Projet de Programmation Web
Vie d'Enseirb

Responsable
Philippe Swartvagher
philippe.swartvagher@enseirb-matmeca.fr

Intervenants
Pierre-Marie Marcille
pierre-marie.marcille@u-bordeaux.fr
Amina Teboubli
amina.teboubli@bordeaux-inp.fr

2023 – 2024

Vous allez développer un clone du célèbre site Vie de Merde[1], que l'on va nommer Vie d'Enseirb. Une Vide de Merde est une courte histoire qui raconte une galère du quotidien ; dans le cas de Vie d'Enseirb, ce sera une galère à l'Enseirb[2].

1 Fonctionnalités principales

Le site Vie d'Enseirb sera composé des éléments suivants :

- Une page d'accueil qui affiche les VdE ;
- Une page qui permet d'ajouter une VdE ;
- Une page qui permet d'afficher les commentaires d'une VdE, cette page contiendra aussi un formulaire pour ajouter un commentaire.

2 Tâches à réaliser

Pour que le code soit correctement testé sur Thor, **il faut scrupuleusement respecter les quelques contraintes techniques indiquées dans la suite**.

Excepté pour la tâche 7, vous utiliserez uniquement les compétences acquises dans ce cours pour réaliser le projet : **pas de programmation orientée objet, pas de framework PHP, ...**

1. https://www.viedemerde.fr/
2. Exemples (d'un point de vue enseignant) : « *Je fais cours à l'ENSEIRB pendant les campagnes et mes étudiants ne me ramènent même pas à manger...* » ou encore « *Je propose des projets super cools à mes étudiants, et ils arrivent quand même en retard...* »

Tous les fichiers sources devront être placés à la racine de votre dépôt Git (*i.e.*, pas dans un dossier `src` ou autre).

Voici les différentes tâches à réaliser **dans l'ordre** pour le projet :

Tâche 1 : création de la structure de la base de données

Depuis phpMyAdmin, concevez la structure de la base de données qui permettra de stocker les données du site Internet.

Une fois les tables nécessaires créées, exportez **uniquement la structure** des tables dans un fichier **nommé `code.sql` que vous placerez à la racine de votre dépôt Git**.

Pour exporter : onglet *Exporter*, *Méthode d'exportation personnalisée*, décochez *Données* et bouton *Exporter* tout en bas.

Pourquoi ? Les tests sur Thor vont exécuter le code SQL contenu dans `code.sql` pour construire les tables utilisées par l'application.

Tâche 2 : paramètres de connexion SQL

Votre dépôt Git contient déjà un fichier `config.dist.php` qui montre à quoi doit ressembler le contenu d'un fichier `config.php`. Vous ne devez pas supprimer le fichier `config.dist.php`.

Votre code PHP doit inclure un fichier `config.php` et utiliser les informations du tableau `$config` pour se connecter à la base de données SQL.

Pourquoi ? Les tests sur Thor vont utiliser leur propre fichier `config.php`, avec les informations de connexion spécifiques à l'environnement de test.

Tâche 3 : page d'accueil

La page d'accueil est accessible à la page `index.php`. Elle contient les éléments suivants :

- un lien pour aller vers la page qui permet d'ajouter une vie d'Enseirb ;
- la liste des vies d'Enseirb (pseudo, date et contenu), les plus récentes en premier ;

Tâche 4 : ajout d'une VdE

Pour ajouter une VdE, on accède à la page `add_vde.php`. Cette page contient un formulaire pour saisir un pseudo et la VdE.

Lorsqu'elle est appelée en `POST`, cette page insère la VdE passée en paramètres (nommés `pseudo` et `content`) dans la base de données. En cas de succès, on est redirigé vers la page d'accueil, sinon la même page est affichée.

Tâche 5 : afficher une VdE

La page `show_vde.php` affiche la VdE avec l'ID passé dans le paramètre `GET` nommé `id`.

Si l'ID ne correspond à aucune VdE, un code d'erreur 404 est renvoyé.

Tâche 6 : commentaires

La page qui affiche une VdE affiche également les commentaires de la VdE, dans l'ordre chronologique.

La page contient également un formulaire pour ajouter un commentaire. Ce formulaire possède trois champs avec **les noms suivants : `vde_id`, `pseudo` et `comment`**. Le champ `vde_id` est de type `hidden` et contient l'ID de la VdE dans l'attribut `value`. **Ce formulaire envoie ses données en `POST` à la page `add_comment.php`.**

La page `add_comment.php` ajoute en base de données le commentaire reçu. En cas de succès, elle redirige vers la page de la VdE commentée.

La page d'accueil affiche maintenant aussi le nombre de commentaires (**sous la forme `x commentaires`**) de chaque VdE. Ce nombre est un lien qui permet d'aller vers la page qui affiche la VdE correspondante.

Tâche 7 : un peu de style

Utilisez le framework CSS Bootstrap (**et rien d'autre**) pour donner un peu de style à votre site.

Tâche 8 : retenir le pseudo

Utilisez les sessions pour retenir le pseudo saisi lors de l'envoi d'une VdE ou d'un commentaire, pour qu'il n'y ait pas besoin de saisir son pseudo à nouveau pour envoyer un autre commentaire ou une autre VdE.

Tâche 9 : publication d'un commentaire en AJAX

Utilisez JavaScript pour valider le formulaire de commentaire et ajouter le nouveau commentaire à la liste des commentaires sans avoir à recharger la page.

Tâche 10 : pagination

Ajoutez une pagination à la page d'accueil : seules `$config["paginate_by"]` VdE seront affichées sur la page et des liens permettant d'aller aux autres pages (**avec une adresse de la forme `index.php?page=x`**) seront présents.

3 Évaluation

Vous serez évalués sur les points suivants.

3.1 Rendu du code

- Quelles sont les tâches réalisées et fonctionnelles (il n'est pas forcément nécessaire de réaliser toutes les tâches : il vaut mieux bien faire 5 tâches que faire de travers 9 tâches)
- Code fonctionnel
- Code propre (style cohérent, commentaires, respect de la sémantique HTML, ...)
- Les tests sur Thor des tâches réalisées sont verts
- Messages de commits clairs et explicites

3.2 Schéma de la base de données

Un document concis (donc pas un rapport) qui contient :

- un schéma de la structure de votre base de données avec les relations entre les tables (clés primaires, clés étrangères)
- une justification de vos choix