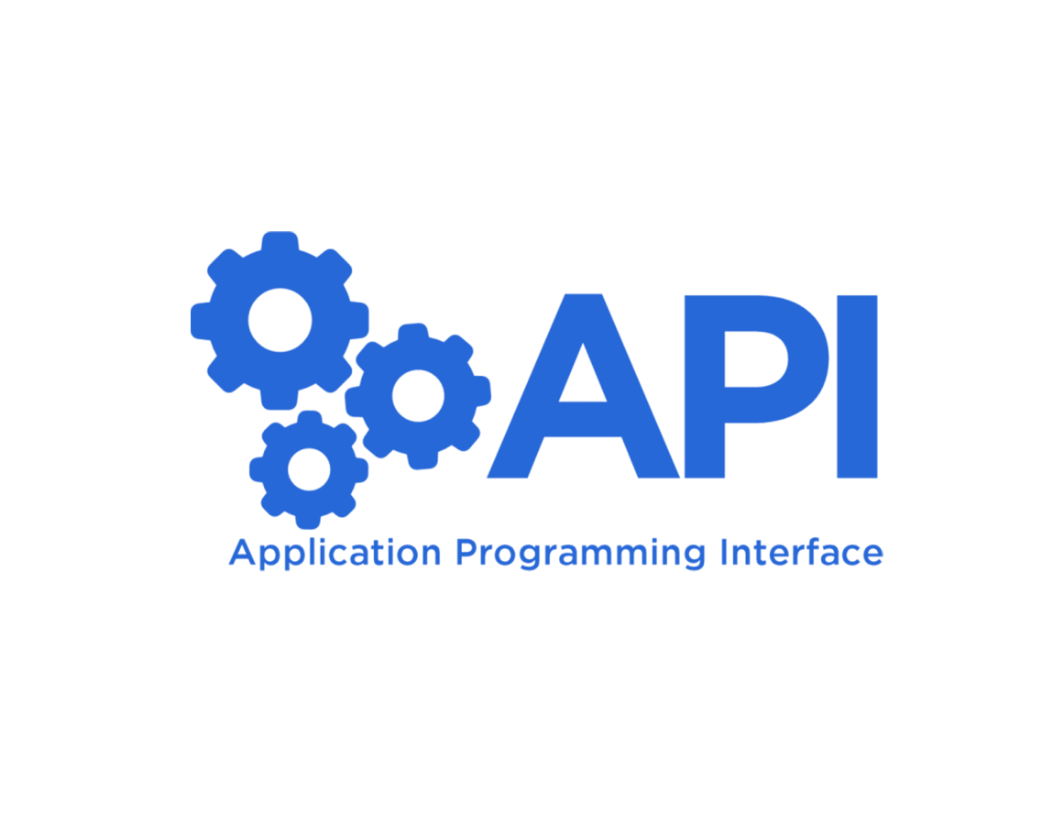
P\_WEB – C295



Sofien Habib Belkhiria - Lucas Lordon

Vennes – ETML

19.02.24 – 12.03.24

Chef de projet : Antoine Mveng

Table des matières

[Introduction 1](#_Toc161177440)

[Analyse 1](#_Toc161177441)

[Routes 1](#_Toc161177442)

[Base de données 3](#_Toc161177443)

[Structure du projet 6](#_Toc161177444)

[Structure dossier codeFinal 7](#_Toc161177445)

[Structure dossier SRC 8](#_Toc161177446)

[Réalisation 8](#_Toc161177447)

[Test 8](#_Toc161177448)

[Conclusion 9](#_Toc161177449)

[Auto-évaluation : 9](#_Toc161177450)

[Critiques 10](#_Toc161177451)

[Webographie/Bibliographie/Glossaire 10](#_Toc161177452)

[7. Utilisation d’IA 10](#_Toc161177453)

# Introduction

Ce projet a été réaliser par Lucas Lordon et Sofien Habib Belkhiria.

Le but de ce projet est de faire une application backend qui consiste à la mise en place d’une API REST. L’API REST doit reprendre les notions suivantes vu dans le cours : les routes, la gestion des statuts, un système de recherche, un système d’authentification JWT, une documentation Swagger, des tests via Insomnia, des tests automatisés avec vitest, GitHub actions et une dockerisation du backend.

Le développement de cette application permettra de consolider les bases de nos connaissances sur le développement Backend et l’utilisation de technologies tels que « Sequilize » et « Insomnia ».

Cette API permet d'ajouter des livres à qui sont stocker dans une base de donnée, de les consulter via des de recherches plus ou moins complexe ce faisant au travers de routes. Les utilisateurs peuvent également rédiger des critiques sur les livres lus, avec ou sans commentaire, et évaluer les livres.

Le projet à été réaliser entre le 19.02.24 et le 12.03.24 pour un total de 24 périodes.

L’ensemble du projet et présent dans le repository GitHub P\_Web295, mis à par la planification qui a été réaliser sur trello.

Avant de commencer notre projet, il est primordial de comprendre le concept d'API (Interface de Programmation d'Application). Une API représente un ensemble de chemins ou de points d'accès qui facilitent la communication entre différentes applications logicielles. En simplifiant l'API permet à divers programmes informatiques de partager des informations.

# Analyse

Durant ce projet, nous avons utilisé GitHub et Trello pour la gestion de nos tâches. Avant chaque séance, nous consacrions un court moment à la répartition des tâches afin d'assurer une collaboration efficace. Pour éviter toute corruption de données, nous avons initialement travaillé individuellement sur nos propres dossiers, en créant des routes conformément aux instructions des documents PDF fournis. Par la suite, nous avons fusionné nos codes. Sur Trello, nous avons simplement dressé une liste des tâches à effectuer. De plus, chacun d'entre nous tenait un journal de travail pour suivre notre progression individuelle.

## Routes

Les routes que nous avons créées sont répertoriées dans notre document Insomnia appelé "Insomnia\_LucasL\_Sofiene.json", situé dans le dossier "data". Pour les utiliser, il suffit de l'importer dans Insomnia. Les routes évaluées sont ensuite disponibles dans le dossier "RequetteHttpDemander" :

Une image contenant texte, capture d’écran, menu, conception

Description générée automatiquement

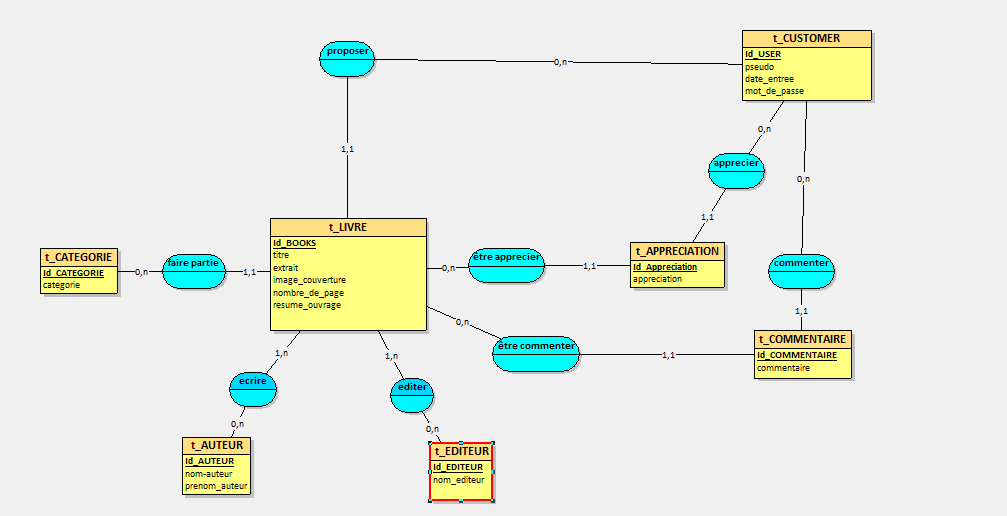
Comme vous pouvez peut-être le constater il nous manque 2 route sur les 17 demander : Une image contenant texte, capture d’écran, nombre, Parallèle

Description générée automatiquement

## Base de données

MCD

Comme pour tout projet nécessitant une base de donné SQL il est important de crée un MCD et un MLD :



MCD

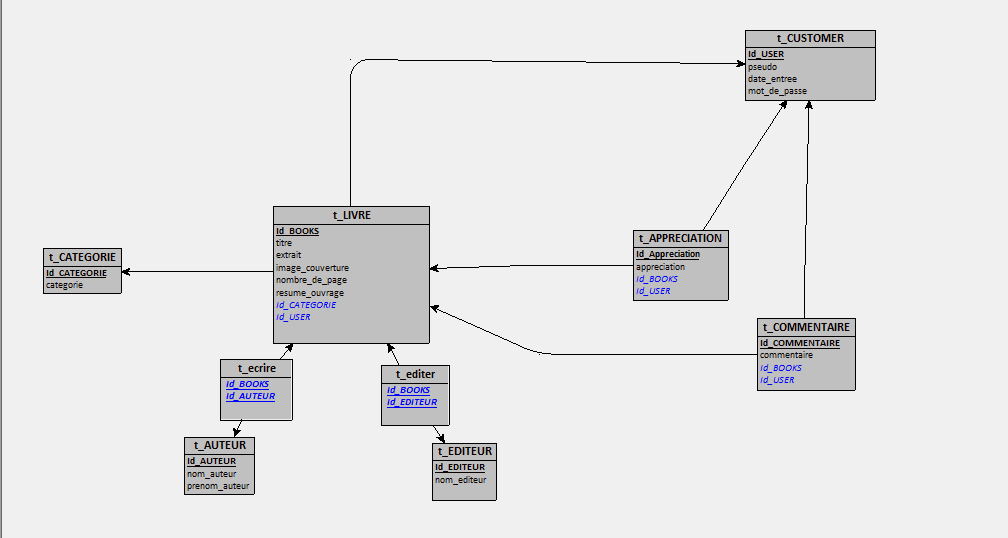
Table

La base de données comprend 7 tables vous trouverez ci-dessous les raisons des cardinalités de chacune d’entre elle ainsi que les raisons de leur emplacement.

* T\_CUSTOMER :
  + Contient le pseudo, le mot de passe et la date d’entrée de l’utilisateur.
  + Est associer à la table livre afin que l’utilisateur puissent proposer des livres, chacun des livres de l’ayant été proposer par un utilisateur a donc un lien vers l’id de l’utilisateur l’ayant proposé.
  + Est associer à la table appréciation afin de permettre à l’utilisateur de pouvoir laisser des appréciations et de connaitre quel utilisateur à laisser quelle appréciation.
  + Est associer à la table commentaire afin de permettre à l’utilisateur de pouvoir laisser des commentaires et de connaitre quel utilisateur à laisser quel commentaire.
* T\_LIVRE
  + Contient le titre, le prix, la/les catégorie(s), le nombre de pages, le résumé d’un livre, la table contient aussi l’ID de l’utilisateur qui à proposer le livre.
  + Est associer à la table éditeur afin de permettre à l’utilisateur de connaitre quel éditeur a édité le livre.
  + Est associer à la table catégorie afin de permettre à l’utilisateur de connaitre quel sont les catégories associer au livre.
  + Est associer à la table auteur afin de permettre à l’utilisateur de connaitre qui est l’auteur du livre.
* T\_AUTEUR
  + Contient le nom et le prénom de l’auteur.
* T\_CATEGORIE
  + Contient le nom des catégories
* T\_EDITEUR
  + Contient le nom de l’éditeur
* T\_COMMENTAIRE
  + Contient le commentaire de l’utilisateur
* T\_APPECIATION
  + Contient l’appréciation de l’utilisateur

MLD

* Vous trouverez-ci-dessous la logique de connexion entre les tables.



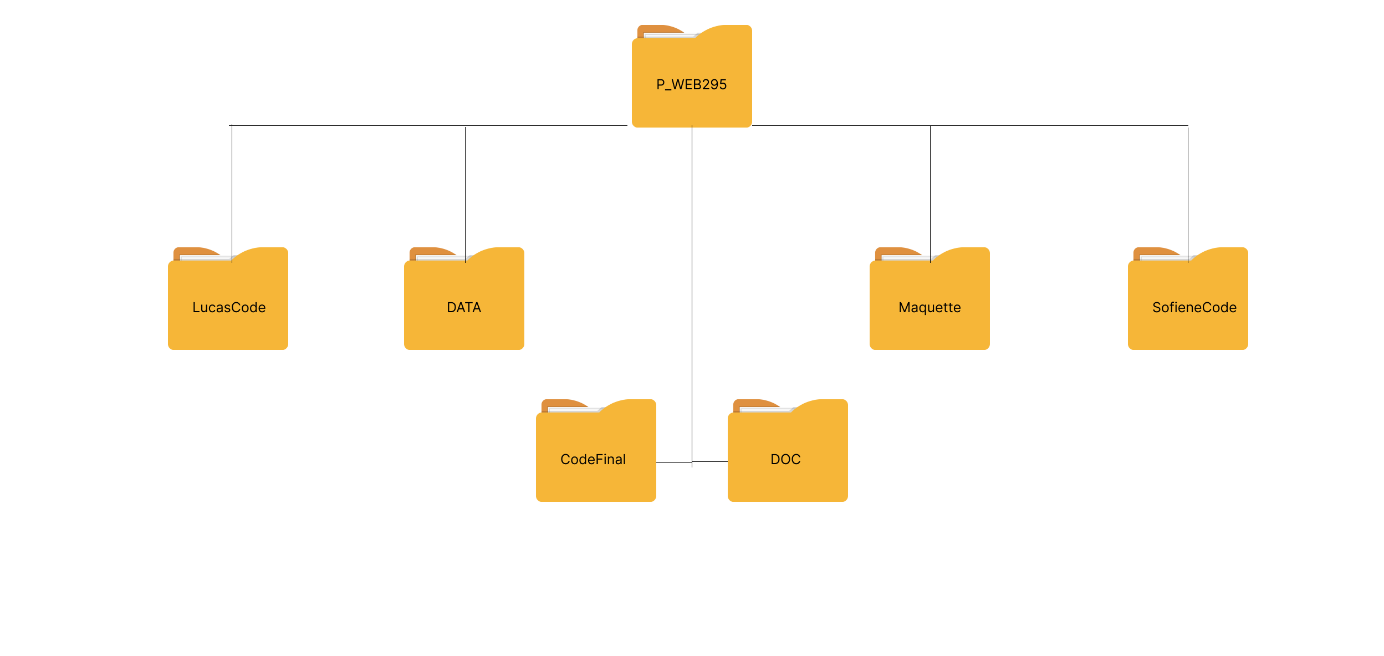
MLD

## Structure du projet

But

* Vous trouverez ci-dessous une analyse du code de l’API en commençant par la structure des fichiers, passant par les modules utiliser et finissant par le contenu des fichiers.

Structure des Dossiers



LucasCode

* Dossier contenant le code de l’API en cours de développement appartenant à Lucas.

SofieneCode

* Dossier contenant le code de l’API en cours de développement Appartenant à Sofiène.

Data

* Dossier contenant tous les dossiers concernant la base de données et les requêtes tel que :
  + MCD/MLD looping.
  + Image du MCD et MLD du looping.
  + Fichier SQL contenant les données de la db.
  + Fichier Excel contenant les informations des livres.
  + Fichier Insomnia contenant toutes les requête de l’API.

Doc

* Dossier contenant la documentation du projet comme le rapport ainsi que des PDF utile à l’API en plus des journaux de travaux.

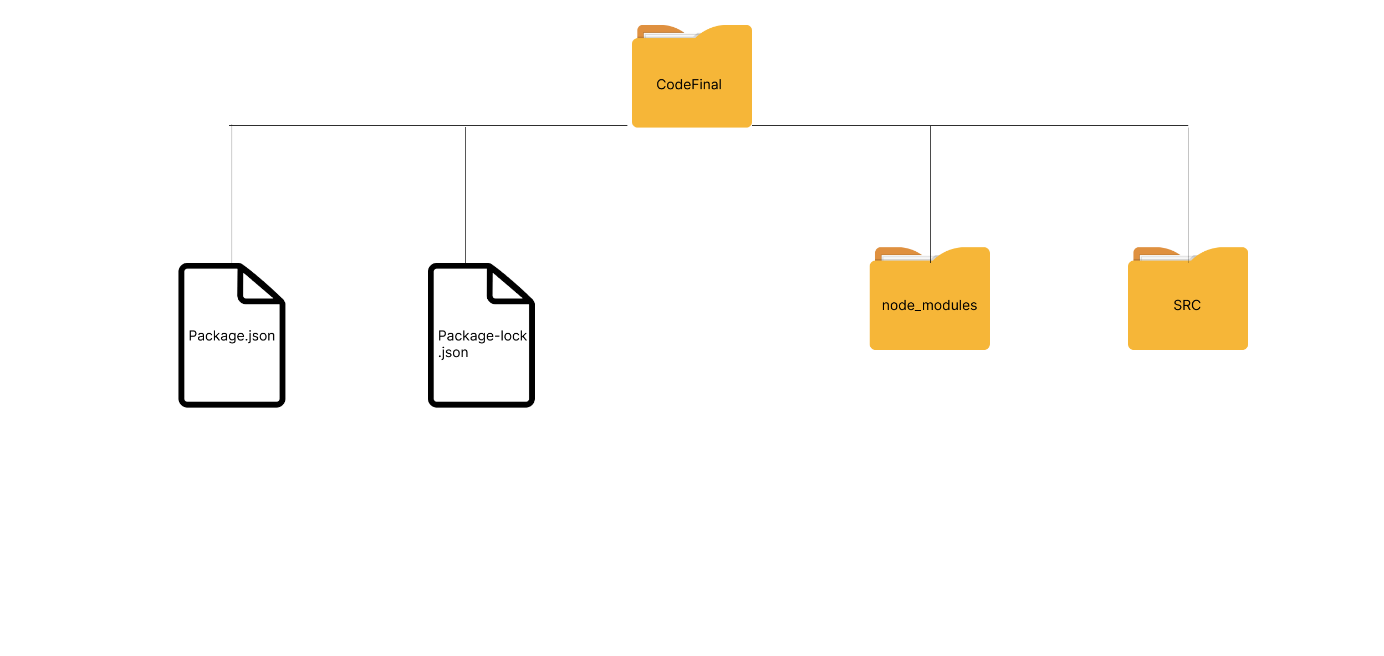
Maquette

* Dossier contenant les maquette du site Web.

Code Final

* Code final de l’API, fusion du code de Lucas et Sofiène.

## Structure dossier codeFinal



Package.json

* Fichier de base de l’API, contient la référence des modules utiliser, ainsi que les informations de base de l’API

Package-lock.json

* Contient tous les imports des différents modules utiliser dans l’API

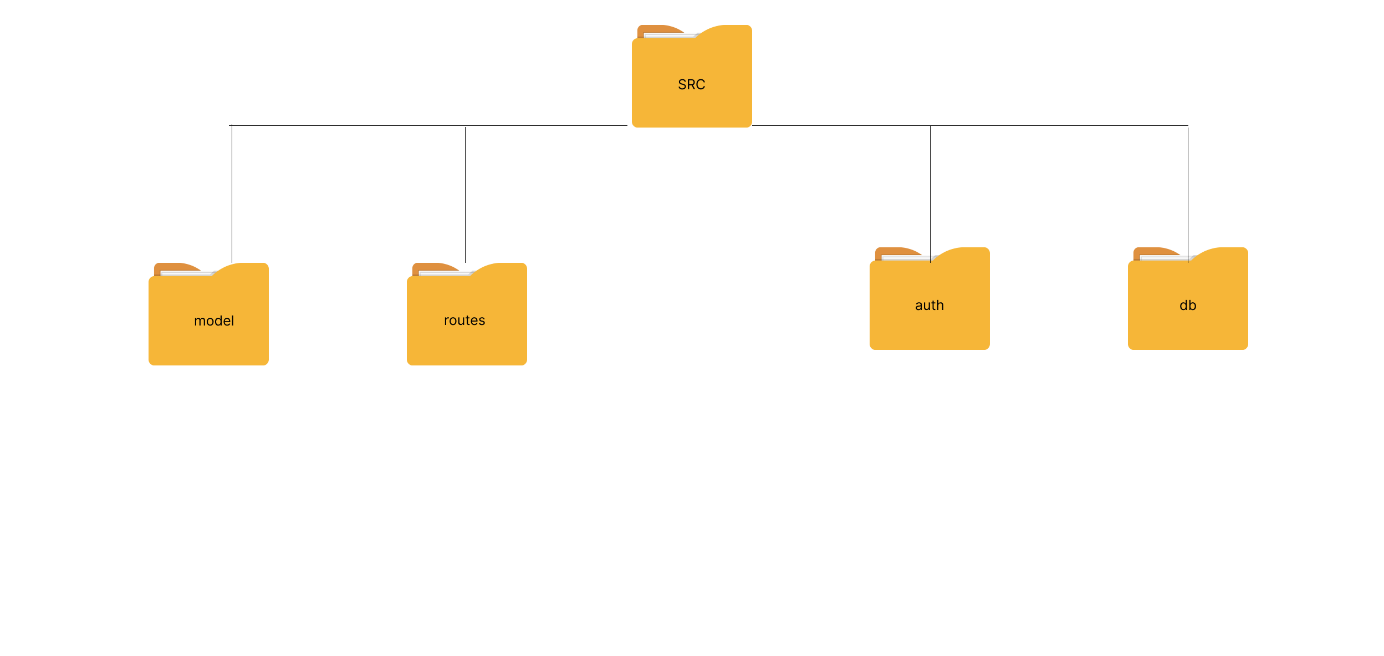
Node\_modules

* Contient tous les fichiers nécessaires au bon fonctionnement des modules installé

SRC

* Contient tous les dossiers du code de l’API (voir précision plus bas).

## Structure dossier SRC



Model

* Contient les différents modèles de chacune des tables de l’API, ces model permette à l’API de connaitre sur quelles tables elle va agir ainsi que les données acceptées sur ces tables.

Routes

* Contient toutes les routes utilisables par l’API, ces dans ce fichier que se trouve le code s’appliquant quand le client exécute une des URI vus plus [haut](#_Routes).

Auth

* Contient le code gérant l’authentification des utilisateurs via leur token, il est aussi responsable d’empêcher certaines routes dans le cas ou l’utilisateur n’est pas authentifier.

DB

* Dossier contenant les fichiers servant de données de base à introduire dans la DB, contient aussi le code permettant de lier l’API à la base de données en plus de créer les tables de la DB en fonction des model vu précédemment.

# Réalisation

## Explication de l'algorithme pour l'authentification et la gestion des rôles.

Algoritme :  
  
import jwt from "jsonwebtoken";

import { privateKey } from "./privateKey.mjs";

const auth = (req, res, next) => {

  const authorizationHeader = req.headers.authorization;

  if (!authorizationHeader) {

    const message = `Vous n'avez pas fourni de jeton d'authentification. Ajoutez-en un dans l'en-tête de la requête.`;

    return res.status(401).json({ message });

  } else {

    const token = authorizationHeader.split(" ")[1];

    const decodedToken = jwt.verify(

      token,

      privateKey,

      (error, decodedToken) => {

        if (error) {

          const message = `L'utilisateur n'est pas autorisé à accéder à cette ressource.`;

          return res.status(401).json({ message, data: error });

        }

        const userId = decodedToken.userId;

        if (req.body.userID && req.body.userID !== userID) {

          const message = `L'identifiant de l'utilisateur est invalide`;

          return res.status(401).json({ message });

        } else {

          next();

        }

      }

    );

  }

};

export { auth };

Cet algorithme est une fonction de middleware qui sert à vérifier l'authenticité d'un utilisateur avant de lui permettre d'accéder à certaines ressources dans une application web ou API. Voici comment il fonctionne :

Tout d'abord, il vérifie si le client a fourni un jeton d'authentification dans l'en-tête de la requête.

Si aucun jeton n'est trouvé, il renvoie une erreur indiquant que l'accès est refusé car aucun jeton n'a été fourni.

Si un jeton est trouvé, il est extrait de l'en-tête de la requête.

Ensuite, le jeton est vérifié en utilisant une clé privée spécifiée dans un fichier appelé "privateKey.mjs". Cela garantit que le jeton est valide et n'a pas été modifié.

Si le jeton est invalide ou a expiré, une erreur est renvoyée indiquant que l'utilisateur n'est pas autorisé à accéder à la ressource.

Si le jeton est valide, l'ID de l'utilisateur est extrait du jeton décodé.

Ensuite, il vérifie si l'ID de l'utilisateur dans la requête correspond à celui extrait du jeton. Si les IDs ne correspondent pas, une erreur est renvoyée indiquant que l'identifiant de l'utilisateur est invalide.

Si tout est correct, la fonction middleware appelle la fonction "next()" pour passer au middleware suivant dans la chaîne de traitement des requêtes.

En résumé, cette fonction middleware assure que seuls les utilisateurs authentifiés et autorisés peuvent accéder aux ressources protégées, en vérifiant l'authenticité du jeton d'authentification et en comparant l'identifiant de l'utilisateur dans le jeton avec celui de la requête.

## Mesures de sécurité mises en place.

Nous n’avons pas pris de mesure de sécurité autres que la validation lors de la création des donnée via « «validate ».

## Explication détaillée de chaque fonctionnalité technique.

| **Verbe HTTP** | **URI** | **Description** | **Exemple d'utilisation** | **Fait** |
| --- | --- | --- | --- | --- |
| GET | /api/books/ | Obtenir la liste des livres | Afficher tous les livres disponibles | Oui |
| GET | /api/books/:id/ | Obtenir un livre | Afficher les détails d'un livre spécifique | Oui |
| GET | /api/books/:id/comments/ | Obtenir tous les commentaires d'un livre | Afficher les commentaires d'un livre | Oui |
| GET | /api/books/:id/notes/ | Obtenir toutes les notes d'un livre | Afficher les notes attribuées à un livre | Non |
| POST | /api/books/ | Ajouter un nouveau livre avec catégorie et auteur | Ajouter un nouveau livre à la bibliothèque | Oui |
| POST | /api/books/:id/comments/ | Commenter un livre | Laisser un commentaire sur un livre | Oui |
| POST | /api/books/:id/notes/ | Évaluer un livre | Donner une note à un livre | Non |
| PUT | /api/books/:id/ | Modifier un livre | Mettre à jour les détails d'un livre | Oui |
| DELETE | /api/books/:id/ | Supprimer un livre | Supprimer un livre de la bibliothèque | Oui |
| GET | /api/categories/ | Obtenir la liste des catégories | Afficher toutes les catégories disponibles | Oui |
| GET | /api/categories/:id | Obtenir une catégorie | Afficher les détails d'une catégorie | Oui |
| GET | /api/categories/:id/books/ | Obtenir tous les livres de cette catégorie | Afficher tous les livres d'une catégorie | Oui |
| GET | /api/authors/ | Obtenir la liste des auteurs | Afficher tous les auteurs disponibles | Oui |
| GET | /api/authors/:id/ | Obtenir un auteur | Afficher les détails d'un auteur spécifique | Oui |
| GET | /api/authors/:id/books/ | Obtenir tous les livres de cet auteur | Afficher tous les livres d'un auteur | Oui |
| GET | /api/users/:id/books/ | Obtenir tous les livres d'un utilisateur | Afficher tous les livres d'un utilisateur | Oui |
| POST | /api/login/ | S'authentifier pour obtenir un token | Se connecter pour obtenir un token d'accès | Oui |

# Test

Nous n’avons pas eu le temps de faire des test via le package "vitest".

# Conclusion

## Organisation de la gestion du code avec Git/GitHub

Pour l'organisation de la gestion du code avec Git/GitHub, nous avons suivi une approche collaborative fluide. Nous n'avons rencontré aucun conflit lors de notre collaboration. Initialement, nous avons travaillé individuellement sur nos propres dossiers, chacun se concentrant sur des parties spécifiques du projet. Après avoir complété nos tâches individuelles, nous avons fusionné nos dossiers pour réunir notre code. Cette approche nous a permis de travailler de manière efficace tout en évitant les conflits de fusion et en maintenant une cohérence dans le développement du projet.

Conclusion générale sur le projet.

Ce projet à été pour nous vraiment important dans notre compréhension des API car avec toutes la théorie reçu d’un coup nous avions du mal a assimiler et notre motivation ce dégradais. Mais durant le projet nous avons eu un regain de motivation car nous avons on eu le temps de refaire les pdf et de bien assimiler la matière. Malheureusement le projet n'a pas pu être complété dans sa totalité en raison de la charge de travail considérable spécifiée dans le cahier des charges. Toutefois, nous avons réussi à mettre en œuvre les fonctionnalités principales telles que la mise en place des routes pour les catégories et les livres, ainsi que le système d'authentification pour obtenir le Token. Les fonctionnalités manquantes incluent la gestion des rôles et les points de sécurité précédemment abordés. En outre, la dockerisation et l'automatisation des tests unitaires n'ont pas été réalisées en raison de contraintes de charge de travail et de temps insuffisant.

## Conclusion personnelle sur le projet.

Lucas :

Ce projet a été un moyen pour moi d'appliquer les connaissances acquises lors du module 295. J’ai bien aimé le faite de pouvoir appliquer la théorie apprise lors développement de l'API. Malgré quelques petits obstacles rencontrés et le gros obstacle de la liaison à la table Customer. A la fin de ce projet je me sens motivé à le continuer lors du module front-end. Je trouve que notre groupe a bien fonctionner.

Partie Sofiene :

La faite de me replonger dans les MCD et MLD ma plus. La faite que ce projet m’ait permis de bien mieux comprendre la matière est un plus. Notre groupe a très bien fonctionner, nous avons bien réussi à répartir les tache.

## Auto-évaluation :

Dans cette partie de la conclusion, la note que nous nous sommes attribués sera expliquer critère par critère :

|  |  |  |
| --- | --- | --- |
| Critère | Note | Explication |
| Rythme de travail | ACQUIS | Nous avons acquis les exigences demandées, mais nous n’avons pas eu le temps de faire toutes les routes bonus. |
| Qualité du travail | ACQUIS | Nous estimons avoir un travail transmissible mais l’absence des commentaires via swagger reste un point à faire. Et la faite que le rapport n’est pas assez complet. Nous considérons avoir manqué de temps notamment à cause des bugs rencontrer pour les associations user |
| Niveau de maîtrise technique | ACQUIS | Nous estimons maitriser l’ensemble des technologies de ce projet mais loin de la maitrise totale du Backend d’API. |
| Autonomie | LARGEMENT ACQUIS | Dans l’ensemble nous avons été très autonomes, nous avons su surmonter les difficultés. |
| Processus de travail | LARGEMENT ACQUIS | Nous avons respecté les contraintes imposer durant ce projet. De plus l’ensemble des documents demander ont été fournis. |
| Expression orale et écrite Technique de présentation | ACQUIS | Dans l’ensemble nous avons bien documenter via insomnia notamment mais le commentaire reste un manque dans notre projet. |
| Approche de la durabilité et de l’innovation | ACQUIS | Nous avons respecté la durabilité sans apporter de réelle innovation. |
| Aptitude au travail en équipe Gestion des conflits Communication | LARGEMENT ACQUIS | Nous sommes contents d’avoir travailler ensemble car il n’y a eu aucun problème est que nous étions les deux autant impliquer. |
| Notes | **ACQUIS** | **/** |

Les fichier PDF de notre autoévaluation est disponible sous :

P\_WEB295\Doc\ AutoEvaluation\_P\_WEB-SBA-LLN

## Critiques

La réalisation de ce projet est probablement faisable en 24 périodes à deux lorsque l'on maîtrise bien le sujet. Cependant, étant donné que nous n'avions eu que ~18 séances de théorie consacrées au sujet et que nous devions souvent copier-coller du code affiché au tableau, nous avons dû consacrer une bonne partie du temps à refaire les étapes des 14 PDF avant de pouvoir réellement commencer à travailler sur le projet.

# Webographie/Bibliographie/Glossaire

- <https://sequelize.org/docs/v6/>

- <https://stackoverflow.com/>

- Antoine Mveng

- Les 14 pdf

# 7. Utilisation d’IA

L'intelligence artificielle ChatGPT a été utiliser dans le rapport. Notamment pour l’explication de l’algorithme d’authentification.