INF5190 – Programmation web avancée
TP1 – Été 2021

Création d'un CMS simple

Vous devez concevoir un logiciel de type CMS (*Content Management System*), en version simplifiée. Un CMS permet de gérer le contenu d'un site web, c'est-à-dire d'y mettre des articles, de spécifier à quel moment ces articles seront publiés, bref de permettre à un non-programmeur de modifier le contenu de son site web.

~~*La base de données*~~

Un script de création de base de données pour SQLite vous sera fournit sur Moodle. La base de données n'a qu'une seule table et permet de contenir l'information sur un article. Voici les champs et leur utilité :

- titre : Une chaîne de caractères représentant le titre de l'article.
- identifiant : Une chaîne de caractères représentant l'identifiant de l'article. L'identifiant sera utilisé pour construire une URL unique pour chaque article. Uniquement des caractères permis dans une URL peuvent être utilisés.
- auteur : Une chaîne de caractères représentant le nom de l'auteur de l'article.
- date_publication : Une date correspondant à la date où l'article doit être publié. La date est un champ TEXT en format ISO8601 (voir la documentation de SQLite).
- paragraphe : Une chaîne de caractère représentant le paragraphe de l'article. Dans votre CMS simplifié, les articles ne peuvent avoir qu'un seul paragraphe chacun.

~~*Les routes*~~

Plusieurs routes doivent être supportées par votre application web. Voici le détail des exigences pour chaque route. Vous avez l'entière liberté de disposer les éléments demandés à votre guise. L'esthétique du site web est importante. À cet effet, vous êtes autorisés à utiliser des gabarits CSS provenant du web, dans la mesure où vous respectez leur licence d'utilisation.

~~GET /~~

Cette route correspond à la page d'accueil du site. Elle doit afficher les ~~5 dernières publications~~ en date du jour (vous ne devez pas afficher les publications avec une date de publication dans le futur). Pour chaque publication, vous devez afficher ~~toutes les données~~ que vous avez sur la publication.

La page d'accueil doit aussi contenir un champ texte représentant un ~~moteur de recherche~~. Le texte entré dans ce champ doit être recherché dans ~~tous les titres et paragraphes~~ connus du CMS. La liste des articles qui contiennent l'expression recherchée sera ~~retournée dans une page~~ (avec une nouvelle route, si vous le désirez). Pour chaque article dans le résultat de recherche, vous devez afficher le ~~titre de l'article et sa date de publication~~. Le ~~titre sera également un lien vers la page de l'article.~~

Utilisez l'opérateur ~~LIKE en SQL~~ pour faire la recherche.

~~GET /article/<identifiant>~~

Cette route correspond à la page d'un article en particulier. Vous devez récupérer les données de l'article identifié par l'URL et afficher les données de l'article. Une page 404 doit être retournée si l'identifiant n'existe pas.

GET /admin

Cette route correspond au point d'entrée pour un administrateur de contenu du site web.

La page sur cette route doit présenter la ~~liste de tous les articles connus du logiciel~~. Pour chaque article, on présente son ~~titre, la date de publication et un lien vers une page pour modifier l'article~~.

La page doit aussi contenir ~~un lien vers une page de création~~ d'un nouvel article.

La route à utiliser pour la page de modification d'un article est à votre discrétion. Lors de la modification d'un article, uniquement le titre et le paragraphe sont modifiables.

~~GET /admin-nouveau~~

Cette route permet d'afficher une page avec un formulaire pour ~~créer un nouvel article~~. Le formulaire doit contenir ~~tous les champs requis~~ par la base de données. La route à utiliser pour envoyer les données au serveur est à votre discrétion. Le serveur doit ~~valider les données~~ selon les exigences spécifiées dans la section sur la base de données. Également, tous les champs sont obligatoires. ~~En cas d'erreur de validation, la page du formulaire doit être affichée de nouveau avec les champs contenant déjà les valeurs soumises (même celles erronées) et les messages d'erreurs appropriés~~.

~~*Technologies*~~

Dans le front-end, vous **devez** utiliser les technologies suivantes :

- HTML 5
- CSS 3

Toujours dans le front-end, vous **pouvez** utiliser les technologies suivantes au besoin :

- Javascript
- ~~JQuery~~
- ~~Underscore.js~~
- Bootstrap

Dans le back-end, vous ne devez utiliser que les technologies disponibles dans l'environnement de développement founit.

Le code ~~Python doit respecter PEP8~~. Tous les fichiers sources doivent être encodés en UTF8.

Remise

Le travail doit être fait individuellement. Le répertoire de travail contenant les fichiers doit être archivé dans un fichier zip et nommé selon le code permanent de l'auteur. L'archive doit être remise par Moodle. Aucun retard ne sera accepté et une pénalité sera appliquée pour une archive non conforme sans le code permanent. Il est fortement recommandé de travailler sous gestion de sources avec git (dans un dépôt privé, bien sûr).

Pondération

Fonctionnalités : 55%
Respect des exigences et standard du web : 25%
Respect de PEP8 et qualité du code (taille des fonctions, qualité de la nomenclature, etc.) : 10%
Esthétique du site web : 10%