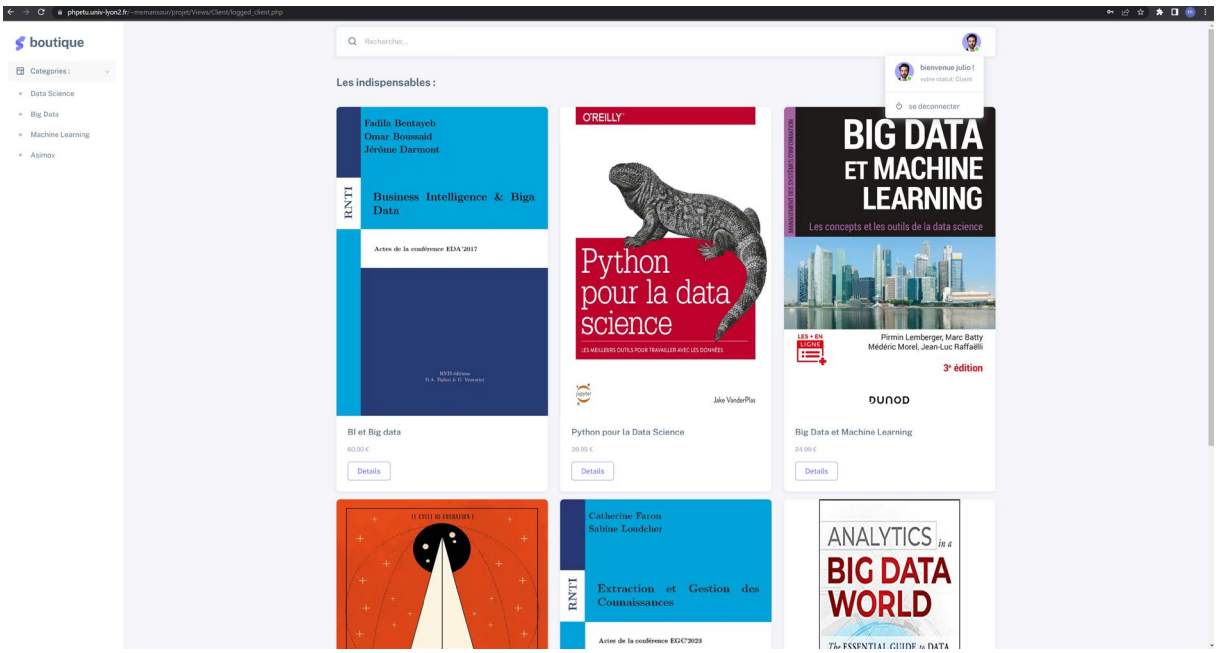
Document de synthèse pour le :

Projet de fin de module en Web Backend

Lien vers la page d'accueil du site :

https://phpetu.univ-lyon2.fr/~memansour/projet/Views/Client/home.php



Etudiant : Mehdi Mansour – N : 5221916

Diplôme préparé : Licence informatique parcours sciences des données

I. Introduction :

Ce travail vient couronner la fin du module web-Backend. Dans ce module nous avons pu nous initier aux fondamentaux de la gestion d'un site web du côté serveur. En particulier l'authentification, l'autorisation des utilisateurs, et la communication avec la base de données. C'est ainsi tout naturellement qu'un soin particulier a été porté à la mise en application des notions de session et de connexion avec la base de données, le tout en adoptant une architecture séparant drastiquement les langages et les finalités des fichiers.

Une stratégie de réalisation s'imposait, afin de ne pas s'éparpiller, d'investir un temps de travail efficace et ciblé. L'un des risques latents eut été de se faire happer par des considérations esthétiques chronophages sur le frontend. Ou de partir d'un modèle trop complexe, menant inexorablement à perdre en lisibilité du code ou à un bug récalcitrant.

Ainsi, sur la partie visible du client, le choix a été fait de recourir à un Template. Le lien fourni par le sujet en offrait une pléthore. Le premier, « Sneat », a été retenu, sur des critères de gratuité et de charte de couleurs.

Sur le plan du schéma relationnel de la base de données, le choix de la simplicité et de la lisibilité primait. Quatre tables ont été utilisées. Elles affèrent aux utilisateurs et aux produits, en abritant des attributs peu nombreux, mais suffisants pour mettre en place une logique métier pertinente du côté serveur.

Pour le reste, l'élaboration des algorithmes de contrôle et de présentation, devait respecter une rhétorique claire et éviter la cuistrerie. En effet, chaque algorithme a été conçu en répondant à la question suivante : « quelles conditions faut-il vérifier ? que faire pour chaque condition ? ». Cette construction, bien qu'optimisable ou verbeuse quelquefois, offrait à mon sens le meilleur compromis de lisibilité et d'évolutivité du code.

En essayant donc de respecter ces règles de travail, ainsi que le cahier des charges du sujet, le site « Boutique » vit le jour. Il s'agit d'un site de vente de livres scientifiques. Ayant plusieurs catégories de livres, deux types d'utilisateurs et trois types d'autorisations. Il déploie de fonctionnalités permettant de créer des comptes, un affichage et des accès différents en fonction du type d'utilisateur. Ainsi qu'une interface dédiée à l'administrateur pour réaliser des opérations crud dans la base de données.

Bonne navigation.

II. Architecture du site :

Le modèle MVC est adopté pour la réalisation de cette application :

- Dossier contrôleurs : contenant respectivement le contrôleur de la base de données, de l'authentification et des produits.

- Dossier des modèles : chaque fichier correspondant à une abstraction d'une (ligne de) table dans la base de données, sous forme d'une classe.

- Dossier des vues : partagées entre les vues admin, les vues client, et les vues d'authentification. Chaque vue est un couple de fichiers portant le même nom, l'un en html contenant le visuel et l'autre en PHP fixant les règles d'affichage.

Le diagramme suivant permet d'appréhender de façon simplifiée l'arborescence du projet :

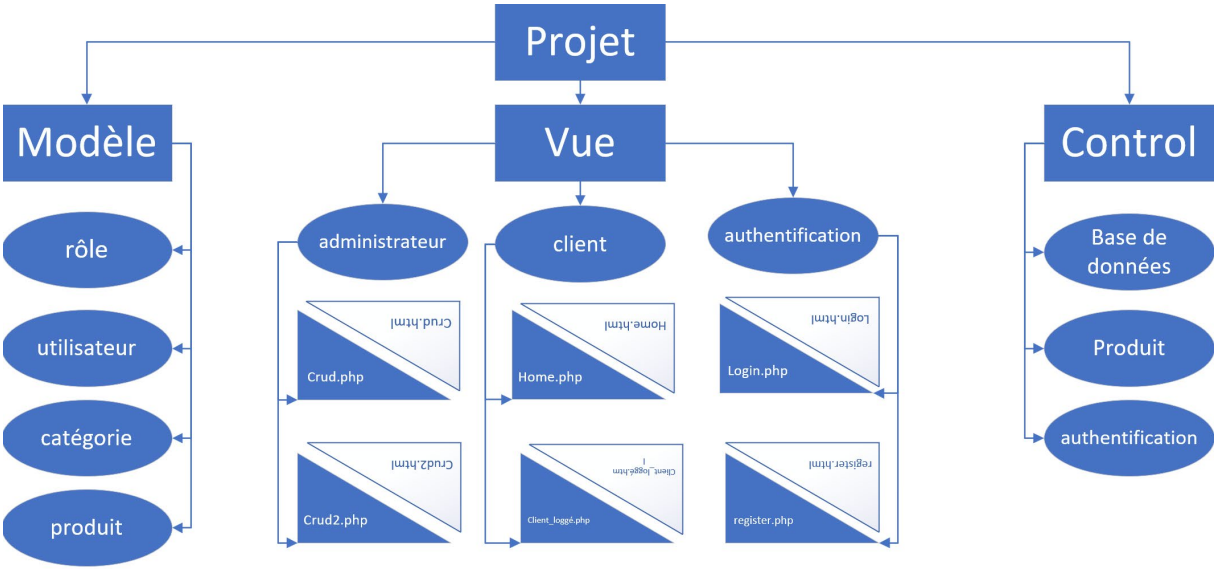


Figure1: Arborescence du projet

III. Schéma conceptuel de la BDD :

Comme évoqué plus haut, le choix de travail est de ne pas complexifier inutilement nos structures de départ. Le but étant de se concentrer sur les notions fondamentales de sessions, de dialogue avec la BDD et d'architecture.

Ainsi, une base de données avec quatre tables était suffisante pour la mise en avant des différentes fonctionnalités à produire.

Les quatre tables sont :

- Roles : permettant de stocker l'Id d'un rôle et sa nomenclature, dans notre cas deux rôles existent, l'administrateur et le client enregistré.

- Users : chaque ligne reprend les attributs d'un client enregistré, ou d'un administrateur. La colonne roleid étant une clé étrangère qui référence l'Id de la table Roles.

- Categories : les différentes catégories de produits par nom et id.

- Produits : stocke chaque produit disponible, a comme clef étrangère l'Id de la table Categories.

Le schéma conceptuel suivant permet d'avoir une vision d'ensemble de notre base de données :

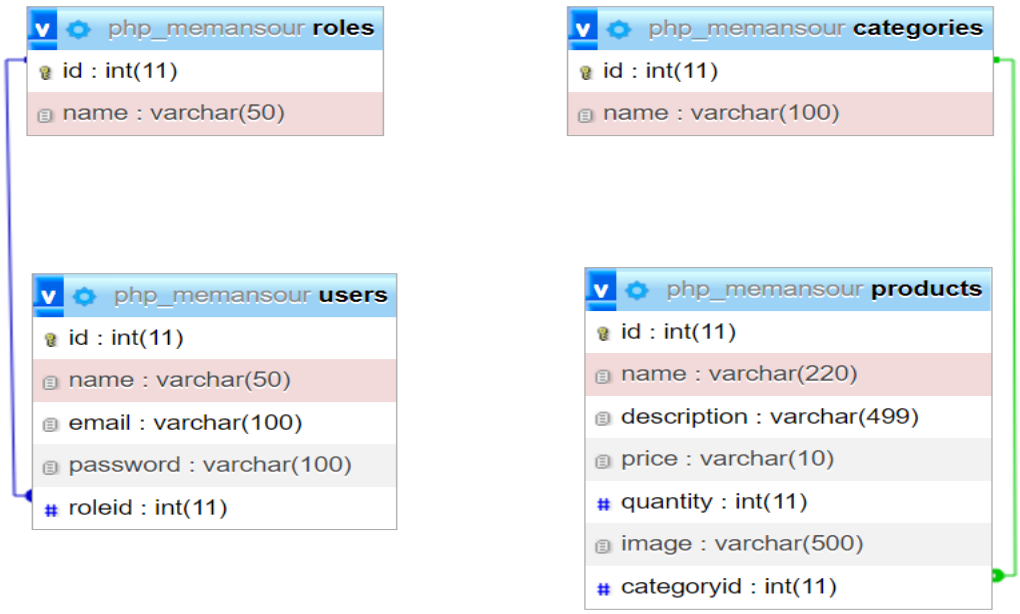


Figure 2 : Schéma conceptuel de la BDD

Dans une première version, dans la table Produits, la photo du produit était stockée sous forme binaire en extension BLOB. Ceci s'est avéré contraire aux bonnes pratiques et source de bugs. Désormais ce que l'on sauvegarde dans la base de données est une chaine de caractères spécifiant le chemin d'accès à l'image.

IV. Méthodes et algorithmes :

La simplicité du code a été une préoccupation de premier ordre lors de la réalisation du projet. D'un côté par respect des bonnes pratiques. Mais surtout pour gérer le risque d'un bug en cascade, qui ne puisse être solutionné, et qui mette en péril la livraison dans les temps. En effet, à mon niveau novice de savoir-faire, j'ai souvent rencontré des blocages qui mobilisent des heures de recherche, et qui demandent l'intervention d'un sachant pour débloquer.

Pour se prémunir contre ce risque, j'ai essayé de mettre en place des remparts méthodologiques :

- Une architecture claire et intuitive : en appliquant le modèle MVC. Ensuite, en consacrant à chaque vue HTML son petit gestionnaire PHP, quitte à répéter du code. En contrepartie de la redondance, un problème touchant une vue, pouvait être rapidement circonscrit et solutionné.

- Associer des modules réutilisables aux opérations qui se répètent : en d'autres termes, profiter du potentiel de la POO, pour créer des blocs logiques autonomes pouvant s'exécuter dans les différentes parties du programme. Les propriétés d'encapsulation et d'abstraction permettent

en plus d'imbriquer ce genre de modules les uns dans les autres. Ces blocs, du moment ou ils fonctionnent, deviennent un outil fiable auquel on fait confiance.

Pour illustrer cette vertu, prenons l'exemple de la méthode d'ajout de produit du contrôleur de Produit :

```php
public function addProduct(Product $product)
{
    $this->db=new DBController;
    if($this->db->openConnection())
    {
        $query="insert into products values ('','$product->name','$product->description','$product->price','$product->quantity','$product->image',$product->categoryid)";
        return $this->db->insert($query);
    }
    else
    {
        echo "Error in Database Connection";
        return false;
    }
}
```

Figure 3 : Encapsulation et modularité en POO, méthode addProduct() en exemple

Cette méthode peut être appelée par une instance de la classe productController, pour réaliser une tâche de routine qui est d'ajouter un produit dans la BDD, elle est utilisée dans « crud2.php » pour accomplir cette tâche. Cette méthode prend en argument un objet Product, qui décrit le nouveau produit à ajouter. Elle se sert de l'objet DBController qui est un attribut de l'instance productController, et qui lui permet le dialogue avec la BDD, car disposant lui-même de méthodes de connexion, de déconnexion et d'exécution de requêtes. Sans exploiter ces propriétés d'encapsulation et de modularité, le code aurait été moins lisible, complexe et moins robuste.

- Utiliser des algorithmes qui décomposent la tâche pas à pas : dans l'élan de la concision, on peut facilement être tenté d'économiser des lignes de code. Ceci nuit généralement à la clarté de la tâche à réaliser et finit par couter.
  La stratégie adoptée ici est de décomposer le code en mini structures logiques, permettant tour à tour de vérifier une condition, et d'appliquer le traitement adéquat. Cette Stratégie permet de lire le code de façon aisée, et d'y retrouver les correspondances « métier », les commentaires associés au code deviennent presque une évidence. Ce choix a permis de facilement retrouver les anomalies de fonctionnement et a facilité la maintenance.

V. Conclusion :

La mise en application des fondamentaux du web Backend lors de ce petit projet, a permis une immersion en conditions réelles et de façon non assistée dans la construction d'application web. Cette confrontation vertueuse aux difficultés pratiques de la réalisation, était une bonne occasion de faire confiance à son savoir-faire et de l'améliorer.

Toutefois, ce fût l'occasion également de revisiter des lieux communs et de s'alléger de certains a priori. Etant en parcours sciences des données, j'avais l'appréhension de la construction web, d'un côté pour son aspect « non exacte », et de l'autre pour l'aspect « laborieux » d'une construction html par exemple. L'utilisation d'un Template CSS, et la fiabilité de la classe TBS, m'ont permis de me réconcilier avec cette discipline, qui pourrait être un atout supplémentaire et valoriser mes compétences sur le marché.

Les problèmes de compatibilité et de dépréciation semblent être une constante de ce type de travail. Il peut être judicieux de les minimiser, en choisissant au préalable une panoplie d'outils techniques dont la compatibilité et l'actualité sont déjà validées.

Ce travail a été en somme une agréable opportunité de devenir un meilleur programmeur.

URLs :

L'Url de la Homepage du site est la suivante :

https://phpetu.univ-lyon2.fr/~memansour/projet/Views/Client/home.php

Un circuit optimal, de découverte du site et de validation des fonctionnalités :

Homepage > se connecter (en tant que client) > se déconnecter > s'enregistrer > se connecter avec le nouveau compte > se déconnecter > se connecter (en tant qu'admin) > supprimer un produit > ajouter un produit ..

Voici deux comptes existants dans la bdd :

- Admin : super_admin@gmail.com / password
- Client : client_lambda@gmail.com/ password

Les autres urls du site sont :

https://phpetu.univ-lyon2.fr/~memansour/projet/Views/Auth/login.php

https://phpetu.univ-lyon2.fr/~memansour/projet/Views/Client/logged_client.php

https://phpetu.univ-lyon2.fr/~memansour/projet/Views/Auth/register.php

https://phpetu.univ-lyon2.fr/~memansour/projet/Views/Admin/crud.php

https://phpetu.univ-lyon2.fr/~memansour/projet/Views/Admin/crud2.php

Afin de vérifier la sécurisation des sessions, il peut être envisagée de se connecter en tant que client, et d'essayer d'accéder directement au crud avec l'url etc.

Clefs de la boutique :

```
public $dbHost="mysql02.univ-lyon2.fr";
public $dbUser="php_memansour";
public $dbPassword="HT_nNV6QW/8Xvky/L633pt2lp";
public $dbName="php_memansour";
```