Logotipo, Icono

Descripción generada automáticamente

**ECOLE TECHNIQUE  
ECOLE DES MÉTIERS LAUSANNE**

Frontend

P\_Web 294



Joachim Berchel & Thomas Nardou

MID2 - 2024

18.03.2024 – 27.05.2024

Antoine Mveng

Vennes - ETML

Table des matières

[1. Introduction 3](#_Toc166588490)

[2. Analyse 4](#_Toc166588491)

[2.1 Planification 4](#_Toc166588492)

[2.2 Base de données 5](#_Toc166588493)

[2.3 Analyse de la structure du code 6](#_Toc166588494)

[2.4 Schéma de l’architecture 8](#_Toc166588495)

[3. Réalisation 9](#_Toc166588496)

[3.1 Mesures prises pour la sécurité 9](#_Toc166588497)

[3.2 Technique 9](#_Toc166588498)

[3.3 GitHub 9](#_Toc166588499)

[4. Test 10](#_Toc166588500)

[5. Conclusion 11](#_Toc166588501)

[5.1 Conclusion générale 11](#_Toc166588502)

[5.2 Conclusion personnelle 11](#_Toc166588503)

[Thomas : 11](#_Toc166588504)

[Joachim : 11](#_Toc166588505)

[5.3 Critiques 11](#_Toc166588506)

[6. Webographie / Bibliographie / Glossaire 12](#_Toc166588507)

[7. Utilisation d’IA 12](#_Toc166588508)

# Introduction

L’objectif de ce projet est de réaliser le front d’une application Web interactive avec Vue. Le travail s’effectue en groupe de 2. Le projet s’effectue du 18 mars 2024 jusqu’au27 mai 2024. Le projet est lié au module C294 « Réaliser le frontend d’une application Web interactive ». Pour ce projet il a été mis à disposition : Un ordinateur avec le système d’exploitation Windows 10, l’éditeur de texte Visual Studio Code, un serveur local pour pouvoir exécuter les routes de l’API, un navigateur Web pour pouvoir effectuer des recherches et un accès à internet. À noter que nous utilisons l’API fait lors du projet du module C295 « Réaliser le backend pour des applications », ainsi ces deux projets sont liés et les rapports de ces deux projets peuvent donc se complémenter.

# 2. Analyse

## 2.1 Planification

Nous avons premièrement voulu utiliser GitHub Projects mais nous n’avons pas trouvé comment récupérer la date à laquelle la tâche a été fini, nous avons donc rapidement reporté notre choix sur Trello afin de gagner du temps.

De plus Trello avait déjà été utilisé auparavant et ce projet allant dans la continuité du projet « P\_Web 295 » nous avons donc récupéré le tableau Kanban anciennement créé.

Lien pour accéder à la planification Trello :

<https://trello.com/invite/b/1XpbzDPe/ATTI15cb97128bf2b139ef7df8a7c1b23815DCDEEB8D/papi>

## 2.2 Base de données

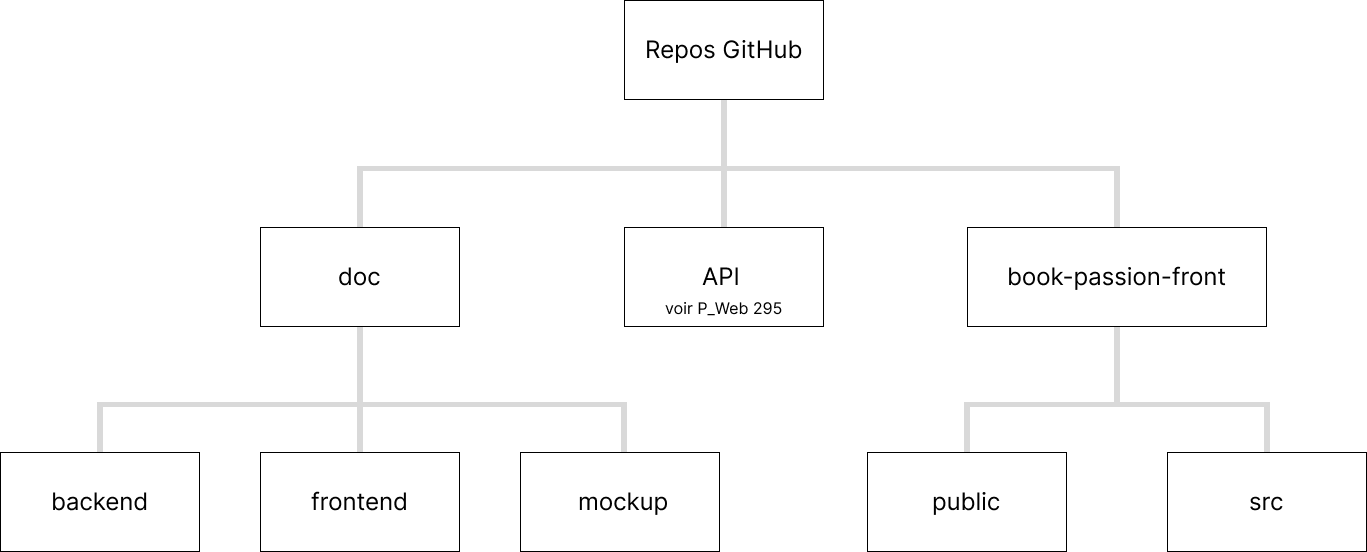
L’aspect base de données ayant déjà été vue lors du projet « P\_Web 295 » et la connexion à la base de données s’effectuant uniquement depuis le backend nous n’avons pas trouvé pertinent d’afficher une copie du rapport de l’autre projet.

**Néanmoins** nous avons rajouté un champ « title » à la table « t\_comments » afin que les commentaires soient plus pertinents.

Se référer au rapport « R\_P\_WEB-RapportAPI » du projet « P\_Web 295 ».

## 2.3 Analyse de la structure du code

Tout d’abord voici un diagramme de notre arborescence :



A la racine se trouve le dossier du repositories de GitHub c’est lui qui contient tout notre travail des projets « P\_Web 295 » et « P\_Web 294 ».

Dans le répertoire GitHub il y a 3 dossiers, le premier doc qui contient :

* La documentation du projet « P\_Web 295 » dans le dossier « backend ».
* La documentation de ce projet (P\_Web 294) dans le dossier « frontend ».
* Les maquettes de l’application dans le dossier « mockup ».

Le second dossier « API » contient le projet « P\_Web 295 », pour plus d’informations se référer au rapport de ce projet (dans « doc/backend »).

Le troisième dossier « book-passion-front » contient le projet « P\_Web 294 », il contient donc la racine du projet (index.html) et les dépendances, il contient aussi le dossier « public » qui stock les images et le dossier src.

Analyse du dossier « src » :

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

Description générée automatiquement

Dans le dossier « assets » on importe le style de font – nous utilisons la police « Kanit »

Dans le dossier « components » il y a plusieurs dossiers :

* « Global » - contient les composants qui sont utilisé partout, donc sur plusieurs pages tel le footer et le header.
* « Lobby » - contient les composants qui sont utilisé dans la page d’accueil.
* « oneBook » - contient les composants qui sont utilisé dans la page d’affichage d’un seul livre.
* « AllBooks » - contient les composants qui sont utilisé dans la page affichant tous les livres.
* « Login » - contient les composants utilisés dans la page d’identification.
* « userPage » - contient les composants utilisés dans la page d’affichage des livres de l’utilisateur.

Dans le dossier « router » se situe le fichier « index.js » qui se charge de faire le routage du site.

Dans le dossier « utils » se situe des fichiers faisant certaines actions que nous voudrions pouvoir appeler afin de simplifier le code.

Dans le dossier « views » se situe les vues de chaque page.

## 2.4 Schéma de l’architecture

Le component Login/LoginSquare fait un appel Axios à l’API :

« POST http://localhost:3000/api/login » - Effectue la connexion à un utilisateur

Le component oneBook/addComment fait un appel Axios à l’API :

« POST http://localhost:3000/api/comments » - Ajouter un commentaire

Le component userPage/userBook fait un appel Axios à l’API :

« DELETE http://localhost:3000/api/books/:bookId » - Supprime un livre grâce à son id

La vue views/addBooksView fait plusieurs appels Axios à l’API :

« GET http://localhost:3000/api/categories » - Récupère toutes les catégories différentes

« POST http://localhost:3000/api/books » - Ajoute des livres dans l’API

La vue views/AllBooksView fait plusieurs appels Axios à l’API :

« GET http://localhost:3000/api/categories » - Récupère toutes les catégories pour faire une recherche imbriquée.

« GET http://localhost:3000/api/categories/:id/books » - Récupère les livre selon leur catégorie

La vue views/LobbyView fait un appel Axios à l’API :

« GET http://localhost:3000/api/books?order=createdAt » - Récupère les livres en les ordonnant selon leurs dates de sortie

La vue views/oneBookView fait des appels Axios à l’API :

« GET http://localhost:3000/api/books/bookId » - Récupère un livre grâce à son id.

« GET http://localhost:3000/api/comments/:commentId?order=createdAt » - Récupère un commentaire grâce à son id et l’ordre selon sa date de création.

« PUT http://localhost:3000/api/books/:bookId » - Modifie la moyenne des notes du livre.

La vue views/updateBook fait des appels Axios à l’API :

« GET http://localhost:3000/api/books/:bookId » - Récupère le livre avant sa modification

« GET http://localhost:3000/api/categories » - Récupère la catégorie du livre.

« PUT http://localhost:3000/api/books/:bookId » - Modifie le livre avec les entrées voulu par l’utilisateur

La vue views/userPage fait un appel Axios à l’API :

« GET http://localhost:3000/api/user/:userId/books » - Récupère les livres de l’utilisateur

## 2.5 Piste d’amélioration

Par manque de temps nous n’avons pas pu implémenter toutes les fonctionnalités demandées ou les peaufiner comme voulu, voici donc nos points d’améliorations :

1. L’envoi de commentaire peut être spammer ce qui permet d’augmenter ou de descendre la moyenne volontairement, afin d’empêcher le spam de bot nous aurions voulu mettre en place un Captcha.
2. Les cookies utilisés ne sont pas suffisamment protégés, nous aurions voulu les stocker dans le local storage.
3. Les livres ajoutés par les utilisateurs n’ont pas de vérification logique, c’est-à-dire qu’un livre sorti à une date postérieur à la date actuelle.
4. Par manque de temps la modification d’un livre n’est pas implémentée.
5. Il faudrait réunir tous les appels Axios à l’API dans un même fichier dans le dossier utils.

## 2.6 Éco-conception

Comme point d’écoconception nous utilisons le lazy loading dans notre routeur et toutes nos images sont en format webp ou en format ico.

# 3. Réalisation

## 3.1 Mesures prises pour la sécurité

Afin de garantir la sécurité de l’utilisateur nous utilisons le système de token JWT lors du login.

Une redirection à lieu lorsqu’un utilisateur non connecté essaie d’accéder à une ressource qui demande un compte utilisateur ou supérieur.

Les entrées effectuées par un utilisateur sont protégées contre les injections.

## 3.3 GitHub

N’étant que deux à travailler sur ce projet nous n’avons pas trouvé d’utilité à la création de branche, nous avons donc tout pousser sur le « main » et nous sommes mis d’accord par oral sur quel fichier nous pouvions travailler.

# 4. Test

Nous n’avons pas eu assez de temps pour mettre en place une stratégie de test.

# 5. Conclusion

## 5.1 Conclusion générale

## 5.2 Conclusion personnelle

### Thomas :

### Joachim :

J’ai beaucoup apprécié ce projet. Il m’a appris à faire du Vue mais m’a aussi aidé à consolider mes bases en HTML, CSS et javascript. J’ai appris à faire un lien entre le backend et le frontend et ainsi j’ai aussi pu consolider mes connaissances sur le backend utilisé et créé lors du projet « P \_Web 295 ». J’ai l’impression d’avoir beaucoup appris lors de ce projet.

# 6. Webographie / Bibliographie / Glossaire

* <https://vuejs.org/>
* <https://www.vuemastery.com/>
* Support de cours du module I-295

# 7. Utilisation d’IA

Nous n’avons utilisé aucune intelligence artificielle pour nous aider lors du projet.