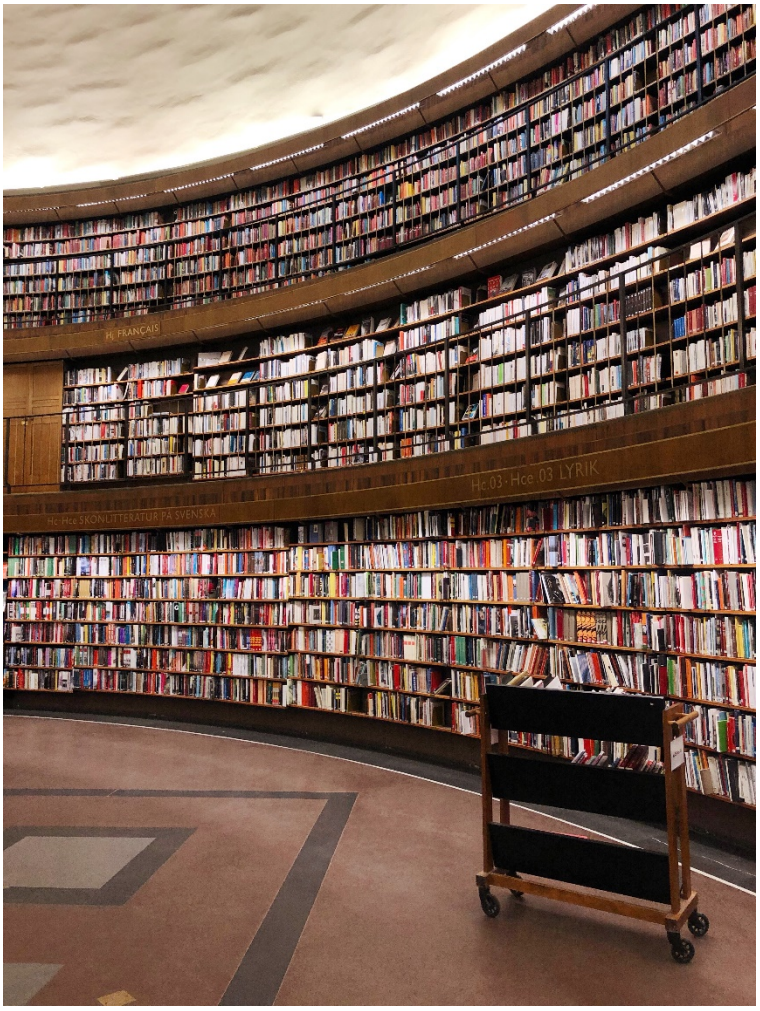
Site web - Livres



Julien Leresche et Simon Guggisberg
ETML

Table des matières

1 INTRODUCTION

Le but de ce projet était 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.

Nous disposions d'un cahier des charges, que nous analysons dans les premières pages de ce rapport, nous avons, par la suite, découpé les taches en plusieurs catégories, celles vitales et celles non vitales, mais intéressantes à avoir.

2 ANALYSE

2.1 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.

2.2 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.

Enfin reste la table t_evaluate qui garde en mémoire les évaluations données par qui et pour quel livre, avec des champs supplémentaires tel qu'un commentaire.

Au final, nous avons créé une classe PHP database qui communique avec la base de données 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.
On distingue au sein de celle-ci plusieurs types de fonctions :

- Les fonctions DataExistsAt : retournent -1 si la valeur n'existe pas dans la database.
- Les fonctions AddData, qui ajoutent une valeur à la database et retournent l'index de cette nouvelle valeur.
- Les fonctions Get qui retournent une valeur du tableau, si ce n'est tout le tableau entier, selon certains critères, défini en fonction.

2.3 UML

Conformément au cahier des charges, nous avons adopté une hiérarchie de fichiers et logique de code MVC. C'est-à-dire que toute la logique du code est exécutée dans les fichiers php dans le dossier controller, et que toute la partie visuelle réside dans les fichiers html du dossier view. Enfin, la partie d'interaction avec les données se fait via le dossier model, et plus spécifiquement la classe Database, dont sa structure a déjà été évoquée dans le point précédent, concernant le MCD/MLD/MPD.

Nous avons également créé un fichier php contenant plusieurs méthodes dont nous nous sommes rendus comptes que leur utilisation revenait à travers les pages du site internet.

3 RÉALISATION

3.1 Algorithme login

Une fois les informations du login entrées par l'utilisateur, l'on vérifie si l'utilisateur existe bien dans la database et, le cas échéant, si le mot de passe entré correspond à celui sauvegardé, sous forme de hash, via la méthode php password_verify.

On sauvegarde à ce moment dans les variables de sessions le nom d'utilisateur ainsi qu'une variable attestant du succès de la connexion.

On vérifiera alors dans chacune des pages nécessitant de s'authentifier la valeur de cette seconde variable.

3.2 Appréciation moyenne

Afin de procéder à ces opérations nous avons dû créer une table concernant les évaluations, les notes, dans la base de données, voir point plus haut concernant la conception du MCD/MLD.

Pour calculer une moyenne, il convient d'abord de recueillir l'entièreté des notes pour un ouvrage, via une requête SQL, puis de parcourir le tableau ainsi reçu. Une variable « total des notes » est initiée à zéro puis est incrémentée de chaque note du tableau susmentionné, jusqu'à la fin.

Enfin, il ne reste plus qu'à diviser le total des notes par la quantité de notes, donnant ainsi une moyenne.

3.3 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 relevait d'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 : notre site ne disposant pas de CSS, nous avons pu faire cela pour les pages HTML uniquement, créant des conflits avec ceux PHP.
- Compression des médias utilisés : Nous n'utilisons pas d'images ou de fichiers sur notre site, excepté ceux envoyés par les utilisateurs concernant les livres. Mais, en raison du temps, nous n'avons pas pu trouver une solution pour compresser ces fichiers, cela reste un point améliorable pour le futur néanmoins
- Utilisation d'un framework : Nous utilisons Bootstrap. Nous aurions aussi pu utiliser Laravel. Mais cela aurait requis une prise de décision dès le début, ce qui n'a pas été fait.
- Limitation du nombre de requêtes http : Le nombre de requêtes http est assez limité, nous utilisons même une seule page pour afficher plusieurs types d'informations, via l'utilisation des modals, afin d'éviter trop de rechargement intempestif de pages.
- Limitation des requêtes SQL : lorsque des requêtes SQL doivent être traitées par la base de données, celles-ci ont été conçues dans le but d'optimiser en tête, exemple, pour la page d'accueil, nous n'affichons que les 5 derniers livres ajoutés, une limite est donc appliquée via la requête SQL, pas via le code PHP.
- Stockage en local des données statiques : Ce point n'est actuellement pas complètement rempli, la faute au CDN de Bootstrap, que nous avons préféré utiliser, pour une raison de légèreté des fichiers de travail.
- 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 : Notre site ne contient que des fonctionnalités utiles et mentionnées sur le cahier des charges.
- 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 : Nous n'utilisons pas de plugins

Au final nous avons tenté de tendre vers l'écoconception autant que faire se peut mais des obstacles techniques se sont dressés sur notre chemin. Pour un prochain projet nous serons au fait de ceux-ci et saurons les éviter.

4 TESTS

Accès et GET :

Afin de vérifier que les accès sont correctement restreints, nous avons tenté de consulter les différentes pages sans être connecté, en donnant des arguments dans l'HTML en GET.

Nous avons ensuite tenté de se diriger sur ces mêmes pages, mais avec un compte connecté, afin de s'assurer que tout fonctionne comme prévu.

Injections SQL, Caractères spéciaux :

Finalement, nous avons testé différents caractères connus pour créer des problèmes dans les champs de formulaire, tels que le simple guillement, le double, le crochet plus petit que/plus grand que, le signe égal, et, point-virgule, etc

Bien entendu, nous avons testé les caractères exclus des expressions régulières, avant de les en exclure pour un second test

Base de données :

Au fur et à mesure de l'avancement du projet, nous avions déjà la preuve que la base de données fonctionnait bien avant des index manquants, mais quid d'une base de données vide ? ou d'une neuve ? Ce sont d'autres tests que nous avons faits, tout comme supprimer certaines valeurs en cours de route, et vérifier que la suppression en cascade se déroule correctement, si celle-ci est bien mise en place.

5 CONCLUSION

5.1 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.

Au final, nous avons réalisé les objectifs nécessaires du cahier des charges, mais ceux considérés comme optionnels n'ont pas pu être atteints dans les temps.

5.2 Personnelle

Julien : A mes yeux, ce projet représentait une bonne synthèse de toute la matière vue lors des deux modules. Étant plutôt adepte du développement en général, j'ai eu donc beaucoup de plaisir à travailler sur ce projet, d'autant plus que le sujet de la bibliothèque de livres me parle tout particulièrement. Il me tarde désormais d'acheter un nom de domaine et de développer mon propre site web personnel… J'ai trouvé que le temps en fin de projet était un peu cours puisque les dernières notions théoriques ont été vues assez tardivement dans les modules, mais je me rend toutefois bien compte qu'il est difficile de voir toute la matière des modules suffisamment rapidement pour bien laisser le temps à l'implémentation dans le projet. Globalement, ce projet m'a amené beaucoup de connaissances pratiques, théoriques et de plaisirs.

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.

5.3 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.

6 ANNEXES

6.1 Mode d'emploi

Le fichier SQL permettant de créer la base de donnée vide crée également automatiquement 4 catégories de base, Bande dessinée, Comic, Manga et Livre, ainsi qu'un utilisateur standard : « Paul » avec comme mot de passe « toto!1234 ».
Un second compte a été créé, afin de tester plusieurs évaluations dans la base de données, le nom de compte et le mot de passe sont respectivement : « root » et « root »

Un second fichier SQL contient une base de données d'une dizaine de livres ainsi que quelques évaluations, FullLibrary.sql

Les fichiers HTML ayant été minifiés, les originaux se trouvent dans src.
Dans le dossier src se trouve également un wireframe, ou une maquette pour les francophones, qui permet ainsi de voir l'évolution du projet dans le temps.

6.2 Webographie

- https://commons.wikimedia.org/wiki/File:Stockholm_Public_Library_stacks.jpg
  Bibliothèque publique de Stockholm – CC0
- https://getbootstrap.com/docs/5.0/getting-started/introduction/
- https://www.gitkraken.com/
- https://github.com/GuggisbergSimon/40-Web-Book