

Damien Jakob

Diogo Vieira Ferreira

**Exercise  
Looper**

2019

# Introduction

Notre application sert à gérer des exercices. Elle permet premièrement de créer de nouveaux exercices. Ensuite, les utilisateurs peuvent répondre aux exercices créés. De plus, ils peuvent revenir modifier les réponses ultérieurement. Enfin, il est possible de consulter les toutes les réponses à un exercice.

# Résultat

L’application web est partagée en trois catégories bien distinctes :

* Take an exercise permet de choisir un exercice ayant le statut Answering, afin d’y répondre.

Sauvegarder l’url[[1]](#footnote-1) permet de modifier nos réponses.

* Create an exercise permet de créer un exercice avec plusieurs questions, puis de le valider afin de pouvoir y répondre.  
  Nous avons aussi la possibilité de quitter la création d’exercice sans valider celui-ci afin de le reprendre plus tard.
* Manage exercises quant à lui permet de reprendre ou supprimer un exercice qui n’est pas encore validé. Leur statut est *Building.* Il est aussi possible de les valider, leur statut passe alors à *Answering*.

Quand un exercice avec le statut *Answering* ne doit plus accepter de réponses, nous pouvons le clore, son statut devient *Closed*.

Il est ensuite possible de supprimer un exercice clos.

Nous avons aussi la possibilité de voir toutes les réponses d’un exercice. Sur la page des résultats, on peut apercevoir :

* En haut chaque question de l’exercice.
* A gauche l’heure à laquelle un des exercices a été rempli, dont nous l’appellerons utilisateur.

Si l’utilisateur a bien respecté les critères nous verrons des vus, s’il a juste rempli le champ un simple vu, si le champ est vide une croix est visible.

En cliquant une question, nous pouvons voir les réponses de tous les utilisateurs pour celle-ci.

Si nous cliquons sur le nom d’utilisateur, nous verrons toutes ses réponses.

# Prérequis

Pour la réalisation de ce projet, nous avons utilisé :

* Serveur Apache[[2]](#footnote-2) 2.4.41
* PHP[[3]](#footnote-3) 7.2.23
* Base de données[[4]](#footnote-4) : MySQL 8.0.17

# Installation

1. Créer la base de données via DB\_script.sql dans Triplice/BD/.
2. Mettre sur votre serveur Apache (à la racine) le contenu du dossier Triplice/Looper/.
3. Dupliquer le fichier configExample.php se situant dans Triplice/Looper/library et le renommer en config.php.
4. Modifier les données de celui-ci avec les informations de votre base de données.

# Contraintes techniques

Pour la réalisation de ce projet, nous avons eu comme restriction de devoir tout coder de A à Z nous-mêmes, sans utilisation de librairies externes.

De plus, nous étions obligés d’utiliser PHP comme langage de programmation, en plus des langages de bases tels que HTML[[5]](#footnote-5) et CSS[[6]](#footnote-6).

# Gestion du projet

Pour la gestion du projet, comme nous n’étions que deux, nous avons opté pour une méthodologie agile. C’est-à-dire qu’au début de chaque semaine, nous faisions une petite réunion permettant de voir les quelques problèmes en cours afin de leur trouver une solution lorsque nous remarquions que ceux-ci prenaient beaucoup de temps, de planifier de nouvelles tâches et de les répartir de manière équitable. A la fin de chaque semaine, toutes nos idées ou problèmes étaient rassemblés en nouvelles tâches pour la semaine suivante.

Au début de notre projet, nous avions commencé par définir notre logiciel de versionning[[7]](#footnote-7). Nous avons opté pour GIT, logiciel simple de compréhension, efficace et surtout utilisé par un grand nombre d’utilisateur dans le domaine du développement d’applications.

Plus tard, nous avons appris la méthode GIT FLOW, celle-ci consiste à séparer les « branches [[8]](#footnote-8)» entre état fonctionnel, nommé release, état en cours de développement, nommé develop et nos prochaines améliorations avec le terme feature. Cette façon de faire étant simple de compréhension est au début très compliquée à mettre en place, car on a tendance à mélanger les branches, chose que nous ne devrions normalement pas faire. Le seul moment où nous pouvons mélanger « merge » les branches, c’est quand nous avons terminé notre feature.

# Les difficultés

Les premières difficultés se sont senties lors de la création de la structure, car nous ne maitrisions pas le système MVC (Modèle-vue-contrôleur, c’est le fait de séparer les différentes interfaces afin d’avoir les données dans le modèle, tout ce qui touche à la présentation dans la vue et le contrôleur permet de lier les deux) où nous pouvons avoir un modèle par vue. Nous avons eu recours à l’expertise de Monsieur Hurni afin de mieux comprendre la structure MVC pour l’implémenter de manière correcte et efficace.

Nous avons eu plusieurs problèmes liés à la configuration de la base de données, plus précisément pour les identifiants. Comme nous travaillions avec les mêmes fichiers, nous n’avions pas pensé à séparer la connexion à la base de donnée afin d’éviter de devoir à chaque fois modifier le fichier principal. Il nous a donc fallu définir un fichier de configuration qui soit propre à chaque développeur.

Vers la fin, nous nous sommes rendu compte du grand nombre de failles disponible sur notre application. Les failles étant un danger pour toute application, car celles-ci permettent soient de modifier l’affichage de l’application de manière indésirable, soit d’accéder à des données qui ne sont normalement pas accessibles à l’utilisateur.

Les premières difficultés étaient de corriger l’insertion de caractères spéciaux qui pouvaient soit modifier l’affichage de manière indésirable, soit bloquer l’insertion des données dans notre base de données. Il a donc fallu traiter ces caractères spéciaux de la manière appropriée.

La seconde faille existante permettait à l’utilisateur d’effectuer des actions qu’il ne devrait pas être autorisé à faire, en entrant manuellement certaines url. Il a donc fallu ajouter une vérification avant chaque ordre donné par l’utilisateur afin de vérifier que l’action était autorisée.

L’apprentissage des REGEX[[9]](#footnote-9), afin de pouvoir analyser le lien afin d’appeler la page correspondante avec les paramètres voulus.

# Acquisition de connaissances

Nous avons profité de la facilité du projet pour créer notre propre Framework[[10]](#footnote-10) pour faciliter la transmission des informations sans que l’utilisateur aie une quelconque information de l’infrastructure, empêchant donc l’utilisateur d’avoir accès aux fichiers confidentiels, et permet une gestion des liens internet plus simple, qui permet d’empêcher l’utilisateur d’accéder à des pages inexistantes.

Nous avons aussi utilisé la technologie SASS, langage de développement pour simplifier l’écriture du CSS.

La mise en place des REGEX a permis de simplifier notre Framework, car nous vérifions que nos liens correspondent à ceux fournis dans le fichier Routes.php.

Quand le lien possède le mot clé « id », nous attendons des chiffres à la place de ce mot et quand nous trouvons le mot clé « text », nous souhaitons récupérer une suite de caractères.

# Notre avis

Nous avons trouvé que ce projet, de prime abord relativement simple, était parfois devenu compliqué suite à certaines de nos lacunes. Nous y avons fait face en faisant beaucoup de recherches, et par conséquent acquis de nouvelles connaissances sur les différentes technologies et méthodes utilisées.

Si à l’avenir nous devions refaire un projet similaire, nous pourrions plus facilement le mettre en place et aurions directement une structure de projet plus adaptée, ce qui nous permettrait d’obtenir plus rapidement les résultats attendus.

Comparé à nos débuts, nous pensons que notre manière de gérer les projets à beaucoup évoluée. Au départ, nous prenions beaucoup de temps pour répartir le travail et décider des tâches à effectuer. Ensuite, nous avons commencé à créer les prochaines tâches en fonction des problèmes que nous avons rencontré avec la précédente, permettant donc une création de tâches beaucoup plus rapide et plus complète. Au final, nous regardions les tâches disponibles et nous prenions celles ayant la priorité la plus élevée, tout en privilégiant leur intérêt pédagogique.

1. Adresse d'un site ou d'une page internet (ex. www.google.ch) [↑](#footnote-ref-1)
2. Serveur permettant d’accéder à notre site web [↑](#footnote-ref-2)
3. Langage de programmation, spécialisé dans la gestion des sites internet, utilisé uniquement par le serveur (l’utilisateur ne reçoit que le résultat produit par PHP) [↑](#footnote-ref-3)
4. Nous pouvons imaginer une base de données comme un gros classeur électronique, Excel en plus avancé [↑](#footnote-ref-4)
5. Langage permettant de structurer le contenu d’une page internet. [↑](#footnote-ref-5)
6. Langage permettant de décrire l’affichage des différents éléments sur notre site. [↑](#footnote-ref-6)
7. Logiciel permettant de faire des sauvegardes de chaque étape importante de notre code, comme des versions de chaque étape fonctionnelle, et de consulter facilement les modifications effectuées entre deux versions. [↑](#footnote-ref-7)
8. Nous pouvons définir une branche comme un dérivé de l’état actuel de notre projet. Une fois celle-ci développée, on peut en reverser le contenu dans le projet original, procédé que l’on appelle « merge ». [↑](#footnote-ref-8)
9. Moyen de vérifier qu’une suite de caractères contienne bien les mots que nous souhaitons, par exemple récupère le nombre de « a » dans une phrase ou vérifier que celle-ci possède bien des « a » [↑](#footnote-ref-9)
10. Canevas nécessitant une certaine structure de l’application, servant à simplifier certaines tâches. [↑](#footnote-ref-10)