Plateforme de Gestion de Bibliothèque en Ligne

Rapport de Projet

Enseignant : Mr Gansbeogo

Étudiant : Kone Zana Jean Baptiste

4 juillet 2025

Table des matières

1 PRÉSENTATION DU PROJET

1.1 Vue d'ensemble

Le système académique requiert plusieurs contraintes favorisant un enfreint à l'évolution de l'apprentissage éducationnel, c'est dans ce contexte que notre plateforme universitaire a été développée permettant aux étudiants de gérer plusieurs interactions avec une bibliothèque smart intelligente de façon digitalisée. Le système offre une interface moderne et intuitive permettant aux étudiants de rechercher, réserver et emprunter des livres en ligne tout en offrant un espace administrateur pour gérer les ressources globales.

1.2 Objectifs stratégiques

Le système mis en place propose les exigences ou fonctionnalités suivantes :

— Une dématérialisation complète des processus d'inscription et de connexion
— Un système de recherche avancé avec filtres multiples
— Une gestion automatisée des emprunts et retours
— Un système de notation et commentaires des livres pour enrichir l'expérience utilisateur
— Un système de notifications proactif pour les retards
— Un tableau de bord administrateur pour la gestion des ressources globales

2 ARCHITECTURE TECHNIQUE

Les technologies utilisées pour la réalisation de notre projet sont :

2.1 Stack technologique

Frontend :

— HTML, CSS, JavaScript (Next.js) pour l'interface utilisateur interactive
— Gestion d'état moderne

Backend :

— Node.js avec Express.js pour l'API RESTful
— Architecture middleware pour la gestion des requêtes
— Authentification JWT pour la sécurité

Base de données :

— MySQL comme système de gestion de base de données relationnelle
— Modélisation normalisée des données
— Intégrité référentielle assurée

Outils de développement :

— GitHub pour le versioning et la collaboration
— Docker & Docker Compose pour la conteneurisation

2.2 Architecture RESTful

Notre plateforme respecte les principes REST avec une utilisation appropriée des verbes HTTP :

— GET pour la consultation des ressources

— POST pour la création de nouvelles entités

— PUT pour la mise à jour des données existantes

— DELETE pour la suppression des ressources

3 MODÉLISATION DE LA BASE DE DONNÉES

3.1 Conception relationnelle

La base de données api présente une architecture bien structurée avec 6 tables principales :

Table users :

— Gestion des utilisateurs avec système de rôles

— Champs obligatoires sécurisés (nom, prénom, email, password)

— Système binaire de rôles (0 = étudiant, 1 = administrateur)

Table livres :

— Catalogue complet des ouvrages

— Identification unique par ISBN

— Gestion du statut de disponibilité

— Catégorisation par genre

Table commentaires :

— Liaison avec les livres via clé étrangère

— Traçabilité temporelle des contributions

Table notations :

— Système d'évaluation de 1 à 5 étoiles

— Contrainte d'unicité utilisateur/livre

— Indexation optimisée pour les performances

Table emprunts :

— Gestion complète du cycle d'emprunt

— Suivi des dates (emprunt, retour prévu, retour effectif)

— Gestion des statuts (en_cours, retourné, en_retard)

Table réservations :

— Système de réservation préalable

— Traçabilité des demandes

3.2 Intégrité référentielle

L'intégrité des données du système est assurée par :

— Contraintes de clés étrangères avec CASCADE
— Contraintes d'unicité (email, ISBN)
— Contraintes de validation (notes entre 1 et 5)
— Index optimisés pour les requêtes fréquentes

4 FONCTIONNALITÉS IMPLÉMENTÉES

4.1 Espace étudiant

Authentification sécurisée :

— Système d'inscription avec validation des données
— Connexion sécurisée via JWT
— Gestion des sessions utilisateur

Gestion du catalogue :

— Consultation des livres disponibles
— Système de filtrage avancé (titre, auteur, genre)
— Affichage des détails complets des ouvrages

Système d'emprunt :

— Réservation en ligne des ouvrages
— Processus d'emprunt digitalisé

Interaction communautaire :

— Système de notation des livres
— Espace commentaires pour le partage d'avis

Notifications proactives :

— Alertes automatiques par email en cas de délais de retour d'ouvrages dépassé
— Rappels de retour avant échéance
— Notifications de retard

4.2 Espace administrateur

Gestion des utilisateurs :

— Vue d'ensemble des étudiants inscrits
— Gestion des profils utilisateur

Gestion du catalogue :

— Ajout de nouveaux ouvrages
— Modification des informations existantes
— Suppression des ouvrages existants

Gestion des emprunts :

— Suivi en temps réel des emprunts
— Traitement des retours

5 STRUCTURE DU PROJET

5.1 Organisation des fichiers

```
/KONE
          backend/
                server.js                # Point d'entr e du serveur
                middleware/              # Middlewares personnalis s
                database.js              # Configuration base de donn es
                Dockerfile              # Configuration Docker backend
          frontend/
                app/                     # Application React
                Dockerfile               # Configuration Docker frontend
                package.json             # D pendances frontend
          .env                          # Variables d'environnement
          .gitignore                   # Fichiers ignor s par Git
          dev.session.sql              # Scripts SQL de d veloppement
          docker-compose.yml           # Orchestration des services
          README.md                    # Documentation projet
          schema_base_de_donn es.png # Diagramme relationnel
          captures/                    # Documentation visuelle
          Rapport_PROJET               # Rapport du projet
```

5.2 Séparation des responsabilités

— Backend : API RESTful et logique métier

— Frontend : Interface utilisateur et expérience client

— Base de données : Persistance et intégrité des données

— Documentation : Guides d'utilisation et captures d'écran

6 DÉPLOIEMENT ET HÉBERGEMENT

6.1 Stratégie d'hébergement initiale

Le projet avait initialement pour objectif un déploiement en production avec :

— Frontend : Hébergement sur Vercel pour optimiser les performances et la distribution

— Backend : Déploiement sur Render dédiée

— Base de données : Hébergement sur un service cloud MySQL spécialisé

6.2 Contraintes rencontrées

Malgré la planification initiale, plusieurs défis techniques ont été rencontrés :

— Configuration complexe des variables d'environnement inter-services

— Gestion des CORS entre domaines différents

— Synchronisation des déploiements entre les trois composants

— Coûts d'hébergement pour un projet de démonstration

6.3 Solution adoptée

Face à ces contraintes, la décision a été prise de privilégier une démonstration locale qui permet :

— Une présentation complète des fonctionnalités
— Un contrôle total de l'environnement d'exécution
— Une configuration simplifiée pour les tests

7 GUIDE D'INSTALLATION ET DÉMARRAGE

7.1 Prérequis système

— Node.js (version 14 ou supérieure)
— MySQL Server
— Git pour le clonage du repository
— Docker (optionnel, pour la conteneurisation)

7.2 Installation backend

```
cd KONE
cd backend
npm install
npm start
```

7.3 Installation frontend

```
cd KONE
cd frontend
npm install
npm start
```

7.4 Configuration base de données

— Création de la base de données "api"
— Exécution des scripts SQL fournis (dev.session.sql)
— Configuration des variables d'environnement (.env)

8 SÉCURITÉ ET AUTHENTIFICATION

8.1 Système d'authentification

— Utilisation de JWT (JSON Web Tokens) pour l'authentification
— Chiffrement des mots de passe
— Validation des données côté serveur
— Gestion des sessions sécurisées

8.2 Gestion des rôles

Le système implémente un contrôle d'accès basé sur les rôles :

— Rôle 0 (Étudiant) : Accès aux fonctionnalités de base

— Rôle 1 (Administrateur) : Accès complet à la gestion

Par défaut, un utilisateur créé se verra attribuer le rôle d'un étudiant.

8.3 Compte administrateur de test

Un compte administrateur a été créé pour les démonstrations :

— Nom : kone (minuscule)

— Prénom : zana (minuscule)

— Email : kone1@gmail.com (minuscule)

— Password : kone1@gmail.com (minuscule)

— Rôle : Administrateur (1)

9 DOCUMENTATION ET CAPTURES

9.1 Documentation technique

— Schéma de base de données (schema_base_de_données.png)

— Scripts SQL complets (dev.session.sql)

— Documentation README complète

— Commentaires détaillés dans le code source

9.2 Documentation visuelle

Le dossier /captures contient :

— Captures d'écran de toutes les fonctionnalités

— Interfaces utilisateur étudiant et administrateur

— Processus d'inscription et connexion

10 PERFORMANCES ET OPTIMISATIONS

10.1 Optimisations base de données

— Index sur les champs fréquemment consultés

— Contraintes d'intégrité pour éviter les données corrompues

— Requêtes optimisées pour les opérations courantes

10.2 Optimisations frontend

— Composants React optimisés

— Gestion d'état efficace

10.3 Optimisations backend

— Middleware optimisé pour les performances
— API RESTful respectant les bonnes pratiques

11 ÉVOLUTIONS FUTURES

11.1 Améliorations techniques

— Implémentation d'un système de recherche full-text
— Intégration d'un système de recommandations
— Amélioration du système de notifications (push notifications)

11.2 Fonctionnalités supplémentaires

— Système de lecture et favoris et listes de lecture
— Intégration avec des APIs externes de livres
— Système de recommandations personnalisées
— Module de statistiques avancées
— Intégration d'un système de liste d'emprunts pour étudiants

11.3 Déploiement en production

— Configuration d'un pipeline CI (intégration continue) / CD (déploiement continu)
— Mise en place d'un monitoring avancé
— Optimisation pour le déploiement cloud
— Sécurisation renforcée pour l'environnement de production

12 CONCLUSION

Ce projet de plateforme de gestion de bibliothèque représente une solution complète et moderne pour la digitalisation des services de bibliothèque universitaire. Malgré les contraintes d'hébergement rencontrées, la solution développée démontre une architecture solide, d'une approche des fonctionnalités complètes et technique professionnelle.

La décision de privilégier une démonstration locale permet une présentation optimale des capacités du système tout en conservant la flexibilité nécessaire pour les évolutions futures. Le projet constitue une base solide pour un déploiement en production ultérieur avec les optimisations et améliorations identifiées.

L'architecture modulaire et les bonnes pratiques implémentées garantissent la maintenabilité et l'évolutivité du système, répondant ainsi aux besoins actuels et futurs d'une bibliothèque universitaire moderne.