CHENE
Valentin

Atelier Pro 4 : Back-office

I. Introduction

A) Contexte :

Il faut créer un site avec un back office, listant dans un tableau des produits choisis, avec une fiche détaillée.

B) Langages utilisés :

- HTML/CSS
- PHP

C) Supports utilisés :

- Bootstrap 5
- SQLite

D) Sommaire :

Vendredi 10 Février 2023

II. Travaux Appliqués

A) Les Premiers Choix

Tout projet a un commencement, et il ne s'agit pas de s'empresser d'aller sur un éditeur de texte et de commencer à coder. Non, il faut premièrement trouver une thématique à ce sujet puisqu'il n'y a rien d'imposé. Étant un adepte des jeux vidéos, mais aussi du développement/modding de jeux, j'ai décidé d'orienter mon site sur les Moteurs de Jeux.
Mon Site Web porte donc sur un descriptif des nombreux Moteurs de Jeux utilisés par les développeurs plus ou moins expérimentés.

Ensuite, il a fallu que je décide de l'organisation du site, des pages, des couleurs, et aussi de l'organisation des fichiers.

Les pages et vérifications

Je suis parti du principe que : toutes pages formulaires feront une requête post vers un fichier de vérification avec une nomenclature précise "manager_page". J'ai donc mes fichiers pages en page.php et mes fichiers de vérification en manager_page.php.

Les classes

Ensuite, je mets mes classes dans le dossier model et elles font toutes parties de l'espace de nommage Model.

Les composants de pages

Comme toutes les pages ont certains points communs comme le footer ou la navbar, j'ai décidé de faire un fichier footer et navbar dans le un dossier component. S'il faut les modifier cela se fera uniquement là bas et non pas dans chaque page.

Stylisation

Pour la stylisation je me suis contenté de quelques lignes de css et de Bootstrap 5.
Je n'ai pas osé ajouter du Javascript que ce soit pour la partie technique que la partie esthétique puisque je trouve déjà mon code trop brouillon.

Les données

Dans ce projet, j'utilise SQLite3 pour stocker les données Utilisateurs et Produits.

A - Les Features

Maintenant, vient à réfléchir sur ce que sera capable de faire ce site.
Premièrement des points essentiels sont demandés :

- back office avec édition/suppression de produit
- listing des produits en accueil
- fiche détaillée du produit

Ce furent les premières choses à faire. Étant perfectionniste, je m'ajoute des tâches supplémentaires comme un backoffice de la gestion des utilisateurs et une page profile

permettant aux utilisateurs de modifier leur compte. Je prête aussi un œil sur l'optimisation, et me documente à ce sujet.

B - Les Caractéristiques

Il ne reste plus qu'à décider de la structure des données avant de commencer à coder.
Un utilisateur aura :

- un id UNIQUE
- un nom
- un email UNIQUE
- un mot de passe

Un produit aura :

- un id UNIQUE
- un nom UNIQUE
- une année
- des langages
- un éditeur
- des jeux
- un site web
- une histoire

Ça y est, on peut commencer à créer les fichiers!

B) Développement

Vient alors à créer les fichiers, il faut :

- model BDD
- model Session
- model produit
- model utilisateur (non obligatoire)
- page index
- page fiche produit
- page edition produit
- page liste utilisateur
- page edition utilisateur
- vérification action de produit
- vérification de login
- vérification d'édition utilisateur (non obligatoire)

A - Les Classes

J'ai commencé par faire mes classes, la classe Engine a été la première car je l'utiliserai dans la classe Engine par la suite. Je pars du principe que mon code tournera autour de mes classes, donc j'en ai besoin en premier pour pouvoir commencer convenablement.

J'ai fait ce choix car je me sens plus à l'aise et organisé à travers la POO. J'ai créé une "boîte à outils" DB qui me sert à interagir avec ma Base de Données avec une meilleure clarté.

B - Débuts de Front

Une fois ceci fait, je me suis empressé de monter mes pages, le front, mais pas en totalité. J'ai mis les éléments nécessaires pour interagir avec le back, car l'esthétique n'est pas ce qui me vient directement en tête puis je préfère faire la partie logique plutôt que visuelle.
Cela veut dire que j'ai ajouté tout ce qui est navigation, tableau, boutons, liens, formulaires etc.

C - Gestion login

Une fois que ça a été fait, je me suis occupé des pages "manager" et ai réfléchi à tous les éléments à vérifier.

La première à traiter fût celle des actions utilisateur, car il me fallait absolument pouvoir me connecter pour compléter les features nécessaires telles que le back-office. Pour le côté cyber, j'ai utilisé des requêtes préparées pour éviter des injections SQL.
Je précise que cette page manager a aussi été utilisée pour déconnecter et enregistrer un utilisateur. J'ai voulu centraliser un "thème" pour chaque manager, par exemple on verra que pour les actions sur un engine c'est géré dans un seul et même fichier.

Pour l'action d'enregistrement, je vérifie si l'adresse mail est déjà existante dans la base puis arrête le processus si c'est le cas. J'ai aussi mis l'email avec une contrainte UNIQUE en BDD pour éviter les duplicatas quoi qu'il arrive. J'ai fait ça de cette façon car je n'ai pas trouvé s'il existait des Callback dans ce cas-ci, qui renvoient une erreur ou des réponses après avoir essayé d'insérer une valeur.

Par la suite je me suis mis sur la connexion d'un utilisateur, pas tellement de difficulté, j'ai utilisé ma classe Session pour y stocker les informations de l'utilisateur, le mot de passe n'est pas compris pour éviter toutes potentielles failles (déjà qu'il n'est pas crypté).

Enfin, j'ai ajouté une action de déconnexion, ici encore pas de difficulté, il fallait seulement détruire la session.

D - Gestion des moteurs

Maintenant qu'il est possible de me connecter en admin, je peux commencer à développer les fonctionnalités admin comme l'ajout, édition et suppression de moteur de jeux.

L'ajout, j'ai développé un formulaire, qui demande tout ce dont un moteur de jeux à besoin (déjà cité au-dessus). A savoir que les jeux connus et l'histoire du moteur sont des entrées facultatives, il est tout à fait possible de valider un formulaire sans ces informations car elles peuvent être difficiles à trouver. Les options obligatoires ont un attribut "required" pour forcer l'utilisateur à mettre une entrée.

Pour l'édition, les champs sont remplis automatiquement par les informations originales du moteur. Une page d'édition permet de supprimer ou d'éditer le produit.

E - Gestion des utilisateurs

Petit bonus que je me suis infligé: une gestion des utilisateurs avec la possibilité d'en supprimer et de modifier leur nom, email et rôle.

F - Implantation des images

Les images des moteurs ont été ajoutées vers la fin, je ne savais pas encore comment j'allais m'y prendre. Finalement, j'ai fait assez simple, le nom des logos est changé en l'id du moteur, de façon à ce qu'ils puissent être facilement accessibles en utilisant l'id précisé dans l'URL d'une fiche. Je stock ces images dans le répertoire *img/uploads/*.

G - Décision du style

Dans ce projet, j'ai changé 3 à 4 fois le style du site, je n'étais jamais satisfait du résultat. Finalement j'ai opté pour cette dernière version grisée qui était selon moi la meilleure de toutes. J'ai donné aux containers une opacité moindre que je trouvais plaisante.

H - Bulles de confirmation

A la fin du projet, j'ai pris la décision d'ajouter des messages de confirmation après chaque action de suppression afin d'éviter toutes mauvaises manipulations. J'ai alors utilisé les modals de bootstrap. J'ai dû forcer le text de celui-ci en couleur noir avec la class "text-black", car ma feuille de style met le texte des pages en blanc par défaut.

J'ai ici un premier bouton Supprimer qui ouvre le modal, demandant la confirmation de la suppression, le submit supprimer est bien celui de la pop-up.

./www/edit.php

```
                    <!-- Delete button -->
                    <input type="button" value="Supprimer" class="btn
btn-danger" data-bs-toggle="modal" data-bs-target="#deleteModal">

                    <!-- Delete modal -->
                    <div class="modal fade" id="deleteModal"
data-bs-backdrop="static" data-bs-keyboard="false" tabindex="-1"
aria-labelledby="deleteModalLabel" aria-hidden="true">
                        <div class="modal-dialog">
                            <div class="modal-content">
                                <div class="modal-header">
                                    <h1 class="modal-title fs-5
text-dark" id="deleteModalLabel">Supprimer <?php echo
$engine->getName() ?> ?</h1>
                                    <button type="button"
class="btn-close" data-bs-dismiss="modal"
aria-label="Close"></button>
                                </div>
```

```
                                <div class="modal-body text-dark">
                                    Souhaitez-vous vraiment supprimer
<?php echo $engine->getName() ?> ?
                                </div>
                                <div class="modal-footer">
                                    <button type="button" class="btn
btn-main" data-bs-dismiss="modal">Fermer</button>
                                    <input type="submit"
value="Supprimer" name="submit" class="btn btn-danger">
                                </div>
                            </div>
                        </div>
                    </div>
```

I - Barre de recherche

On m'a aussi proposé de faire une barre de recherche qui filtre les produits de la page d'accueil, j'ai cru qu'il s'agissait de tout un casse tête compliqué, pourtant il ne s'agissait que d'une requête get via un form qui target la même page, hors je ne savais que c'était possible. Ça m'a appris le rôle "search" des formulaires.

./www/index.php

```
        <!-- Search bar -->
        <form class="d-flex" role="search" method="get">
            <input class="form-control me-2" type="search"
value="<?php echo $filter ? $filter : "" ?>" name="filter"
placeholder="Recherche..." aria-label="Search">
            <button class="btn btn-main"
type="submit">Rechercher</button>
        </form>
```

J - Gestion des permissions de page

C'est à la fin de mon projet que j'ai pensé à la sécurité des pages, elles étaient en réalité accessibles par tous si l'on tapait leur nom dans l'URL. J'ai réfléchi à deux différentes façons de procéder : je connais l'existence des fichiers .htaccess, cependant je n'ai pas assez de connaissances pour savoir ce que cela peut me permettre concrètement. Je me suis quelque peu baladé sur des forums pour connaître ses capacités, mais ça me prenait trop de temps.

J'ai alors abandonné les recherches et ai opté pour une version plus simplifiée de procéder: une redirection php si l'utilisateur n'a pas le rôle d'admin.
Dans chaque pages censées être admin uniquement, j'ai ajouté cette portion de code au début des fichiers.

```php
if (!Session::isAdmin()) {
    header("refresh:0;url=index.php");
    exit;
}
```

III. Conclusion

Pour conclure, ce travail m'a beaucoup appris sur le PHP et sur le développement Web en général, qui, finalement je ne trouve pas si terrible qu'auparavant, malgré que ça me plaise moins que le développement pur et dur…

Pour mener à bien mon projet, j'ai appris à utiliser Bootstrap et sa longue liste de fonctionnalités. J'ai utilisé les pages responsives, les tableaux, les boutons, les zones d'insertion, les nav et bien plus.

De plus, j'ai encore plus manié le CSS, j'ai découvert notamment qu'il y avait des variables que l'on pouvait définir. Je l'ai cherché moi-même car je me disais qu'il était impensable de toujours mettre la même couleur à chaque fois, je m'étais dit qu'il y avait forcément la possibilité de définir des variables.

En ce qui concerne le PHP tel quel, j'ai appris à faire des redirections, des paramètres pré configurés, à utiliser les Sessions, à mêler le PHP au HTML…

Ce travail m'a bien servi dans mon apprentissage, et je n'ai aucun regret de m'être donné corps et âme dans ce devoir. Pour finir, je vois une progression dans mes capacités à rédiger, probablement parce que dans ce devoir il y avait beaucoup de contenu à présenter. Cependant je me trouve encore très loin du but.