



Rapport – Développement d'une API de gestion de bibliothèque

Travail réaliser par l'étudiant Anunu Daniel

1.Introduction

Dans un contexte de digitalisation croissante des services, la gestion informatisée des bibliothèques est devenue une nécessité afin d'optimiser l'organisation des ressources, le suivi des utilisateurs et la gestion des emprunts. Les systèmes traditionnels, souvent manuels ou peu automatisés, présentent des limites en termes de fiabilité, de rapidité et de traçabilité.

Ce projet consiste à concevoir et développer une **API REST de gestion de bibliothèque**, permettant l'administration des utilisateurs, des livres et des prêts, tout en assurant la sécurité des accès grâce à un système d'authentification. L'application a été développée avec des technologies modernes du développement web backend, garantissant modularité, évolutivité et simplicité d'utilisation.

Objectifs du projet

L'objectif principal de ce projet est de développer une **application backend fonctionnelle et sécurisée** permettant la gestion complète d'une bibliothèque.

Les objectifs spécifiques sont les suivants :

- Mettre en place une API REST conforme aux bonnes pratiques.
- Gérer les utilisateurs (création, consultation, modification, suppression).
- Gérer les livres (ajout, consultation, modification, suppression).
- Gérer les prêts de livres avec des règles métier précises.
- Implémenter un système d'authentification sécurisé basé sur JWT.
- Utiliser une base de données relationnelle pour stocker les informations.
- Fournir une interface frontend simple (HTML) pour tester le bon fonctionnement de l'API.

3.Démarche méthodologique

La réalisation du projet s'est faite suivant une démarche progressive et structurée :

1. Analyse des besoins

Dans un premier temps, les fonctionnalités essentielles d'un système de bibliothèque ont été identifiées : gestion des utilisateurs, gestion des livres et suivi des emprunts.

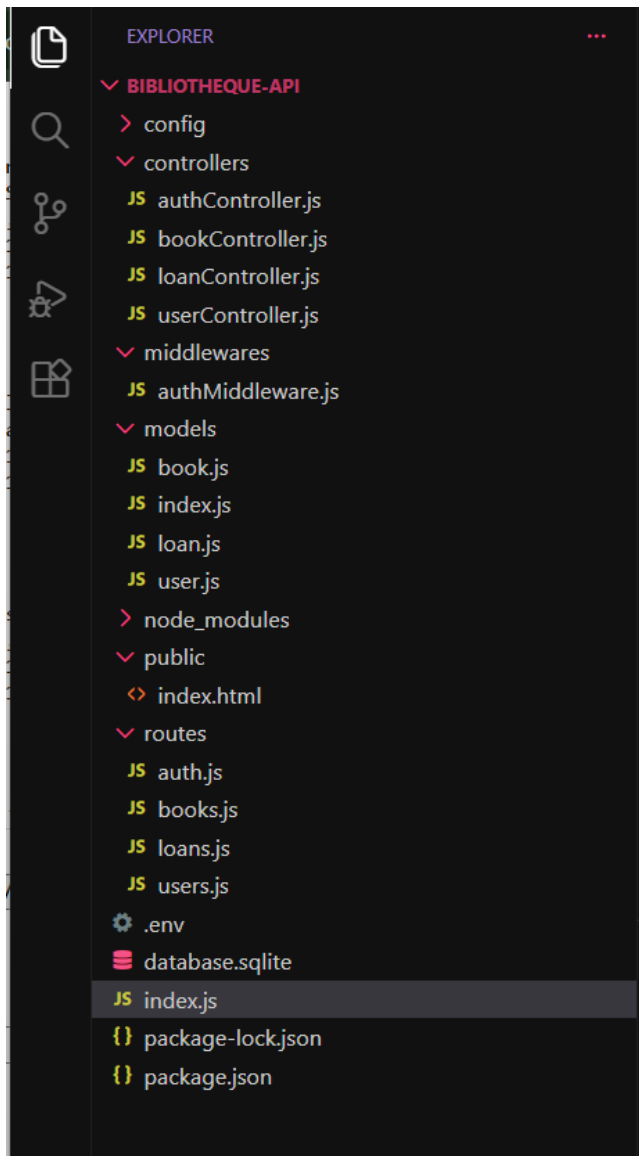
2. Choix technologiques

- Les technologies suivantes ont été utilisées :
- **Node.js** et **Express.js** pour le serveur backend.
- **Sequelize ORM** pour la gestion de la base de données.
- **SQLite** comme système de gestion de base de données, simple et léger.
- **JWT (JSON Web Token)** pour l'authentification.
- **bcrypt** pour le hachage des mots de passe.
- **HTML + JavaScript** pour un frontend minimal de test.

- **3. Conception de l'architecture**

Le projet a été structuré selon une architecture MVC (Modèle – Contrôleur – Routes) :

- **Models** : définition des entités User, Book et Loan.
- **Controllers** : logique métier.
- **Routes** : exposition des endpoints API.
- **Middlewares** : gestion de l'authentification.



4. Implémentation

Les différentes fonctionnalités ont été implémentées progressivement :

- Authentification (signup/login).
- CRUD des utilisateurs et des livres.
- Gestion des prêts avec contraintes :
 - Maximum 2 livres empruntés par utilisateur.
 - Durée maximale de prêt de 15 jours.

```

1 require('dotenv').config();
2 const express = require('express');
3 const path = require('path');
4 const bcrypt = require('bcrypt');
5 const app = express();
6
7 // ♦ Importation sequelize et modèles
8 const { sequelize, User, Book } = require('./models');
9
10 // ♦ Routes
11 const authRoutes = require('./routes/auth');
12 const userRoutes = require('./routes/users');
13 const bookRoutes = require('./routes/books');
14 const loanRoutes = require('./routes/loans');
15
16 // ♦ Middleware
17 app.use(express.json());
18
19 // ♦ Routes API
20 app.use('/api/auth', authRoutes);
21 app.use('/api/users', userRoutes);
22 app.use('/api/books', bookRoutes);
23 app.use('/api/loans', loanRoutes);
24

```

5. Tests et validation

Les fonctionnalités ont été testées à l'aide :

- d'un frontend HTML simple,
- de données préchargées (seed) intégrées directement dans le script,
- de requêtes HTTP simulées.

Résultats obtenus

À l'issue du développement, l'API est pleinement fonctionnelle et répond aux objectifs fixés :

- Authentification sécurisée avec génération de token JWT.
- Création automatique d'un utilisateur administrateur et de livres par défaut.
- Gestion complète des utilisateurs, livres et prêts.
- Application des règles métier sur les emprunts.
- Base de données synchronisée automatiquement.

Interface HTML simple permettant de visualiser et tester les fonctionnalités.

L'API est stable, claire et facilement extensible pour des évolutions futures.

Test Backend Library

Connexion

Liste des utilisateurs

```
[
  {
    "id": 1,
    "nom": "Admin",
    "prenom": "Biblio",
    "age": 30,
    "ecole": "Université Fès",
    "email": "admin@biblio.com",
    "password": "$2b$10$oQg2UqBurpvzoHVN8KMNEeHvdmiV80iaE.mhmFLxWeEhXHx.JxN7e",
    "createdAt": "2025-12-14T12:14:56.136Z",
    "updatedAt": "2025-12-14T12:14:56.136Z"
  }
]
```

Liste des livres

```
[
  {
    "id": 1,
    "titre": "Le Petit Prince",
    "auteur": "Antoine de Saint-Exupéry",
    "edition": "Gallimard",
    "createdAt": "2025-12-14T12:14:56.149Z",
    "updatedAt": "2025-12-14T12:14:56.149Z"
  },
]
```

Liste des prêts d'un utilisateur


```
[
  {
    "id": 1,
    "startDate": "2025-11-11",
    "endDate": "2025-11-12",
    "createdAt": "2025-12-14T21:20:44.066Z",
    "updatedAt": "2025-12-14T21:20:44.066Z",
    "userId": 1,
    "bookId": 1,
    "User": {
      "id": 1,
      "nom": "Admin",
      "prenom": "Biblio",
      "age": 30,
      "ecole": "Université Fès",
      "email": "admin@biblio.com",
      "password": "$2b$10$Og2UqBurpvzoHVN8KMNEeHvdmiv80iaE.mhmFLxWeEhXHx.JxN7e",
      "createdAt": "2025-12-14T12:14:56.136Z",
      "updatedAt": "2025-12-14T12:14:56.136Z"
    },
    "Book": {
      "id": 1,
      "titre": "Le Petit Prince",
      "auteur": "Antoine de Saint-Exupéry",
      "edition": "Gallimard",
      "createdAt": "2025-12-14T12:14:56.149Z",
      "updatedAt": "2025-12-14T12:14:56.149Z"
    }
  }
]
```

Créer un prêt


```
{
  "id": 1,
  "userId": "1",
  "bookId": "1",
  "startDate": "2025-11-11",
  "endDate": "2025-11-12",
  "updatedAt": "2025-12-14T21:20:44.066Z",
  "createdAt": "2025-12-14T21:20:44.066Z"
}
```

Conclusion

Ce projet a permis de mettre en pratique les concepts fondamentaux du développement backend moderne, notamment la conception d'API REST, la sécurisation des accès, l'utilisation d'un ORM et la structuration d'un projet professionnel.

La solution développée répond efficacement aux besoins d'une bibliothèque de petite à moyenne taille et constitue une base solide pour des améliorations futures telles que :

l'ajout de rôles utilisateurs (admin, lecteur),
une interface frontend plus avancée (React, Vue),
la gestion des retards et pénalités,
le déploiement sur un serveur distant.

En conclusion, ce projet démontre la faisabilité et l'efficacité d'une solution backend moderne pour la gestion des bibliothèques, tout en offrant une architecture claire et évolutive.