# Description de l'Application Manga Reader

Malek BOUZARKOUNA, Yacine KESSAL, Lyes LAÏMOUCHE

April 3, 2025

## 1 Description de l'application

L'application **Manga Reader** est une plateforme disponible sur **web**, permettant aux utilisateurs de consulter, rechercher et lire des mangas en ligne. Elle intègre un système de gestion de compte utilisateur, de suivi des lectures et une communauté interactive via un système de commentaires.

L'objectif principal de l'application est d'offrir une **expérience immersive** et **personnalisée** pour les lecteurs de mangas, avec des recommandations basées sur leurs préférences et un accès facilité aux nouvelles sorties.

## 2 Fonctionnalités

### 2.1 Fonctionnalités principales

- Consultation des mangas tendances
- Recherche de mangas et lecture
- Gestion de l'historique de lecture
- Système de commentaires
- Personnalisation du compte utilisateur
- Suivi des mangas favoris
- Gestion de la base de données des utilisateurs et mangas

### 2.2 Idées d'intégration d'IA

Pour améliorer l'expérience utilisateur, une fonctionnalité basées sur l'intelligence artificielle ont été intégrées :

- Chatbot

## 3 Stack Technique

L'application repose sur une architecture moderne avec les technologies suivantes :

### 3.1 Backend

- **Langage** : Go
- **Base de données** : MongoDB
- **APIs utilisées** : MangaDex (mangas), Cloudinary (gestion des images), Spotify (musique) , aimlapi/OpenAI (Chatbot)

### 3.2 Frontend

- **Web** : React
- **Style** : CSS

### 3.3 Infrastructure

- Déploiement via Docker
- GitHub

### Justification des choix techniques

Les choix technologiques effectués pour le développement de Manga Reader ont été guidés par des critères de performance, de scalabilité, de simplicité d'intégration, et de pertinence vis-à-vis des fonctionnalités prévues. Voici les principales justifications :

**Go (Backend)** Le langage Go a été choisi pour le développement du backend en raison de sa **performance élevée**, de sa **gestion efficace des routines concurrentes**, et de sa **syntaxe simple**. Go permet de créer des API REST performantes avec une latence faible, ce qui est idéal pour une application dynamique comme Manga Reader.

**MongoDB (Base de données)** Le choix de MongoDB s'est imposé naturellement pour une application orientée contenu, avec une structure de données **souple et évolutive**. Le stockage en documents JSON permet une représentation intuitive des entités comme les mangas, chapitres ou utilisateurs.

**React (Frontend Web)** React permet de concevoir une **expérience utilisateur fluide** grâce à sa gestion efficace de l'état et à son système de composants. C'est une solution moderne et largement adoptée pour les interfaces dynamiques.

**CSS (Style)** L'utilisation de CSS pur donne un **contrôle total sur le design**. Ce choix a été fait pour une meilleure compréhension du comportement visuel sans dépendre d'un framework externe.

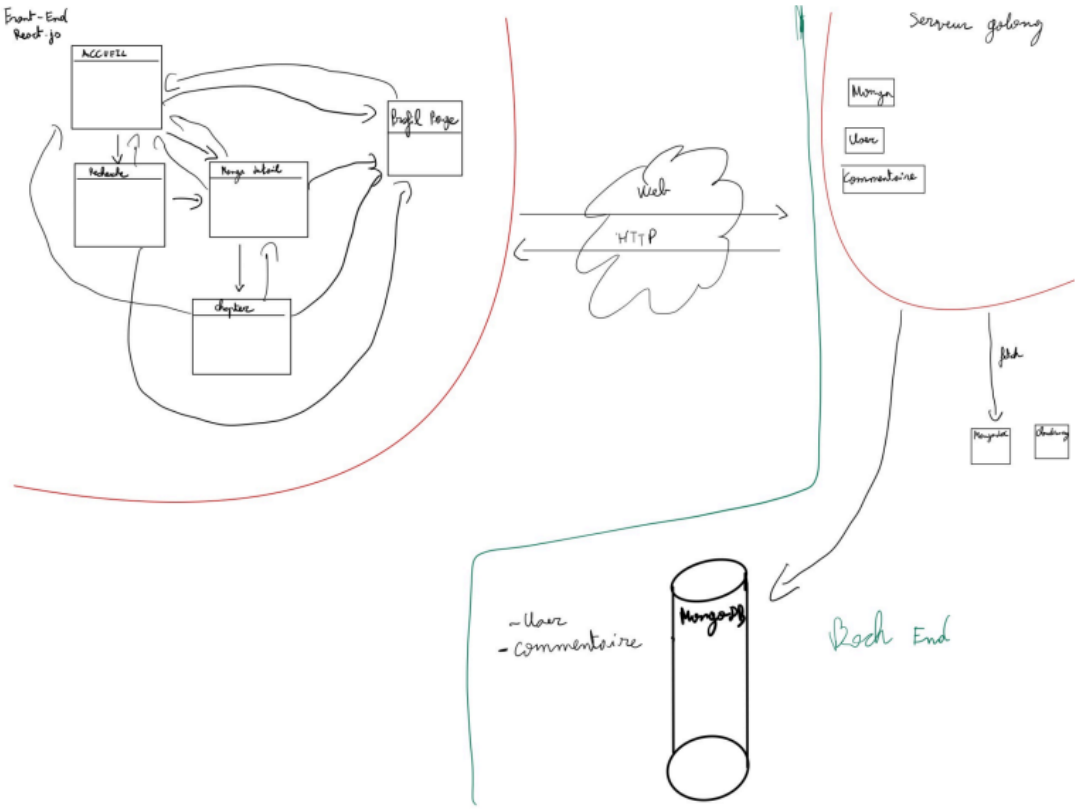
**APIs externes**

- **MangaDex** : API spécialisée fournissant les données de mangas (titres, images, chapitres).
- **Cloudinary** : Pour la gestion et l'optimisation des images (compression, CDN, redimensionnement).
- **OpenAI / AIML API** : Pour l'intégration d'un chatbot interactif, capable de recommander des mangas ou de répondre à des questions sur l'univers manga.

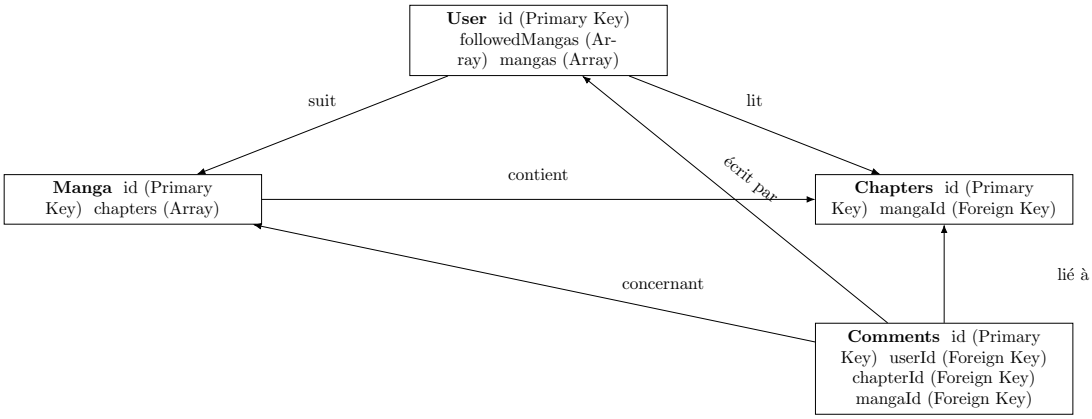
**Docker** Docker simplifie la **gestion des environnements de développement et de production**. Il permet une portabilité optimale entre machines, et facilite les tests ainsi que le déploiement.

**GitHub** GitHub offre un excellent support pour le travail en équipe, le versioning du code, et s'intègre facilement avec des plateformes comme Vercel ou Render pour le déploiement continu.

# 4 Architecture de l'application




# 5 Schéma des relations entre les collections




# 6 Difficultés :

- Intégration de musique avec l'API Spotify
- Deploiement sur internet de l'appli. Nous avons utilisé Vercel pour le Frontend, Render pour le backend et la BDD était déjà déployée avec MongoDB Atlas.
  Nous avons déployé le backend sur Render avec succès, cependant, l'intégration de l'API Cloudinary ne fonctionne pas correctement sur l'environnement déployé, bien qu'elle soit opérationnelle en local.