CFPT

Application web de e-commerce, partie administration

TPI 2021

MORRONE Flavio
20/05/2021

Table des matières

1. Introduction

Ce TPI est un projet réalisé par 3 candidats. Le projet seul était une charge de travail trop importante pour qu'il soit réalisé seul en 3 semaines. Nous avons donc mis en commun une architecture afin de remettre le projet en un afin qu'il soit totalement fonctionnel.

L'application est un site de e-commerce contenant 3 parties :

- Une partie administration et de gestion des utilisateurs
- Une partie de gestion du panier
- Une partie de gestion des stocks

Ma partie était celle de l'administration et de gestion des utilisateurs. Cette partie consiste à gérer les catégories, les utilisateurs ainsi que la connexion et enregistrement tout en étant sécurisant au maximum le site.

2. Cahier des charges

I. Matériels et logiciels à disposition

- Un PC avec Windows 10
- Deux écrans
- Un IDE au choix (Visual Studio Code, Netbeans, Notepad++)
- Un serveur web au choix (Laragon, EasyPhP, Wamp, Xamp)
- Un navigateur web (Firefox, Chrome, Edge)
- Un outil de versionning (Github)
- Un logiciel de gestion de base de données (PHPMyAdmin, MySQL Workbench)
- Des logiciels de bureautiques (Word, Excel, Figma)

II. Livrables

- Planning prévisionnel fourni le premier jour du TPI
- Rapport de projet
- Manuel utilisateur
- Résumé du rapport du TPI
- Journal de travail

-

III. Organisation

Élève ➔ Flavio MORRONE, email : flavio.mrrn@eduge.ch

Maître d'apprentissage ➔ Pascal COMMINOT, edu-comminotp@eduge.ch

Experts ➔

Pascal COURT, email : experts.tpige@outlook.com

Carol QUARROZ, email : experts.tpige@outlook.com

IV. Descriptif complet du projet

A. Description de l'application

Le but de ce projet est de développer une application de commerce en ligne, comportant un catalogue, des articles classés dans une hiérarchie de catégories, une gestion de panier de commandes, une gestion de stock et de livraison, en plus de la traditionnelle gestion des utilisateurs.

Présenté ainsi, le projet est trop important et complexe pour être réalisé dans le contexte d'un TPI, et de ce fait le travail a été réparti en trois modules indépendants, chacun des candidats travaillant sur ce projet se concentrant sur le module qui lui a été désigné.

Les modules prévus sont les suivants :

- Gestion des utilisateurs et des catégories du site

- Gestion des articles et des commandes

- Gestion du stock et des livraisons

La facturation et la gestion des paiements ne sont pas traitées.

Ce TPI se réalise la gestion des utilisateurs et des catégories du site.

B. Product Backlog

Titre	Inscription
Description	Faire en sorte qu'un utilisateur puisse créer son compte avec un email, nom, prénom, adresse et un mot de passe
Test	1, 2, 3, 4

Titre	Vérification de l'email
Description	Une fois l'inscription réalisée, l'utilisateur peut se connecter, mais son rôle est « NotVerified » afin de devenir « Customer ». Il a reçu un mail au moment de son inscription afin de confirmer son adresse email. Il lui suffit de cliquer sur le lien et de rentrer son email.
Test	9, 10, 11

Titre	Identification
Description	Après avoir créé son compte, l'utilisateur souhaite se connecter afin de pouvoir accéder au contenu du site. Il lui suffit d'entrer son email et son mot de passe. Si l'utilisateur se trompe trois fois la connexion, cela est noté dans les logs.
Test	5, 6

Titre	Récupération du mot de passe
Description	Si l'utilisateur perd son mot de passe, il a la possibilité de le récupérer. Il rentre son email dans un formulaire et reçoit un mail de réinitialisation du mot de passe si le compte existe. Il clique sur le lien et doit ensuite mettre son email pour vérifier son identité ainsi que son nouveau mot de passe.
Test	12, 13, 14, 15

Titre	Le profil
Description	Chaque utilisateur (vérifier ou non) peut apercevoir. Un utilisateur non vérifié ne peut pas le modifier.
Test	16, 17, 18, 19,20, 21, 22, 23, 24, 25, 26

Titre	Consultation des logs
Description	L'administrateur est dans la capacité de consulter les logs de l'application
Test	38, 39 ,40

Titre	Liste des utilisateurs
Description	L'administrateur peut voir la liste de tous les utilisateurs.
Test	41

Titre	Ajouter des utilisateurs
Description	L'administrateur peut ajouter des utilisateurs.
Test	44, 45

Titre	Modifier les utilisateurs
Description	L'administrateur peut modifier n'importe quels champs de tous les utilisateurs sauf lui
Test	43

Titre	Supprimer les utilisateurs
Description	L'administrateur peut supprimer les utilisateurs dont la date de validation est expirée
Test	42

Titre	CRUD Catégories
Description	L'administrateur a accès à un CRUD sur la table " categories ". Il peut créer des catégories et les rattacher à n'importe quelle autre catégorie. Il peut les déplacer ou il veut dans l'arborescence sauf dans ses catégories enfants. Il peut supprimer celles qui n'ont pas d'enfant.
Test	27, 28, 30, 31, 32, 33, 34, 35, 36, 37

Titre	Ligne d'Ariane
Description	Lorsqu'on souhaite voir le contenu d'une seule catégorie, une ligne contenant les parents de la catégorie doit s'afficher.
Test	29

C. Planning prévisionnel

Le planning prévisionnel est l'assemblage des taches que je devais réaliser avec l'ordre dans lequel je pensais les réaliser. Le but est de le respecter au maximum afin de bien organiser son travail et savoir si on est dans les temps. Il peut ne pas être respecté en fonction de si on a bien estimé le temps des taches ou pas.

Tâches à réaliser	Temps
Analyse	04:00
Inscription générale (formulaire simple)	01:00
Vérification de l'email	02:00
Formulaire d'identification (email - mot de passe) / login	01:30
Ajout dans les logs après 3 echecs	01:00
Formulaire récupération du mot de passe / envoie du mail	02:30
Modification du mot de passe après la redirection	02:00
Faire la page du profil	04:00
Modification du profil	02:30
Consultation des logs pour l'admin	02:00
Afficher les utilisateurs avec filtre de recherche par nom, prénom, email(admin)	02:30
Modification des users pour l'admin / mail pour informer	03:00
Suppression d'un utilisateurs (pas possible si il a des commandes et actifs il y a moin d'un an)	02:30
CRUD sur la table catégorie	04:00
Documentation	30:00
Tests	20:00
Journal de bord	03:25
	87:55

Figure 1 tâches du planning

1er jour	2e jour	3e jour	4e jour	5e jour	6e jour	7e jour	8e jour	9e jour	10e jour	11e jour	Total
04:00											04:00
01:00											01:00
02:00											02:00
	01:30										01:30
			01:00								01:00
	02:30										02:30
	02:00										02:00
		04:00									04:00
			02:30								02:30
			02:00								02:00
				02:30							02:30
					03:00						03:00
					02:30						02:30
						04:00					04:00
01:00	01:00	03:00	02:00	04:30	02:30		04:00	04:00	04:00	04:00	30:00
						04:00	04:00	04:00	04:00	04:00	20:00
00:15	00:25	00:25	00:25	00:25	00:15	00:15	00:15	00:15	00:15	00:15	03:25
08:15	07:25	07:25	07:55	07:25	08:15	08:15	08:15	08:15	08:15	08:15	
										87:55	87:55

Figure 2 répartition des tâches sur la semaine

D. Modèle logique de données

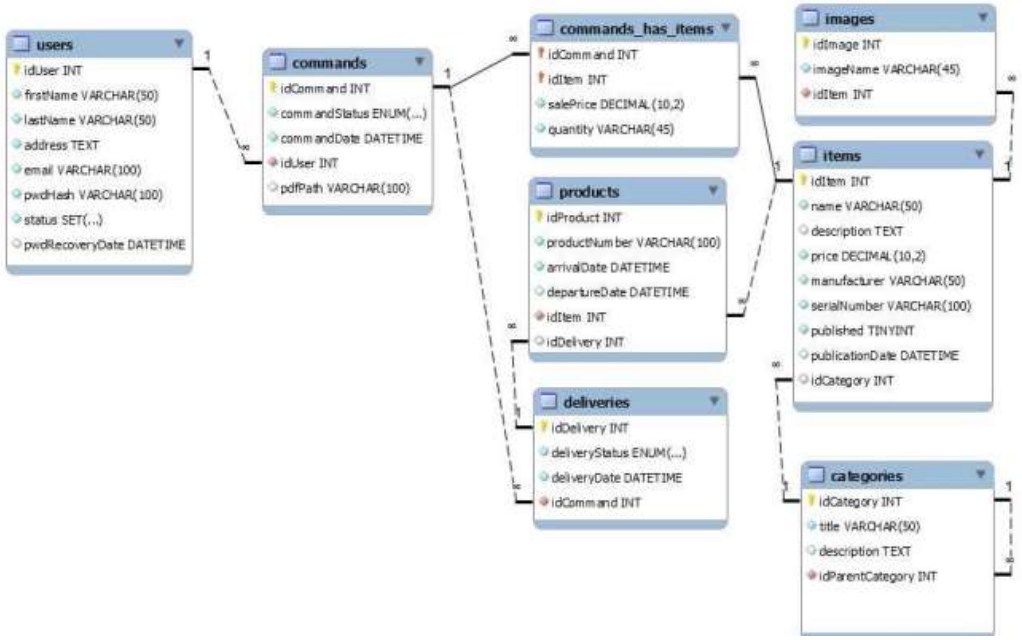

Figure 3 modèle de données fournis

Le modèle ci-dessus est celui qui m'a été fourni avec l'énoncé. Il contient tout ce qu'il faut pour que le projet puisse fonctionner avec les fonctionnalités de base. Mais pour les fonctionnalités que je devais faire, j'ai dû rajouter des champs ainsi qu'une table.

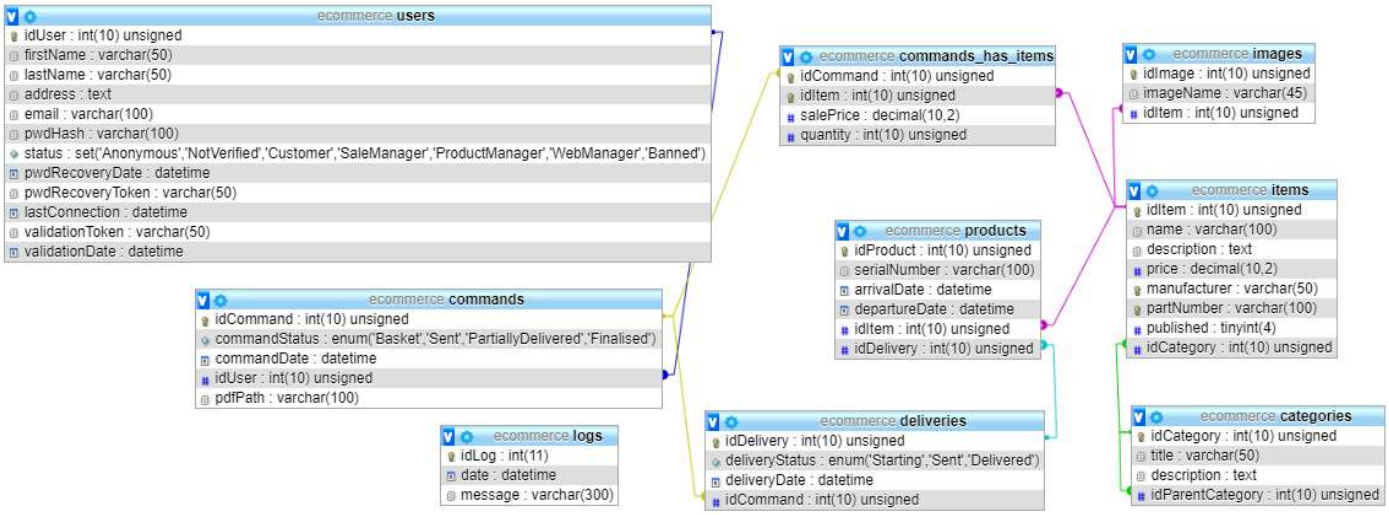

Figure 4 modèle de données après modification

Voici à quoi ressemble maintenant mon modèle de base de données. Comme nous sommes trois sur le projet et que nous avions chacun une partie à développer, je n'ai pas eu besoin d'utiliser toute la base de données. Pour cela, je vais mettre en avant les tables qui me sont nécessaires afin de réaliser ma partie.

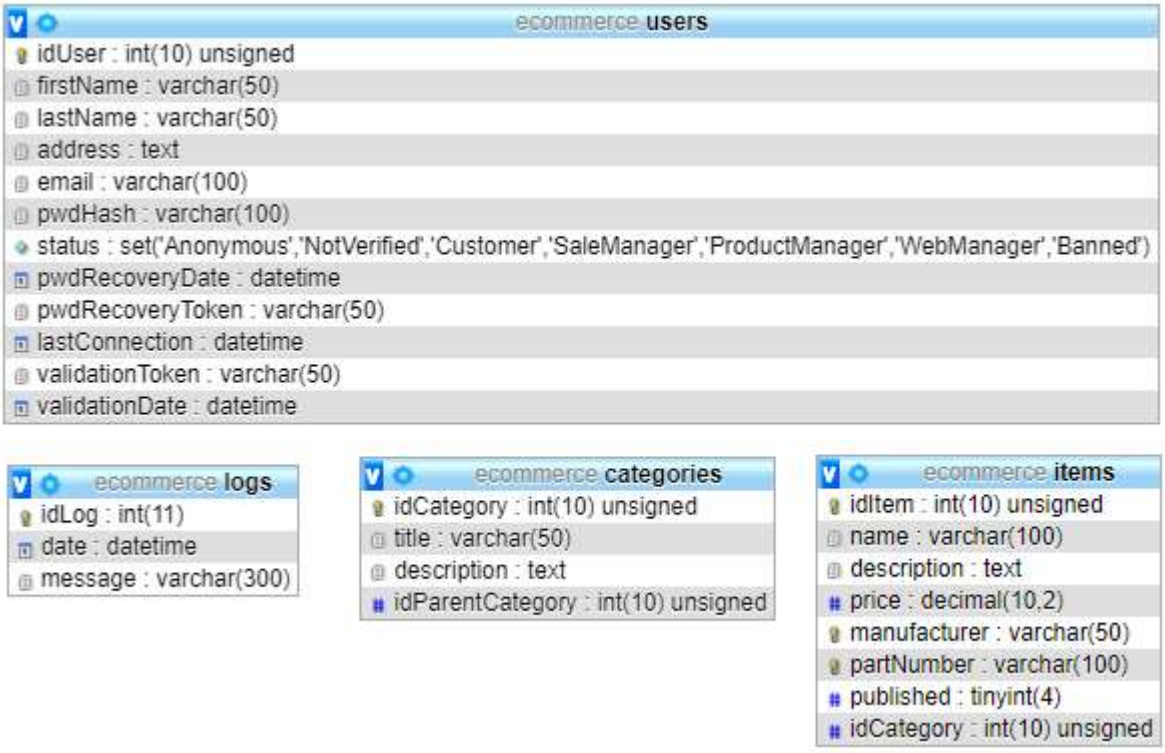

Figure 5 tables importantes

Les tables ci-dessus sont les seules que je manipule dans ma partie. On peut remarquer dans la table « users » que des champs ont été rajoutés par rapport à celui que l'on m'a donné de base.

Ils m'ont été utiles pour la validation de l'email et la récupération de mot de passe. Le but était de faire en sorte d'avoir une date limite ainsi qu'une clé permettant de prouver la demande et de sécuriser un maximum les demandes. C'était pour moi la meilleure façon de faire afin que cela ne soit pas complexe et sécurisé.

De plus, il m'était demandé de savoir la date de la dernière connexion afin de voir l'activité de l'utilisateur.

De plus, la table « logs » a été rajoutée. J'avais le choix entre deux possibilités. Soit enregistrer les logs dans la base de données, soit les enregistrer dans un fichier texte. J'ai choisi de créer une table dans la base de données pour plusieurs raisons :

- L'architecture utilisée dans ce programme (MVC) facilitait l'utilisation de la base de données.
- Plus propre et plus facile dans la base de données que dans un fichier texte.
- J'ai plus souvent eu l'habitude de faire comme ceci par le passé.

V. Points techniques évalués

A14 : Les fonctionnalités de l'administrateur du site concernant la gestion des utilisateurs sont réalisées.

A15 : Les fonctionnalités des autres utilisateurs, en lien avec la table users, sont réalisées.

A16 : Les mots de passe sont gérés de manière sécurisée.

A17 : La gestion des catégories du site est réalisée.

A18 : Les tests sont réalisés et documentés.

A19 : L'architecture mise en place avant le TPI est respectée, et permet l'intégration avec les autres modules du projet.

A20 : L'application est protégée contre les injections SQL et XSS.

3. Méthodologie

La méthodologie que j'ai utilisée se nomme **la méthodologie en 6 étapes**. Je l'avais expérimenté auparavant lors de différents modules. Elle consiste à séparer la réalisation du programme demandé en 6 étapes différentes. Ces étapes se nomment s'informer, Planifier, Décider, Réaliser, Contrôler et Évaluer.

Figure 6 méthodologie en 6 étapes

I. S'informer

La première étape de cette méthodologie était de lire en profondeur mon énoncé afin de bien comprendre le sujet. De plus, j'ai dû faire des recherches afin de savoir comment je pouvais faire certaines choses telles que la vérification de l'email ou encore le changement de mot de passe.

Puis j'ai posé mes questions à mon maître de TPI pour être certain d'avoir bien compris certaines choses.

II. Planifier

La planification a pour principe de prévoir l'organisation de son travail. Pour cela, je me suis légèrement inspiré de la méthodologie **SCRUM** afin de faire un Backlog complet contenant toutes mes *user-stories*. Je me suis inspiré de cette méthodologie uniquement pour le Backlog étant donné que je travaille seul, le reste ne m'était pas très utile.

Une fois le **Backlog** terminé, j'ai mis en place mon planning prévisionnel afin de savoir quand je pensais faire mes taches au départ du projet.

III. Décider

Durant tout mon TPI, il a fallu que je prenne des décisions sur la manière de faire certaines choses. Je pense par exemple au changement d'email ou j'ai pris au du temps à prendre la décision sur quelle procédure je devrais utiliser pour être le plus efficace possible.

IV. Réaliser

La réalisation représente la partie la plus importante de cette méthodologie. Elle consiste à réaliser les tâches mises en place lors de la planification et de la manière qui a été choisie lors de l'étape Décider.

V. Contrôler

Chaque partie et fonctionnalité du site ont été testées et documentées dans ce fichier afin que n'importe quelle personne, interne ou externe au métier, puisse les réaliser sans difficulté.

Chaque test a été effectué sous plusieurs navigateurs afin de vérifier qu'il n'y a pas de problème de compatibilités.

VI. Évaluer

La dernière étape est l'évaluation. Cette étape est incontournable pour analyser comment c'est dérouler le projet dans son ensemble et avoir une pensée critique sur la réalisation.

On peut se poser les questions suivantes :

- Qu'est-ce qui peut être amélioré ?
- Qu'est-ce qui a été positif ?
- Qu'est-ce qui a été négatif ?
- Comment le modifier ?

Le journal de bord est une pièce très importante pour la réalisation de cette étape. Il nous permet de revoir comment nous avons fait chaque chose et nos problèmes.

4. Outils utilisés

I. Laragon

Laragon est un environnement de développement web uniquement disponible sous Windows. Il intègre de différentes technologies telles qu'apache, MySQL et PHP. Il est très pratique pour installer des packages et des librairies. De plus, il facilite l'envoi de mails afin de ne pas être mis en spam.

Figure 7 laragon

II. Visual Studio Code

Figure 8 VS Code

Visual studio code est un IDE puissant disponible sous Windows, Linux et Ubuntu. Il possède de base un support intégrant JavaScript, TypeScript et Node.js. Mais il dispose d'un très riche nombre d'extensions permettant de prendre en compte d'autres langages tels que C++, C#, Java, Python PHP et bien d'autres.

Les extensions que j'ai choisies ne sont autres que « PHP Intelephense » et « PHP Debug ».

III. MySQL Workbench

MySQL Workbench est un outil visuel pour les architectures de bases de données MySQL. Il fournit des outils de modélisations de données, de développement SQL et d'administration pour la configuration d'un serveur. Il est disponible sous Windows, Linux et Mac.

Figure 9 MySQL Workbench

IV. Github

Figure 10 Github

Github est un service d'hébergement et outil de versionnage de logiciel. Il utilise le programme git afin de gérer les versions. Cela m'aura servi de gestion de versions ainsi que de sauvegarde afin de ne pas tout perdre en cas de problème technique.

V. Antidote

Antidote est un correcteur, de langue, avancé avec filtres intelligents. Il permet de facilement corriger ses fautes de français et même d'autres langues. Il m'a beaucoup servi pour la relecture de la documentation

Figure 12 Antidote

VI. Xdebug

Figure 11 Xdebug

Xdebug est une extension pour PHP permettant, comme son nom l'indique, de débugger du code PHP. Elle m'a permis d'avancer plus vite sans rester bloqué sur des bugs pendant trop longtemps.

5. Architecture

L'architecture de l'application se nomme Modèle Vue Contrôleur (MVC). Elle consiste à séparer le programme en 4 parties.

- Le routeur
- Les contrôleurs
- Les modèles
- Les vues

Le routeur est le point central du projet. Il permet la redirection sur les différentes pages selon les actions de l'utilisateur. Cela permet de gérer les accès plus facilement.

Les contrôleurs s'occupent de la liaison entre le modèle et la vue. Il effectue le traitement des données reçu par les vues (formulaires) et les modèles afin de mettre à jour la vue si nécessaire ou la base de données.

Les vues permettent l'affichage des données envoyées par les contrôleurs. Elles n'ont aucune liaison directe avec la base de données.

Les modèles contiennent les données que l'on va manipuler. Ils vont assurer leurs gestions. C'est le modèle qui contient les données de la base.

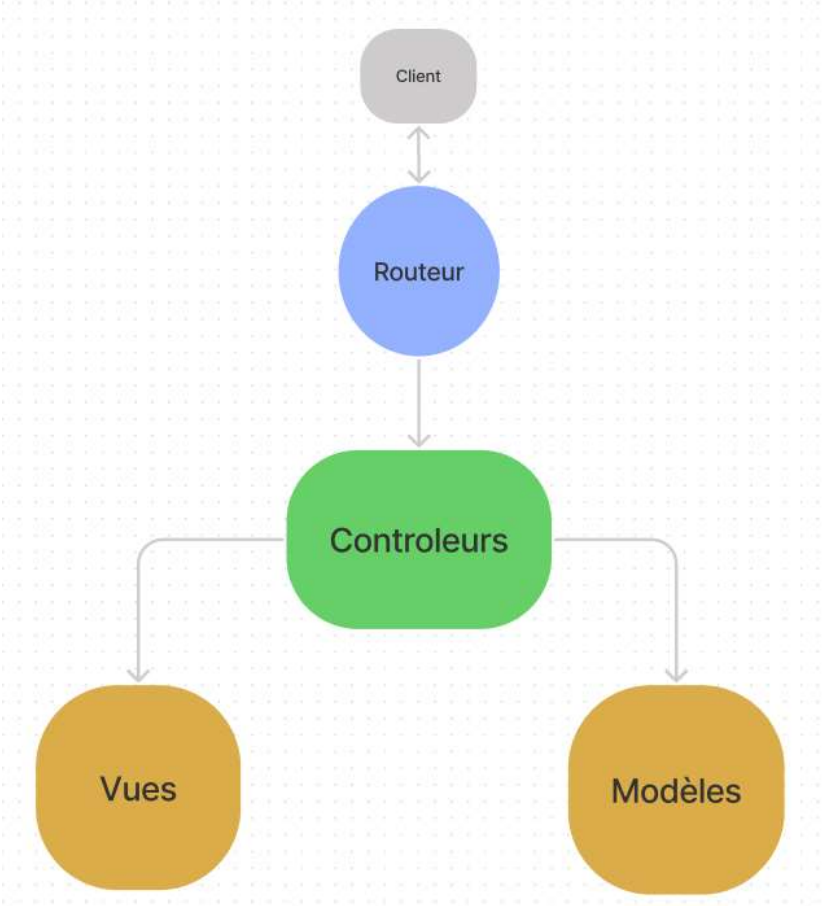

Figure 13 architecture MVC

6. Analyse fonctionnelle

I. Fonctionnalités intégrées

01 Enregistrement d'un utilisateur

La première fonctionnalité de ma partie est l'enregistrement d'un utilisateur. Chaque nouvelle personne arrivant sur le site est considérée comme une personne avec le rôle « Anonymous ». Ses personnes ont donc la possibilité de s'inscrire sur le site en renseignant les informations suivantes :

- Prénom
- Nom
- Email
- Adresse
- Mot de passe

L'utilisateur doit confirmer son mot de passe afin que cela soit confirmé et qu'il ne se soit pas trompé.

02 Vérification de l'email

Après son inscription, l'utilisateur créé peut désormais se connecter. Mais il possède le rôle « NotVerified » ce qui le bride dans l'utilisation du site.

Afin de se débarrasser de ce rôle, il doit vérifier son email. Pour cela, un mail lui a été envoyé lors de son inscription pour la valider.

Il lui suffit de cliquer sur le lien, ce qui le redirigera vers une page du site. Sur cette page se trouve un simple formulaire comprenant un champ pour mettre son email, ainsi qu'un champ en lecture seule avec la clé de vérification.

Il doit remplir le champ avec l'email qu'il a fourni afin de vérifier que cela ne soit pas quelqu'un de malveillant essayer de mettre des clés aléatoires.

Une fois l'email fourni, il clique sur le bouton valider et son compte sera mis à jour. Il pourra désormais apercevoir que son compte possède à présent le rôle « Customer ».

Pour cela, il dispose de 24h au maximum, après son compte pourra être supprimé par les admins et il ne pourra plus le valider.

03 La connexion

Après l'enregistrement, l'utilisateur peut se connecter. Si son email est vérifié, il possède le rôle « Customer ». Sinon, son rôle est « NotVerified ».

Pour se connecter, il doit cliquer sur le bouton « Se connecter » puis d'entrer son email ainsi que son mot de passe.

04 *Affichage et modification du profil*

Toutes les personnes connectées ont un accès à leur profil.

Pour tout le monde, le profil affiche les informations personnelles de l'utilisateur. C'est-à-dire son nom, son prénom, son adresse, son email. Le mot de passe n'est pas affiché dans un but de sécurité.

Pour les personnes bannies ou non vérifiées, l'accès à la modification de valeurs est impossible.

La modification des informations personnelles est disponible pour tous les autres rôles.

Pour modifier son nom, son prénom, son adresse, il suffit de modifier les champs et de cliquer sur le bouton « Modifier », disponible que si nos rôles le permettent.

Pour le mot de passe, il y a 3 champs dans la page. Un pour vérifier que l'on connaisse bien le mot de passe actuel, deux autres pour mettre le nouveau mot de passe. Si tous les champs sont bien remplis et que le mot de passe actuel est correct, lorsque l'on cliquera sur le bouton « Modifier », le mot de passe sera changé.

Pour la modification de l'email, c'est un peu plus sécurisé. Lorsque l'on modifie le champ de l'email, cela va vérifier si l'email n'est pas déjà pris dans la base. S'il n'est pas déjà utilisé, un mail sera alors envoyé, avec un lien, sur notre email pour confirmer que nous voulons vraiment changer d'email. Lorsque l'on clique sur le lien, on est redirigé sur une page nous demandant notre email actuel suivi de notre nouvel email et une clé de validation.

Lorsque vous rentrez votre email actuel puis votre nouvel email, cliquez sur modifier et elle sera alors modifié. Vous recevrez un courriel de confirmation.

05 *Récupération du mot de passe*

Sur la page de connexion, il y a la possibilité de cliquer sur « Mot de passe oublié » ce qui permet de renseigner son email afin de faire une demande de modification de mot de passe.

Pour cela, lorsque vous rentrez votre email, vous allez recevoir un courriel avec un lien. Si vous cliquez sur le lien, vous aurez la possibilité de rentrer un nouveau de passe. Et comme pour les autres modifications et vérifications, vous devez entrer votre email pour vérifier que c'est bien vous. Si toutes les données sont correctes, le mot de passe sera modifié.

06 Création/Affichage/Modification/Suppression des utilisateurs

Dans ce cite, uniquement pour les administrateurs, il est possible de voir la liste des utilisateurs. Grâce à cette liste, il a la possibilité de modifier leurs données. Ceux-ci seront prévenus par mail.

De plus, il peut aussi supprimer les utilisateurs qui n'ont pas validé leurs emails 24h après leurs demandes d'inscription. Il a aussi la permission de retirer ceux qui n'ont pas de commandes en cours et qui sont inactifs depuis plus d'un an.

Pour finir, les admins peuvent créer un nouvel utilisateur. Cet utilisateur sera créé en bypassant les vérifications d'email.

07 Les logs

Il est aussi possible, via un compte admin de voir les logs du site. Les logs sont enregistrés dans la base de données lorsqu'un utilisateur essaie de se connecter 3 fois de suite ou plus avec le même email et qu'il échoue à chaque fois.

Elles sont aussi enregistrées à chaque demande de récupération d'email.

08 Création/Affichage/Modification/Suppression des catégories

Tous les utilisateurs du site peuvent voir les catégories. Ils ont la possibilité de soit voir la liste des catégories, soit afficher tous les objets contenus dans chacune d'entre-elle à l'aide d'un menu déroulant.

Le menu affiche les catégories selon les parents/enfants.

Une ligne d'Ariane permet de remonter facilement les catégories parentes de celle-ci.

Les WebManager peuvent, contrairement aux autres rôles, modifier, ajouter, supprimer les catégories. Le titre ne doit pas être vide.

II. Structure

Mon programme est développé avec une architecture MVC. Ce qui signifie que la séparation des fichiers est bien précise et permet une compréhension plus facile.

- annexes/ : ce dossier contient uniquement les documents tels que la documentation les plannings, etc. Aucun lien avec le code.
- commons/ : L'use case de base du programme. Ce dossier était fourni avec l'énoncé et uniquement la classe Session a été modifiée. Il a les fichiers utiles à la forme de base du site.
  - commons/controller : Ce sont tous les contrôleurs de base et nécessaires pour le projet (accessDenied, home, Routes).
  - commons/model : possède les modèles FlashMessages, Menu, Session, DbConnection. Ils seront utiles pour tout le projet.
  - commons/views : contient les vues pouvant être appelées à tout moment afin d'afficher une erreur ou des flashMessages, etc…
- css/ : toute la css du projet
- db/ : contient une exportation de la base de données
- images/ : possède les images du site
- js/ : tous les scripts JavaScript du site
- uc/ : Tous les cas d'utilisation de la méthode MVC se trouvent dans de dossier. À chaque fois qu'une nouvelle table de la base de données est utilisée dans le code, on créer un nouveau cas d'utilisation à l'intérieur. Tous les sous-dossiers de uc/ fonctionne de la même manière. Pour cela, je vais expliquer uniquement le fonctionnement général.
  - uc/*/ : Pour chaque cas d'utilisation, on retrouve une architecture MVC ainsi qu'un register.php . Le register est indispensable, car il permet d'enregistrer l'use case à l'index du site. Et de gérer les droits de chaque Routes.
    - uc/*/controllers : le sous dossier controllers contient tous les contrôleurs utiles pour manipuler les données de la base
    - uc/*/model : Les modèles de chaque cas se trouvent dans le dossier modèle. Ils serviront à stocker les données ainsi qu'à les récupérer de la base et faire des manipulations dessus. Les modèles ont un fonctionnement hybride. Ils stockent les valeurs de la base grâces

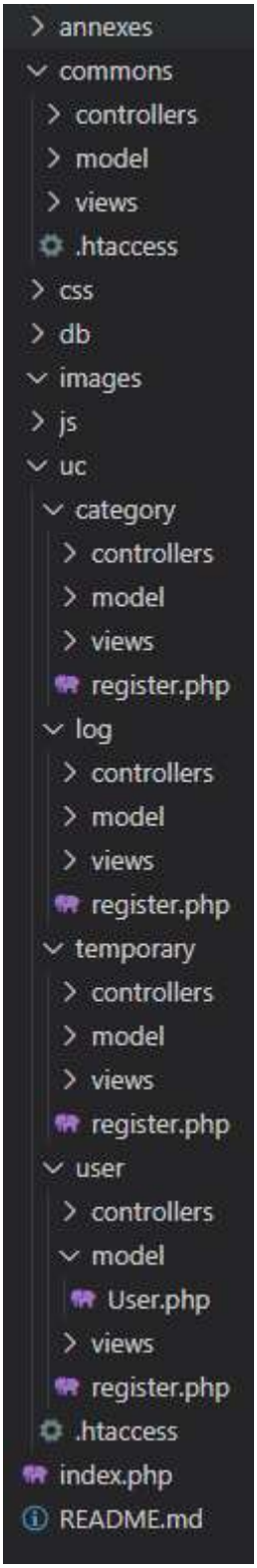

Figure 14 structure du projet

à des instances, mais peuvent aussi exécuter des opérations grâce à des fonctions statiques
        - uc/*/views : Toutes les vues de chaque use cases se trouvent dans ce dossier. Elles seront appelées afin d'afficher les données traitées.
    - index.php : l'index est le routeur du projet. Il redirige vers chaque contrôleur selon la requête et importe tous les cas d'utilisations.

Tous les .htaccess ont uniquement « deny from all » comme contenu. Cela servira à bloquer toute personne essayant de se balader dans les données à l'aide de l'URL ou autre. Uniquement l'index.php peut être appelé et il inclura les fichiers selon les requêtes.

III. Les uses cases

Comme expliqué aux dessus, notre travail se répartit sur différents uses cases. Ils servent à séparer le code de façon logique afin qu'il soit plus facile à comprendre et à utiliser.

Personnellement, j'ai utilisé 3 use cases différents.

- Users
- Category
- Log

Chaque cas est constitué de 3 sous-dossiers constituant l'architecture MVC ainsi que d'un fichier register.php.

A. Users

Ce cas d'utilisation est celui sur lequel j'ai le plus travaillé. Comme mon travail consistait à faire le côté administration, les utilisateurs sont beaucoup mis en avant dans cette partie.

01 Model

Pour gérer plus facilement cette partie, la classe « User » m'a été fournie. Cette classe contenait les champs du premier modèle de données que j'ai reçu.

Étant donné que la table « users » a dû être modifiée afin de pouvoir réaliser mes tâches, j'ai aussi dû rajouter les champs supplémentaires dans cette classe ainsi que ses getters setters.

Cette classe contient aussi plusieurs méthodes statiques permettant de récupérer ou d'insérer des données dans la base. Mais certaines nécessitent un peu plus d'explication.

(1) verifyRecoverTokenEmail

```php
/**
 * Vérifie si le token de récupération de mot de passe correspond a l'email
 * @param string le token a vérifier
 * @param string l'email du compte
 * @return bool true si il correspond false si correspond pas
 */
public static function verifyRecoverTokenEmail(string $token, string $email): bool
{
    $sql = "SELECT * FROM users WHERE pwdRecoveryToken = :token AND email = :email;";
    $req = DbConnection::getInstance()->prepare($sql);
    $req->bindParam(":email", $email);
    $req->bindParam(":token", $token);
    $req->execute();

    if ($req->fetch() === false) {
        return false;
    }

    return true;
}
```

Cette fonction est utilisée afin de sécuriser la récupération de mot de passe. En effet, elle oblige l'utilisateur à avoir un token et de connaitre son email afin d'éviter à un hacker de faire une demande de vérification de mot de passe sans avoir accès à l'email. En effet elle va vérifier dans la base de données si l'email entré possède bien le token entré par l'utilisateur (le token est mis automatiquement si on a accès au lien depuis son email).

(2) askRecover

```php
/**
 * Ajoute une date limite pour la demande de modification de mot de passe ainsi qu'un token unique
 * @param string email de l'utilisateur
 * @param string token unique
 * @return void
 */
public static function askRecover(string $email, string $token)
{
    $date = date_format(new DateTime("NOW"), 'Y-m-d H:i:s');
    $date = date("Y-m-d H:i:s", strtotime('+2 hours', strtotime($date)));
    $sql = "UPDATE `ecommerce`.`users` SET `pwdRecoveryDate` = :date, `pwdRecoveryToken` = :token WHERE (`email` = :email);";
    $req = DbConnection::getInstance()->prepare($sql);
    $req->bindParam(":email", $email);
    $req->bindParam(":token", $token);
    $req->bindParam(":date", $date);
    $req->execute();
}
```

AskRecover est la méthode obligatoire afin de pouvoir utiliser celle présentée juste avant. Elle permet justement d'ajouter dans la base le token de vérification ainsi que la date de limite.

Ses deux méthodes permettent une sécurité particulière au moment de la modification du mot de passe. Elles permettent d'éviter qu'un hacker tombe sur un lien facilement. La création d'un token unique contre cela.

(3) searchUser

```php
/**
 * SearchUser effectue une recherche sur la table dico
 * Le mot recherché peut se trouver dans le champ nom, pérnom ou email
 *
 * @param  mixed $word
 * @param int offset
 * @param int limit
 * @return array
 */
public static function SearchUser(?string $word, int $offset, int $limit): array
{
    $word = "%$word%";
    $sql = 'SELECT * FROM users
            WHERE firstName LIKE :word
            OR lastName LIKE :word
            OR email LIKE :word
            OR status LIKE :word
            LIMIT :offset, :limit';
    $req = DbConnection::getInstance()->prepare($sql);
    $req->setFetchMode(PDO::FETCH_CLASS, 'User');
    $req->bindParam(":word", $word, PDO::PARAM_STR);
    $req->bindParam(":offset", $offset, PDO::PARAM_INT);
    $req->bindParam(":limit", $limit, PDO::PARAM_INT);
    $req->execute();
    return $req->fetchAll();
}
```

Voici la dernière méthode pour laquelle je pense qu'il faut montrer plus d'attention.

Cette méthode permet de faire la recherche d'un mot sur plusieurs champs. Elle permet donc la recherche des users dans la page réservée aux admins. Cela facilite le travail de l'administrateur.

02 Controllers

(1) addUser.php

C'est un contrôleur simple permettant à l'admin d'ajouter un utilisateur. Il va uniquement filtrer que tous les champs soient remplis et que l'email ne soit pas déjà pris.

(2) deleteUser.php

Celui-là demande légèrement plus d'attention. Il sert à supprimer un utilisateur avec un id donné. La comparaison de date n'est pas quelque chose que l'ont fait tous les jours.

```php
$date = new DateTime("NOW");
$lastYear = $date->modify('-1 year');
$validateDate = $user->getValidationDate();
$validateDate = DateTime::createFromFormat('Y-m-d H:i:s', $validateDate);
$lastConnection = strtotime('Y-m-d H:i:s', $user->getLastConnection());
$countCommands = User::countCommands($user->getIdUser());
if ($validateDate !== false) {
    if ($validateDate < $date && $user->hasRole(User::USER_ROLE_NOT_VERIFIED)) {
```

Comme on le voit je créer une date avec la date d'aujourd'hui et je lui retire une année. Cela permet après de regarder si la personne c'est connecté il y a moins d'un pour la dernière fois avec uniquement l'opérateur « < ».

(3) editUser.php

EditUser.php est un contrôleur complexe permettant la modification de tous les champs d'un utilisateur par l'administrateur.

```php
if (filter_has_var(INPUT_POST, 'edit')) {
    $id = filter_input(INPUT_POST, 'id', FILTER_VALIDATE_INT);
    $user = User::findById($id);
    if ($user !== null) {
        Session::Set(USER_UPDATE_ID, $id);
    } else {
        FlashMessage::AddMessage(FlashMessage::FLASH_RANKING_DANGER, 'Une erreur est survenue.');
        header("Location: " . Routes::PathTo('user', 'showUsers'));
        exit;
    }
}
```

Étant donné que l'on parle de la modification d'un utilisateur, la sécurité est plus importante ici que dans tous les autres. C'est pourquoi dès que je rentre dans le contrôleur je stocke l'utilisateur concerné dans la session afin d'éviter toute manipulation frauduleuse.

```php
foreach ($verifyStatus as $s) {
    if (
        trim($s) != User::USER_ROLE_UNDEFINED
        && trim($s) != User::USER_ROLE_NOT_VERIFIED
        && trim($s) != User::USER_ROLE_CUSTOMER
        && trim($s) != User::USER_ROLE_SALE_MANAGER
        && trim($s) != User::USER_ROLE_PRODUCT_MANAGER
        && trim($s) != User::USER_ROLE_WEB_MANAGER
        && trim($s) != User::USER_ROLE_BANNED
    ) {
        header("Location: " . Routes::PathTo('user', 'showUsers'));
        FlashMessage::AddMessage(FlashMessage::FLASH_RANKING_DANGER, 'Le status n\'est pas correct.');
        exit;
    }

    if (trim($s) == User::USER_ROLE_BANNED && $user->getEmail() == Session::getUser()->getEmail()) {
        header("Location: " . Routes::PathTo('user', 'showUsers'));
        FlashMessage::AddMessage(FlashMessage::FLASH_RANKING_DANGER, 'Vous ne pouvez pas vous bannir.');
        exit;
    }
}
```

De plus, je dois vérifier s'il essaie de se bannir lui-même ou pas et si les rôles sont corrects.

(4) Login.php

Il effectue la connexion de l'utilisateur et enregistre dans les logs au bout de 3 essais manqués.

```php
$previousEmail = Session::Get('previousEmailTry');
if ($email != $previousEmail) {
    $currentTryNumber = 0;
} else if ($currentTryNumber >= 2) {
    Log::addLog("La connexion a echoué " . ($currentTryNumber + 1) . " fois de suite sur le même poste avec l'email $email.");
}
Session::addTry($email, $currentTryNumber);
```

Pour cela, je m'aide de la session afin de sauvegarder le nombre d'essais même après un rafraichissement.

(5) modifyEmail.php

Dans cette partie, on s'occupe de la modification de l'email. Pour cela, la sécurité doit être maximale, car si l'on perd l'accès à l'email du compte, on perd tous les moyens de récupération du mot de passe et de connexion.

Pour ajouter une sécurité, j'ai décidé d'ajouter un champ dans la base de données contenant une clé unique. Cette clé est liée à l'utilisateur et donc il est le seul à la connaitre. Pour cela, on lui envoie un lien par email afin qu'il soit le seul à le connaître. Et donc au moment de la vérification, on vérifie si la clé correspond à l'email avant le changement.

```php
if (User::verifyRecoverTokenEmail($token, $email)) {
```

(6) profil.php

Ce fichier permet la modification du profil si l'utilisateur est vérifié et pas banni.

(7) recoverPassword.php

La récupération de mot de passe est gérée ici. Il a fallu faire comme pour la modification d'email. C'est-à-dire gérer la vérification de la personne avec une clé unique.

(8) register.php

Ce script permet l'enregistrement des utilisateurs. La seule contrainte est de vérifier si l'email est déjà pris par un utilisateur avant l'enregistrement.

(9) showUsers.php

L'affichage des utilisateurs est légèrement différent d'un affichage normal, car j'ai fait une pagination ainsi qu'une recherche.

```php
$pages = array();
for ($i = 1; $i <= ceil($count / PAGE_SIZE); $i++) {
    $pages[$i] = Routes::PathTo("user", "showUsers") . "&page=$i";
}
```

Pour la pagination, je dois créer un tableau qui contient les routes vers toutes « pages d'utilisateurs » avec leurs id afin de le passer ensuite à page suivante.

(10) verify.php

Le script de vérification d'email utilise encore le principe de clé. Afin de s'assurer que l'utilisateur possède vraiment cet email, on lui envoie un mail avec un lien contenant sa clé de validation.

03 Views

(1) adduserform.php

Formulaire pour créer un utilisateur

(2) askRecoverPassword.php

Formulaire avec comme champ juste l'email.

(3) loginform.php

Formulaire de connexion au site.

(4) modifyEmail.php

Formulaire pour modifier son email.

(5) modifyPassword.php

Formulaire pour modifier son mot de passe.

(6) profil.php

Affiche le profil de l'utilisateur.

(7) registerform.php

Formulaire d'enregistrement pour les utilisateurs.

(8) updateuserform.php

Formulaire de modification d'utilisateur.

(9) usertable.php

Affiche tous les utilisateurs

(10) verifyform.php

Formulaire de vérification d'email.

B. Category

L'use case gérant les catégories ne m'était pas fournis. J'ai donc dû le créer de la même façon qu'était fait celui de l'utilisateur c'est-à-dire en modèle/vues/contrôleurs.

01 Model

J'ai donc commencé par créer la classe Category. Elle me permettra comme pour l'utilisateur de stocker les données de la base. Mais aussi de les récupérer et de les manipuler grâce à des méthodes statiques.

Je vais donc vous présenter celles que je trouve particulièrement intéressantes.

(1) buildArrayWithChild

Cette fonction est essentielle dans l'affichage de l'arborescence des catégories. Cela été un point bloquant, car il me fallait trouver une logique afin de créer un tableau qui contenait les parents avec ses enfants comme sous tableau.

```php
/**
 * Créer un tableau propre avec les parents contenants les enfants
 * @param array les catégories de la base
 * @return array
 */
public static function buildArrayWithChild(array $categories): array
{
    $sous_menu = [];
    foreach ($categories as $c) {
        $sous_menu[$c->getIdCategory()] = $c;
    }

    foreach ($sous_menu as $key => &$c) {
        if ($c->getIdParent() !== null) {
            $sous_menu[$c->getIdParent()]->children[] = $c;
        }
    }

    $menus = [];
    foreach ($sous_menu as $m) {
        if ($m->getIdParent() == null) {
            if (!self::isInArray($m, $menus)) {
                $menus[] = $m;
            }
        }
    }
    return $menus;
}
```

Je passe en paramètre la liste de toutes les catégories de la table afin de pouvoir toutes les parcourir.

Je les parcours toutes une fois et je les mets dans un tableau vide avec comme clé les identifiants leurs id.

Je parcours alors ce nouveau tableau. Je vérifie pour chaque catégorie si l'id de son parent n'est pas nulle (s'il n'est pas dans la première ligne de l'arborescence). Si cela n'est pas le cas, j'ajoute dans le tableau « children » de son parent la catégorie enfant.

Pour finir, je parcours encore une fois le tableau et je retire du premier niveau les catégories qui ont un parent afin d'éviter de les avoir deux fois dans l'arborescence.

Cette fonction m'a ensuite été d'une grande aide pour l'affichage.

(2) hasChild

Voici une méthode simple, mais que je trouve pratique. Elle me permet de vérifier si une catégorie possède des enfants juste avec une requête SQL.

```php
/**
 * Vérifie si une catégorie possède un enfant dans la base de données
 * @param int l'id de la catégorie
 * @return bool
 */
public static function hasChild(int $id)
{
    $sql = 'SELECT * FROM categories WHERE idParentCategory = :id;';
    $req = DbConnection::getInstance()->prepare($sql);
    $req->bindParam(':id', $id);
    $req->execute();
    $r = $req->fetchAll();
    if ($r == false) {
        return false;
    } else {
        return true;
    }
}
```

Elle va chercher dans la base toutes les catégories possédant comme parent la catégorie parent. Pour cela, je passe l'id en paramètre.

Si le résultat trouve quelque chose, elle retournera vraie sinon elle retournera fausse.

(3) hasCategoryChild

Et pour finir, une fonction qui vérifie si une catégorie possède une autre catégorie précise comme enfant.

La complexité de ce petit script est que la catégorie peut posséder un enfant qui lui-même possède un autre enfant et cela à l'infini.

```php
/**
 * Vérifier si une catégorie possède une autre catégorie comme enfant
 * @param int l'id du parent
 * @param int l'id de l'enfant
 * @return bool
 */
public static function hasCategoryChild(?int $idParent, ?int $idChild)
{
    $child = self::findById($idChild);
    if ($child->getIdParent() != null) {
        if ($child->getIdParent() == $idParent) {
            return true;
        } else {
            return self::hasCategoryChild($idParent, $child->getIdParent());
        }
    }
    return false;
}
```

Comme on peut le voir, je passe l'id du parent et de l'enfant. Je récupère l'enfant dans la base, grâce à l'id, afin d'avoir accès à toutes ses données (surtout pour savoir qui est son parent).

Je vérifie si l'id parent passé en paramètre est le même que celui qui est donné par l'enfant.

Si c'est le cas, cela veut dire que ce que l'on cherche est vérifié donc je retourne vrai. Mais si cela n'est pas le cas, on va vérifier si l'enfant à encore des enfants et si un d'eux correspond à celui que l'on cherche en rappelant ce qui fait que cela peut aller jusqu'à l'infini.

02 Controllers

(1) addCategory.php

Ce contrôleur va permettre d'ajouter une catégorie dans la base de données. Pour cela, il faut juste vérifier que la catégorie parent que l'utilisateur a choisie existe bien. Cela poserait problème au niveau de l'affichage de l'arborescence.

(2) deleteCategory.php

Ce script sert à supprimer une catégorie. Avant de la supprimer, il faut vérifier plusieurs choses.

- Qu'elle ne possède pas d'enfant
- Qu'elle ne possède pas d'items publiés
- Et de bien supprimer les items non publiés avant de la supprimer.

```php
if (!Category::hasChild($id)) {
    $count = Category::CountItems($id);
    if ($count[0] <= 0) {
        Category::deleteNotPublishedItems($id);
        FlashMessage::AddMessage(FlashMessage::FLASH_RANKING_SUCCESS, "La catégorie à bien été supprimé.");
        Category::deleteCategory($id);
    }
} else {
    FlashMessage::AddMessage(FlashMessage::FLASH_RANKING_DANGER, "Il est impossible de supprimer cette catégorie.");
}
```

(3) showCategory.php

Récupère toutes les catégories et les envois à la vue.

(4) showOneCategory.php

Ce contrôleur envoie à la vue la catégorie à afficher. Mais il permet aussi la création de la ligne d'Ariane.

Pour cela, j'ai encore du utilisé la récursivité afin de parcourir enfant par enfant et vérifié s'il n'y a pas de sous-enfant.

```php
/**
 * Créé un tableau contenant, pour chaques catégories, toutes ses catégories mères
 * @param Category la catégorie à analyser
 * @return
 */
function makeBreadCrumb($category)
{
    $result = [];
    $result[] = $category->getIdCategory();
    if ($category->getIdParent() != null) {
        $c = Category::findById($category->getIdParent());
        $result[] = makeBreadCrumb($c);
    }
    return $result;
}
```

La première fonction me permet de créer le tableau avec toutes les catégories à afficher.

Je créer un tableau ou j'ajoute la première catégorie et je vérifie si elle à un parent. Si elle en possède un, je rappelle cette méthode en passe le parent en paramètre.

```php
/**
 * Permet de créer la ligne d'arianne avec un tableau de catégory
 * @param array le tableau avec les catégories mères
 * @return string
 */
function buildBreadCrumb($bc)
{
    $result = '';
    if (isset($bc[1])) {
        $result .= buildBreadCrumb($bc[1]);
    }
    $id = $bc[0];
    $category = Category::findById($id);
    $result .= '<li class="breadcrumb-item"><a href="' . Routes::PathTo('category', 'showOneCategory') . '&id=' . $category-
    return $result;
}
```

La deuxième retourne un texte contenant du code HTML.

Comme le tableau est construit « à l'envers », on doit commencer par la fin. Je commence donc la fonction par vérifier s'il y a un enregistrement dans le tableau après le premier et s'il y en a un, je rappelle la méthode en partant de ce nouveau. Il va donc commencer par afficher le dernier puis revenir un par un.

(5) updateCategory.php

Il permet à l'administrateur de modifier une catégorie. Pour cela il faut vérifier que l'on ne mette pas comme parent son propre enfant.

03 *Views*

(1) addcategoryform.php

Formulaire pour ajouter une catégorie.

(2) categoriestable.php

Liste de toutes les catégories.

(3) showOneCategory.php

Affiche le contenu d'une catégorie choisie et sa ligne d'Ariane.

(4) updatecategoryform.php

Formulaire d'édition d'une catégorie.

C. Logs

01 *Model*

Le modèle servant à la gestion de logs et le plus facile de l'application. Il contient les champs de la base, leurs getters setters ainsi que deux méthodes :

- Un servant à récupérer tous les logs de la base de données
- L'autre sert à ajouter un nouveau log dans la base.

02 *Controllers*

(1) showLogs.php

Le seul contrôleur de ce use case sert à récupérer tous les logs de la base et à les envoyer à la vue.

03 *Views*

(1) logtable.php

Affiche tous les logs de la base de données.

7. Pan de tests

J'ai décidé de mettre en place un plan de tests afin de vérifier plus facilement que tout fonctionne. De plus, il est rédigé afin que n'importe qui puisse exécuter les tests sans problèmes de compréhension sauf pour quelques tests de sécurité que je voulais mettre en avant ou il faut légèrement de connaissance.

N° Test	Descriptif	Résultat attendu
1	**Inscription :**Faire une inscription avec des données valables (nom, prénom, adresse, email, mot de passe)	Redirection sur la page login avec un message de confirmation + envoi d'un mail
2	**Inscription :** Faire une inscription sans remplir toutes le données	Affichage d'un message d'erreur
3	**Inscription :** Inscription avec un email déjà pris	Redirection sur la page login avec un message d'erreur
4	**Inscription :** Inscription avec le mot de passe et confirmation de mot de passe différente	Affichage d'un message d'erreur
5	**Connexion :**Connexion sur un compte avec l'email non vérifié	Connexion avec comme rôle « NotVerified »
6	**Connexion :** Connexion avec des mauvais identifiants	Affichage d'un message d'erreur
7	**Déconnexion :** Cliquer sur le bouton déconnexion	Déconnecte l'utilisateur et le redirige sur la page d'accueil
8	**Changement de rôle :** Cliquer sur un rôle non sélectionné et disponible	Change le rôle de l'utilisateur
9	**Vérification d'email :** Entre le bon email lors de la vérification	Valide le compte + message de succès
10	**Vérification d'email :** Entre un mauvais email lors de la vérification	Message d'erreur
11	**Vérification d'email :** Modifie le token lors de la vérification	Message d'erreur
12	**Changement de mot de passe :** Mets son email dans le premier formulaire	Reçois un mail avec un lien

13	**Changement de mot de passe :** clique sur le lien	Redirigé vers un formulaire
14	**Changement de mot de passe :** rentre le bon email et le nouveau mot de passe dans les deux champs	Redirigé vers la page login avec message de succès
15	**Changement de mot de passe :** mets le mauvais email	Message d'erreur
16	**Profil :** affichage du profil	Affiche les données de l'utilisateur
17	**Modification du profil :** Change le nom et/ou le prénom et/ou l'adresse puis clique sur modifier	Modifie les données et affiche un message de succès
18	**Modification du profil :** change l'email puis clique sur modifier	Envoie un mail avec un lien. Le lien redirige sur une page pour la modification.
19	**Modification du profil :** modifie l'email en entrant son email actuel et son nouvel email.	Modifie le mail du compte. Envoie un mail au nouvel email pour confirmer.
20	**Modification du profil :** vérifie l'email en entrant un faux email actuel et son nouvel email.	Message d'erreur
21	**Modification du profil :** modifie le token de validation	Message d'erreur
22	**Modification du profil :** mets son ancien mot de passe ainsi que le nouveau mot de passe	Modifie le mot de passe et affiche un message de succès.
23	**Modification du profil :** mets le mot de passe actuel, mais pas de nouveau mot de passe	Message d'erreur
24	**Modification du profil :** mets le nouveau mot de passe, mais pas l'actuel	Message d'erreur
25	**Modification du profil :** met le mauvais mot de passe actuel	Message d'erreur
26	**Modification du profil :** tente de modifier le profil avec le rôle NotVerified ou Banned	impossible
27	**Catégories :** Voir l'arborescence des catégories dans le menu	Affiche l'arborescence dans le menu catégorie avec les sous-menus, etc.

28	**Catégories :** Clique sur une catégorie de l'arborescence	Affiche la page de la catégorie avec la ligne d'Ariane.
29	**Catégories :** Clique sur le parent de la catégorie dans la ligne d'Ariane	Redirige vers la page de la catégorie
30	**Affiche toutes les catégories :** En tant que WebManager	Des boutons " modifier " s'affichent sur toutes les catégories. Ainsi que des boutons " supprimer " pour celles qui n'ont pas de sous-catégorie.
31	**Affiche toutes les catégories :** En tant que NotVerified, Customer, SaleManager, ProductManager et Banned	Toutes les catégories s'affichent sans moyen de modification et suppression
32	**Modifie une catégorie :** clique sur le bouton modifier	Affiche un formulaire de modification
33	**Modifie une catégorie :** laisse le titre vide et essaie de modifier	Message d'erreur
34	**Modifie une catégorie :** modifie l'id de la catégorie parent dans le select avec une valeur impossible	Message d'erreur
35	**Modifie une catégorie :** modifie en laissant du texte dans le titre et avec un parent valable (ceux proposés).	Change la catégorie et redirige vers toute les catégories avec un message de succès.
36	**Supprime une catégorie :** modifie l'id avec celui d'une catégorie non supprimable ou inexistante	Message d'erreur
37	**Supprime une catégorie :** supprime une catégorie valable (celles qui ont les boutons)	Supprime la catégorie et affiche un message de succès.
38	**Logs :** échoue 3 fois (ou plus) la connexion avec la même adresse email	Enregistre une log
39	**Logs :** effectue une demande de récupération de mot de passe	Enregistre une log
40	**Logs :** le WebManager va voir la liste de log	Affiche toutes les logs du site
41	**Utilisateurs :** Le WebManager veut voir la liste d'utilisateurs	Affiche la liste des utilisateurs avec un bouton suppression pour ceux valables et bouton un supprimer.

42	**Utilisateurs :** Le WebManager veut supprimer une catégorie	Si elle n'a pas d'enfant et pas d'article publié, un bouton s'affiche. Il clique et cela supprime.
43	**Utilisateurs :** Le WebManager veut modifier un utilisateur. Il peut modifier tout ce qu'il veut tant qu'il ne laisse pas les champs vide et qu'il met des rôles et un email valables.	Modifie, clique sur le bouton et un message de succès s'affiche avec une redirection dans la liste des utilisateurs. De plus, cela envoie un mail à l'utilisateur pour le prévenir.
44	**Utilisateurs :** Le WebManager peut ajouter un utilisateur sans vérification d'email.	Rempli les champs correctement, message de succès et envoie d'un mail.
45	**Utilisateurs :** Le WebManager ne remplit pas tous les champs	Message d'erreur

8. Rapport de test

Le rapport de test ci-dessous est la réalisation du plan ci-dessus. Il m'a permis pour plusieurs fonctionnalités de résoudre des problèmes que je n'avais pas remarqués au cours du développement.

Date	N° Test	Résultat obtenu	OK / KO
04.05.2021	1	Redirection sur la page login avec un message de confirmation + envoi d'un mail	OK
04.05.2021	2	Rafraichissement + message d'erreur	OK
04.05.2021	3	Redirection sur la page login + message d'erreur	OK
04.05.2021	4	Rafraichissement + message d'erreur	OK
04.05.2021	5		OK
04.05.2021	6	Message d'erreur	OK
04.05.2021	7	Redirection à la page d'accueil déconnecté	OK
04.05.2021	8	Changement de rôle	OK
04.05.2021	9	Reçois un mail avec un lien+ redirection sur le login	OK
04.05.2021	11	Message d'erreur	OK
04.05.2021	12	Message d'erreur	OK
04.05.2021	13	Reçois un mail	OK
04.05.2021	14	Redirection sur une page avec un formulaire	OK
04.05.2021	15	Redirection + message succès	OK
04.05.2021	16	Affiche les données de l'utilisateur	OK
17.05.2021	17	Change le profil et message de succès	OK
17.05.2021	18	Envoie le mail	OK
17.05.2021	19	Change l'email et redirige à l'écran principal	OK
17.05.2021	20	Message d'erreur	OK
17.05.2021	21	Message d'erreur	OK
17.05.2021	22	Modifie le mot de passe et message de succès	OK

17.05.2021	23	Message d'erreur	OK
17.05.2021	24	Message d'erreur	OK
17.05.2021	25	Message d'erreur	OK
17.05.2021	26	Impossible	OK
17.05.2021	27	Affiche l'arborescence des catégories	OK
17.05.2021	28	Affiche la page de la catégorie avec la ligne d'Ariane	OK
17.05.2021	29	Redirige vers la catégorie parent	OK
17.05.2021	30	Affiche le tableau de catégories avec les boutons, mais les boutons supprimés s'affichent même si la catégorie a des items publiés	KO
17.05.2021	30.2	Problème réglé : les catégories ayant des articles publiés ne sont pas affichées et les articles non publiés sont supprimés avec la catégorie	OK
17.05.2021	31	Affiche les catégories sans boutons	OK
18.05.2021	32	Affiche le formulaire de modification	OK
18.05.2021	33	Message d'erreur	OK
18.05.2021	34	Message d'erreur	OK
18.05.2021	35	Change la catégorie et redirige vers toutes les catégories. L'arborescence dans le menu est mise à jour.	OK
18.05.2021	36	Message d'erreur	OK
18.05.2021	37	Supprime et message de succès	OK
18.05.2021	38	Commence après 4 essais et ne recommence pas si on essaie avec un nouvel email	KO
18.05.2021	38.2	Correction du problème	OK
18.05.2021	39	Ajoute dans les logs	OK
18.05.2021	40		OK
18.05.2021	41	Affiche la liste avec les boutons	OK
18.05.2021	42	Supprime l'utilisateur sélectionné	OK

18.05.2021	43	Message d'erreur si des champs sont vides et succès si rien n'est vide (sauf le mot de passe)	OK
18.05.2021	44	L'utilisateur est ajouté. Un mail lui est envoyé pour le prévenir de la création du compte.	OK
18.05.2021	45	Message d'erreur	OK

9. Conclusion

I. Planification effective

Le planning effectif est la façon dont j'ai vraiment réalisé mon programme. Cela signifie qu'il peut y avoir des différences avec le planning prévisionnel que j'ai réalisé au début du projet.

Tâches à réaliser	Temps
Analyse	04:00
Inscription générale (formulaire simple)	01:00
Vérification de l'email	02:00
Formulaire d'identification (email - mot de passe) / login	01:30
Ajout dans les logs après 3 echecs	01:00
Formulaire récupération du mot de passe / envoie du mail	02:30
Modification du mot de passe après la redirection	02:00
Faire la page du profil	04:00
Modification du profil	02:30
Consultation des logs pour l'admin	02:00
Afficher les utilisateurs avec filtre de recherche par nom, prénom, email(admin)	02:30
Modification des users pour l'admin / mail pour informer	03:00
Suppression d'un utilisateurs (pas possible si il a des commandes et actifs il y a moin d'un an)	02:30
L'admin peut créer un utilisateur	01:00
CRUD sur la table catégorie	04:00
ligne d'Arianne	00:30
Documentation	00:30
Tests	20:00
Journal de bord	03:25
	59:55

1er jour	2e jour	3e jour	4e jour	5e jour	6e jour	7e jour	8e jour	9e jour	10e jour	11e jour	Total
03:00											03:00
01:00											01:00
02:30											02:30
	01:00										01:00
			01:30								01:30
	01:00										01:00
	02:00										02:00
		02:00									02:00
		03:25									03:25
			01:40								01:40
			01:00	01:00							02:00
					02:00						02:00
				01:20							01:20
					01:00						01:00
					05:00	07:00					12:00
							01:30				01:30
01:30	02:00	02:10	03:00	04:00			03:30	04:00	08:00	08:00	36:10
	02:00						03:00	04:00			09:00
00:15	00:15	00:25	00:25	00:25	00:15	00:25	00:15	00:15	00:15	00:15	03:25
08:15	08:15	08:00	07:35	06:45	08:15	07:25	08:15	08:15	08:15	08:15	
										87:30	87:30

On peut déjà remarquer deux grosses différences dans les tâches à réaliser. Il y a deux taches en plus qui sont l'ajout d'un utilisateur par l'admin et la ligne d'Ariane. L'ajout de l'utilisateur par l'administrateur a été rajouté dans le planning, car c'était une information que je n'avais pas vu dans le cahier des charges.

La deuxième est arrivée en plus, car c'était une tache qui pour moi n'était pas explicite, mais que je devais comprendre.

Ces deux rajouts me prouvent que j'aurais dû passer plus de temps sur la lecture et la compréhension du cahier des charges.

De plus, j'ai eu de grosses différences notables dans la planification. Des tâches que je pensais longues m'ont parfois pris 2 fois moins de temps que prévu. Je pense par exemple à la page du profil ou encore la consultation des logs. Mais la plus grosse différence se trouve pour la manipulation de la table catégorie. Ou j'ai eu beaucoup de mal à analyser que cela me prendrait autant de temps.

II. Difficultés rencontrées

La plus grosse difficulté que j'ai eue est l'affichage de l'arborescence des catégories. C'est ce qui m'a fait faire la plus grosse différence de temps dans le planning. Cela m'aurait pris nettement moins de temps si je pouvais modifier la structure de la base de données. Mais malheureusement le fait que nous travaillons à plusieurs dessus m'a empêché de faire les modifications voulues.

De plus, j'ai eu plusieurs problèmes de compréhension de l'énoncé que j'ai dû souvent mettre au clair avec mon responsable de TPI.

Sinon dans l'ensemble je n'ai pas eu de grosse difficulté me bloquant et me posant de gros problèmes.

III. Amélioration possible

Pour commencer, je n'ai pas eu le temps de faire une pagination dans la page des logs. Elle n'était pas demandée, mais je pense que cela serait une bonne amélioration. De plus, il serait pratique de faire sur cette même page un système de recherche afin de pouvoir rechercher des emails ou des actions serait. Cela ferait un gros point positif.

Il y a aussi la fonctionnalité des sous-menus qui serait améliorable. En effet, quand un menu possède un sous-menu, il ne peut pas rediriger sur une autre page alors que dans mon programme, cela était mon but lors de l'arborescence des catégories.

IV. Remerciements

Pour la réalisation de ce TPI, je souhaite remercier :

- Monsieur COMMINOT, qui était trouvait toujours le temps pour se libérer 10 minutes si j'avais un problème et des questions.
- Adrian SPYCHER, qui a su régulièrement me conseiller et me donner son avis sur la forme et la façon de faire certaines choses.

V. Bilan personnel

Ce TPI fut une expérience très prolifique. En effet, il m'aura permis pour une fois de réaliser un projet concret. Il m'a aussi fait apprendre une architecture que je connaissais de nom, mais que je n'avais jamais pu mettre en place réellement. J'ai d'ailleurs remarqué qu'elle me sera utile pour la suite de mon activité, car je la trouve très pratique et simple d'utilisation une fois comprise.

De plus, j'ai, durant ses 11 jours, commis quelques erreurs de jugement des tâches à cause de ma précipitation au début à vouloir absolument commencer à développer le plus rapidement possible. Cela m'a fait oublier quelques informations du cahier des charges que j'ai dû vérifier plusieurs fois en fin de projet.

Ces erreurs ont été commises une fois et j'espère que cela me permettra de ne plus les commettre et de ne plus me précipiter pour rien.

10. Glossaire

MVC	MVC est l'abréviation de modèle/vues/contrôleur. C'est une façon de développer qui consiste à séparer le code en trois parties. Le modèle stocke et manipule les données de la base, les contrôleurs traitent et envoient les données à la vue et les vues affichent les données.
CRUD	CRUD signifie create/read/update/ delete, en français : créer/lire/modifier/supprimer. Ce sont des actions effectuées sur la base de données.
Logs	Les logs sont l'ensemble des actions effectuées sur une application que l'on souhaite recenser.
Ligne d'Ariane	C'est une aide à la navigation. Cela permet à l'utilisateur de garder une trace de son emplacement dans une arborescence.
Scrum	Méthodologie agile de réalisation de projet.

11. Code source