Création d’une api pour gérer une site de critique de livres

Chef de projet : M. Mveng

24 périodes (8 semaines)

Table des matières

[Introduction 2](#_Toc160543747)

[Analyse 3](#_Toc160543748)

[API (routes) 3](#_Toc160543749)

[Routes utilisateurs 3](#_Toc160543750)

[Routes livres 3](#_Toc160543751)

[Base de données (MCD, MLD) 4](#_Toc160543752)

[Architecture de la base de données 4](#_Toc160543753)

[Explications relations 4](#_Toc160543754)

[Réalisation 5](#_Toc160543755)

[Test 6](#_Toc160543756)

[Conclusion 7](#_Toc160543757)

[Gestion du code 7](#_Toc160543758)

[Qu’y a-t-il à retenir de ce projet 7](#_Toc160543759)

# Introduction

Ce projet a pour but d’approfondir nos connaissances sur les API et leurs constructions acquise en module, il consiste à créer une API comportant diverses routes permettant la gestion des données ainsi qu’un système d’authentification basé sur les jetons JWT, tout cela avec une gestion des statuts (200,3xx,4xx,5xx). Le bon fonctionnement de l’API est certifié par des tests via Insomnia. Finalement il y a aussi une documentation automatique crée grâce à Swagger (voir annexe).

Dans ce rapport nous allons vous expliquer comment nous avons fait pour travailler en groupe, la manière de gérer les tâches ainsi que la gestion du code, ensuite nous allons analyser et expliquer les divers incréments que nous proposons tel que l’API ou encore l’architecture de la base de données. Après nous allons expliquer la réalisation de l’API en passant par les points techniques et les fonctionnalités demandée. Finalement on conclura ce rapport en repassant sur la gestion du projet, et en l’analysant en globalité afin d’émettre des critiques constructives.

# Analyse

## API (routes)

### Routes utilisateurs

|  |  |  |  |
| --- | --- | --- | --- |
| **Action** | **Verbe http** | **URI** | **Informations à transmettre** |
| Sélectionne tous les utilisateurs | GET | /api/utilisateurs |  |
| Sélectionne un utilisateur selon ID | GET | /api/utilisateurs/ :id |  |
| Créer un utilisateur | POST | /api/utilisateurs | pseudo, mot de passe au format json |
| Supprimer un utilisateur | DELETE | /api/utilisateurs/ :id |  |

### Routes livres

|  |  |  |  |
| --- | --- | --- | --- |
| **Action** | **Verbe http** | **URI** | **Informations à transmettre** |
| Sélectionne tous les livres | GET | /api/livres |  |
| Sélectionne un livre selon son ID | GET | /api/livres/ :id |  |
| Ajout d’un livre | POST | /api/livres | Nom, extraitpdf(string), resume, anneeEdition, imageCouverture(string)  En json  A REVOIR (après fk etc) |
| Supprimer un livre | DELETE | /api/livres/ :id |  |
| Mettre à jour un livre | PUT | /api/livres/ :id | Informations que l’on veut modifier en json |

**A REVOIR « informations à transmettre » peut-être un exemple à la place d’explication ?**

## Base de données (MCD, MLD)

### Architecture de la base de données

Une image contenant texte, capture d’écran, diagramme, ligne

Description générée automatiquement

Une image contenant texte, capture d’écran, diagramme, ligne

Description générée automatiquement

L’architecture de notre BDD (base de données) a été conçu en ayant en tête d’avoir une compréhension facile, des informations utiles et cohérentes.

### Explications relations

La table LIVRE contient les données des livres et est en relation avec toutes les autres tables, ces relations permettent d’obtenir plus d’informations sur certains de ses champs, par exemple on peut avoir plus d’informations sur l’auteur ou même d’obtenir des informations via des agrégations simples par exemple on peut obtenir la moyenne d’appréciation d’un livre grâce à la table commentaire

## Structure du code

Nous organisons notre code en regroupant tous les fichiers selon 4 fonctions, les codes des routes, les modèles de données (normes), ce qui est en lien avec la db et l’authentification.

Une image contenant texte, capture d’écran, Police, Rectangle

Description générée automatiquement

**A CONTINUER (Auth + relation/liaison)**

# Réalisation

# Test

# Conclusion

## Gestion du code

Afin de pouvoir travailler simultanément sur notre API nous avons choisi d’utiliser Github où nous travaillions sur la même API, cependant nous nous répartissions les tâches afin de ne pas travailler sur les mêmes parties du codes, grâces à cela nous avons rencontré peu de conflits facilement réglables directement.

**A REVOIR**

## Qu’y a-t-il à retenir de ce projet

Webographie