez le formulaire pour ajouter un nouveau livre à votre bibliothèque

Titre du Livre *

Entrez le titre du livre

Auteur *

Sélectionnez un auteur

Sélectionnez un auteur

Antoine de Saint-Exupéry

Année de Publication *

2024

Genre *

Sélectionnez un genre

Description

Décrivez le livre... (facultatif)

Bug n°3 — Ajout du bouton « Voir » :

1. Ajout du bouton « Voir » :

```
142         <div class="book-actions">
143             <button class="btn btn-view" onclick="viewBook('${book.id}')">👁 Voir</button>
144             <button class="btn btn-edit" onclick="editBook('${book.id}')">✎ Modifier</button>
145             <button class="btn btn-delete" onclick="deleteBook('${book.id}')">🗑 Supprimer</button>
146         </div>
147     </div>
148     `).join('');
149 }
```

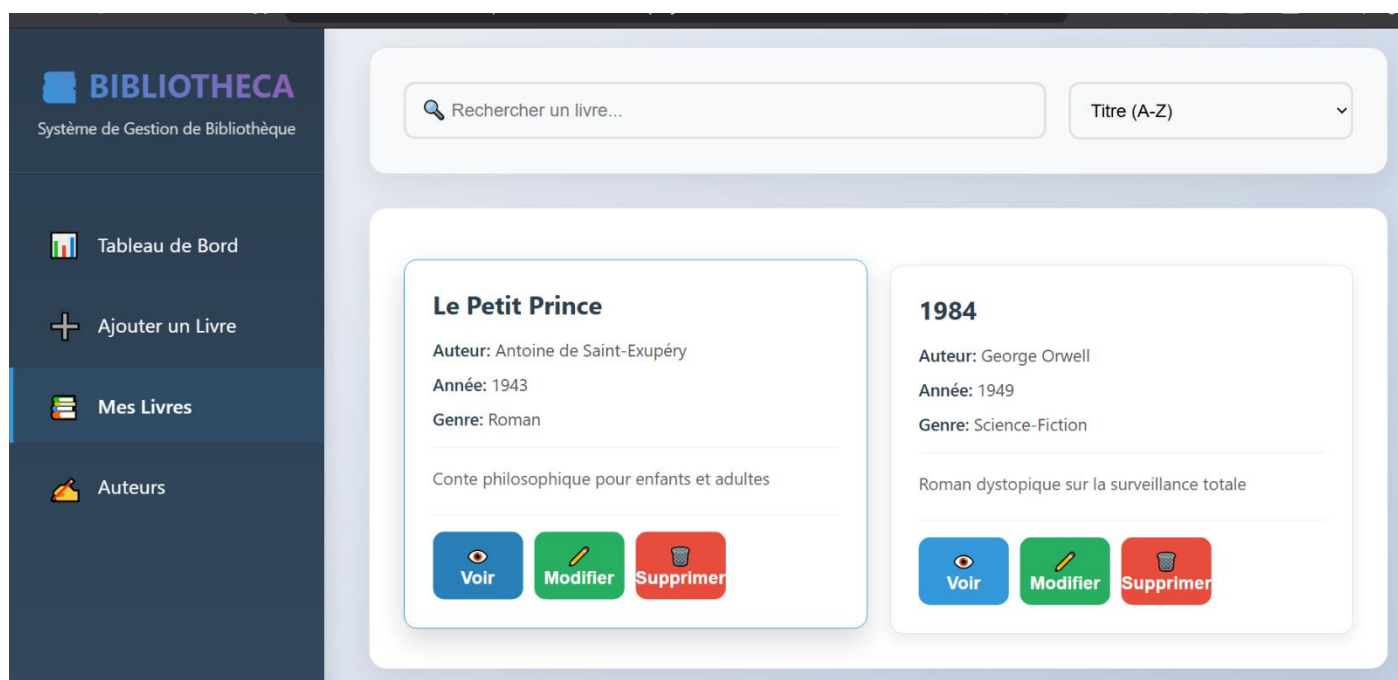
2. Consultation des informations d'un livre :

```
index.html JS script.js X # style.css 1
JS script.js > viewBook
688 function viewBook(id) {
689     console.log(`👁 Viewing book ID: ${id}`);
690
691     const book = books.find(b => b.id === id);
692     if (!book) return;
693
694     // Créer un popup/modal pour afficher les détails
695     const modalHTML = `
696         <div class="modal-overlay">
697             <div class="modal-content">
698                 <h2>👁 Détails du Livre</h2>
699
700                 <div class="book-details">
701                     <p><strong>📖 Titre:</strong> ${book.title}</p>
702                     <p><strong>👤 Auteur:</strong> ${book.author}</p>
703                     <p><strong>📅 Année:</strong> ${book.year}</p>
704                     <p><strong>📖 Genre:</strong> ${book.genre}</p>
705                     ${book.description ? `<p><strong>📖 Description:</strong> ${book.description}</p>` : ''}
706                 </div>
707
708                 <div class="modal-actions">
709                     <button class="btn btn-edit" onclick="editBook('${book.id}'); closeModal()">✎ Modifier</button>
710                     <button class="btn" onclick="closeModal()">>Fermer</button>
711                 </div>
712             </div>
713         </div>
714     `;
715     // Ajouter le modal au HTML
716     document.body.insertAdjacentHTML('beforeend', modalHTML);
717 }
```

3. Gestion de la fermeture de la fenêtre :

```
function closeModal() {
    const modal = document.querySelector('.modal-overlay');
    if (modal) {
        modal.remove();
    }
}
```

Après correction :



Jusqu'à présent, nous avons réalisé :

Le système a été amélioré par l'ajout d'un bouton « **Voir** » permettant d'afficher les détails d'un livre dans une fenêtre modale sans passer en mode modification. Un système de **sélection contrôlée des auteurs** a été mis en place grâce à une liste déroulante automatiquement synchronisée avec la base de données. L'interface a également été optimisée pour charger dynamiquement les données lors de la navigation entre les sections. La **barre de recherche** a été améliorée afin d'offrir un meilleur affichage et une meilleure ergonomie sur écran PC (desktop).

Bug n°4 — Ajout ajouter un KPI :

1. Ajouter un nouveau KPI dans le dashboard (HTML) :

Dans la section dashboard, ajoutez ce nouveau KPI après le 4ème KPI :

```

106
107 <!-- KPI 5: API Authors -->
108 <div class="kpi-card">
109   <div class="kpi-icon"><img alt="API icon" data-bbox="245 68 265 80"/></div>
110   <div class="kpi-content">
111     <h3>Auteurs API</h3>
112     <div class="kpi-value" id="api-authors-kpi">0</div>
113     <p class="kpi-desc">Auteurs différents disponibles via API</p>
114   </div>
115 </div>
116 </div>

```

2. Mettre à jour la fonction searchExternalBooks() pour extraire les auteurs :

La fonction de recherche externe a été améliorée par l'augmentation du nombre de résultats récupérés depuis l'API, ainsi que par l'extraction automatique des auteurs uniques.

Un nouvel indicateur (KPI) affichant le nombre d'auteurs distincts a été ajouté afin d'enrichir l'analyse des données.

L'interface des résultats a été restructurée pour séparer l'affichage des livres et des auteurs.

Ces améliorations permettent une recherche plus intelligente, plus lisible et plus exploitable sur le plan fonctionnel et pédagogique.

3. Ajouter du CSS pour le nouveau KPI et l'affichage des auteurs :

La mise en page a été conçue pour organiser clairement les indicateurs (KPI) et les résultats de recherche de manière structurée et lisible.

L'interface sépare visuellement les sections « livres » et « auteurs » afin de faciliter la compréhension des informations affichées.

Le design s'adapte automatiquement aux différents écrans (PC, tablette, mobile) pour garantir une bonne lisibilité et une navigation fluide.

4. Pour la présentation, ajoutez des données de démonstration :

Dans la fonction init(), ajoutez un KPI initial pour l'API :

```
// Initialiser les KPI API
document.getElementById('api-authors-kpi').textContent = '0';
document.getElementById('api-books-kpi').textContent = '0';
```

La fonction `init()` est la fonction de démarrage qui s'exécute automatiquement quand la page web se charge. Elle joue le rôle de "moteur de démarrage" de l'application : elle charge les données sauvegardées dans le navigateur (livres et auteurs), affiche ces données à l'écran, prépare le tableau de bord avec des valeurs initiales, et ajoute quelques livres d'exemple si la bibliothèque est vide. En bref, `init()` prépare tout pour que l'application soit immédiatement utilisable dès son ouverture.

Logique informatique :

txt Copy code

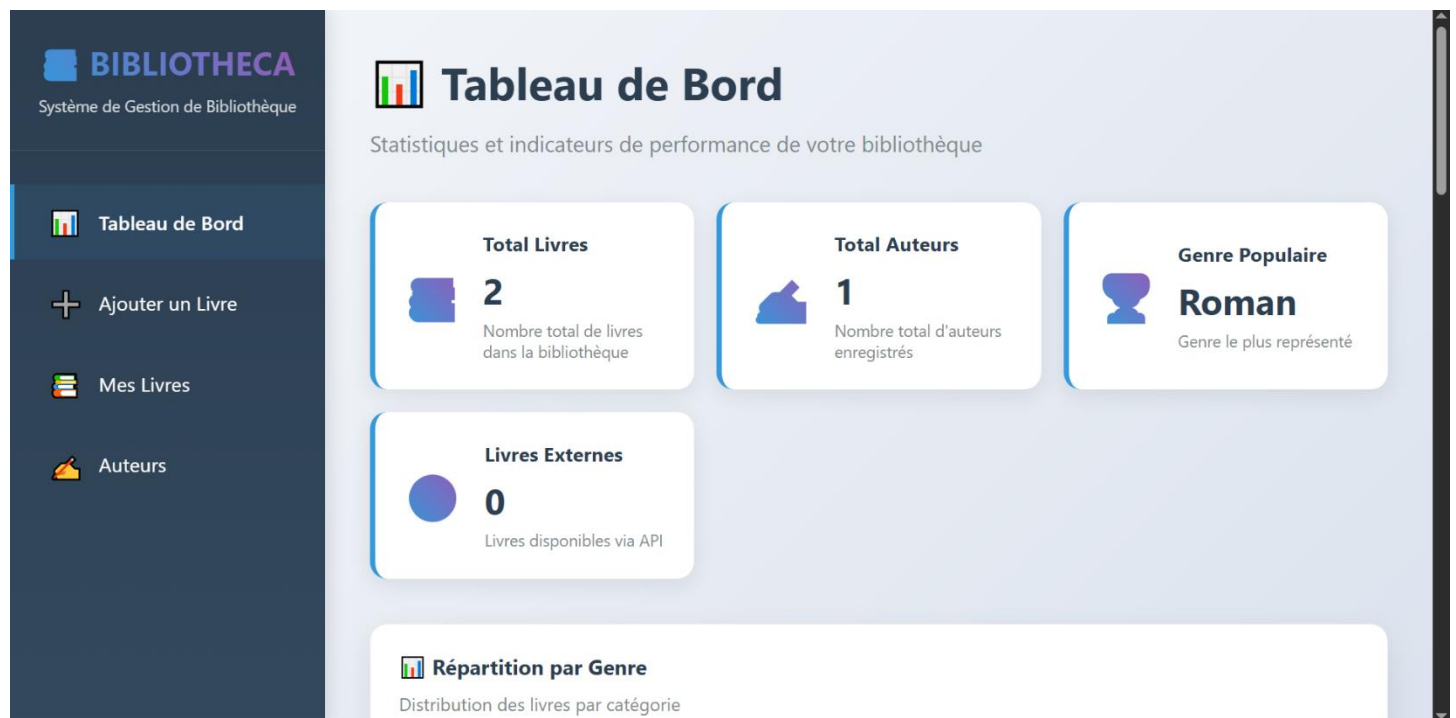
AVANT :

API → livres → affichage

APRÈS :

API → livres + auteurs → analyse → KPI → affichage structuré

Avant mise à jour :



Après mise à jour :



18. Version propre à insérer avant la conclusion :

De nombreux projets académiques sont réalisés uniquement dans un cadre pédagogique, sans application réelle dans le monde professionnel.

Une approche plus efficace consisterait à développer des projets basés sur des problématiques réelles, des besoins concrets du marché ou des idées pouvant avoir une valeur économique ou sociale.

Les projets les plus formateurs sont ceux qui permettent un contact direct avec des entreprises réelles, des utilisateurs réels ou des partenaires professionnels. Cette approche favorise l'apprentissage pratique, l'innovation et la préparation réelle à l'insertion professionnelle.

19. Conclusion

Le projet **BIBLIOTHECA** est une application web complète qui démontre :

- la maîtrise du JavaScript
- moderne la gestion dynamique
- du DOM l'utilisation d'API REST la
- visualisation de données une
- bonne organisation du code

Il constitue une base solide pour un futur projet professionnel ou une extension vers une application full-stack.