

# **Dossier Projet Professionnel**

Titre RNCP Développeur Web & Web Mobile

**Toréa Patissier**

---

Réalisation d'un site vitrine et e-shop pour Marika Coiffure, salon de coiffure et barbier basé en Corse à Calvi.

# Sommaire:

## 1. Introduction

- 1.a Compétences visées par le projet 4
- 1.b Résumé du dossier de projet 4

## 2. Présentations

- 2.a Présentation du client 5
- 2.b Présentation de l'école 6
- 2.c Présentation de l'équipe 7

## 3. Préparations du projet

- 3.a Quels sont les besoins 8
- 3.b Qu'est-ce qui a été mis en place 9
- 3.c Analyse concurrentielle 9

## 4. Réalisation du cahier des charges

- 4.a Hébergement / nom de domaine 10
- 4.b Arborescence du site 10
- 4.c Back-office 11
- 4.d Maintenance et évolution du site 11

## 5. Spécifications technique du projet

- 5.a Environnement de travail 12
- 5.b Choix des technologies 12

<b>6. Réalisations et code</b>	
- 6.a Header et Footer	13
- 6.b Formulaire d'inscription	14
- 6.c Formulaire de connexion	15
- 6.d Page profil	17
- 6.e Back-office	18
- 6.f Mise en place du responsive	34
<b>7. Veille sur la sécurité</b>	
- 7.a Diagnostic	36
<b>8. Recherche sur site anglophone</b>	
- 8.a Situation ayant demandé une recherche sur un site anglophone	39
<b>9. Structure du site</b>	
- 9.a Classes et méthodes	40
<b>10. Jeux d'essai</b>	
10.a Elaboration d'un jeux d'essai	42

# 1. Introduction

## 1.a - Compétences visées par le projet

Le projet recouvre les compétences suivantes :

Activité type 1 “Développer la partie front-end d’une application web ou web mobile en intégrant les recommandations de sécurité”:

- **Maquetter** un application
- Réaliser un **interface** utilisateur web **statique** et adaptable
- Développer une **interface** utilisateur web **dynamique**

Activité type 2 “Développer la partie back-end d’une application web ou web mobile en intégrant les recommandations de sécurité”:

- Développer les composants **d'accès** aux **données**
- Développer la **partie back-end** d’une application web ou web mobile

## 1.b - Résumé du dossier de projet

Ce dossier traite le projet qui sera présenté afin de valider le titre RNCP, il sera détaillé les compétences qui sont visées. Une présentation de l'équipe de développement, de l'école qui m'a permis d'acquérir les acquis nécessaires à la réalisation de ce projet, et du client que nous avons démarché afin de répondre à ses besoins suite à une analyse de la concurrence dans le même secteur.

Nous aborderons ensuite la réalisation du cahier des charges qui traite de l'hébergement du site, de sa structure, de la charte graphique imposé par le client, et du back office lui permettant de gérer la vente de ses produits mais aussi de sa galerie photos qui présentera les coupes de cheveux, homme, femme et enfant.

La maintenance du site sera faite si un problème de la part du client remonte. Il sera présenté les moyens mis en place pour atteindre notre

objectif tel que l'environnement de travail et le choix des technologies. J'aborderais de manière concise les différentes pages du site en mettant en avant certaines parties du code.

Enfin il sera présenté des captures d'écran explicites pour soutenir le traitement de la sécurité par exemple, mais aussi une situation où la recherche sur un site anglophone a été nécessaire étant donné que la majorité de la documentation en terme de développement web est anglaise.

## 2. Présentations

### 2.a - Présentation du client

Fondé en l'an 2000 le salon **MARIKA Coiffure** est l'aboutissement d'un parcours professionnel qui a conduit à ancrer l'espace de travail au centre de la ville de Calvi, Cité Semper Fidelis, sous les Remparts de la Citadelle.

Le salon a fidélisé une clientèle calvaise exigeante et toute empreinte de nouveautés.

Aux fondamentaux de la coiffure mixte se sont greffés, au fil du temps, des techniques que savent apprécier les connaisseurs au travers de l'espace barbier et les élégantes auprès du pôle extensions capillaires de très grandes marques comme **GREAT LENGTHS**.

Le salon, ces dernières années s'est aussi résolument tourné vers les techniques naturelles, et notamment de la **coloration végétale**.

Le salon MARIKA à Calvi vous propose un espace dynamique et chaleureux où il fait bon passer du temps auprès de son équipe.

Vous qui passez par là, poussez la porte d'un monde de douceur et de bien être.

## 2.b - Présentation de l'école

La Plateforme est une école du numérique et des nouvelles technologies co-fondée avec le Club Top 20 réunissant les grandes entreprises de la Métropole Aix Marseille. La Plateforme forme les étudiants dans différents domaines du numérique, la **coding school** forme des développeurs, la **cyber security school** forme à la sécurité dans le monde du numérique, **IA school** forme les étudiants dans l'intelligence artificiel.

Étant étudiant à la coding school, je vais développer sur cette partie de l'école.

La formation au sein de la **Coding School** se déroule en 2 ans.

La première année est consacrée à l'apprentissage de différents langages (**HTML, CSS, Javascript, PHP**) et aux **technologies du web**.

Les **acquis** de ces langages sont validés en réalisant des **projets** de plus en plus techniques concernant le langage visé. Un projet débloque des points, et ces points sont nécessaires pour pouvoir travailler sur le prochain projet, jusqu'à arriver au **dernier projet** du langage étudié (le dernier projet pour boucler le HTML par exemple était un blog sur la ville de Marseille, si ce projet n'est pas terminé alors l'étudiant ne pourra pas passer au langage suivant, c'est à dire PHP).

Les étudiants s'informent et apprennent de manière **autonome**, cependant les membres de la pédagogie sont disponibles pour déboguer les étudiants à tout moment.

Ils ont la possibilité de demander un **how to**, c'est à dire un speech ou un membre pédagogique va expliquer de manière concise le problème rencontré par la majorité (par exemple un how to sur flexbox a été demandé, puis mis en place en septembre pour les étudiants qui avaient du mal à l'utiliser correctement).

Enfin il y a aussi des **cours d'expression orale et écrite** ainsi que des **cours d'anglais**. Ces cours d'anglais traitent sur un sujet (écologie, sport, technologies), à l'issue de 3 ou 4 semaines les étudiants doivent présenter le projet à l'oral avec un powerpoint en support. Cette présentation dure en général 10 à 15 minutes.

## 2.c - Présentation de l'équipe

### Nuno Barbosa :

J'ai 25 ans et je suis originaire de Calvi en Corse. J'entame depuis Septembre 2020 une reconversion dans le monde du web au sein de l'école du numérique La Plateforme à Marseille où j'ai intégré la Coding School.

La première nous a permis de nous familiariser avec le web en apprenant des langages de programmation par ordre chronologique (HTML/CSS -> PHP -> JAVASCRIPT).

La deuxième année sera consacrée au développement d'applications, nous allons aborder des langages comme ReactJs, Python ainsi que la création d'API.

L'année se déroule en alternance école/entreprise.

J'ai auparavant travaillé en tant que cuisinier pendant une période de 4 ans puis je me suis reconverti dans le BTP , où j'ai exercé pendant 2 ans le métier d'aide maçon avant de passer au poste de conducteur d'engins que j'ai occupé pendant 1 année.

### Toréa Pâtissier:

J'ai **25 ans** et je suis originaire de **Calvi en Corse**. J'entame depuis Septembre 2020 une **reconversion** au sein de **l'école du numérique La Plateforme** à Marseille où j'ai intégré la **Coding School**.

La **première année** m'a permis de me familiariser avec le web en apprenant des langages de programmation tels que **HTML,CSS,PHP et JAVASCRIPT**.

La **deuxième** année sera consacrée au développement d'applications, nous allons aborder **ReactJs,Symfony, Python** ainsi que la création **d'API**. L'année se déroule en **alternance** école/entreprise à partir du mois de septembre pour la durée d'un an.

Le **but** de cette alternance est avant tout de **découvrir** le monde de **l'entreprise**, mais aussi de mettre en œuvre les compétences acquises durant notre année à l'école tout en acquérant l'expérience dans le poste occupé.

## 3. Préparations du projet

### 3.a -Quels sont les besoins

Il y aura un travail en amont sur le **SEO** pour s'assurer que le site se soit le plus proche possible du premier résultat lors des recherches Google et autres moteurs de recherche.

Les clients pourront savoir quels services sont proposés avant même de se rendre sur place, ainsi que les différents produits vendus.

La **Galerie** photo ou des photos des différentes coupes, couleurs et autres services effectués par le staff seront visibles par tout visiteur du site. Le client pourra donc voir comment travaille l'équipe et la qualité de leur savoir-faire.

Nous allons aussi mettre en place un système de **Click & Collect**, relié à la boutique du site qui contiendra tous les différents produits vendus au détail dans le salon.

Le client pourra ainsi payer son produit en ligne et le récupérer sur place, grâce à une facture reçue par e-mail.

Un espace avis sera disponible dans l'accueil du site, les clients pourront donc décrire leur expérience dans le salon ce qui influencera et rassurera les nouveaux clients qui visitent le site.

Les éléments mis en place ont pour but de répondre à deux besoins, **la e-réputation et l'attraction des clients.**

### 3.b - Qu'est-ce qui a été mis en place



Comme la majorité de commerces dans l'air du temps, Marika Coiffure est présente sur les réseaux sociaux.

Pour gagner en visibilité, un compte **facebook** et **instagram** ont été créés ou sont présentés les différentes coupes de cheveux (homme femme et enfant) du salon de coiffure.

Le but étant d'avoir une certaine **e-réputation**, donc un atout **concurrentiel** de taille, mais surtout d'attirer la clientèle qui aura déjà un avis sur la qualité des services proposés. En dehors du numérique, bien entendu il y a le **bouche à oreille** étant donné que le salon existe depuis 20 ans maintenant.

### 3.c - Analyse concurrentielle

Marika Coiffure est un des **sept salons** de coiffure présents à Calvi, il faut donc prendre des **initiatives** pour se **démarquer** et c'est aussi pour cela que la **création** de ce **site** a lieu.

**Aucun** de ses **concurrents** ne dispose d'un **site internet** en ligne, ce sera donc une grande opportunité pour fidéliser les nouveaux clients qui cherchent un salon de coiffure sur internet.

Les différents **concurrents** sont :

- Mv Coiffure
- Altea
- Nuance
- Vogue S
- Le Carré
- Fanny Coiffure

La **gamme** de prix de **tous** les salons est **moyenne**, à l'exception du salon Le Carré qui propose ses services pour des prix plus élevés.

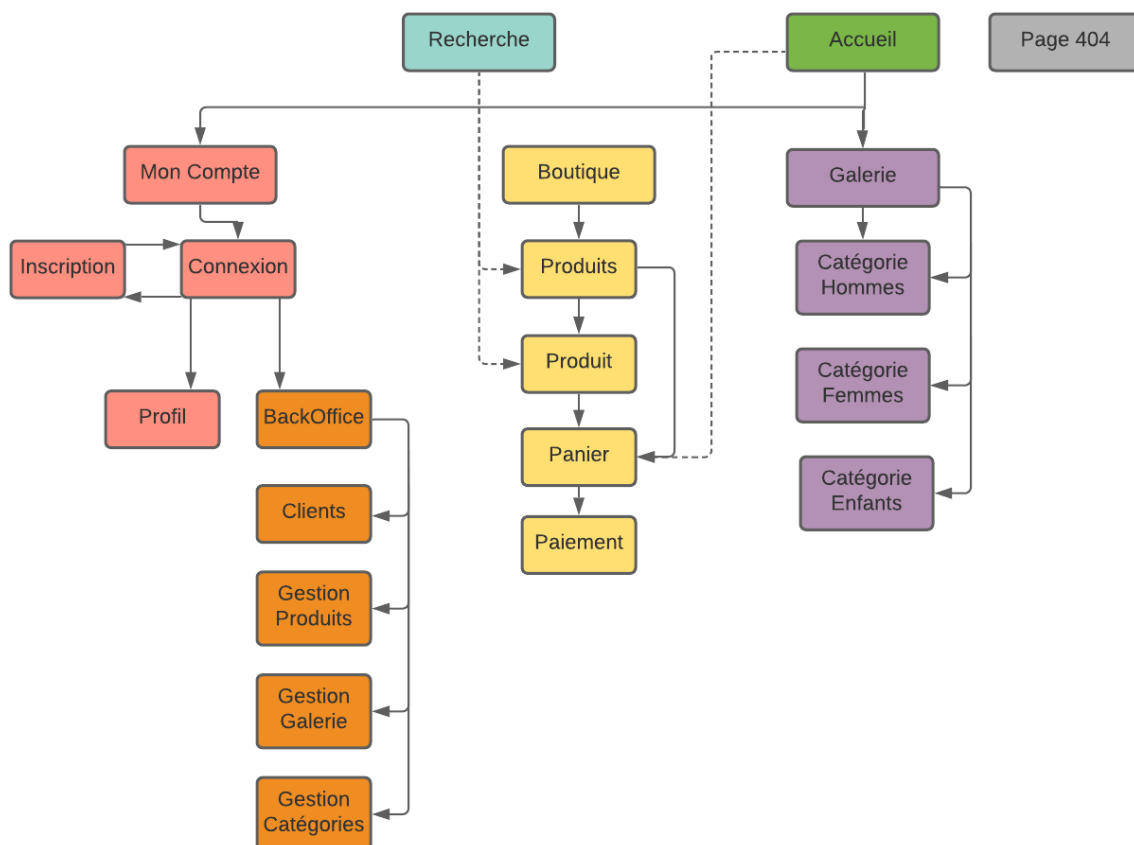
## 4. Réalisation du cahier des charges

## 4.a - Hébergement / nom de domaine

Pour l'**hébergement** du site nous allons le mettre en ligne chez **OVH Cloud**, anciennement appelé OVH, une entreprise française qui pratique initialement de l'hébergement de serveur, et est un fournisseur d'accès à Internet, puis opérateur de télécommunications pour les entreprises.

Effectivement, le **client** dispose **déjà** d'un **nom de domaine** chez l'entreprise en question qui était pour son ancien site internet actuellement hors service ([www.marika.fr](http://www.marika.fr)).

## 4.b - Arborescence du site



## 4.c - Back-office

Le back office sera une **interface** du site accessible **uniquement** aux utilisateurs ayant pour **droit** “**Administrateur**”.

Cette interface permettra à l'administrateur de **gérer** tout le **contenu** de son site.

Le but étant de créer une interface **intuitive** et **simple** d'utilisation utilisable même par quelqu'un qui n'est pas très familiarisé avec le Web.

Un design **épuré** sera utilisé, avec des couleurs simples mais bien contrastées pour que tous les **éléments** soient **visibles**.

Cette interface sera **contrôlée** par des **conditions** qui s'assureront au chargement de la page, que **l'utilisateur** en question bénéficie des **droits nécessaires** pour y accéder.

Elle sera accessible depuis la barre de navigation du site, une fois l'administrateur connecté.

## 4.d - Maintenance et évolution du site

Une fois le site mis en ligne et fonctionnel, la partie du back-office pour la gestion du contenu, que ce soit les produits ou la galerie, sera expliqué étape par étape.

Si le client rencontre un ou plusieurs **problèmes**, ils seront communiqué et l'objectif sera donc de les **résoudre** afin de garder l'expérience utilisateur optimale.

Concernant **l'évolution du site**, si avec le temps le client souhaite modifier une ou plusieurs pages, en changeant le **design**, la **mise en page**, donc la disposition de certains **éléments**, ou encore la **charte graphique** il le communiquera et les **modifications** se feront après un entretien afin de connaître et répondre aux nouveaux besoins du site.

## 5. Spécifications technique du projet

### 5.a - Environnement de travail

**Pour la partie code**, j'ai utilisé l'IDE Visual Studio Code

- Github et Github Desktop pour le versionning de fichiers
- Un gestionnaire de base de données (PhpMyAdmin)
- Un serveur web local (MAMP)

**Pour la partie collaboration** j'ai eu recours à Trello pour la planification des tâches à faire, en cours et terminé. Cet outil est utile pour avoir un aperçu sur l'avancement du projet. J'ai utilisé également Discord durant la période de télétravail ce qui m'a permis de partager mon écran et de d'être ou de débbugger sur certaines parties du code avec la personne avec qui j'ai réalisé le site internet.

**Pour le maquettage** j'ai utilisé Figma, c'est un outil puissant qui permet aux personnes ayant accès au projet de pouvoir ajouter/ modifier et supprimer du contenu de manière instantané.

**Pour la réalisation de la base de données** j'ai utilisé LucidApp en mettant en place un MCD (Modèle Conceptuel de Données)

### 5.b - Choix des technologies

Pour la réalisation de ce projet, les technologies utilisées sont:

**Front-end: HTML** pour la mise en page, **CSS** pour y ajouter du style.

Le framework **Materialize** permet d'accéder à un panel de fonctionnalités, ainsi qu'une gamme de couleurs visant à rendre le site ergonomique pour l'utilisateur.

Materialize a permis également au site d'être responsive grâce à son système de grilles de 12 colonnes, même si d'autres moyens tels que les medias queries étaient possible.

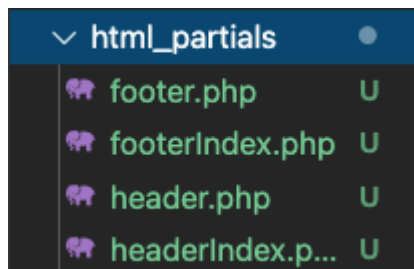
La bibliothèque **Jquery** de **Javascript** pour ajouter du dynamisme au site.

**Back-end: PHP** pour créer les différentes classes et méthodes nécessaires au fonctionnement du site, ainsi que **SQL** pour les requêtes avec la base de données, ce qui est primordial pour que l'utilisateur puisse se connecter et acheter un ou plusieurs produits.

Enfin **AJAX** qui m'a permis de faire des requêtes à la base de données de façon asynchrone c'est-à-dire de récupérer ou envoyer des éléments d'une page sans avoir à la recharger.

## 6. Réalisations et codes

### 6.a - Header et Footer



J'ai stocké le Header et footer dans un dossier afin de ne pas avoir à rappeler 30 lignes de codes à chaque page mais uniquement la page contenant le header et du footer.

Les fichiers header contiennent les balises HTML jusqu'à `</header>`.

### Code:

On peut voir que si un utilisateur est connecté, l'onglet Profil sera affiché, sinon Connexion, de plus si il dispose du droit nécessaire, il y aura l'onglet Admin en plus.

```
32 <a href="#" class="brand-logo"></a>
33 <ul class="right hide-on-med-and-down">
34   <li><a href="index.php">Accueil</a></li>
35   <li><a href="#">Boutique</a></li>
36   <?php
37     if(isset($_SESSION['user'])) {
38       ?<li><a href="users/profil.php">Profil</a></li> <?php
39     } else {
40       ?<li><a href="users/connexion.php">Connexion</a></li> <?php
41     }
42     if(($_SESSION['user']['id_droits']) == 20260) {
43       ?<li><a href="backoffice/backoffice.php">Admin</a></li> <?php
44     }
45   ?>
```

## Navbar dynamique:

Donc, selon le statut de l'utilisateur (connecté, déconnecté, admin, visiteur) la navbar affichera des onglets différents.



## Footer:

Le footer quant à lui contient les balises `<footer></body></html>` et `<script>` pour les fichiers Javascript.

## 6.b - Formulaire d'inscription

```
12  if (isset($_POST["inscription"])) {  
13      $pageInscription->register();  
14  }
```

Au moment où l'utilisateur envoie le formulaire, la **fonction register** est lancée pour insérer les éléments entrés en base de données si toutes les conditions sont respectées.

*usersClass.php*

```
$confpassword = trim(htmlspecialchars($_POST['iConfPassword']));  
$chiffre = 1;  
$testpwd = preg_match("#[A-Z]#", $password) + preg_match("#[a-z]#", $password) + preg_match("#[0-9]#", $password) + preg_match("#[^a-zA-Z0-9]#", $password);  
// Hashage mdp  
$options = ['cost' => 12,];  
$hash = password_hash($password, PASSWORD_BCRYPT, $options);
```

Chaque `$_POST` du formulaire (il y en a 6) sont disposés comme **\$confpassword** (confirmation de mot de passe).

La fonction **htmlspecialchars** permet de se protéger contre la **faille XSS** en échappant les caractères en entité HTML. Quant à **trim** il supprime les espaces en début et fin de caractères

**\$chiffre** est **\$id\_droits** par défaut lors d'une inscription de la part de l'utilisateur

**\$testpwd** permet d'intégrer une expression régulière, dans ce cas l'utilisateur devra obligatoirement avoir un mot de passe qui contient 7 caractères, une majuscule, une minuscule, un chiffre et un caractère spécial.

**\$hash** contient **PASSWORD\_BCRYPT** l'algorithme qui va hacher le mot de passe

En amont **\$e-mail** entré par l'utilisateur sera **vérifié** en base de données pour s'assurer qu'aucun compte n'a été créé avec cet e-mail.

#### **Ce qui est vérifié avant de valider l'inscription:**

- Que l'adresse mail n'est pas déjà utilisée
- Que le numéro de téléphone est valide
- Que le mot de passe respecte les conditions requises
- Que le mot de passe et la confirmation de mot de passe est identique
- Un des 6 input ou seront insérés les données clients

*inscription.php*

```
<div class="row">
  <div class="col col s6 m6 l6">
    <label>Nom</label><br /><br />
    <input type="text" name="iNom" id="iNom" value="<?php if($_POST)echo $_POST['iNom'];?>"
    autofocus required><br /><br />
```

## 6.c -Formulaire de connexion

```
if(isset($_POST['connexion']) && !empty($_POST['connexion'])){

    $email = trim(htmlspecialchars($_POST['email']));
    $password = trim(htmlspecialchars($_POST['password']));

    if( $email != '' && $password != ''){

        $con = $this->connectDb();
        $req = $con->prepare("SELECT * FROM utilisateurs ");
        $req->execute();
        $result = $req->fetchAll();
```

J'ai créé la fonction **connexion()** qui va me permettre de **connecter** mon **utilisateur**.

Dans un premier temps j'ai récupéré **l'e-mail** et le **mot de passe**.

**Si** ces 2 variables ne sont **pas vides** alors je peux lancer ma **requête**. **Sinon**, un message **d'erreur** s'affiche.

```
for ($i = 0; isset($result[$i]); $i++) { // Boucle for pour parcourir le tableau
    $logcheck = $result[$i]['email']; // On recupère le login dans le tableau parcouru
    $passcheck = $result[$i]['password']; // Et ici le MDP
    if ($email == $logcheck and password_verify($password, $passcheck) == TRUE) { // Si Login et MDP ==
        le tab alors co + Verify pass
        $_SESSION['user'] = $result[$i];
        header('location: http://localhost:8888/projet\_pro/users/profil.php');
    }
}

if ($email == $logcheck and password_verify($password, $passcheck) == FALSE) { // Si Login et MDP =
    le tab alors co + Verify pass
    echo '<p class="container red-text"><b>Identifiant ou mot de passe incorrect.</b></p><br />';
    return FALSE;
}
```

Avec la **boucle for**, je parcours ma base de données pour comparer les variables e-mail et mot de passe.

Si la **comparaison est bonne**, alors j'ouvre une **session** et redirige vers la page profil.

**Si** il n'y en a **pas**, alors j'affiche un **message d'erreur**.



## 6.d - Page profil

# Bonjour Toréa !

Pour modifier vos informations, veuillez remplir les champs ci-dessous

Nom :	Email :
Patissier	p.torea@icloud.com
Prénom :	Modifiez votre mot de passe :
Toréa	Mot de passe
Tel :	Confirmation de mot de passe :
681472329	Confirmez ici

MODIFIER

DÉCONNEXION

Une fois **connecté**, l'utilisateur arrive sur sa page **profil**. Il peut y trouver différentes **informations** présentes en placeholder. Il peut **modifier** ses informations personnelles.

```
if (isset($tel) && !empty($tel)) {  
    if(is_numeric($tel) && strlen($tel) == 10){  
        $sql = $con->prepare("UPDATE utilisateurs SET tel = :tel WHERE id = :id ");  
        $sql->bindValue('tel', $tel, PDO::PARAM_STR);  
        $sql->bindValue('id', $id, PDO::PARAM_INT);  
        $sql->execute();  
        $_SESSION['tel'] = $tel;  
    }else{  
        echo '<br />' . '<p class="center-align red-text"><b>Rappel : Le numéro n\'est pas valide.</b></p>';  
    }  
}
```

Par **exemple**, si je veux **modifier** mon **numéro** de téléphone, ce sera le code ci-dessus en interaction avec la base de données qui sera exécutée si un numéro de téléphone différent est entré. La mise à jour sera alors faite si le numéro est au bon format (10 chiffres maximum).

## 6.e - Back Office

Le backoffice est composé de quatre pages de gestion différentes, Gestion utilisateurs, Gestion catégories, Gestion produits et Galerie.

### Gestion d'utilisateurs

- La **page gestion des utilisateurs** du site web dans laquelle l'administrateur du site peut **consulter le nom, prénom, email, téléphone et les droits des utilisateurs**. Les informations sont affichées dans un **tableau** qui récupère toutes les informations directement depuis la base de données avec une requête SQL.

```
<table class="responsive-table">
  <thead>
    <tr>
      <th>Prénom</th>
      <th>Nom</th>
      <th>E-Mail</th>
      <th>Téléphone</th>
      <th>Droits</th>
    </tr>
  </thead>
  <tbody>
    <?php $gestionUtilisateurs->showTableUsers(); ?>
  </tbody>
</table>
```

*Code HTML du tableau de la page users.php*

```
public function showTableUsers(){

    $con = $this->connectDb();
    $query = $con->prepare( query: "SELECT * FROM utilisateurs");
    $query->execute();
    $resultats = $query->fetchAll();
}
```

*Requête exécutée dans la fonction appelée dans le HTML*

Je stock les résultats récupérés dans la variable **\$resultats**. **fetchAll()** nous retourne chaque ligne en bdd sous forme d'un tableau.

```

1 =>
array (size=14)
  'id' => string '2' (length=1)
  0 => string '2' (length=1)
  'nom' => string 'Barbosa' (length=7)
  1 => string 'Barbosa' (length=7)
  'prenom' => string 'Nuno' (length=4)
  2 => string 'Nuno' (length=4)
  'email' => string 'kns@kns.com' (length=11)
  3 => string 'kns@kns.com' (length=11)
  'tel' => string '0623468138' (length=10)
  4 => string '0623468138' (length=10)
  'password' => string '$2y$12$eDZkr4fxUkvDOPRLcUTf0ODcq1BtcAN.hDhMtI90blvd21lf0R7q' (length=60)
  5 => string '$2y$12$eDZkr4fxUkvDOPRLcUTf0ODcq1BtcAN.hDhMtI90blvd21lf0R7q' (length=60)
  'id_droits' => string '20260' (length=5)
  6 => string '20260' (length=5)

```

J'utilise ensuite un **foreach** qui me permet de récupérer chacune des informations de l'utilisateur avec l'index de la colonne de la base de données.

Il ne me reste donc plus qu'à afficher ses informations dans des **<tr>** **et <td>** pour pouvoir ensuite l'afficher dans la page HTML en instanciant la méthode.

```

foreach($resultats as $result){
    ?>
    <tr>
    <td><?php echo $result["prenom"] ?></td>
    <td><?php echo $result["nom"] ?></td>
    <td><?php echo $result["email"] ?></td>
    <td><?php echo $result["tel"] ?></td>
    <td><?php echo $result["id_droits"] ?></td>
    <td>
        <a href='?show=?<?php echo $result["id"] ?>'>Modifier</a><br />
        <a href='?action=delete&id=?<?php echo $result["id"] ?>'>Supprimer</a>
    </td>
    </tr>

```

*Informations envoyées dans le HTML par la méthode.*

Les deux hyperliens (**Modifier et Supprimer**) servent comme leur nom l'indique à **supprimer** un utilisateur ou bien **modifier** ses informations via un formulaire.

Pour cela on injecte dans l'URL **"?show=?\$result["id"]"** qui correspond à l'id de l'utilisateur du quel on veut modifier les informations.

On récupère ensuite l'id via la méthode `$_GET` pour charger les informations dans les placeholder des input du formulaire de modification.

Les informations sont quant à elles récupérées avec une requête préparée en sélectionnant toutes les informations du user auquel appartient l'id en question.

On affiche ensuite le formulaire dans notre page.

```
<?php
    if (isset($_GET['show'])) {
        $gestionUtilisateurs->modifierUser();
    }
?>
```

*Récupération de l'id de l'user*

```
function modifierUser(){
    $id = $_GET["show"];
    $con = $this->connectDb();
    $query = $con->prepare( "SELECT * FROM utilisateurs WHERE id = :id");
    $query->bindValue( param: "id", $id, type: PDO::PARAM_INT);
    $query->execute();
    $resultats = $query->fetchAll();

    foreach($resultats as $result){
        $nom = $result["nom"];
        $prenom = $result["prenom"];
        $email = $result["email"];
        $tel = $result["tel"];
        $id_droits = $result["id_droits"];
    }
    var_dump($resultats);
?>

<div class="container">
    <div class="row">
        <form id='modifierArticle' class="col s12" action="" method="post">

            <div class="input-field col s12 m4 l4">
                <label>Nom :</label><br/><br />
                <input type="text" name="nom" value="<?php echo $nom;?>" required><br/><br />
            </div>
        </div>
    </div>
</div>
```

*Récupération des informations de l'user et affichage dans les différents input*

Une fois les modifications effectuées, on clique sur le bouton "Modifier" qui lance une suite de code qui récupère toutes les informations des input avec la méthode **POST**. J'utilise **trim()** pour

supprimer les espaces avant et après les informations entrées pour éviter des problèmes pour les insertions ainsi que **htmlspecialchars()** qui transforme tous les caractères en entités **HTML** pour éviter toute injection malveillante en base de données.

```
if (isset($_POST['modifier'])) {  
    $newPrenom = trim(htmlspecialchars($_POST['prenom']));  
    $newNom = trim(htmlspecialchars($_POST['nom']));  
    $newEmail = trim(htmlspecialchars($_POST['email']));  
    $newTel = trim(htmlspecialchars($_POST['tel']));  
    $newId_droits = trim(htmlspecialchars($_POST['id_droits']));  
}
```

*Récupération et sécurisation des informations du formulaire.*

On vérifie ensuite qu'aucun des inputs n'est vide et on exécute pour chacun d'entre eux une requête **UPDATE** pour mettre à jour l'information modifiée en base de données avec une requête préparée.

```
if (!empty($_POST['prenom'])) {  
    $reqID = $con->prepare( query: "UPDATE utilisateurs SET prenom = :newPrenom WHERE id = :id ");  
    $reqID->bindValue( param: 'newPrenom', $newPrenom, type: PDO::PARAM_STR);  
    $reqID->bindValue( param: 'id', $id, type: PDO::PARAM_INT);  
    $reqID->execute();  
}
```

*Requête UPDATE, exécutée pour chacun des différents inputs.*

Pour la méthode qui supprime un utilisateur on procède de la même façon, on récupère l'id de l'utilisateur en question en effectuant une requête **DELETE** de l'utilisateur auquel correspond l'id.

```
if (isset($_GET['id']) and !empty($_GET['id'])) {  
  
    if ($_SESSION['user']['id'] == $_GET['id']) {  
  
        echo '<div class="container">Impossible de supprimer un compte qui est connecté.'.</div>';  
    } else {  
  
        $id = $_GET['id'];  
        $supp = $con->prepare( query: "DELETE FROM utilisateurs WHERE id = :id ");  
        $supp->bindValue( param: 'id', $id, type: PDO::PARAM_INT);  
        $supp->execute();  
        header( header: 'Refresh 0;');  
    }  
}
```

Avant d'effectuer la requête, j'impose une condition qui empêche la suppression de l'utilisateur connecté.

## Gestion des catégories

- La **page de gestion des catégories de produits**, permettant **d'ajouter différentes catégories et sous catégories** qui sont ensuite attribuées aux différents produits mis en ligne dans la e-boutique du site internet.  
Pour cela, un formulaire est mis en place, avec deux input text, un pour Catégories et l'autre pour sous catégorie.

Nom de la nouvelle Catégorie :	Nom de la nouvelles Sous-Catégorie
Ajouter Catégorie	Ajouter Sous-Catégorie

AJOUTER

*Rendu des inputs et bouton dans la page*

```
<form name="Ajouter_Catégorie" action="" id="formcategorie" method="POST">
  <div class="container">
    <div class="row">
      <div class="input-field col s12 m6 l6">
        <label>Nom de la nouvelle Catégorie :</label><br />
        <input type="text" name="newCatégorie" placeholder="Ajouter Catégorie"><br /><br />
      </div>
      <div class="input-field col s12 m6 l6">
        <label>Nom de la nouvelles Sous-Catégorie</label><br />
        <input type="text" name="newSCatégorie" placeholder="Ajouter Sous-Catégorie">
      </div>
    </div>
    <input id="submitForm" class="btn black" type="submit" name="valider" value="Ajouter">
  </div><br/>
</form>
```

*Code des inputs & button*

J'utilise la méthode **trim()** pour supprimer les espaces avant et après les informations entrées pour éviter des problèmes pour les insertions ainsi que la méthode **htmlspecialchars()** qui transforme tous les caractères en entité **HTML** pour éviter toute injection malveillante en base de données  
Un seul bouton submit nous permet d'ajouter les deux informations, ensemble ou séparément grâce à une condition établie en début de page qui vérifie que le input text en question soit rempli avant

d'effectuer la requête, évitant ainsi les insertions à vide et la saturation de la bdd.

```
if (isset($_POST["valider"])) {  
    if (!empty($_POST["newCategorie"])) {  
        $gestionCategories->AjouterCategorieBdd();  
        header( header: 'location:http://localhost/projet_pro/backoffice/gestion_categories.php');  
    }  
    if (!empty($_POST["newSCategorie"])) {  
        $gestionCategories->AjouterSCategorieBdd();  
        header( header: 'location:http://localhost/projet_pro/backoffice/gestion_categories.php');  
    }  
}
```

*Condition de vérification des inputs*

Une requête préparée INSERT est ensuite effectuée pour stocker les informations en bdd.

```
public function AjouterCategorieBdd()  
{  
    $newCategorie = trim(htmlspecialchars($_POST["newCategorie"]));  
    $con = $this->connectDb();  
    $req = $con->prepare( query: "INSERT into categories (categorie) value (:newCategorie)");  
    $req->bindValue( param: "newCategorie", $newCategorie, type: PDO::PARAM_STR);  
    $req->execute();  
}
```

*Requête d'ajout Catégorie en BDD*

```
public function AjouterSCategorieBdd()  
{  
    $newSCategorie = trim(htmlspecialchars($_POST["newSCategorie"])); //  
    $con = $this->connectDb(); // Connexion Db  
    $stmt = $con->prepare( query: "INSERT INTO sous_categories (sous_categorie) values (:nom)");  
    $stmt->bindValue( param: 'nom', $newSCategorie, type: PDO::PARAM_STR);  
    $stmt->execute();  
}
```

*Requête d'ajout Sous-catégorie en BDD*

Une fois des valeurs ajoutées aux deux tables en base de données, on retrouve au-dessus des inputs, deux listes, contenant les différentes catégories et sous catégories présentes en base de données, suivies d'un hyperlien supprimer. L'affichage est géré par l'instanciation d'une méthode qui les récupère en base de données.

## Catégories :

Baume Supprimer  
Brume Supprimer  
Crème Supprimer  
Crème de beauté Supprimer  
Crème de jour Supprimer  
Gel douche Supprimer  
Huile Supprimer  
Lotion Supprimer  
Masque Supprimer  
Masque - Soin Supprimer  
Mousse Supprimer  
Shampooing Supprimer  
Shampooing - Masque Supprimer  
Shampooing - Soin Supprimer  
Shampooing sec Supprimer  
Soin Supprimer  
Soin de nuit Supprimer  
Spray Supprimer  
Traitement anti chute Supprimer  
Velouté Supprimer  
Vinaigre Supprimer

## Sous Catégories :

Absolue Kératine Supprimer  
Astera Fresh Supprimer  
Astera Sensitive Supprimer  
Curbicia Supprimer  
Forticea Supprimer  
Karinga Supprimer  
Karité Hydra Supprimer  
Karité Nutri Supprimer  
Lissea Supprimer  
Lumicia Supprimer  
Melaleuca Supprimer  
Naturia Supprimer  
Okara Blond Supprimer  
Okara Color Supprimer  
Okara Silver Supprimer  
Solaire Supprimer  
Style Supprimer  
Sublime Curl Supprimer  
Tonucia Supprimer  
Triphasic Progressive Supprimer  
Triphasic Reactional Supprimer  
Volumea Supprimer

*Rendu dans la page du site*

```
public function AfficherCategoriesBdd()
{
    $con = $this->connectDb();
    $req = $con->prepare( query: "SELECT * FROM categories ORDER BY categorie ASC");
    $req->execute();
    $result = $req->fetchAll();

    echo "<h2> Catégories : </h2>";
    foreach ($result as $resultat) {
        echo $resultat['categorie'] . ' ' . ' <a class="href_admin" href="gestion_categories.php?id=' . $resultat['id'] . '>' . ' <b>Supprimer</b>' . '</a>' . "<br />";
    }
}
```

*La méthode qui appelée qui affiche les listes, ici celle des catégories.*

Les catégories sont récupérées en ordre **ASC** (ascendant) pour faciliter la lecture.

**L'hyperlien "Supprimer"** nous permet, grâce à l'**id** de la donnée qui le précède récupéré dans l'url avec la méthode **GET**, d'effectuer une requête qui se charge **d'effacer la valeur correspondante**. La condition fait partie de la même méthode.

```
if (isset($_GET['id']) and !empty($_GET['id'])) {
    $id = htmlspecialchars($_GET['id']);
    $supp = $con->prepare( query: "DELETE FROM categories WHERE id = :id ");
    $supp->bindValue( param: 'id', $id, type: PDO::PARAM_INT);
    $supp->execute();
    header( header: 'location:http://localhost/projet_pro/backoffice/gestion_categories.php');
}
```

*Condition de suppression de catégorie*



## Gestion des produits

- La **page gestion de produits** permet à l'administrateur d'ajouter ou supprimer des produits à la vente, ainsi qu'une fonction modifier qui se charge de modifier les informations (prix, stock, description..) des produits déjà existants.

En arrivant sur la page, on trouve tout d'abord un tableau où tous les produits stockés en base de données sont affichés.

Le tableau est affiché par une méthode récupérée dans le fichier **backOfficeClass.php**.

```
$query = $con->prepare( query: "SELECT * FROM produits LIMIT :limite OFFSET :debut");
//On lie les valeurs
$query->bindValue( param: 'limite', $limite, type: PDO::PARAM_INT);
$query->bindValue( param: 'debut', $debut, type: PDO::PARAM_INT);
$query->execute();

echo "<br /><br /><br />";
echo "<div class='row'>";
echo "<table id='tableProducts' class='responsive-table' ><thead>";
echo "<th>Image Produit</th>";
echo "<th>Nom Produit</th>";
echo "<th>Prix Produit</th>";
echo "<th>Volume</th>";
echo "<th class='hide-on-med-and-down'>Description Produit</th>";
echo "<th>N° en Stock</th>";
echo "</thead><tbody>";

while($r = $query->fetch( mode: PDO::FETCH_OBJ)){

    echo "<tr>";
    echo "<td><img src='../Images/' . $r->image_nom . '.jpg' width='100px' height='100px' /></td>";
    echo "<td> . $r->nom . "</td>";
    echo "<td> . $r->prix . "€</td>";
    echo "<td> . $r->volume . "</td>";
    echo "<td class='hide-on-med-and-down'> . $r->description . "</td>";
    echo "<td> . $r->stock . "</td>";
    echo "<td><a id='modifyProduct' href='?show=" . $r->id . "' onclick='modifyProductsHideForms()'>Modifier</a><br/>";
    echo "<a href='?action=delete&id=" . $r->id . "'>Supprimer</a></td>";
    echo "</tr>";
}

echo "</tbody></table></div><br /><br />";
```

*Affichage du tableau après récupération des données en base de données.*

Un système de pagination a été mis en place pour réduire la taille de l'affichage du tableau, affichant que 5 articles par page.

```
//Connexion Bdd
$con = $this->connectDb();
$page = intval($_GET['page']); //conversion forcée en entier
//Si le nombre est invalide, on demande la première page par défaut
if ($page <= 0) {
    $page = 1;
}

$limite = 5;

$resultFoundRows = $con->query( statement: "SELECT count(id) FROM produits");
$nombreElementsTotal = $resultFoundRows->fetchColumn();
$debut = ($page - 1) * $limite;
// Partie "Requête"
// On construit la requête, en remplaçant les valeurs par des marqueurs. Ici, on
// n'a qu'une valeur, limite. On place donc un marqueur là où la valeur devrait se
// trouver, sans oublier les deux points « : »

$query = $con->prepare( query: "SELECT * FROM produits LIMIT :limite OFFSET :debut");
//On lie les valeurs
$query->bindValue( param: 'limite', $limite, type: PDO::PARAM_INT);
$query->bindValue( param: 'debut', $debut, type: PDO::PARAM_INT);
$query->execute();
```

### *Début pagination*

```
//On calcule le nombre de pages
$nombreDePages = ceil( num: $nombreElementsTotal / $limite);

/* Si on est sur la première page, on n'a pas besoin d'afficher de lien*/
/* vers la précédente. On va donc ne l'afficher que si on est sur une autre*/
/* page que la première*/

if ($page > 1) :
    ?<button class="btn black"><a class="aFooter" href="?page=?php echo $page - 1; ?"></a></button>
<?php endif;
echo " ";
for ($i = 1; $i <= $nombreDePages; $i++) :
    ?<u><a href="?page=?php echo $i; ?"><?php echo $i; ?</a></u>
<?php endfor;
echo " ";
//Avec le nombre total de pages, on peut aussi masquer le lien vers la page suivante quand on est sur la dernière//
if ($page < $nombreDePages) :
    ?<button class="btn black"><a class="aFooter" href="?page=?php echo $page + 1; ?"></a></button>
<?php endif; ?<?php
```

### *Suite Pagination*

Sous la tableau sont présents deux boutons "Précédent" et "Suivant" qui nous permettent d'avancer dans les différentes pages du tableau.

Un script javascript gère l'affichage des différents éléments de la page. Si le tableau Produits est visible, on cache le formulaire d'ajout et vice versa.

```
function showHideAddProduct() {
  let divAddProducts = document.getElementById( elementId: "formAddProduct");
  let tableViewProducts = document.getElementById( elementId: "tableProduct");
  let speechAjoutProduit = document.getElementById( elementId: "divAjouterProduit");
  if (divAddProducts.style.display === "block") {
    divAddProducts.style.display = "none";
    tableViewProducts.style.display = "block";
    speechAjoutProduit.style.display = "block";
  } else {
    divAddProducts.style.display = "block";
    tableViewProducts.style.display = "none";
    speechAjoutProduit.style.display = "none";
  }
}
```

### *Script page produits*

On commence par sélectionner les éléments à traiter en déclarant une variable pour chacun d'entre eux.

**document.getElementById()** permet de sélectionner l'élément présent dans le document qui a pour id le paramètre que l'on rentre dans les parenthèses.

Une condition if s'occupe de changer le style css des éléments en fonction de celui qui est affiché.

**L'ajout de nouveaux produits** se fait via un formulaire qui devient visible en cliquant sur un bouton "Ajouter un produit" présent au début de la page.

On y retrouve plusieurs input text dans lesquels on renseigne des informations concernant le nouveau produit (Nom, Prix, Description, Volume, Stock)

ainsi que des input select permettant de choisir la catégorie et sous catégorie que l'on souhaite attribuer au produit.

Les valeurs <option> présentes dans le select sont récupérées en base de données avec une requête **SELECT** présente dans une méthode. Pour chacun des résultats obtenus on retourne un nouvel <option> qui a comme **value l'id** de la catégorie et comme valeur affichée dans le select son nom.

```
function selectCategory()
{
    $con = $this->connectDb(); // Connexion Db
    $stmt = $con->prepare( query: "SELECT * FROM categories ORDER BY categorie ASC");// Requete
    $stmt->execute();//J'exécute la requete
    $result = $stmt->fetchAll( mode: PDO::FETCH_ASSOC);//Result devient un tableau des valeurs obtenues
    echo'ok';
    foreach($result as $resultat){
        $categorie = $resultat["categorie"];
        $idCategorie = $resultat["id"];

        echo "<option value='$idCategorie'>$categorie</option>";
    }
}
```

*Méthode qui permet d'afficher les différentes valeurs du <select>*

Pour l'ajout de l'image un simple input File est disposé en fin de formulaire.

Le une fois une image ajoutée dans l'input, le fichiers est soumis à plusieurs vérifications.

On commence tout d'abord par récupérer le nom du fichier et son nom temporaire et on les attribue a deux variables.

```
$Img = $_FILES["Img"]["name"];
$img_Tmp = $_FILES["Img"]["tmp_name"];
```

A partir de là on commence les vérifications.

Une condition **if** s'assure que le nom temporaire existe ("tmp\_name").

On sépare la chaîne de caractères (nom du fichier) en deux avec la fonction **explode()**. Le premier élément sera donc le nom attribué à l'image et la deuxième son extension.

On attribue à une variable ici appelée **\$img\_Ext** la dernière valeur du tableau grâce à la **fonction end()**, ici la valeur de l'extension du fichier.

```
$img_Name = explode( separator: ".", $Img); //On sépare la chaîne de caractères en deux parties, avant et après le point
$img_Ext = end( &array: $img_Name); //On attribue à img_Ext la valeur du dernier élément du tableau, l'extension du fichier
```

*Récupération de l'extension de l'image*

Par la suite, une condition **if** vérifie à l'aide de la **fonction in\_array()** si l'extension de l'image correspond une des extensions acceptées.

La **fonction strtolower()** converti toute la chaîne de caractères en lettres minuscules.

Si le contenu de notre variable ne correspond pas, on affiche un message d'erreur demandant à l'utilisateur de choisir un fichier du bon format. Si c'est le cas, on continue le traitement.

```
if(in_array(strtolower($img_Ext),array("png", "jpg", "jpeg")) === false){ //si le dernier element du tableau  
    echo "L'image insérée doit avoir pour extension : .png, .jpg, .jpeg"; //On affiche un message d'erreur
```

### *Vérification de l'extension du fichier*

Pour la suite du traitement j'ajoute à la variable **\$img\_Size** la valeur du poids de l'image avec la **fonction getimagesize()**. On soumet ensuite des conditions avant de recréer une nouvelle image à partir du fichier que l'on vient d'upload avec les fonctions

**imagecreatefromjpeg()** & **imagecreatefrompng()**

En cas d'échec, on affiche un message d'erreur informant que le fichier n'est pas valide.

```
$img_Size = getimagesize($img_Tmp); //On attribue a la varirable img_Size la valeur du poids de l'image  
if($img_Size["mime"] == "image/jpeg"){ //si l'extension d'image est egale a jpeg  
    $img_Src = imagecreatefromjpeg($img_Tmp); //On recrée une nouvelle image a partir du jpeg  
}else if ($img_Size["mime"] == "image/png"){ //Si l'extension d'image est egale a png  
    $img_Src = imagecreatefrompng($img_Tmp); //On recrée une image a partir du png  
}else{ //sinon  
    $img_Src = false; //Source d'image = false  
    echo "Veuillez rentrer une image valide"; //On affiche le message d'erreur comme quoi l'image n'est pas valide  
}
```

### *Création d'une nouvelle image à partir du fichier uploadé*

Si la condition est remplie, on entame la dernière étape du traitement du fichier.

On détermine les dimensions souhaitées pour la nouvelle image et avec la **fonction imagecreatetruecolor()** on recrée une image avec des couleurs vraies.

La **fonction imagecopyresampled()** Copie, redimensionne, rééchantillonner une image et prend comme paramètres pour cela 10 paramètres ( Ressource cible de l'image, Ressource source de l'image, X : coordonnées du point de destination, Y : coordonnées du

point de destination, X : coordonnées du point source, Y : coordonnées du point source, Largeur de la destination, Hauteur de la destination, Largeur de la source et Hauteur de la source) .  
 Enfin on choisit le nouveau nom pour le fichier fraîchement recréé, j'ai pris le choix de utiliser la **fonction md5()** combiné avec la **fonction uniqid()** car ça me permet d'attribuer à la nouvelle image un nom unique et crypté empêchant ainsi que deux images se retrouvent avec le même nom, pouvant créer des complications pour les récupérations d'image plus tard lorsque je devrais les afficher.  
 Pour finaliser la phase de traitement du fichier je le déplace dans le dossier souhaité à l'aide de la **fonction imagejpeg()**, qui prend comme paramètres l'image traitée et le chemin de la destination du fichier.

```
if($img_Src != false){ //Si l'image est au bon format
    $img_Width = 1000;
    if($img_Size[0] == $img_Width){
        $img_Finale = $img_Src;
    }else{
        $new_Width[0] = $img_Width;
        $new_Height[1] = 1000;
        $img_Finale = imagecreatetruecolor($new_Width[0], $new_Height[1]);
        imagecopyresampled($img_Finale, $img_Src, dst_x: 0, dst_y: 0, src_x: 0, src_y: 0, $new_Width[0], $new_Height[1], $img_Size[0], $img_Size[1]);
    }
    $encName = md5(uniqid());
    imagejpeg($img_Finale, filename: "../Images/" . $encName . ".jpg");
}
```

### *Phase finale du traitement d'image*

Pour récupérer l'image par la suite, je n'aurais qu'à récupérer le enc\_name de l'image en base de données et l'insérer dans le href de la balise <img> de la façon suivante :

```
<img src='../Images/" . $r->enc_name . ".jpg' />
```

Enfin pour en finir avec l'ajout du nouveau produit, une requête préparée **INSERT** est effectuée pour stocker toutes les différentes informations concernant le produit en base de données.

```

$con = $this->connectDb();
$req = $con->prepare( query: "INSERT INTO produits(nom,description,prix,volume,id_categorie,id_sous_categorie,stock,image_nom) values (:nom,:description,:prix,:vol
$req->bindValue( param: "nom", $nomProduit, type: PDO::PARAM_STR);
$req->bindValue( param: "prix", $prixProduit, type: PDO::PARAM_STR);
$req->bindValue( param: "volume", $volumeProduit, type: PDO::PARAM_STR);
$req->bindValue( param: "description", $descriptionProduit, type: PDO::PARAM_STR);
$req->bindValue( param: "id_categorie", $idCategorie, type: PDO::PARAM_INT);
$req->bindValue( param: "id_sous_categorie", $idScategorie, type: PDO::PARAM_INT);
$req->bindValue( param: "stock", $stockProduit, type: PDO::PARAM_INT);
$req->bindValue( param: "encname", $encName, type: PDO::PARAM_STR);

$req->execute();

```

### Requete insertion en base de données

La modification des produits existants est gérée par une formulaire accessible depuis un hyperlien présent devant les informations de chacun des articles.

Cet hyperlien :

```

<a id='modifyProduct' href='?show=" . $r->id . "'
onclick='modifyProductsHideForms();'>Modifier</a>

```

Il nous recharge la même page mais ajoute dans l'URL ?show= + l'id de l'article en question qui est au rechargement de la page récupéré par la méthode \$\_GET["show"] et à partir du quel je récupère toutes les informations du produit visé pour les afficher dans les placeholder des input du formulaire de modification.

Un script javascript est aussi exécuté pour cacher les autres éléments de la page laissant place à l'affichage du formulaire uniquement.

### Code du formulaire de modification de produit

```

$id = $_GET["show"];
$con = $this->connectDb();
$product = htmlspecialchars($_GET['show']);
$request = $con->prepare( query: "SELECT * FROM produits WHERE id = :id"); // Requête SQL
$request->bindValue( param: "id", $id, type: PDO::PARAM_INT);
$request->execute(); // On execute

$s = $request->fetch( mode: PDO::FETCH_OBJ); // Résultat stocké dans la $s

?>
<br /><br /><br />
<div class="container">
  <div class="center-align">
    <br/><br/>
    <br/><br/>
  </div>
</div>
<div class="row">
<form id='modifierArticle' class="col s12" action="" method="post">
  <div class="input-field col s12 m4 l4">
    <label>Titre :</label><br/><br />
    <input type="text" name="nom" value="<?php echo $s->nom;?>" required><br/><br />
  </div>
  <div class="input-field col s12 m4 l4">
    <label>Description :</label><br/><br />
    <textarea name="description" rows="4" cols="50" required><?php echo $s->description;?></textarea><br/><br />
  </div>
  <div class="input-field col s12 m4 l4">
    <label>Prix :</label><br/><br />
    <input type="text" name="prix" value="<?php echo $s->prix;?>" required><br/><br />
  </div>
  <div class="input-field col s12 m4 l4">
    <label>Stock :</label><br/><br />
    <input type="text" name="stock" value="<?php echo $s->stock;?>" required><br/><br />
  </div><br/>

```

Le formulaire est affiché par l'instanciation de la méthode qui comporte ce code.

Une fois les informations modifiées à souhait, au clique de l'input submit, une requête préparée est effectuée pour mettre à jour les informations concernant le produit, en base de données.

```
<?php
// Si on appuie sur modifier
if(isset($_POST['envoyer'])){

    $nom = trim(htmlspecialchars(addslashes($_POST['nom'])));
    $description = trim(htmlspecialchars(addslashes($_POST['description'])));
    $prix = trim(htmlspecialchars($_POST['prix']));
    $stock = trim(htmlspecialchars($_POST['stock']));

    // Requête de modification
    //UPDATE produits SET nom = 'coco', prix = 10, description = 'Oui', id_categorie = 21, id_sous_categorie = 22, stock = 23, chemin_image = 0 WHERE id = 36

    $update = $con->prepare("UPDATE produits SET nom = :nom, prix = :prix, description = :description, stock = :stock WHERE id = :id");
    $update->bindValue( param: "nom", $nom, type: PDO::PARAM_STR);
    $update->bindValue( param: "prix", $prix, type: PDO::PARAM_INT);
    $update->bindValue( param: "description", $description, type: PDO::PARAM_STR);
    $update->bindValue( param: "stock", $stock, type: PDO::PARAM_INT);
    $update->bindValue( param: "id", $id, type: PDO::PARAM_INT);
    $update->execute();

    //REFRESH / HEADER LOCATION AVEC JAVASCRIPT
    $gestionProduits = new backOffice;
    echo "<script language='JavaScript'> document.location.replace('http://localhost/projet_pro/backoffice/gestion_produits?page=1.php'); </script>";
    $gestionProduits -> viewAllProduits();
}
```

## Gestion galerie

- La page Galerie, ou l'administrateur du site peut ajouter des images qui seront visibles dans la page Galerie du site internet permettant aux clients de voir les différents services effectués dans le salon de coiffure.

Il pourra aussi ajouter et supprimer des catégories d'image.

La page comporte donc 2 formulaires, le premier permet d'ajouter des nouvelles catégories d'images et le deuxième pour ajouter une nouvelle image à la galerie.

### Formulaire d'ajout de catégorie

Le formulaire d'ajout de catégorie d'images ne comporte que deux éléments, un input texte et un bouton submit.

Une fois le nom de la catégorie renseigné dans le input de type texte et que l'on clique sur le bouton d'ajout de catégorie, une vérification est effectuée avec un **if()** qui vérifie que la catégorie renseignée n'existe pas déjà en base de données. cela permet d'éviter les doublons et d'alléger la base de données.



Si la condition est respectée, une requête **INSERT** est effectuée pour insérer la nouvelle catégorie en base de données.  
La requête est exécutée par l'instanciation d'une méthode.

```
public function addNewGalerieCategory(){
    $newCategory = trim(htmlspecialchars($_POST["newGalerieCategory"]));

    $con = $this->connectDb();
    $request = $con->prepare( query: "SELECT * FROM categories_galerie");
    $request->execute();
    $resultat = $request->fetchAll();

    foreach($resultat as $result){
        if($result["categorie"] == $newCategory){
            return false;
        }
    }

    $insertRequest = $con->prepare( query: "INSERT INTO categories_galerie (categorie) VALUES (:newCategory)");
    $insertRequest->bindValue( param: "newCategory", $newCategory, type: PDO::PARAM_STR);
    $insertRequest->execute();
}
```

Toute catégorie de photo ajoutée en base de données est ensuite affichée dans une liste positionnée au dessus du formulaire, avec la possibilité de la supprimer en cliquant sur un hyperlien "Supprimer".

```
public function actualCategory(){
    $con = $this->connectDb();
    $request = $con->prepare( query: "SELECT * FROM categories_galerie");
    $request->execute();
    $resultat = $request->fetchAll();

    echo "<p> Categories galerie existantes : </p>";
    foreach ($resultat as $result) {
        echo $result["categorie"] . " . " . "<a class='href_admin' href='gestion_galerie.php?id= ' . $result['id'] . '> . " . <b>Supprimer</b> . " . </a> . " . <br />";
    }

    if (isset($_GET['id']) and !empty($_GET['id'])) {
        $id = htmlspecialchars($_GET['id']);
        $supp = $con->prepare( query: "DELETE FROM categories_galerie WHERE id = :id ");
        $supp->bindValue( param: 'id', $id, type: PDO::PARAM_INT);
        $supp->execute();
        header( 'location:http://localhost/projet_pro/backoffice/gestion_galerie?page=1.php' );
    }
}
```

### *Méthode d'affichage des différentes catégories*

Si l'on clique sur l'hyperlien disposé devant une catégorie, l'id de la catégorie en question est injecté dans l'URL. La méthode `$_GET["id"]` se charge ensuite de le récupérer pour procéder à la requête de suppression en base de données.  
La page est ensuite rechargée pour rafraîchir la liste des catégories.

Le deuxième formulaire quant à lui nous sert à ajouter les nouvelles photos à notre site.\*

Il comporte **4 éléments**, un **input texte** pour renseigner le nom de la photo, un **select** où l'on peut sélectionner la catégorie à laquelle appartient l'image, un **input File** pour l'insertion de la photo et un **input submit** pour valider le tout qui déclenche aussi la méthode d'ajout en base de données.

Pour se faire dans notre méthode on attribue les valeurs Nom et Catégorie à deux variables puis on effectue la vérification et traitement de l'image. Le processus de traitement d'image est exactement le même que pour l'ajout de d'un nouveau produit, je ne vais donc pas le détailler à nouveau ici.

Si toutes les différentes conditions sont remplies, on procède à l'insertion des données en base de données en envoyant une requête INSERT.

```
$con = $this->connectDb();  
$request = $con->prepare( query: "INSERT INTO images_galerie(nom_image, id_categorie, enc_name) VALUES (:nom, :id_categorie, :encname)");  
$request->bindValue( param: "nom", $nomProduit, type: PDO::PARAM_STR);  
$request->bindValue( param: "id_categorie", $photoCategory, type: PDO::PARAM_INT);  
$request->bindValue( param: "encname", $encName, type: PDO::PARAM_STR);  
$request->execute();
```

Les différentes photos sont consultables depuis un tableau en début de page.

Toutes les pages sont responsives au format mobile, tablette, ordinateur et grands écrans.

## 6.f - Responsive

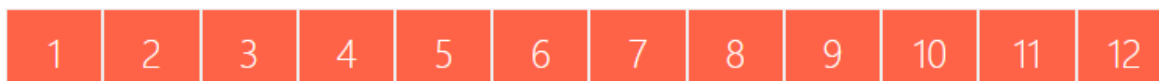
Le site sera responsive et adaptable à toute taille d'écrans. Pour se faire nous allons utiliser le framework CSS Materialize qui grâce à son système de grid, nous permet de redimensionner chacun des éléments selon les différentes tailles d'écran.

La fonctionnalité se base sur une division de la page web en 12 colonnes de même taille, contrôlables très simplement par une classe donnée à une div qui entoure nos éléments en question.

Take a look at this section to quickly understand how the grid works!

### 12 Columns

Our standard grid has 12 columns. No matter the size of the browser, each of these columns will always have an equal width.



To get a feel of how the grid is used in HTML, take a look at the code below which will produce a similar result to the one above.

language-markup

```
<div class="row">
  <div class="col s1">1</div>
  <div class="col s1">2</div>
  <div class="col s1">3</div>
  <div class="col s1">4</div>
  <div class="col s1">5</div>
  <div class="col s1">6</div>
  <div class="col s1">7</div>
  <div class="col s1">8</div>
  <div class="col s1">9</div>
  <div class="col s1">10</div>
  <div class="col s1">11</div>
  <div class="col s1">12</div>
</div>
```

Dans cet exemple, on voit que une div de class "row" englobe plein d'autres petites divs. La classe "row" permet d'afficher son contenu en ligne. Dans cette div, 12 autres divs de class "col s1" qui veut dire que dans un affichage en petit écran la div en question occupera l'espace d'une colonne.

## 7. Veille sur la sécurité

## 7.a - Diagnostic

Durant l'année, au cours des projets, nous avons abordé un point important, la sécurité. Nous avons identifié des éléments tels que les failles xss, l'injection sql mais aussi le fait de sécuriser l'upload de fichiers. Je vais aborder ces différents points qui ont été mis en place sur le projet afin d'avoir un site sécurisé pour l'utilisateur mais aussi pour l'administrateur de ce site.

### Côté client : Faille XSS (Cross-site scripting)

#### Définition:

Le **cross-site scripting** (abrégé XSS) pour ne pas être confondu avec le CSS (feuilles de style) est un type de **faille** de **sécurité** des sites web permettant **d'injecter** du **contenu** dans une **page**, provoquant ainsi des **actions** sur les **navigateurs web** visitant la page. Les possibilités des XSS sont très larges puisque l'attaquant peut utiliser tous les langages pris en charge par le navigateur (JavaScript, Java...) et de nouvelles possibilités sont régulièrement découvertes notamment avec l'arrivée de nouvelles technologies comme HTML5. Il est par exemple **possible** de **rediriger** vers un **autre site** pour de **l'hameçonnage** ou encore de **voler** la session en récupérant les **cookies**.

### Quels sont les moyens pour contrer une faille XSS?

Pour contrer la faille XSS, il existe différentes fonctions.

- **stripetags()** a pour objectif de retirer toutes balise html.
- **htmlspecialchars()** à pour objectif de convertir les caractères spéciaux en entités HTML. C'est aussi la fonction que j'utilise le plus souvent pour contrer les failles XSS.
- **htmlentities()** est identique à la fonction *htmlspecialchars()* sauf que tous les caractères qui ont des équivalents en entités HTML sont effectivement traduits.

## Côté serveur : L'injection SQL

### Définition:

Une **injection SQL** est une forme de **cyberattaque** lors de laquelle un **pirate utilise** un morceau de code **SQL** (« Structured Query Language », langage de requête structurée) **pour manipuler** une **base de données** et **accéder et détourner des informations** potentiellement **importantes**. La requête par exemple peut être dans l'URL du site visé.

### Quels sont les moyens pour contrer une injection SQL?

Pour contrer les injections SQL, je favorise les **requêtes préparées**. A l'aide du **bindValue** je vais venir **associer** une **valeur** à un **paramètre**.

```
$infoUser = $con->prepare("INSERT INTO `utilisateurs`  
    (`nom`, `prenom`, `email`, `password`, `id_droits`, `tel`)  
VALUES  
    (:nom, :prenom, :email, :password, :id_droits, :tel )");  
$infoUser->bindValue('nom', $nom, PDO::PARAM_STR);  
$infoUser->bindValue('prenom', $prenom, PDO::PARAM_STR);  
$infoUser->bindValue('email', $email, PDO::PARAM_STR);  
$infoUser->bindValue('password', $hash, PDO::PARAM_STR);  
$infoUser->bindValue('id_droits', $chiffre, PDO::PARAM_INT);  
$infoUser->bindValue('tel', $tel, PDO::PARAM_STR);  
  
$infoUser->execute();
```

*Requête pour insérer un nouvel utilisateur en base de données*

### **Gestion du back office:**

**Chaque utilisateur** du site aura un **id\_droits** qu'il soit **administrateur** ou **utilisateur**. Pour un administrateur il sera de **20260** et pour un utilisateur **1**. De cette façon, chaque **page réservée à l'administrateur** aura une condition qui contrôlera l'id\_droits de la personne qui tente de venir sur la page administrateur. Si et seulement si l'id\_droits est de 20260 alors la personne est **autorisée** à rester sur le back office, dans le cas contraire il est redirigé vers la page index.

```
if ($id_droits != 20260) {
    header('location:http://localhost:8888/projet_pro/index.php');
    exit();
}
```

*Condition d'accès présent sur chaque page du back office*

### Sécurisation des mots de passe utilisateurs:

Concernant la sécurité des mots de passe utilisateurs j'ai utilisé la fonction **passwordhash()** qui va **hacher** le **mot de passe** utilisateurs en **base de données**. Entre autres, cette fonction permet de ne pas laisser à nue, donc **vulnérable**, les mots de passe des utilisateurs du site. La **fonction** requiert **3 paramètres**, la variable **\$password** qui contient le mot de passe entré, **PASSWORD\_BCRYPT**, ainsi qu'une variable **\$options**, ce sera le salt.

PASSWORD\_BCRYPT va créer une clé de hachage en utilisant l'identifiant '\$2y\$', ainsi le résultat sera toujours une chaîne de 60 caractères ou false si une erreur survient.

Le salt (grain de sel) est une petite donnée additionnelle qui renforce significativement la puissance du hachage pour le rendre beaucoup plus difficile à cracker.

```
// Hashage mdp
$options = ['cost' => 12,];
$password = trim(htmlspecialchars($_POST['iPassword']));
$hash = password_hash($password, PASSWORD_BCRYPT, $options);
```

*La partie de la méthode qui va hacher le mot de passe*

### password

\$2y\$12\$9Ch466gHmO1eCLbrTnSr6uZDzp5u48hY/jlmwPIV8Il...

\$2y\$12\$KUI3On.AMyuv/u7VoEveROiyucxRjy9j97xGTE14gP/...

*Le mot de passe une fois envoyé en base de données et haché.*

## 8. Recherche sur site anglophone


### 8.a - Situation ayant demandé une recherche

Pour la page produits, j'ai recherché sur internet une façon de cacher du contenu en javascript à partir d'une injection dans l'url.

En quelques mots, lorsque je souhaite modifier un produit, ?show="id\_produit" est injecté dans url de la page affichant ensuite un formulaire de modification. Le problème était que lorsque le formulaire s'affichait, le restant du contenu de la page s'affichait aussi plus bas.

J'ai donc mené ma recherche pour que je puisse cacher le contenu de la page normalement affiché.

J'ai trouvé un post sur le site [www.stackoverflow.com](http://www.stackoverflow.com) qui m'a permis de le faire.



How could I do something like this:

406

111

```
<script type="text/javascript">
$(document).ready(function () {
  if(window.location.contains("franky")) // This doesn't work, any suggestions?
  {
    alert("your url contains the name franky");
  }
});
</script>
```

javascript jquery url

Share Improve this question Follow

edited Jan 3 '13 at 16:42 Smi 12.6k 9 53 61

asked Jan 4 '11 at 18:31 RayLoveless 16.3k 19 70 91

3 "window.location.contains is not a function" – gene b. Sep 3 '19 at 14:34

Add a comment

19 Answers

Active Oldest Votes

You need add href property and check indexOf instead of contains

737

```
<script src="https://cdnjs.cloudflare.com/ajax/libs/jquery/3.3.1/jquery.min.js"></script>
<script type="text/javascript">
$(document).ready(function() {
  if (window.location.href.indexOf("franky") > -1) {
    alert("your url contains the name franky");
  }
});
</script>
```

Run code snippet Expand snippet

## 9. Structure du site

### 9.a - Classes et méthodes

Pour la création de ce site j'ai utilisé la **programmation orientée objet** ainsi que les **classes de PHP**. Une classe est une **entité** qui regroupe des **attributs** et des **méthodes**.

On peut retrouver au sein d'une classe des **attributs**, par exemple pour un **utilisateur** ce sera des variables qui vont contenir **son nom, prénom, âge**. Ensuite on peut y trouver des **méthodes**, c'est-à-dire des fonctionnalités qui pourront être réutilisées dans n'importe quelle page, par exemple dans la page **inscription** sur le formulaire d'inscription avec la **méthode inscription()** je vais pouvoir ajouter un nouvel utilisateur en base de données.

Pour **utiliser** une **classe**, il va falloir **instancier un objet**, sans quoi on ne pourra pas utiliser ses méthodes.

*Instanciation d'une classe*

```
require_once('../html_partials/header.php');  
include '../autoloader.php';  
$pageInscription = new users();
```

Grâce à la notion **d'héritage** la classe enfant hérite de toutes les méthodes publiques et protégées, propriétés et constantes de la classe parente.

*La classe users hérite des fonctionnalités de bdd sa classe parent*

```
<?php  
require_once('dbClass.php');  
class users extends bdd  
{
```



Les **propriétés** de **classe** sont définis comme public, protected ou private.

**Public:** Les méthodes seront accessibles depuis la classe mère, enfants et l'extérieur de la classe.

**Protected:** Les méthodes seront accessibles uniquement depuis la classe mère et enfant.

**Private:** Les méthodes seront accessibles uniquement depuis la classe en question.

*Méthode register pour inscrire l'utilisateur en base de données, cette méthode est publique*

```
// Méthode pour s'inscrire
public function register(){
    if(isset($_POST['inscription']) && !empty($_POST['inscription'])){
```

## 10. Jeu d'essai

Pour la représentation du jeu d'essai je vais me servir du formulaire d'inscription du site.

Avant d'insérer le nouvel utilisateur en base de données, plusieurs essais et vérifications sont effectués.

Quand l'utilisateur soumet le formulaire d'inscription, les données du formulaire sont récupérées et envoyées en POST au fichier PHP. Tout d'abord on s'assure que tous les champs soient remplis avec la méthode `!empty()`.

Je récupère dans des différentes variables en utilisant `htmlspecialchars()` pour stocker les informations entrées par l'utilisateur dans les différents champs .

J'effectue une requête SQL pour récupérer tous les identifiants de connexion (emails) en base de données pour vérifier par la suite si l'identifiant n'est pas déjà utilisé.

### **Vérifications**

#### **Téléphone**

La première vérification s'agit de celle du numéro de téléphone entré par l'utilisateur.

```
if(!is_numeric($tel) || strlen($tel) != 10 ){  
    echo '<br/> <b><p class="container center-align  
red-text">Veuillez rentrer un numéro valide.</p></b>';  
    return false;  
}
```

Pour cela on utilise la méthode `is_numeric()` pour vérifier qu'il s'agit bien de caractères numériques et `strlen()` pour vérifier le nombre de chiffres que possède le numéro.

Si le numéro ne correspond pas aux critères de vérification, on envoie un message d'erreur demandant à l'utilisateur d'entrer un numéro valide.

## **Identifiant / E-mail**

On a décidé d'utiliser l'email comme identifiant pour ce projet. Une requête en base de données est effectuée au préalable et essaye de récupérer l'email entré par l'utilisateur en base de données. Si le retour de cette requête = true, cela signifie que l'email est déjà présent, on affiche donc un message d'erreur.

```
if ($stmt->fetch(PDO::FETCH_ASSOC) == true) {  
    // Si il existe déjà echo message d'erreur  
    echo '<br/> <b><p class="container center-align  
red-text">Email déjà existant.</p></b>';  
    return false;
```

## **Mot de passe**

Avant d'être inséré en base de données, le mot de passe est traité et haché avec la méthode password\_hash().

Avant l'inscription, on demande à l'utilisateur un mot de passe comportant une lettre majuscule, une minuscule, un chiffre, un caractère spécial et au minimum 7 caractères en tout.

La vérification se fait avec la méthode preg\_match() qui renvoie 1 si la chaîne de caractères traitée comporte bien la vérification à laquelle elle est soumise.

Une variable est créée et appelée \$testpwd.

```
$testpwd = preg_match("#[A-Z]#", $password) +  
preg_match("#[a-z]#", $password) + preg_match("#[0-9]#",  
$password) + preg_match("#[^a-zA-Z0-9]#", $password);
```

Si le mot de passe respecte tous les critères, la valeur de cette variable sera donc de 4 dû à l'ajout de toutes les vérifications.

Si ce n'est pas le cas, on envoie un message d'erreur rappelant les critères demandés.

```
elseif ($testpwd < 4) {  
    echo '<br />' . '<b><p class="container center-align  
red-text"><b>Rappel : Votre mot de passe doit contenir au  
minimum 7 caractères, incluant une majuscule, un chiffre et  
un caractère spécial.</p></b></b>';  
    return false;  
}
```

Si la vérification est acceptée, on s'assure que le mot de passe et la confirmation de mot de passe soient identiques avec une condition if()).

```
elseif ([ $password ] != [ $confpassword ]) {  
    echo '<br />' . '<b><p class="container center-align  
red-text">Les mots de passe ne correspondent pas.</p></b>';  
    return false;  
}
```

Si ce n'est pas le cas un nouveau message d'erreur sera affiché à l'écran de l'utilisateur.

Si toutes les conditions sont respectées on procède donc à l'insertion des données en BDD avec une requête préparée INSERT.

L'utilisateur est maintenant redirigé vers la page connexion et peut se connecter à son compte et naviguer librement sur le site internet.