Année 2021 -2022

SAE203 - Site Web et BDD

Jacques Battaglini
BUT METIERS DU MULTIMEDIA ET DE L'INTERNET

Introduction

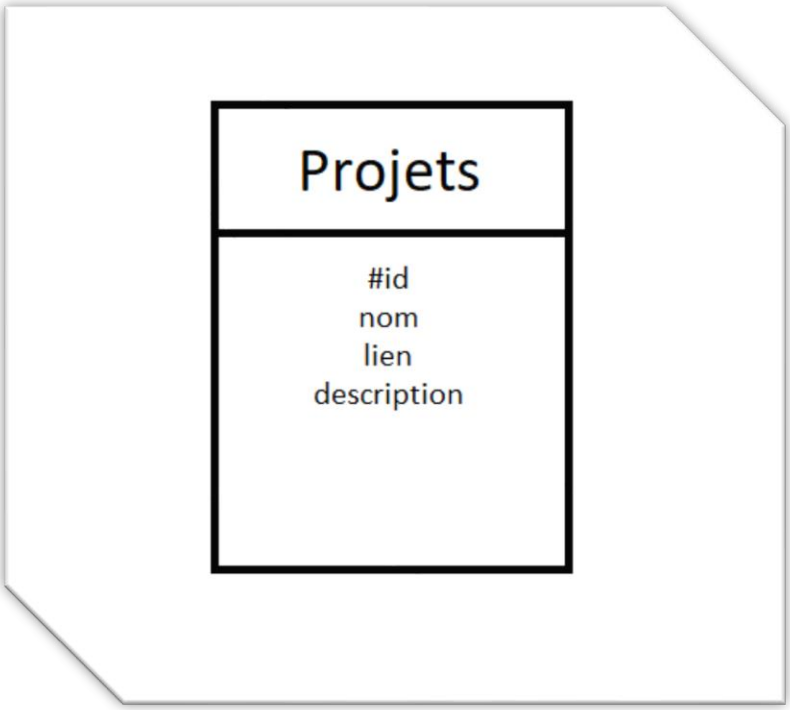
L'exercice demandé dans cette SAE est de reprendre le portfolio que nous avions crée pour une SAE du semestre précédent, et de l'améliorer en lui ajoutant une interface d'administration permettant de modifier et d'agrémenter le portfolio sans devoir passer par le code en lui-même. Pour cela, il faudra que cette interface soit reliée a une base de données qui regroupera toutes les informations nécessaires au portfolio (images, textes, vidéos, etc…). Un autre objectif de cette SAE est de mettre en place un patron de conception Modèle-Vue-Contrôleur (MVC) pour le portfolio, qui consiste a regrouper les fichiers du portfolio en 3 parties : le modèle, qui permet la connexion a la base de données, la vue qui regroupe toute la partie graphique du portfolio et le contrôleur qui s'occupe de faire les requêtes entre la BDD et le site.

Nous allons voir en détail les différentes étapes qui ont servi à l'amélioration du portfolio dans ce document.

I. Modélisation de la BDD

Avant de rentrer dans le vif du sujet, il faut tout d'abord créer un modèle conceptuel de données pour structurer sa BDD.

Dans mon cas, j'ai beaucoup réfléchi à comment faire pour intégrer efficacement une base de données à mon portfolio, et après plusieurs essais j'ai adopté une solution simple qui minimise le nombre de tables nécessaires et ne requiert aucune association. Ma base de données finale ne contient que trois tables, une pour la connexion au backoffice et une pour la messagerie mais nous reviendrons sur ces deux tables plus tard, et une dernière pour stocker tout le contenu nécessaire dans mon portfolio. Cette table s'appelle « projets », et stocke toutes sortes d'informations (texte, images et vidéo) dans avec quatre critères : un id et un nom pour identifier l'élément et l'appeler à l'aide de fonctions, un lien qui représente le chemin nécessaire au serveur pour trouver l'élément à afficher et une description qui contient soit un descriptif de l'élément, soit un bloc de texte a part entière.



Modèle conceptuel de données

Mon schéma entité-association est de ce fait très simpliste et se résume par une entité unique qui contient tout le contenu de mon portfolio.

II. Implémentation de la base de données

Pour créer ma base de données, j'ai utilisé PhpMyAdmin qui est inclus dans Wamp. Une fois mes trois tables créées (admin, formulaire et projets), je les ai remplies avec tous les éléments que contenait mon portfolio, en prévision du passage de ce dernier en architecture MVC.

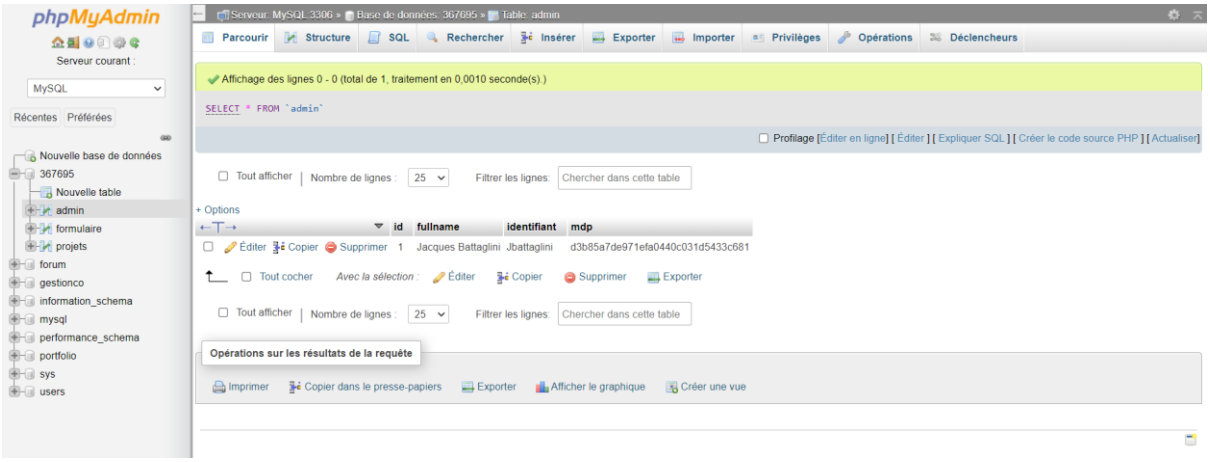


Table admin

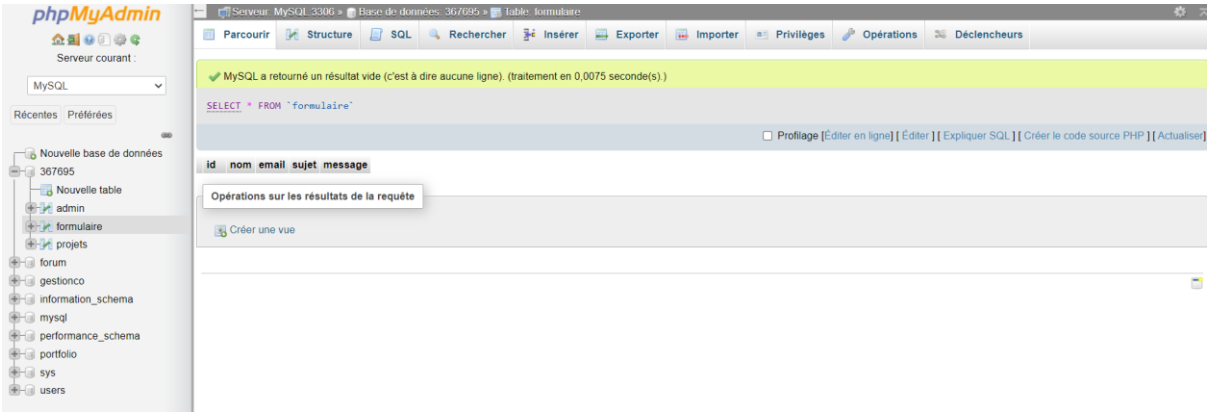


Table formulaire

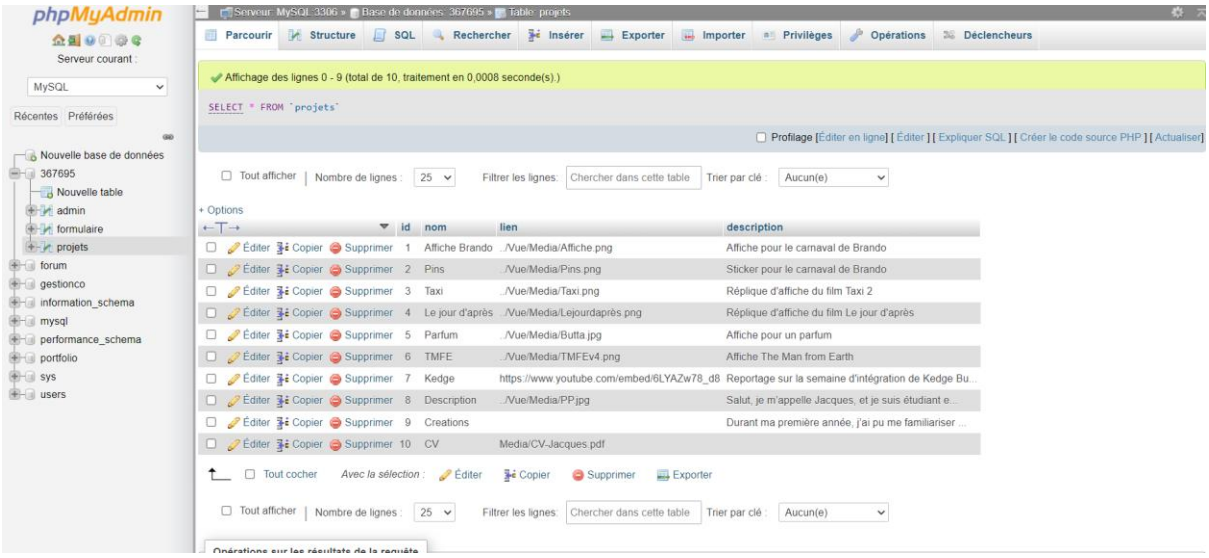
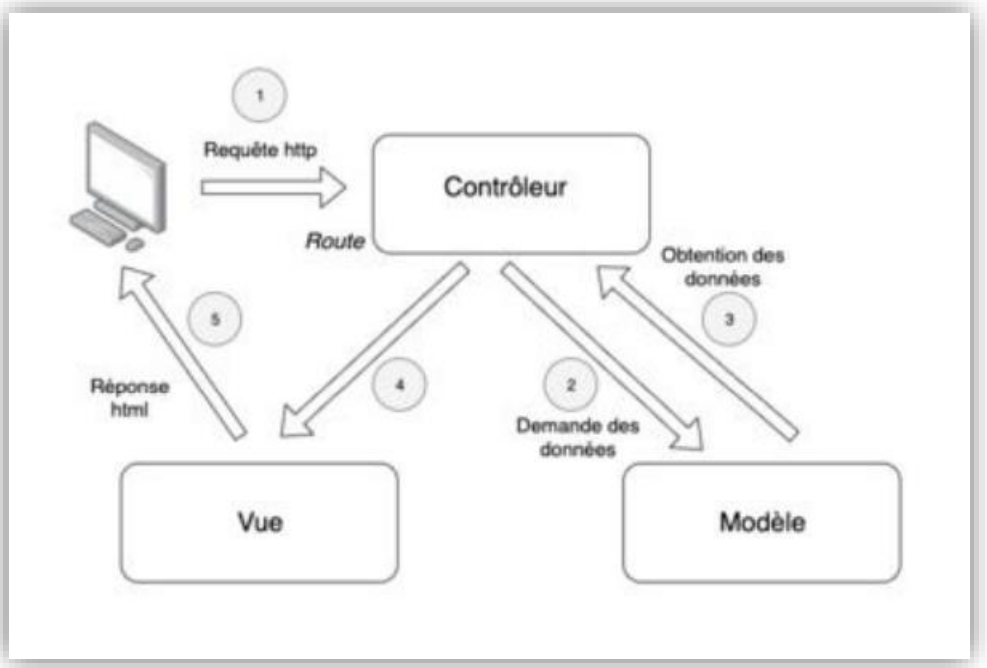


Table projets

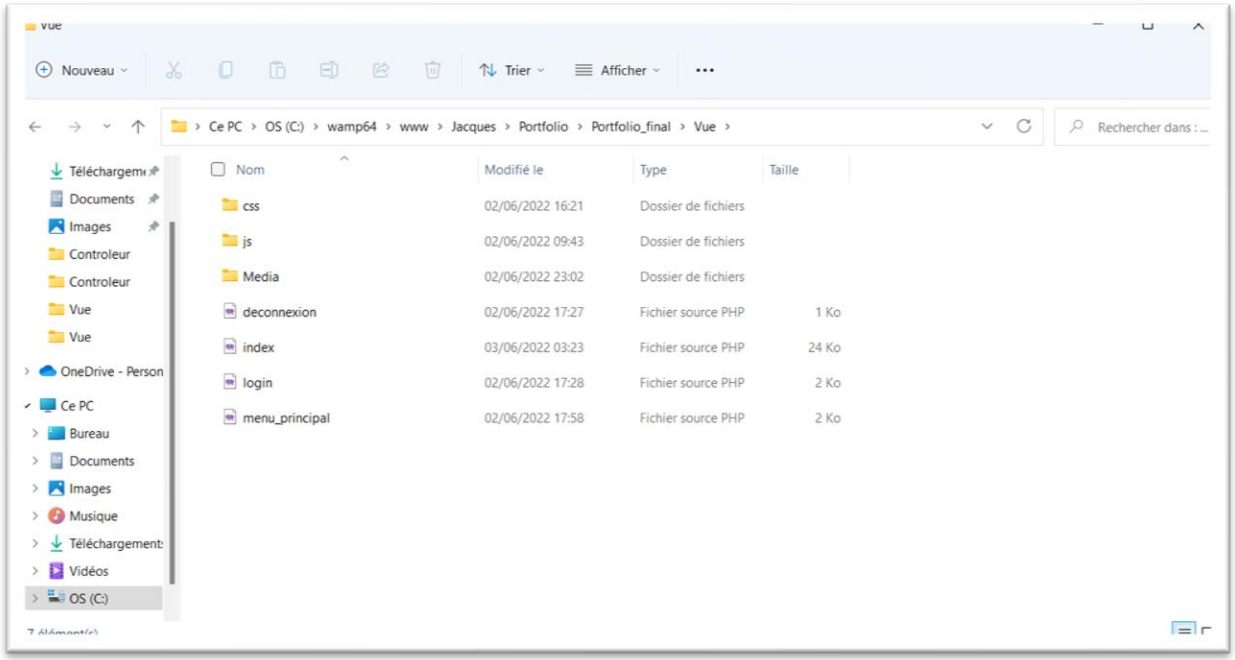
III. Architecture MVC

Maintenant que la base de données est prête, il est temps de modifier l'architecture du portfolio pour lui faire adopter le modèle de conception MVC.

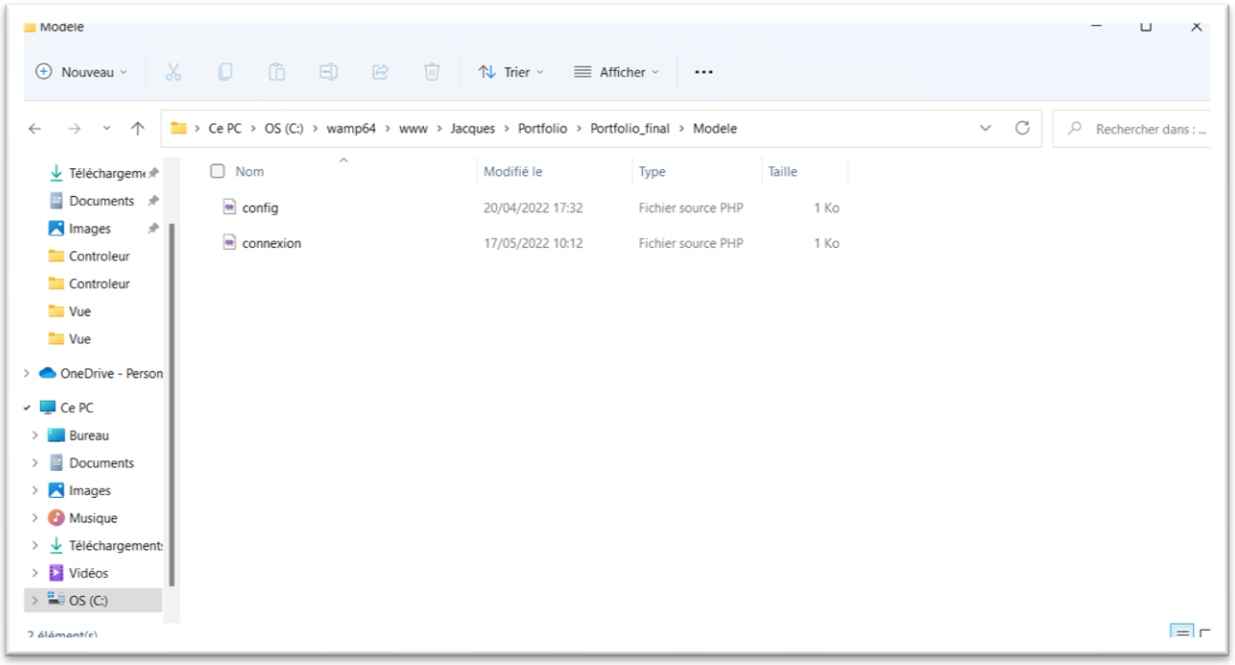


Fonctionnement de l'architecture MVC

Comme le montre ce schéma, le principe du MVC se base sur la répartition des fichiers en 3 parties, chaque partie correspondant à un rôle :

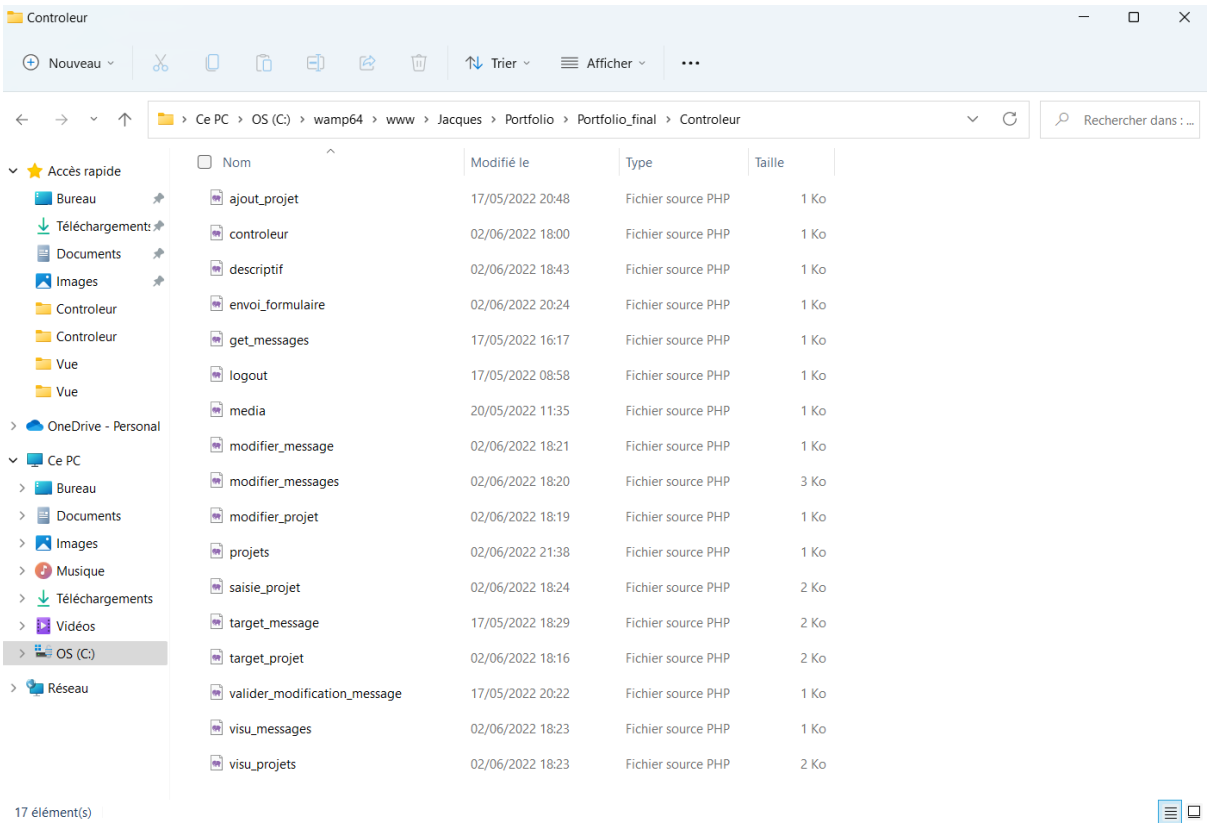


La vue, qui recense tous les fichiers qui constituent l'aspect graphique du portfolio. On y retrouve entre autres l'index et tous les fichiers css et javascript ainsi qu'un dossier contenant les images et le CV que la base de données ira chercher pour les afficher sur le portfolio.



Le modèle, qui contient un fichier connexion.php et connect.php. Ces derniers servent à se connecter a la BDD et sont donc indispensables au bon fonctionnement du portfolio. Le fichier connexion.php contient la requête de connexion, tandis que connect.php contient les identifiants de la base qui seront utilisés dans la requête. Ce système permet de sécuriser la base, car même si quelqu'un parviendrait a accéder au contenu du fichier connexion.php, il n'aura pas les identifiants de la BDD qui sont contenus dans connect.php,

et cela permet aussi de modifier facilement les coordonnées de la base si jamais on veut changer de BDD.



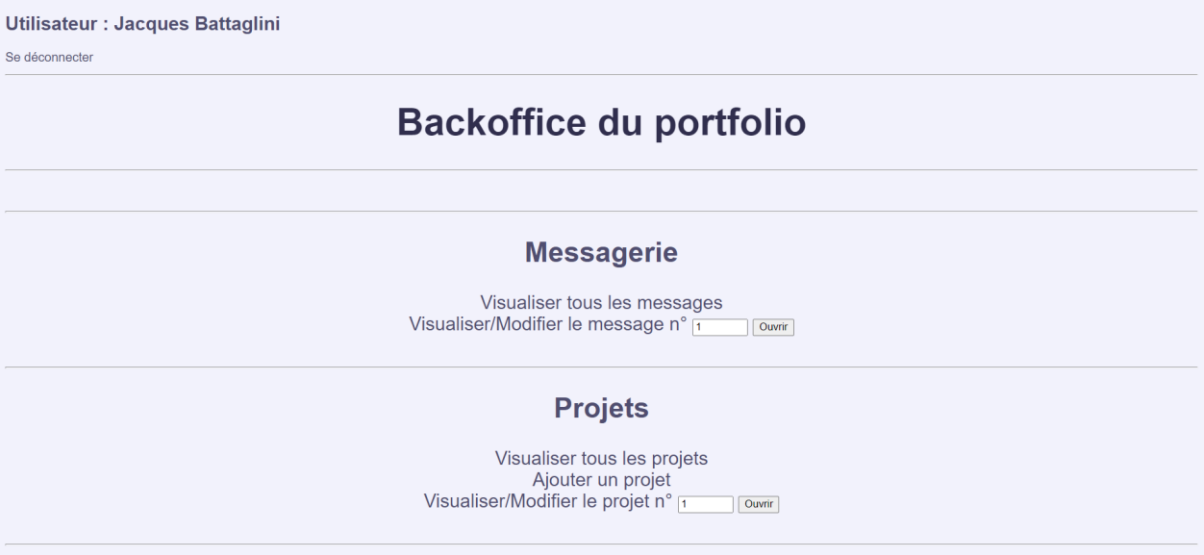
Enfin, le contrôleur contient tous les fichiers effectuant des requêtes à la base de données afin d'afficher son contenu dans le portfolio. On y retrouve le fichier contrôleur.php, qui contient une liste d'include vers des fichiers remplis de fonctions (comme projets.php ou descriptif.php) afin de servir de passerelle entre les fichiers de la vue et les fichiers contenant les requêtes SQL du contrôleur.

```php
<?php
include_once('../modele/connexion.php');
include_once('descriptif.php');
include_once('projets.php');
include_once('get_messages.php');
include_once('target_message.php');
include_once('target_projet.php');
?>
```

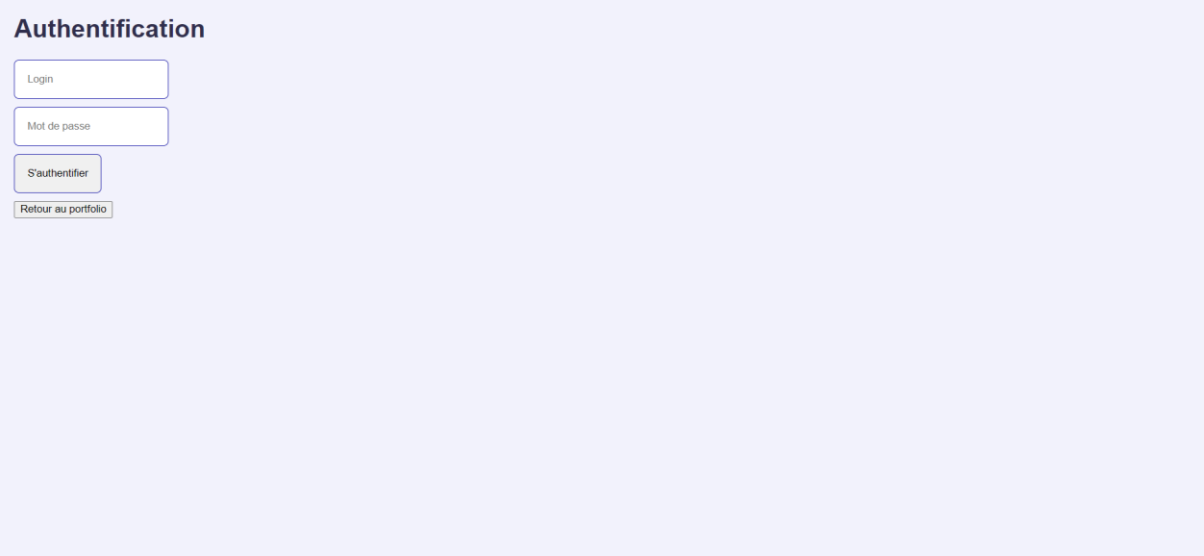
Le fichier controleur.php, contenant une liste d'include pour faire le lien entre les parties vue et controleur.

IV. Backoffice

Ensuite, j'ai commencé à créer la partie backoffice de mon portfolio, pour l'instant déconnectée de cette dernière, dont le but est de pouvoir visualiser, modifier et ajouter des projets a ma base de données et aussi visualiser les messages envoyés depuis le formulaire du portfolio, le tout sans avoir à toucher directement à la base de données. Enfin, j'ai créé une page login pour faire le lien être le portfolio et le backoffice et créer une connexion sécurisée afin que seul l'administrateur du portfolio puisse avoir accès au backoffice.



Menu principal du backoffice



Page de connexion au backoffice

V. Ajout d'éléments multimédia depuis la BDD

Pour pouvoir afficher des éléments multimédias sur mon portfolio, j'ai créé deux fichiers contenant des fonctions qui effectuent des requêtes SQL, puis affiche le contenu récupéré par la requête grâce à un echo. Le premier est descriptif.php qui contient les fonctions des parties Accueil et Présentation de mon site, et le deuxième est projets.php qui continent celles nécessaires pour la partie portfolio où sont exposés mes projets.

```
<div class="container">
    <div class="row">
        <div class="home-info padd-15">
            <h3 class="hello">Salut, je m'appelle <span class="name">Jacques Battaglini</span></h3>
            <h3 class="my-profession">Je suis <span class="typing"></span></h3>
            <p><?php
            require ('../Controleur/controleur.php');
            get_description('Description');
            ?></p>
            <a href="<?php get_cv('CV'); ?>" class="btn">Télécharger mon CV</a>
        </div>
        <div class="home-img padd-15">
            <?php get_image('Description'); ?>
        </div>
    </div>
</div>
</section>
<!-- Home Section End -->
<!-- About Section Start -->
<section class="about section" id="about">
    <div class="container">
        <div class="row">
            <div class="section-title padd-15">
                <h2>A propos de moi</h2>
            </div>
        </div>
        <div class="row">
            <div class="about-content padd-15">
                <div class="row">
                    <div class="about-text padd-15">
                        <h3>Je m'appelle Jacques Battaglini et je suis <span>un étudiant</span></h3>
                        <p><?php get_description('Creations'); ?></p>
                    </div>
                </div>
                <div class="row">
```

Sur cette image, on peut voir 4 fonctions soulignées en rouge. Ces quatre fonctions sont stockées dans le fichier descriptif.php, et permettent d'aller chercher un élément d'une ligne de la table projets en particulier. get_description() permet de récupérer le contenu de la colonne description de la table, a la ligne où figure le nom passé en argument de la fonction. get_CV() est une fonction permettant d'afficher un fichier au format PDF et get_image() permet d'afficher toutes sortes d'images comme des .png, des .jpg ou encore des .gif.

```php
<?php

$pdo = require ('../modele/connexion.php');

function get_image($nom){
    global $pdo;
    $sql = "SELECT lien FROM projets WHERE nom='$nom'";
            $req = $pdo->query($sql);
            $data = $req->fetch(PDO::FETCH_ASSOC);
            echo "<img src=".$data['lien']." />";
}

function get_description($nom){
    global $pdo;
    $sql = "SELECT description FROM projets WHERE nom='$nom'";
            $req = $pdo->query($sql);
            $data = $req->fetch(PDO::FETCH_ASSOC);
            echo $data['description'];
}
function get_cv($nom){
    global $pdo;
    $sql = "SELECT lien FROM projets WHERE nom='$nom'";
            $req = $pdo->query($sql);
            $data = $req->fetch(PDO::FETCH_ASSOC);
            echo $data['lien'];
}
?>
```

Le fichier descriptif.php, contenant les fonctions dont il était question dans le paragraphe précédent.

```php
<?php

$pdo = require ('../modele/connexion.php');

function get_projet($id)
{
    global $pdo;
    $sql = "SELECT * FROM projets WHERE id='$id'";
    $req = $pdo->query($sql);
    $data = $req->fetch(PDO::FETCH_ASSOC);
    echo "<img src=".$data['lien']." alt=".$data['nom']."/>";
}
function get_projets()
{
    global $pdo;
    $sql = "SELECT * FROM projets";
    $req = $pdo->query($sql);
    $data = $req->fetchAll(PDO::FETCH_ASSOC);
    return $data;
}
function get_titre($id){
    global $pdo;
    $sql = "SELECT description FROM projets WHERE id='$id'";
    $req = $pdo->query($sql);
    $data = $req->fetch(PDO::FETCH_ASSOC);
    echo"<p style='color:white;'>".$data['description']."</p>";
}
function get_media($nom){
    global $pdo;
    $sql = "SELECT lien FROM projets WHERE nom='$nom'";
    $req = $pdo->query($sql);
    $data = $req->fetch(PDO::FETCH_ASSOC);
    echo "<iframe width='640' height='360' src=".$data['lien']." title='YouTube video player'
    allow='accelerometer; autoplay; clipboard-write; encrypted-media; gyroscope; picture-in-picture'
    allowfullscreen></iframe>";
}
```

Le fichier projets.php

Conclusion

Au cours du développement de ce projet, j'ai appris à utiliser avec aisance les requêtes SQL ainsi que le langage PHP qui m'ont permis de mettre en place cette architecture MVC. Cette dernière est quant a elle très utile d'un point de vue sécurité mais aussi organisation, et offre aussi une certaine flexibilité pour modifier son site, ce qui peut faire défaut à une architecture web classique. J'ai aussi pu revoir la méthodologie concernant la création de base de données, notamment via le modèle conceptuel de données et le schéma entité-association.

Je tiens a remercier nos enseignants qui ont été là pour répondre à nos interrogations et ainsi nous aider autant que possible a réaliser ce projet.