P\_040-Web2



David Dieperink, Stefan Petrovic  
Robustiano Lombardo, Alexis Rojas

CID2a

ETML

Gregory Charmier

Table des matières

[1 Spécifications 4](#_Toc103956062)

[1.1 Titre 4](#_Toc103956063)

[1.2 Description 4](#_Toc103956064)

[1.3 Matériel et logiciels à disposition 4](#_Toc103956065)

[1.4 Prérequis 4](#_Toc103956066)

[1.5 Cahier des charges 4](#_Toc103956067)

[1.5.1 Objectifs et portée du projet (objectifs SMART) 4](#_Toc103956068)

[1.5.2 Fonctionnalités requises (du point de vue de l’utilisateur) 4](#_Toc103956069)

[1.5.3 Travail à réaliser par l'apprenti 5](#_Toc103956070)

[1.5.4 Points supplémentaires 6](#_Toc103956071)

[1.5.5 Si le temps le permet … 6](#_Toc103956072)

[1.5.6 Méthodes de validation des solutions 6](#_Toc103956073)

[1.6 Les points suivants seront évalués 6](#_Toc103956074)

[1.7 Validation et conditions de réussite 6](#_Toc103956075)

[2 Planification Initiale 7](#_Toc103956076)

[3 Analyse 7](#_Toc103956077)

[3.1 Opportunités 7](#_Toc103956078)

[3.1.1 Difficultés potentielles 7](#_Toc103956079)

[3.1.2 Solutions éventuelles 7](#_Toc103956080)

[3.2 Document d’analyse et conception 7](#_Toc103956081)

[3.2.1 Base de données 7](#_Toc103956082)

[3.2.2 Mind Map 9](#_Toc103956083)

[3.2.3 Maquette site web 9](#_Toc103956084)

[3.2.4 Différences maquette et site web 10](#_Toc103956085)

[3.3 Conception des tests 11](#_Toc103956086)

[3.3.1 Tests d’intégration 11](#_Toc103956087)

[3.3.2 Tests fonctionnels 12](#_Toc103956088)

[3.4 Planification détaillée 13](#_Toc103956089)

[4 Réalisation 13](#_Toc103956090)

[4.1 Dossier de Réalisation 13](#_Toc103956091)

[4.1.1 Version des outils logiciels utilisés 13](#_Toc103956092)

[4.1.2 Configurations spécifiques 13](#_Toc103956093)

[4.1.3 Heroku 13](#_Toc103956094)

[4.1.4 Notre utilisation de Laravel 14](#_Toc103956095)

[4.1.4.1 Vues 14](#_Toc103956096)

[4.1.4.2 Controller 15](#_Toc103956097)

[4.1.4.3 Model 15](#_Toc103956098)

[4.1.4.4 Migrations 17](#_Toc103956099)

[4.1.4.5 Middleware 17](#_Toc103956100)

[4.1.4.6 Factories / Seeds 17](#_Toc103956101)

[4.1.5 Notre utilisation de Tailwind 18](#_Toc103956102)

[4.1.5.1 Composants 18](#_Toc103956103)

[4.1.5.2 CDN 19](#_Toc103956104)

[4.1.6 Organisation – Répartition des tâches 19](#_Toc103956105)

[4.2 Modifications 19](#_Toc103956106)

[5 Ecoconception 20](#_Toc103956107)

[5.1 Réduction de la taille du DOM 20](#_Toc103956108)

[5.2 Le Scroll 20](#_Toc103956109)

[5.3 Animation 20](#_Toc103956110)

[6 Tests 20](#_Toc103956111)

[6.1 Dossier des tests 20](#_Toc103956112)

[6.1.1 Tests d’intégrations 20](#_Toc103956113)

[6.2 Tests CI 21](#_Toc103956114)

[7 Conclusion 22](#_Toc103956115)

[7.1 Bilan des fonctionnalités demandées 22](#_Toc103956116)

[7.1.1 Fonctionnalités requises 22](#_Toc103956117)

[7.2 Bilan de la planification 22](#_Toc103956118)

[7.3 Bilan personnel 22](#_Toc103956119)

[7.3.1 Avis David 22](#_Toc103956120)

[7.3.2 Avis Alexis 22](#_Toc103956121)

[7.3.3 Avis Stefan 22](#_Toc103956122)

[7.3.4 Avis Robustiano 23](#_Toc103956123)

[8 Divers 23](#_Toc103956124)

[8.1 Journal de travail 23](#_Toc103956125)

[8.2 Webographie 23](#_Toc103956126)

[9 Annexes 23](#_Toc103956127)

# Spécifications

## Titre

Passion Lecture

## Description

Ce site est une bibliothèque en ligne, il permet de retrouver des informations sur des livres que les utilisateurs auront mis en ligne.

De plus les utilisateurs ont la possibilité de noter les livres.

## Matériel et logiciels à disposition

* Microsoft Windows 10
* PHP Storm
* Visual Studio Code
* UwAmp
* Internet

## Prérequis

Modules : 101, 431, 104, 302, 403, 404, 226A, 226B, 214, 133 et 151.

## Cahier des charges

### Objectifs et portée du projet (objectifs SMART)

Ce projet vise à mettre en œuvre les connaissances apprises dans les modules 133 et 151, qui se déroulent en parallèle au projet. Au final, l’application réalisée devra être exploitable et livrable. Dès lors, on attend un rendu professionnel et un soin particulier dans la documentation du projet.

### Fonctionnalités requises (du point de vue de l’utilisateur)

Le site web / application aura les pages suivantes :

* Une page d’accueil comprenant une explication de l’utilité du site ainsi que les cinq derniers ouvrages ajoutés (accès tout public).
* Une page comprenant la liste des ouvrages par catégorie (accès tout public avec restrictions sur les liens).
* Une page d’ajout d’un ouvrage (accès utilisateur).
* Une page permettant d’ajouter une appréciation à un ouvrage (accès utilisateur).

PS : le pied-de page du site doit faire mention de la personne qui a créé l’application ainsi que le moyen de la contacter.

### Travail à réaliser par l'apprenti

* Introduction
  + Comprend une brève explication du projet (½ page)
* Analyse
  + Contiendra une analyse quant à la réalisation et à la mise en page du HTML (1 page)
  + Contiendra une analyse de de la base de données à réaliser (MCD, MLD, MPD) (1 page)
  + Contiendra une analyse de la structure du code qui sera effectuée (Schéma UML, découpe du code …) (2 pages)
* Réalisation
  + Comprend une explication de l'algorithme utilisé pour gérer l’identification (1 page)
  + Comprend une explication sur l’appréciation moyenne d’un ouvrage (1/2 page)
  + Comprend un manuel d’utilisation du site du point de vue admin ou utilisateur (comment ajouter un ouvrage, comment ajouter une appréciation) (2 pages)
* Test
  + Comprend une explication des tests réalisés (Unit Test) (1 page)
* Conclusion
  + Comprend une conclusion générale sur le projet (½ page)
  + Comprend une conclusion personnelle sur le projet (½ page)
  + Comprend une critique constructive sur la planification du projet (½ page)
* Webographie / Bibliographie / Glossaire

Concernant la méthode de projet, elle devra être annexée au rapport et sera imposée par le chef de projet. Pour la partie pratique, il sera rendu terminé à 100% au plus tard à la fin de l’antépénultième séquence. Néanmoins, les échéances suivantes doivent être respectées durant le projet :

* Démonstration de l'avancement de l'application : Echéance -> Séquence 4
* Démonstration de l'avancement de l'application et auto-évaluation intermédiaire : Echéance -> Séquence 8
* Démonstration de l'avancement de l'application : Echéance -> Séquence 12

Une présentation et l'auto-évaluation finale doivent être rendues à l'enseignant au début du cours de l’avant-dernière séquence. Les présentations commencent à ce moment.

Règles pour la présentation :

* min. 10 minutes de présentation
* max. 5 minutes de démonstration

### Points supplémentaires

* MVC avec Laravel
* Git / github
* Méthodologie SCRUM
* CSS avec Tailwind
* Tests unitaires + CI
* Déploiement de l’application sur HEROKU en mode Production

### Si le temps le permet …

* Ajout d'une recherche pour retrouver un ouvrage
* Modification d'un ouvrage
* Suppression d’un ouvrage
* Ajout d’une personne
* Ajout d’un commentaire accompagnant une appréciation

### Méthodes de validation des solutions

Les tests d’intégrations seront écrits sous formes d’un tableau lors de l’analyse et ils seront réécris lors de réalisation avec le résultat obtenu.

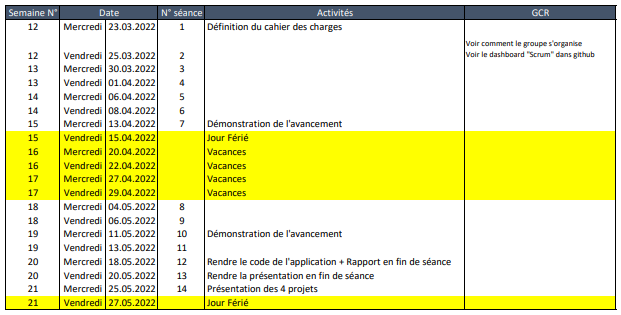
## Les points suivants seront évalués

* Le rapport
* Les planifications (initiale et détaillée)
* Le journal de travail
* Le code et les commentaires
* Les documentations de mise en œuvre et d’utilisation

## Validation et conditions de réussite

* Compréhension du travail
* Possibilité de transmettre le travail à une personne extérieure pour le terminer, le corriger ou le compléter
* Etat de fonctionnement du produit livré

# Planification Initiale



# Analyse

## Opportunités

### Difficultés potentielles

* Comprendre le fonctionnement de Laravel + Tailwind
* Organisation du projet -> répartition des tâches à effectuer
* Mise en production du site web avec Heroku

### Solutions éventuelles

* Lire la documentation + faire l’initialisation avec les membres du groupe
* Répartition des tâches en fonctions des connaissances (au départ)

## Document d’analyse et conception

### Base de données

Nous avons créé le modèle conceptuel de données et le modèle logique pour avoir une représentation du stockage des données du site web. Pour créer ces 2 modèles nous nous sommes mis à 4 pour prendre l’avis de tout le monde est que tout fonctionne bien lors de l’implémentation de la base de données.

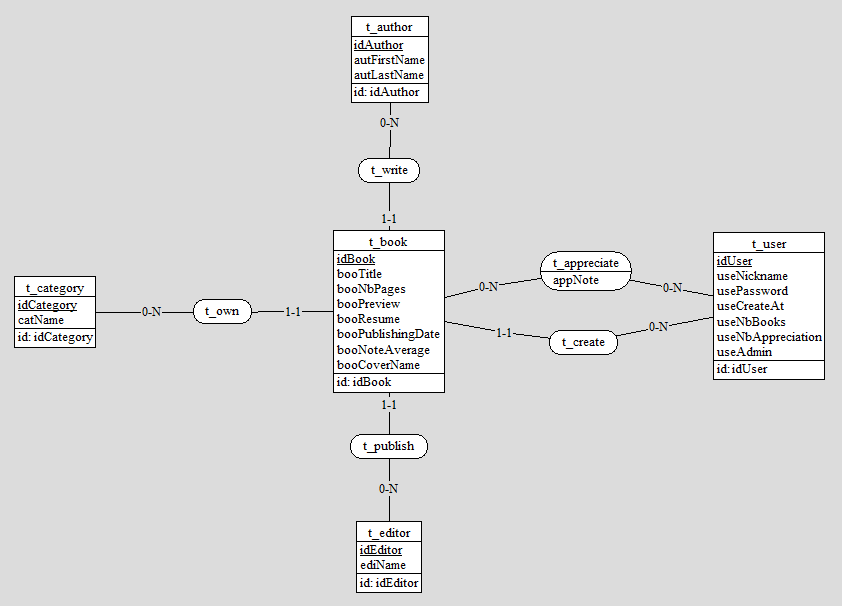
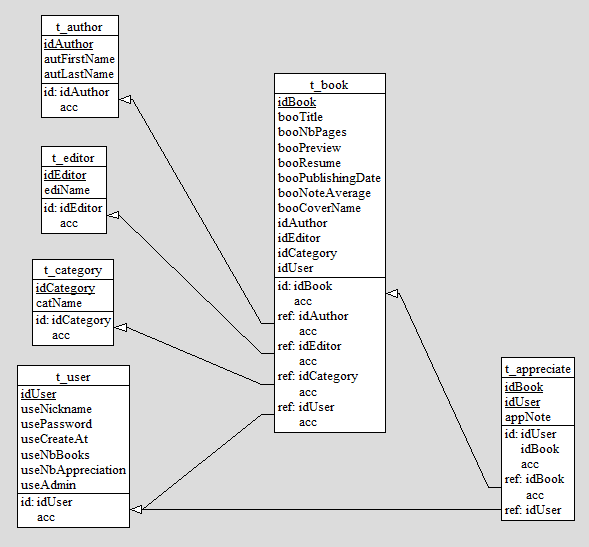
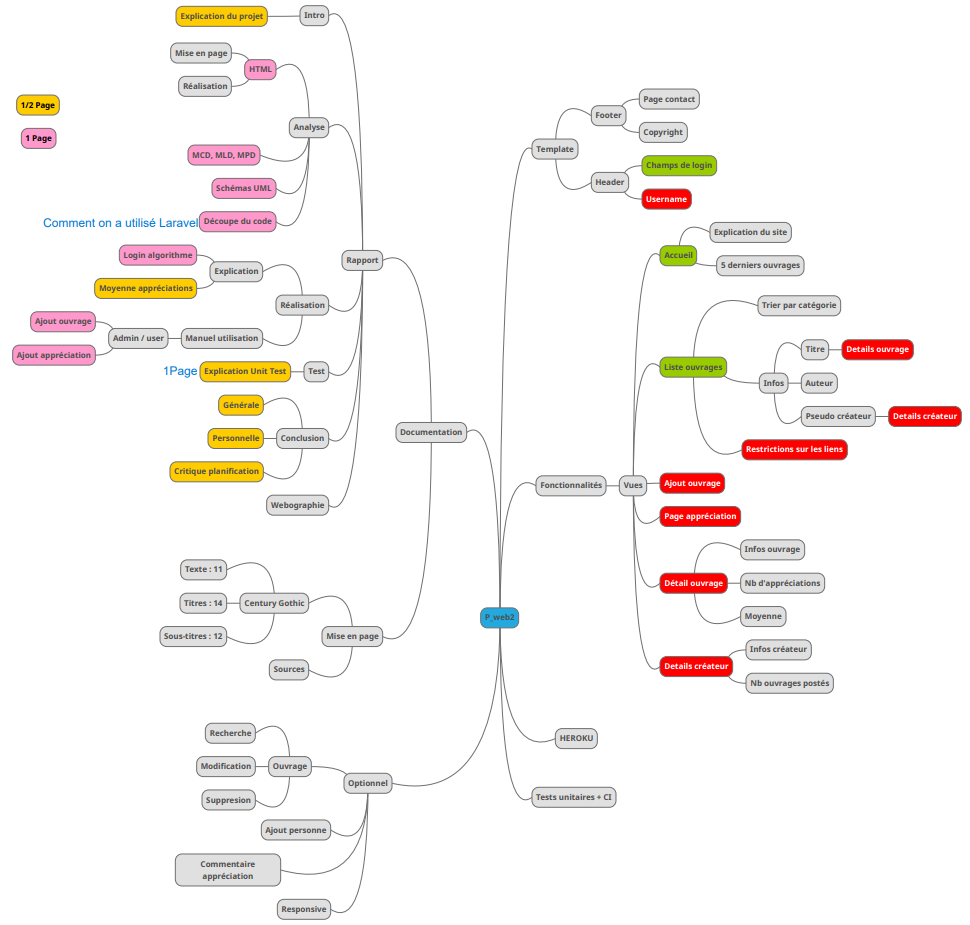


Figure 1 : MLD - Version 1

Figure 2 : MCD - Version 1

### Mind Map



### Maquette site web

Nous avons utilisé Figma pour faire notre maquette. La maquette est composée de plusieurs pages car nous avons fait chaque page de site web pour pouvoir les implémenter plus facilement. Voici le lien de la maquette :

<https://www.figma.com/file/KDZvY8uQ0cxHLlYnnxV2W4/Untitled?node-id=0%3A1>

### Différences maquette et site web

|  |  |  |  |
| --- | --- | --- | --- |
| Elément concerné | Maquette | Site | Raison |
| Carte d’affichage des ouvrages |  |  | Implémentation trop compliquée, mise en place problématique |
| Footer |  |  | Pas très esthétique après réflexion |
| Formulaire login |  |  | Le bouton back a été remplacé par une croix de fermeture. |
| Formulaire inscription |  |  | Le bouton back a été remplacé par une croix de fermeture. |
| Bouton de la barre de navigation |  |  | Problème de couleurs. |
| Formulaire d’ajout d’ouvrage |  |  | Retrait du bouton back et ajout d’un titre, uniformisation de la longueur des champs. |

## Conception des tests

### Tests d’intégration

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Titre | Fonctionnalité | Description | Etapes | Résultat attendu |
| Recherche livre réussie | Recherche d’un livre | Recherche d’un livre avec son nom | 1. Entrer le livre dans la barre de recherche 2. Appuyer sur le bouton « Search » | Le livre s’affiche lors de la recherche |
| Recherche livre échouée | Recherche d’un livre | Recherche d’un livre avec son nom | 1. Entrer le nom du livre dans la barre de recherche 2. Appuyer sur le bouton « Search » | Aucun livre s’affiche car le nom est faux |
| Filtrage livre par catégorie réussi | Filtrer les livres par catégories | Filtrer l’affichage des livres par catégories | 1. Choisir une catégorie dans la liste déroulante 2. Appuyer sur le bouton « Search » | Les livres de la catégories s’affichent |
| Filtrage livre par catégorie échoué | Filtrer les livres par catégories | Filtrer l’affichage des livres par catégories | 1. Choisir une catégorie dans la liste déroulante 2. Appuyer sur le bouton « Search » | Les livres de la catégories ne s’affichent pas car aucun livre n’est dans la catégorie. |
| Redirection sur la page de détail créateur | Redirection sur la page detailsCreator | Redirection sur la page de détail du créateur | 1. Appuyer sur le créateur du livre | La page de détail du créateur s’affiche |
| Redirection sur la page d’ajout d’un livre | Redirection sur la page addBook | Redirection sur la page d’ajout d’un livre | 1. Appuyer sur le bouton « Add book » | La page pour l’ajout d’un livre s’affiche |
| Redirection sur la page détail du livre | Redirection sur la page bookDetails | Redirection sur la page de détail du livre | 1. Appuyer sur le titre du livre | La page de détail du livre s’affiche |

### Tests fonctionnels

Scénario 1 : Login

|  |  |  |
| --- | --- | --- |
| Etape | Description | Remarque |
| Arrange | Informations de l’utilisateur |  |
| Act | Vérifier les informations reçues avec celle de la base données |  |
| Assert | Les informations reçues doivent correspondre à celle de la base de données |  |

Résultat : Le test fonctionnel fonctionne parfaitement.

Remarque :

|  |
| --- |
|  |

Scénario 2 : Confirmation du mot de passe (inscription)

|  |  |  |
| --- | --- | --- |
| Etape | Description | Remarque |
| Arrange | Mot de passe |  |
| Act | Vérifier que les deux champs soient les mêmes |  |
| Assert | Les deux champs doivent correspondre |  |

Résultat : Le test fonctionnel fonctionne parfaitement.

Remarque :

|  |
| --- |
|  |

Scénario 3 : Validation des données d’un formulaire

|  |  |  |
| --- | --- | --- |
| Etape | Description | Remarque |
| Arrange | Données du formulaire |  |
| Act | Vérifier qu’elles correspondent aux critères |  |
| Assert | Les données doivent correspondre aux critères demandés |  |

Résultat : Le test fonctionnel fonctionne parfaitement.

Remarque :

|  |
| --- |
|  |

## Planification détaillée

Aucun journal de travail n’a été demandé par le chef du projet car cela prend trop de temps et il préfère favoriser l’apprentissage de la technique.

# Réalisation

## Dossier de Réalisation

### Version des outils logiciels utilisés

|  |  |
| --- | --- |
| Outils / Logiciels | Version |
| PHP Storm | 2021.3.3.PS-213.7172.28 |
| UwAmp | 3.1.0 |
| PHP | 8.1.4 |
| MySQL | 5.7.11 |
| Laravel | 9 |
| Tailwind | 3.0.24 |
| Git Bash | 2.34 |
| Windows | 10 |

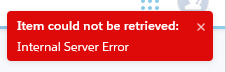
### Configurations spécifiques

Pour les configurations spécifiques nous avons créé des espaces de discussions sur le dépôt GitHub, cela permet que tous les membres du groupe puissent trouver l’information.

* Configuration de UwAmp en Host Virtual -> [lien](https://github.com/Alexis1476/etml-p040-web2/discussions/46)
* Configuration version de php à installer -> [lien](https://github.com/Alexis1476/etml-p040-web2/discussions/37)
* Configuration du ficher .env de laravel -> [lien](https://github.com/Alexis1476/etml-p040-web2/discussions/58)

### Heroku

La mise en production du site web grâce à Heroku n’a pas fonctionné. La liaison entre GitHub et Heroku ne fonctionne pas (18.05.2022). Lors de la connexion une erreur est générée, nous n’avons aucune idée de la provenance de celle-ci.



Suite à cela, nous avons essayé en utilisant le CLI de Heroku. Voici la marche à suivre que nous avons suivie.

1. **npm install –g heroku** cette commande installe heroku sur la machine.
2. **heroku login** cette commande permet de faire la connexion au site de heroku.
3. **heroku create** cette commande génère un projet heroku en local et sur le site d’heroku.
4. **git push heroku main** cette commande push heroku sur la branch main du dépôt GitHub.

Après cette dernière commande une erreur se produit, cela empêche donc de push la configuration heroku sur le dépôt GitHub.



Le push est rejeté, le dépôt refuse le push car la configuration de github côté serveur à un problème. [Lien](https://stackoverflow.com/questions/7986139/git-error-when-trying-to-push-pre-receive-hook-declined) stackoverflow de l’erreur.

Toute la marche à suivre se trouve sur le site de [Heroku](https://devcenter.heroku.com/articles/heroku-cli) et nous avons également regardé un tutoriel sur [Youtube](https://www.youtube.com/watch?v=KD9OaryS1Kw) pour essayer mais rien à faire. Aucune solution n’a été trouvée.

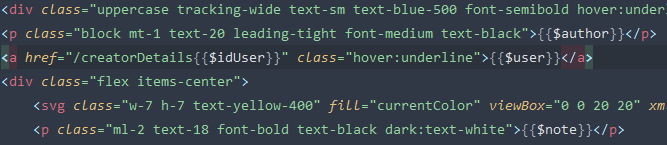
Nous avons également regardé avec le groupe de Thomas et Damien pour savoir s’ils avaient des solutions, malheureusement ils n’en avaient pas.

### Notre utilisation de Laravel

Dans cette partie nous allons expliquer les concepts de laravel que nous avons utilisé pour le projet.

#### Vues

Pour les vues, nous avons décidé d’utiliser des partials et des layout. Les partials nous sont très utiles, c’est des petits blocs de code que l’on peut implémenter sur toutes les pages en envoyant des informations. Par exemple pour l’affichage de nos livres nous avons fait comme ceci :



Les champs avec les $ sont des paramètres que le partial reçoit lorsqu’il est implémenté sur une page, les « {{}} » représente un echo. Les informations sont envoyées comme ceci :



Nous faison un include du partial et grâce à une boucle forelse nous allons récupérer toutes les informations du livre et les envoyer au partial.

Le layout est un fichier qui est composé des éléments de base d’un site web, c’est-à-dire un header, un footer, les liens sur les fichiers css, etc. C’est une template qui est implémentée sur toutes les vues du site web pour que toutes les vues aient les mêmes paramètres. C’est le fichier qui structure les pages du site.

#### Controller

Nous avons 3 controllers dans le projet. Le controller pour les livres, celui pour la page d’accueil et celui pour l’utilisateur.

Le controller pour les livres regroupe toutes les fonctions qui concernent les livres, par exemple la fonction pour rechercher un livre grâce à son nom.



Le paramètre d’entrée est le nom du livre, ensuite la variable $books contiendra le contenu que le model « BookModel » lui retournera. Le contenu retourner sera le résultat de la requête, la requête cherche dans tous les livres quel livre contient le paramètre demandé.

#### Model

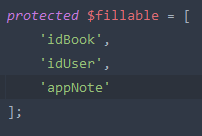
Nous avons un model par table dans la base de données. Donc un model pour les livres, les auteurs, etc.

Pour les model et tout ce qui concerne la base de données nous avons utilisé le mappeur relationnel d’objet Eloquent qui est inclus dans Laravel. Eloquent permet d’interagir plus facilement entre les models et les tables liées.

Dans chaque model il faut définir le nom de la table, la clé primaire et les champs de la table.



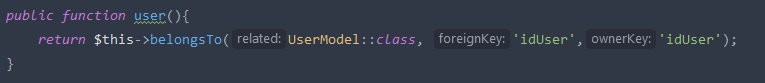
Dans ce cas précis, la table « t\_appreciate » est une table pivot, donc elle ne possède pas sa propre clé primaire. Elle sert à faire la liaison entre la table t\_user et t\_book. Cela veut dire qu’elle possède une clé primaire composée pour faire la liaison.



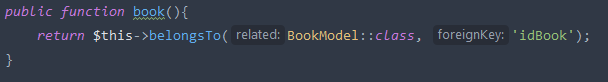
Le $fillable est le tableau qui regroupe les champs de la table.

**Nous avons dû spécifier tous les champs car nous avons utilisé les conventions de nommages de l’ETML et laravel ne les comprends pas car il utilise le « snake\_case ». Si nous avions utilisé les conventions de nommages de laravel nous n’aurions en aucun cas eu besoin de définir la clé primaire et les champs de la table.**

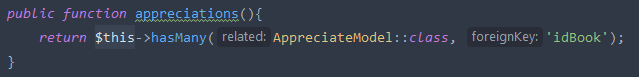
Il faut également savoir que chaque model possède ses propres fonctions, elles dépendent des liaisons entre les tables. Pour le model de la table t\_appreciate nous auront 2 fonctions dans le model.



La fonction user() permet de faire la relation entre la table t\_appreciate et la table t\_user. Le belonsTo veut dire qu’une appréciation appartient à un utilisateur unique.

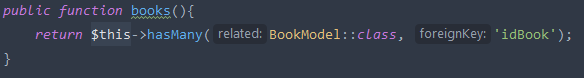


La fonction book() permet de faire la relation entre la table t\_appreciate et la table t\_book. Dans ce cas-là le belongsTo veut dire qu’une appréciation appartient à un livre spécifique. Par contre un livre peut posséder plusieurs appréciations, cette fonction se trouve dans le model de la table t\_book.



Le hasMany veut dire que le livre possède potentiellement plusieurs appréciations.

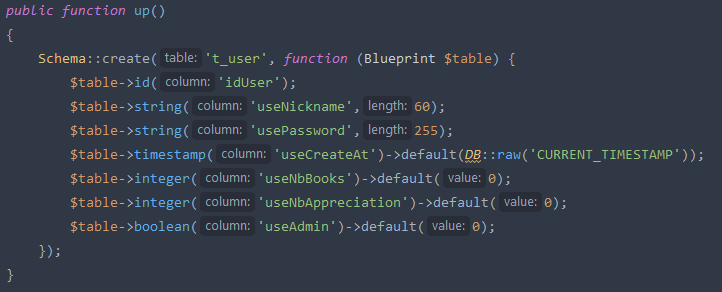
Dans un autre cas de figure nous avons utilisé le hasMany. Cette fonction se trouve dans le model de la table t\_author, donc cela représente les auteurs.



Le hasMany veut dire qu’un auteur possède potentiellement plusieurs livres.

#### Migrations

Les migrations nous ont servi pour créer la base de données sans utiliser un script SQL.

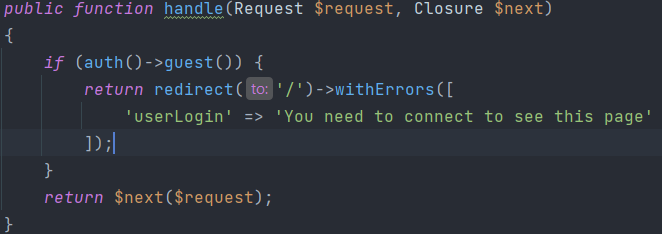


La migration pour la création de la table t\_user, tous les types sont spécifiés ainsi que les valeurs par défaut.

#### Middleware

Un middleware a été créée pour sécuriser les routes du site web auxquelles seulement les utilisateurs connectés peuvent accéder.

S’il y a un utilisateur connecté, le middleware renvoie la page recherchée ; sinon, il le renvoie vers la page d’accueil et affiche le formulaire de connexion.

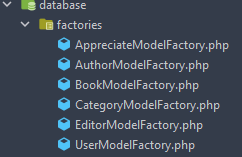
****

#### Factories / Seeds

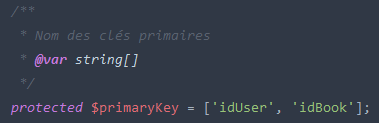
Des factories ont été créées pour insérer des données aléatoires dans la base de données afin de la tester.

On utilise également le DatabaseSeeder pour insérer un tas de données dans chaque table à chaque fois qu’on fait une migration.

Php artisan migrate:refresh --seed

****

Pour que la factory « AppreciateModelFactory » fonctionne il faut mettre en commentaire la définition des clés primaires dans le model de la table t\_appreciate. Etant donné que la clé primaire est une clé composée, c’est-à-dire qu’elle comporte 2 clés primaires, la factory génère des erreurs avec liaisons des clés primaires / étrangères.



C’est la seule solution que nous avons trouvée pour que la factory fonctionne avec cette table. En faisant de cette manière les clés étrangères entre les table n’étaient pas présente donc les appréciations n’étaient pas liée à un utilisateur ni à un livre.

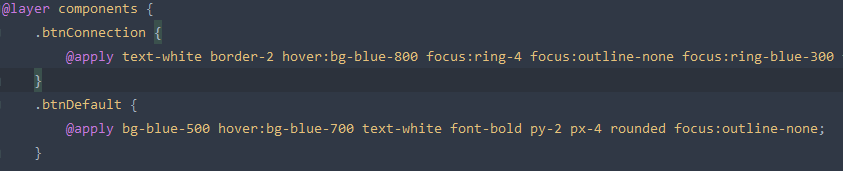
Pour toutes les autres tables les données ont été générées sans problèmes.

### Notre utilisation de Tailwind

Nous avons utilisé tailwind pour le css. Tailwind a été installé sur un PC pour pouvoir créer des composants. Le CDN a également été utilisé car nous n’avons pas eu le temps pour créer tous les composants.

#### Composants

Les composants servent à définir le style pour un élément précis, par exemple pour un bouton, nous définissions un style comme ça lors de la création d’un bouton le style est déjà en place.



Le « .btnConnection » est le style pour les boutons qui concerne la connexion. Pour les inclure dans le php il faut faire cela.



La class du bouton doit avoir le nom du composant.

#### CDN

Le CDN c’est l’utilisation de tailwind avec un lien web, sans le cdn le style de notre site ne fonctionnerais pas sauf pour les éléments qui possède le style d’un composant.



Pour utiliser le CDN il faut utiliser la balise script dans le layout du projet, étant donné que le layout est inclus sur toutes les pages le CDN peut être utilisé sur toutes les pages.

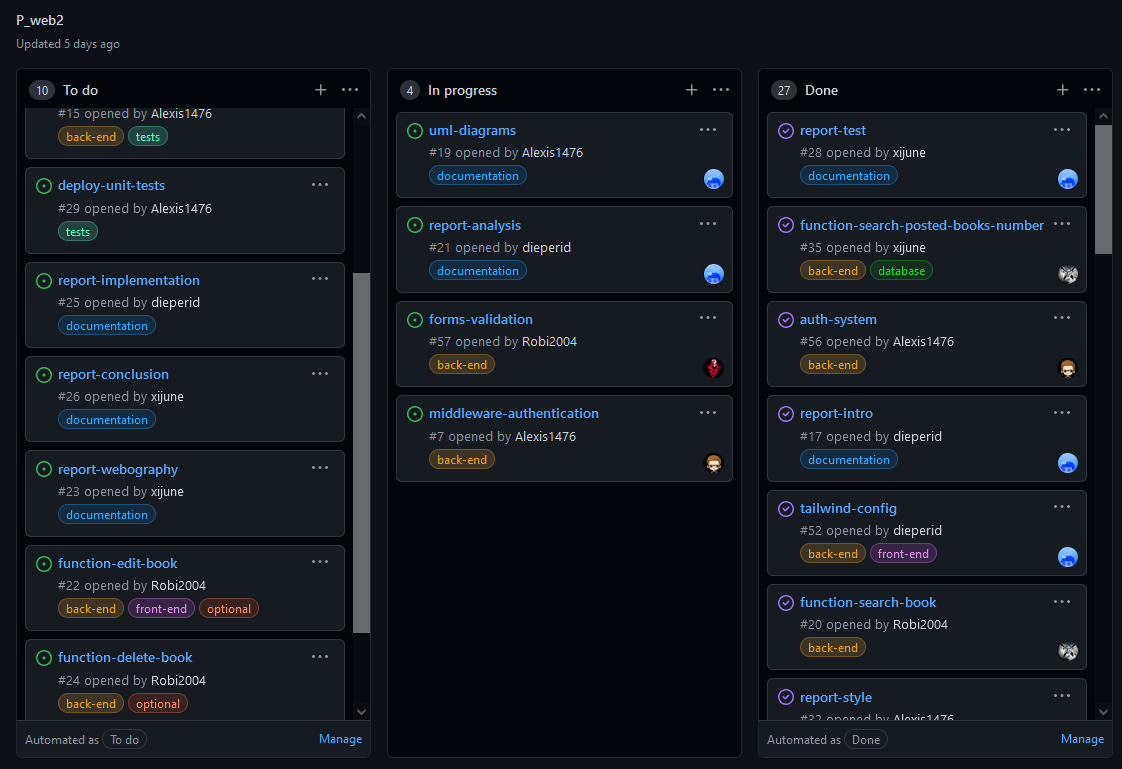
### Organisation – Répartition des tâches

En terme d’organisation nous avons réalisés un document en markdown qui sert de conventions de nommages pour le projet. Ce document a été réalisé au départ du projet pour que toute l’équipe utilise les mêmes nommages. Le document se trouve sur [GitHub](https://github.com/Alexis1476/etml-p040-web2/discussions/36).

Pour la répartition des tâches nous avons utilisé un trello sur GitHub. Les issues ont été créé au départ du projet.

Dès le début nous nous sommes mis à 4 pour créer un mind map qui nous a permis de nous représenter toutes les choses à faire sur le projet. Suite à cela nous avons créé à 4 le mcd et le mld pour la base de données.

Après tout cela, chaque membre du groupe a pris une tâche sur le trello qui l’intéressait et c’était comme ça jusqu’à la fin du projet.

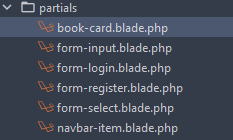


## Modifications

Les modifications sont représentées par les commit sur le dépôt GitHub. Chaque personne a effectué des commit pour pousser ses modifications sur le dépôt distant.

# Ecoconception

## Réduction de la taille du DOM

Nous avons réduit la taille du DOM (Data Object Model). Cela réduit le temps de chargement et le temps de la latence entre les pages. Elles possèdent moins de contenu donc elles sont plus rapides à charger. Pour réduire cela nous avons créée des vues partielles, cela nous permet de réduire la répétition d’un élément dans un page de code ainsi que la taille du fichier.

## Le Scroll

Nous avons également fait en sorte de réduire le nombre de pages ou l’utilisateur peut scroller. Les pages sont donc plus petites et moins conséquente à l’affichage. Cela aide à la réduction du DOM.

## Animation

Les animations (gifs, carrousels, chatbots, etc.) inutiles ont été retirée. De plus nous avons limité les animations pour permettre une meilleure expérience à l’utilisateur et c’est bénéfique à l’écologie.

# Tests

## Dossier des tests

### Tests d’intégrations

|  |  |  |  |
| --- | --- | --- | --- |
| Titre | Résultat attendu | Résultat obtenu | Date |
| Recherche livre réussie | Le livre s’affiche lors de la recherche | Le livre s’affiche lors de la recherche | 18.05.22 |
| Recherche livre échouée | Aucun livre s’affiche car le nom est faux | Aucun livre s’affiche car le nom est faux | 18.05.22 |
| Filtrage livre par catégorie réussi | Les livres de la catégories s’affichent | **PAS IMPLEMENTE**  **MANQUE DE TEMPS** | 18.05.22 |
| Filtrage livre par catégorie échoué | Les livres de la catégories ne s’affichent pas car aucun livre n’est dans la catégorie. | **PAS IMPLEMENTE**  **MANQUE DE TEMPS** | 18.05.22 |
| Redirection sur la page de détail créateur | La page de détail du créateur s’affiche | La redirection fonctionne, les détails du créateur s’affichent. | 11.05.22 |
| Redirection sur la page d’ajout d’un livre | La page pour l’ajout d’un livre s’affiche | La redirection fonctionne, la page d’ajout de livre s’affiche. | 11.05.22 |
| Redirection sur la page détail du livre | La page de détail du livre s’affiche | La redirection fonctionne, les détails du livre s’affiche. | 11.05.22 |

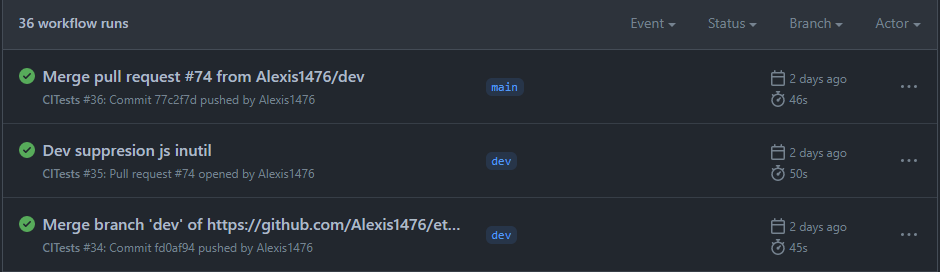
## Tests CI

Pour l’implémentation des tests fonctionneles CI, on a utilisé les **Actions Github.**

Les [tests fonctionnels](#_Tests_fonctionnels) sont testés à chaque fois qu’on fait un push sur la branche *main* ou *dev.*

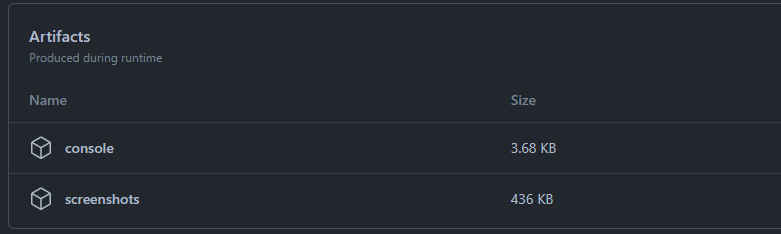
Un fichier **yml** est créé dans le répertoire *github* à la racine du projet **Github** où se trouve la démarche pour lancer les tests fonctionnels :

1. Création du fichier « **env** »
2. Création de la base de données
3. Installation des dépendances composer
4. Génération d’une clé pour l’application
5. Mis à jour du driver **Chrome** + installation **Dusk**
6. Initialisation Chrome Driver
7. Démarrage Laravel Server + Définition variables d’environnement
8. Exécution des migrations
9. Exécution des tests **Dusk**
10. Captures d’écran en cas d’erreur
11. Log en cas d’erreur

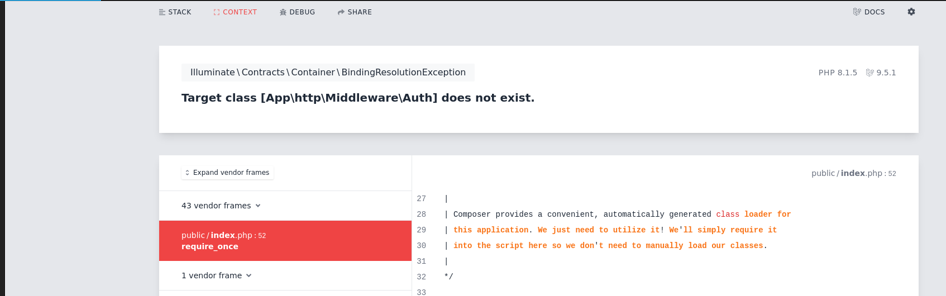


En cas d’erreur, un screenshot de l’erreur en question peut être téléchargé. Exemple :

3 Github > Actions > Runs



4 Exemple de screenshot



# Conclusion

## Bilan des fonctionnalités demandées

Les fonctionnalités qui possèdent un « OK » en vert sont les fonctionnalités qui ont été atteintes lors du projet.

### Fonctionnalités requises

Le site web / application aura les pages suivantes :

* Une page d’accueil comprenant une explication de l’utilité du site ainsi que les cinq derniers ouvrages ajoutés (accès tout public). **OK**
* Une page comprenant la liste des ouvrages par catégorie (accès tout public avec restrictions sur les liens). **OK**
* Une page d’ajout d’un ouvrage (accès utilisateur). **OK**
* Une page permettant d’ajouter une appréciation à un ouvrage (accès utilisateur). **OK**

PS : le pied-de page du site doit faire mention de la personne qui a créé l’application ainsi que le moyen de la contacter. **OK**

Les fonctionnalités sont atteintes car, tous les éléments demandés figurent sur le site web.

## Bilan de la planification

Le projet a pu être fini dans les temps. Nous n’avons pas réellement effectué de planification. Nous avons réalisé des user story qui concernait les éléments à réaliser dans le projet. Elles ont toutes été terminées sauf les optionnelles.

## Bilan personnel

### Avis David

Personnellement j’ai beaucoup aimé faire ce projet. Il m’a permis de voir deux framework web (laravel et tailwind). Sachant que je souhaite devenir développeur web fullstack cela m’a grandement aidé dans mon choix d’avenir et m’a fait prendre énormément de connaissances.

### Avis Alexis

J’ai eu beaucoup de plaisir à faire ce projet vu qu’on a utilisé deux frameworks différents (Laravel et Tailwind), ce qui m’a apporté beaucoup de connaissances et m’a permis de mettre en pratique plusieurs concepts et outils de Laravel que j’avais étudié au préalable.

Si j’avais à refaire ce projet je le ferais en respectant les conventions de nommage Laravel et en utilisant les composants blade.

### Avis Stefan

Je trouve ce projet très utile pour la suite de ma formation, j’ai énormément appris durant ce projet. De plus, j’ai beaucoup apprécié le travail en équipe qui m’a permis de mieux comprendre le fonctionnement de GitHub en groupe.

### Avis Robustiano

J’ai trouvé ce projet très intéressant il nous en a plus appris sur le php et surtout nous a permis d’apprendre quelques bases sur Laravel. J’ai également appris Tailwind mais aussi les tests unitaires en php. En bref, un projet très intéressant pour travailler en groupe et qui nous pousse à sortir de notre zone de confort.

# Divers

## Journal de travail

Le journal de travail se trouve sur [GitHub](https://github.com/Alexis1476/etml-p040-web2/projects/1), c’est un trello de plus il y’a les commit log.

## Webographie

PhpStorm : <https://www.jetbrains.com/phpstorm/>

Laravel : <https://laravel.com/>

Tailwind : <https://tailwindcss.com/>

Flowbite : <https://flowbite.com/>

# Annexes

Laraguide : <https://www.youtube.com/c/ThibaudDauce/videos>