Site web - Livres

(Une image originale représentant le projet)

Julien Leresche et Simon Guggisberg

ETML

Table des matières

[1 Introduction 3](#_Toc58566886)

[2 Analyse 4](#_Toc58566887)

[2.1 HTML 4](#_Toc58566888)

[2.2 MCD/MLD/MPD 5](#_Toc58566889)

[2.3 UML 6](#_Toc58566890)

[3 Réalisation 7](#_Toc58566891)

[3.1 Algorithme login 7](#_Toc58566892)

[3.2 Appréciation moyenne 8](#_Toc58566893)

[3.3 Explication écoconception 9](#_Toc58566894)

[4 Tests 10](#_Toc58566895)

[5 Conclusion 11](#_Toc58566896)

[5.1 Générale 11](#_Toc58566897)

[5.2 Personnelle 12](#_Toc58566898)

[5.3 Critique méthode projet 12](#_Toc58566899)

[6 Divers 13](#_Toc58566900)

[6.1 Bibliographie 13](#_Toc58566901)

[6.2 Webographie 13](#_Toc58566902)

[6.3 Glossaire 13](#_Toc58566903)

# Introduction

Le but de ce projet étant de réaliser un site web dynamique, permettant de se login, d’ajouter des éléments à celui-ci et de voir ces éléments se retrouver sur le site.

Nous avons choisi comme thème celui des livres, où chaque utilisateur peut ajouter ses propres livres à la base de donnée, où chaque utilisateur peut noter, ou apprécier, un livre, en raison de notre grande affection pour les mots, et jeux autour de ceux-ci.

# Analyse

## HTML

Après lecture du cahier des charges, nous avons décidé qu’une page principale, celle d’accueil, devrait être celle n’affichant que 5 premiers livres, une mise en bouche, si vous préférez.

La seconde page serait celle contenant tous les livres, mais plutôt similaire à celle d’accueil sinon.

En cliquant sur un livre nous aboutirions alors à la page de détails dudit livre, mais, en découvrant le framework Bootstrap, nous avons découvert un outil, le modal, permettant d’afficher ces détails de manière bien plus élégante qu’une page à part, cela permettrait de rester sur la page principale/liste de livres et de consulter les livres, avec, certes, un coût de calcul pour le serveur plus important, mais cela ne consultant que du texte essentiellement, cela ne nous a pas posé problème.

Reste encore la page d’ajout de livre, accessible uniquement une fois connecté, via le header de la page principale.

La page permettant d’afficher la liste de livres ajoutés par chaque utilisateur, respectivement. Ainsi, l’utilisateur « Paul » aura sa propre page, dédiée, permettant de visionner les livres qu’il a ajouté à la base de données.

De la même manière, en inspectant les détails d’un livre, il est possible de voir les évaluations dudit livre, sur une page dédiée, encore une fois. Là, il est possible de voir les évaluations individuelles ajoutées par les utilisateurs, et d’en écrire une nouvelle, en indiquant le score que l’on souhaite attribuer.

## MCD/MLD/MPD

En suivant le cahier des charges, plusieurs points ont rapidement été relevés comme importants pour la base de données, certains du point de vue livres, d’autres du point de vue utilisateur.

Nous avons décidé de créer une table pour les catégories et les éditeurs, puisqu’ils représentent des valeurs pouvant revenir de livre en livre. Et pouvant être utile, dans un futur, de trier les livres, ou de les grouper en fonction de ces arguments.

La table t\_author contient le nom et le prénom de l’auteur, ainsi que son id.

La table t\_category contient le nom de la catégorie et l’id.

La table t\_editor contient le nom de l’éditeur et l’id.

La table t\_user contient le pseudo utilisé par l’utilisateur, son mot de passe, haché afin de le rendre illisible, et son id.

La table t\_book, elle, contient le titre, le nombre de pages, le lien vers le pdf d’un extrait du livre, un court résumé, l’année d’édition, un lien vers l’image de couverture, son id et des clés étrangères appartenant à t\_author, t\_category, t\_editor et t\_user.

À relever le cas des notes des utilisateurs, qui sont un cas à part. Nous avons créé une table à part pour les évaluations, possédant deux clés étrangères, vers t\_book et t\_user.

Au final, nous avons créé une classe PHP database qui communique avec la database avec quelques méthodes génériques, où les arguments passés, tels que le nom de la table, le nom des colonnes, etc, changent le comportement du tout au tout de la fonction.

## UML

# Réalisation

## Algorithme login

## Appréciation moyenne

## Explication écoconception

Pour l’écoconception, nous avons décidé de lister les différentes choses que l’on pouvait faire dans ce sens au début du projet, puis les implémenter au fur et à mesure, ou orienter le projet dans ce sens si cela relève une plus grande organisation.

En effet, les points varient, allant de « Minification des fichiers » à « utilisation d’un framework ». Autant le premier est trivial, et ne demande qu’un peu de temps alloué sur la fin du projet, autant le dernier nécessite du travail en amont et une vision d’ensemble.

Je vais donc lister les différents points que nous avons relevé et ce qui a été fait dans ce sens pour chacun de ceux-ci :

* Minification des fichiers :
* Compression des médias utilisés :
* Images sprite :
* Utilisation d’un framework :
* Limitation du nombre de requêtes http :
* Stockage en local des données statiques :
* Design simple et épuré : Puisque nous utilisons Bootstrap comme base, cela nous donne une bonne ligne directrice en terme de design limpide et simple.
* Élimination des fonctionnalités inutiles :
* Réduction du nombre de vidéos et des animations : Puisqu’aucune vidéo ou animation n’est présente sur le site, à l’exception de quelques effets en Javascript via Bootstrap, Nous considérons ce point comme aisément acquis.
* Limiter le recours aux plugins :

# Tests

# Conclusion

## Générale

Ce projet nous a laissé une grande liberté, mais, en raison des critères comme la nécessité de le faire en MVC, ou d’inclure une connexion à une base de données, tous deux vu plus tard lors de la théorie, des mauvaises décisions d’organisation ont été prises. Nous avons, en effet, décidé d’implémenter une connexion à la base de données, puis le reste des fonctionnalités, avec la conversion vers MVC vers le milieu de projet, et enfin la finition des pages en Bootstrap vers la fin du projet.

L’ordre idéal aurait été, MVC puis Bootstrap, puis PDO, mais, la faute à un manque d’organisation et de connaissances, cela ne s’est bien évidemment pas réalisé ainsi.

Un autre point à relever serait la complexité de travailler à plusieurs sur un même projet. Cela concerne autant la division du travail de manière équitable et intelligente, éviter que deux personnes bossent sur la même partie de manière simultanée, que la gestion des conflits entre le travail effectué, notamment via un logiciel de gestionnaire de versions, ou encore la mise en commun du travail, une fois celui-ci effectué.

Autant de points que Git, et son interface visuelle Gitkraken, ont su grandement aider, à différents niveaux. Réussite aussi due à la bonne entente régnant entre les membres de l’équipe du projet.

## Personnelle

Julien :

Simon : En raison de la grande liberté laissée lors de ce projet, ce fut un plaisir de travailler sur ce projet. Je déplore néanmoins l’implémentation tardive au sein du projet, et une organisation globale qui laissait à désirer, car aucun de nous deux n’étaient familiers avec le développement web dynamique.

Mais, au final, j’ai pu apprendre de nouvelles choses sur la programmation web, et ai pu translater ces connaissances dans d’autres projets ou modules, tel que celui fil rouge ou le P\_PROD, très proches de celui-ci dans l’idée.

Ce fut également un plaisir de ne pas avoir travaillé avec le classique journal de travail, je considère que la méthode SCRUM est plus adaptée pour le développement, nécessitant des feedbacks réguliers afin de corriger erreurs et modifier le tir.

## Critique méthode de gestion de projet

Nous avons décidé d’utiliser une méthode inspirée de Kanban pour suivre la quantité de travails à effectuer, la répartition de celui-ci, et enfin l’évolution au cours du temps du projet.

Comme nous utilisions Gitkraken pour tout ce qui est gestionnaire de versions, nous avons également décidé d’utiliser l’outil directement inclus dans Gitkraken, à savoir, Boards. Sorte de Trello, avec des features en moins, d’autres en plus pensées spécialement pour tout le développement Git.

Nous avons décidé d’un code couleur pour les cartes : rouge, ce qui est nécessaire, orange, ce qui est un bug, bleu, ce qui relève de la documentation, et vert, ce qui est facultatif/serait bien à avoir.

En ayant 4 colonnes, To Do, In Progress, Testing, Done, Nous avons largement su gérer les objectifs par priorité. Mais observer l’évolution du projet sur le temps est plus ardu, certaines fonctionnalités comme des Story Points ou une Burndown Chart manquant à l’outil Boards.

Reste que cette méthode de gestion de projet s’est révélée assez dynamique et adaptée pour un projet de cette ampleur, constitué uniquement de deux personnes et sur une période de moins de six mois.

# Divers

## Bibliographie

## Webographie

## Glossaire