05/03/2024

Tiago Rodrigues Sousa, Evin Paramanathan

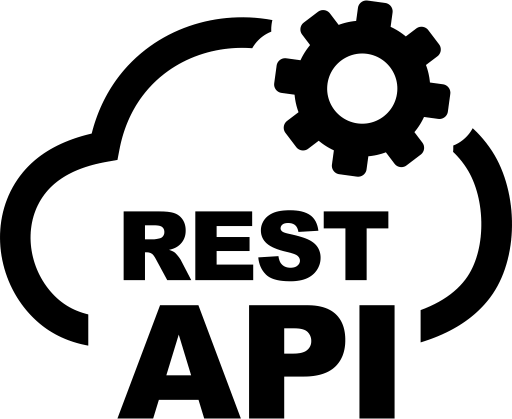
ETML – Vennes

Chef de projet : Antoine Mveng

Salle : B11

Rapport de projet

P\_Web\_C295



Rapport de projet

Table des matières

[Introduction 2](#_Toc161134421)

[Analyse 2](#_Toc161134422)

[1. Planification 2](#_Toc161134423)

[2. Verbe http 2](#_Toc161134424)

[3. Routes : 2](#_Toc161134425)

[4. Base de données 3](#_Toc161134426)

[5. Structure du code 4](#_Toc161134427)

[6. Schéma de l’architecture 5](#_Toc161134428)

[Réalisation 5](#_Toc161134429)

[1. Authentification et la gestion des rôles 5](#_Toc161134430)

[2. Sécurité 5](#_Toc161134431)

[Test 6](#_Toc161134432)

[ChatGPT 6](#_Toc161134433)

[Conclusion 6](#_Toc161134434)

[1. Gestion du code 6](#_Toc161134435)

[2. Général 6](#_Toc161134436)

[3. Personnelle 6](#_Toc161134437)

[a) Tiago 6](#_Toc161134438)

[b) Evin 6](#_Toc161134439)

[4. Planification du projet 7](#_Toc161134440)

[Webographie 7](#_Toc161134441)

[1. Docker : 7](#_Toc161134442)

# Introduction

L'objectif de ce projet est de mettre en pratique le module ICT C295, qui consiste à réaliser le backend pour des applications. Pour ce faire, nous devons mettre en place une API qui permet de partager notre passion pour la lecture. Nous travaillons en binôme pour mener à bien ce projet.

Pour réaliser ce projet, nous disposons d'un PC, de VS Code, d'un accès à Internet, de supports de cours, ainsi que de technologies telles que Node.js, npm, Express.js, Sequelize, MySQL, Swagger et Insomnia. Nous avons à notre disposition 24 périodes pour mener à bien ce projet.

Avant d'entrer dans les détails, regardons simplement ce qu'est une API.

API est l’abréviation d'Application Programming Interface ou Interface de Programmation d'Application en français. Une API est une interface logicielle qui permet de relier un service à un autre ou un logiciel à un autre. Cela permet d'échanger des données et des fonctionnalités.

# Analyse

## Planification

La planification des taches à réaliser pour le projet a été fait sur Trello, ça permet de créer un tableau et de l’adapter pour faire un suivit du projet plus personnalisé.

<https://trello.com/invite/b/CKSDCYuq/ATTI9eb058082f456ec9f8f1b74df1999ac35786C719/pweb295>

## Verbe http

|  |  |
| --- | --- |
| Verb | Action |
| GET | Recherche |
| POST | Créer |
| PUT | Mettre à jours |
| DELETE | Supprimer |

## Routes :

Livre :

|  |  |  |  |
| --- | --- | --- | --- |
| Nom du fichier | Verbe | URI | Action |
| addLivre | POST | http://localhost:3000/api/livres | Permet d’ajouter un livre |
| deleteLivre | DELETE | http://localhost:3000/api/livres/3 | Permet de supprimer un livre |
| getAllLivresCategorie | GET | http://localhost:3000/api/categories/1/livres | Permet de rechercher un le livre spécifique d’une catégorie |
| getLivre | GET | http://localhost:3000/api/livres | Permet de rechercher un livre spécifique |
| getLivreId | GET | http://localhost:3000/api/livres/2 | Permet de rechercher un livre spécifique grâce à son IP |
| updateLivre | PUT | http://localhost:3000/api/livres/2 | Permet de mettre à jour un livre |

Catégorie :

|  |  |  |  |
| --- | --- | --- | --- |
| **addCategorie** | POST | http://localhost:3000/api/categories | Permet d’ajouter une catégorie |
| **getCategorie** | GET | http://localhost:3000/api/categories | Permet de rechercher une catégorie |
| **getCategorieId** | GET | http://localhost:3000/api/categories/1 | Permet de rechercher une catégorie à partir d’un id |

Commentaire :

|  |  |  |  |
| --- | --- | --- | --- |
| **addCommentaires** | POST | http://localhost:3000/api/commentaires | Permet d’ajout un commentaire |
| **getCommentaire** | GET | http://localhost:3000/api/commentaires | Permet de rechercher des commentaire |

Login :

|  |  |  |  |
| --- | --- | --- | --- |
| **Login** | POST | http://localhost:3000/api/login |  |

## Base de données

Pour faire la base de données nous avons décider de faire 6 tableaux, comme on peut le voir dans l’image ci-dessous.

Premièrement, il y a la table principale "ouvrage", celle-ci est reliée à 5 autres tables. Ensuite, il y a la table "catégorie" que nous avons décidé de ne pas mettre directement dans "ouvrage", car si un livre a plusieurs catégories, cela pourrait poser un problème pour les recherches.

Deuxièmement, il y a les deux tables "écrivain" et "éditeur", pour chaque écrivain et chaque éditeur qui contient le nom pour les deux tables, et dans la table "écrivain", il y a le prénom en plus du nom.

Ensuite, il y a la table "utilisateur" dans laquelle il y a le pseudo, le mot de passe, la date d’entrée, le nombre d’ouvrages proposés, le nombre de commentaires et leur rôle.

Et pour finir, il y a la table "commentaire" dans laquelle il y a l'appréciation et le commentaire.

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

Description générée automatiquementUne image contenant texte, capture d’écran, diagramme, Post-it

Description générée automatiquement

## Structure du code

Le projet (partie code) est organisé de cette façon : dans le dossier API qui est la racine de l’API il y a un gitignore, les packages json et un dossier src, à l’intérieur de celui-ci il y a un fichier app.mjs et 4 dossiers.

Premièrement il y a le dossier auth contient deux fichiers pour faire le système d’authentification auth.mjs et private\_key.mjs.

Ensuite il y a le dossier db qui contient aussi deux fichiers, mocks-livre.mjs et sequelize.mjs

L’avant dernière dossier est le dossier models qui contient 6 fichiers, chaque fichier est une table de la base données : donc il y a les fichiers suivant t\_categorie.mjs, t\_commentaire.mjs, t \_ecrivain.mjs, t\_editeur.mjs, t\_ouvrage.mjs et t\_utilisateur.

Pour finir il y a le dossier routes qui contient le plus de fichier. Elle contient toutes les fonctionnalités de l’API donc il y a 12 fichiers dans ce dossier voici la liste de fichiers :

* addCategorie.mjs
* addCommentaires.mjs
* addLivre.mjs
* deleteLivre.mjs
* getAllLivreCategorie.mjs
* getCategorie.mjs
* getCategorieId.mjs
* getCommantaire.mjs
* getLivre.mjs
* getLivre.mjs
* getLivreId.mjs
* helper.mjs
* login.mjs
* updateLivre.mjs

## Schéma de l’architecture

# Réalisation

## Authentification et la gestion des rôles

Pour l’instant, il n’y a qu’un champ utiRole dans la table utilisateur où il y a "admin" et "user". Ils ne sont pour l’instant pas utilisés pour faire la gestion des rôles. Cela n’a pas été fait par manque de temps.

## Sécurité

Le système de token a été mis en place. Il est obligatoire pour pouvoir utiliser l'API. Nous avons également mis en place un hachage pour le mot de passe afin d'éviter que celui-ci soit stocké en clair dans la base de données. Ce sont les seules sécurités qui ont été mises en place en raison d'un manque de temps.

# Test

# ChatGPT

Pour la partie code, ChatGPT nous a servi à comprendre des détails que nous ne savions pas ou que nous ne comprenions pas. ChatGPT nous a également servi à créer des données factices pour les insérer dans la base de données. Du côté du rapport, ChatGPT nous a seulement aidés à corriger des fautes d’orthographe et de grammaire.

# Conclusion

## Gestion du code

Nous avons utilisé GitHub comme gestionnaire de versionning. Il y avait 4 branches :

* main : rassemble toutes les branches
* MiseEnPlaceAPI : c’est là où est stocké le code de l’api et insomnia
* Database : il y a le MCD/MLD/MPD
* Documentation : stocke la documentation

## Général

Le projet est plutôt abouti, toutes les fonctionnalités créées dans le dossier fonctionnent, mais le projet n’est pas fini à 100% car la charge de travail est plutôt élevée par rapport au nombre de périodes dont nous disposons (24 périodes) et aussi par rapport au nombre de personnes par groupe. Il y a plusieurs choses qui pourraient être améliorées : la documentation avec Swagger qui n’existe pas pour l’instant, plusieurs routes (celles qui étaient obligatoires ont été faites), un système de gestion de rôles efficace, la mise en place de tests unitaires et la dockerisation.

## Personnelle

### Tiago

Ce projet m’a permis d’approfondir mes connaissances acquises lors du module C-295 et m’a fait découvrir la gestion des relations entre les tables. Le projet c’est bien passé et la communication avec mon collègue c’est très bien passé. Je trouve dommage de ne pas avoir eu plus de temps pour aboutir à une api complète.

### Evin

Pour ma part ce projet m’as permis d’apprendre de nouvelle chose et d’appliquer ce qu’on vu durant le module. Ça m’a aussi permis de me replonger un peu du coté base de données avec mcd mld et les inserts.

Du côté gestion du projet, je trouve qu’on a réussi à bien nous organiser et que la communication entre mon collègue est moi était claire.

## Planification du projet

Pour la planification nous avons utilisé Trello comme dit dans le point 1 dans analyse, et cela nous a permis de décortiquer le projet pour mieux s’organiser, et vers la fin du projet quand on a compris que ça va être compliqué de tout finir, le fait d’avoir découper le projet auparavant nous a permis de nous réorganiser pour faire les points les plus important pour atteindre le minimum pour réussir ce projet.

# Webographie

## Docker :

<https://welovedevs.com/fr/articles/docker-node-api/>