

BELOUCIF ANICET

dossier de projet



Compétence couverte par le projet	3
Résumé du dossier	4
Cahier des charges	5
les objectifs	5
la cible	5
le périmètre du projet	6
l'arborescence du site	6
Charte graphique	6
Organisation et background de développement	10
Technologies	7
Workflow	8
Modélisation de la base de données	9
Méthode MERISE	9
Modèle conceptuel de données (MCD)	9
Modèle Logique de Données (MLD)	10
Modèle Physique de Données (MPD)	11
Extraits de code significatifs	12
Intégration	21
Responsive design	22
Sécurité	24
Faible Include	25
Injection SQL	26
Faible Upload	26
Faible XSS	27
Attaque force brute	27
Conclusion	28

Introduction

Compétences couvertes par le projet

Mon projet professionnel est une boutique en ligne, ou site e-commerce réalisé en trinôme.

Je vais vous présenter les compétences du référentiel que j'ai couvert avec ma partie du développement.

Ce projet m'a permis de couvrir les compétences du référentiel suivantes :

- Activité type 1:

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

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

Les compétences liées à la partie “FRONT” sont abordées au travers de :

- Développement d’un parcours utilisateur qui influence le développement de l’UX/UI
- La réalisation dans un premier temps d’une maquette, puis d’un prototype
- L’intégration de la maquette dans le projet jusqu’à son développement

- Activité type 2:

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

- Créer une base de données
- Développer les composants d’accès aux données
- Développer le back-end d’une application web / web mobile

Les compétences liées à la partie “BACK” sont abordées au travers de :

- La conception et la modélisation de base de données utilisant la méthode **MERISE**.
- L'élaboration d'une base de données fonctionnelle via **MySQL**
- L'utilisation de la **POO**(Programmation Orientée Objet) permettant à l'utilisateur d'interagir avec les données présentes en **BDD** pour l'utilisateur

Résumé du dossier de projet

Ce dossier présente une partie du travail que j'ai effectué au sein de l'élaboration de la boutique en ligne "BikeAccess", qui correspond à la boutique en ligne de la concession de motocyclette fictive du même nom.

Le site web BikeAccess permet à ses utilisateurs de visualiser les articles et véhicules disponibles, triés par ordre de nouveauté ou alors par catégories et sous-catégories ; les catégories correspondant au type d'article (véhicules, accessoires, équipements de protection, vêtements) et les sous-catégories aux marques de ces articles (Harley Davidson, Kawasaki, ...) puis ensuite de passer commande.

Pour ce faire, les fonctions suivantes ont été implémentées : la possibilité de s'inscrire, de se connecter, d'accéder à son profil, de modifier son profil mais aussi d'accéder à son historique de commande.

Le site est fonctionnel sans nécessité d'inscription, mais la possibilité de commander est réservé aux utilisateurs inscrit

Un espace administrateur a été intégré, disposant d'un CRUD (Create, read, update, delete) complet.

Cette boutique en ligne est en quelque sorte l'achèvement de l'année, de par son importance capitale concernant le passage du titre professionnel de développeur web et web mobile.

Elle a été réalisée en trinôme au cours de ma formation à l'école LaPlateforme_.

Cahier des charges

Présentation de l'entreprise

BikeAccess est une concession fictive de motocyclette choisie pour la réalisation de ce projet de site e-commerce.

Ce site web propose un large éventail de produits comme des véhicules, des accessoires, des équipements de protection ou encore des vêtements de différentes grandes marques tels que Harley-Davidson ou encore Kawasaki.

Les besoins clients

L'objectif

L'objectif est de mettre en place une boutique en ligne permettant à la concession de poursuivre et développer son activité commerciale via le web : une transition numérique.

La boutique sera responsive et donc adaptée à tous types d'User Agent.

Les cibles

Les visiteurs : ce qui comprend les utilisateurs non inscrits sur le site, qui pourront consulter les produits et potentiellement devenir de nouveaux utilisateurs, voire de nouveaux clients

Les inscrits : ce qui comprend les utilisateurs qui pourront consulter les produits, avoir accès au panier, prendre commande auprès du site, ainsi que modifier son profil contenant ses informations personnelles et avoir un historique des commandes passées.

les administrateurs : ce qui comprend les utilisateurs avec des droit d'accès spéciaux au site via une combinaison id/pswd prédéfinie : cet administrateur à accès à l'historique des commandes réalisées, à l'ajout,

la suppression, la modification d'un produit, ou encore le changement des droits d'accès d'un utilisateur.

Le périmètre du projet

Le site sera disponible uniquement en français.

Par ailleurs, il devra être adapté à tous les supports (ordinateurs, tablettes, mobiles) pour permettre à tous les utilisateurs d'avoir une expérience de navigation optimale, c'est à dire être Responsive

L'arborescence

Il est capital de modéliser le parcours utilisateur, c'est -à -dire l'ensemble des chemins que peut prendre l'utilisateur connecté à une page donnée.

Le parcours utilisateur est en quelque sorte le circuit qu'emprunte l'utilisateur. Il peut varier selon certains cas de figure ; par ex : en fonction du statut de connexion de l'utilisateur : non-connecté, utilisateur lambda ou administrateur.

La charte graphique

Il est important de définir en amont la charte graphique du projet, c'est -à -dire l'identité visuelle du site ; cela englobe les couleurs, les logos, les polices d'écritures et bien plus encore.

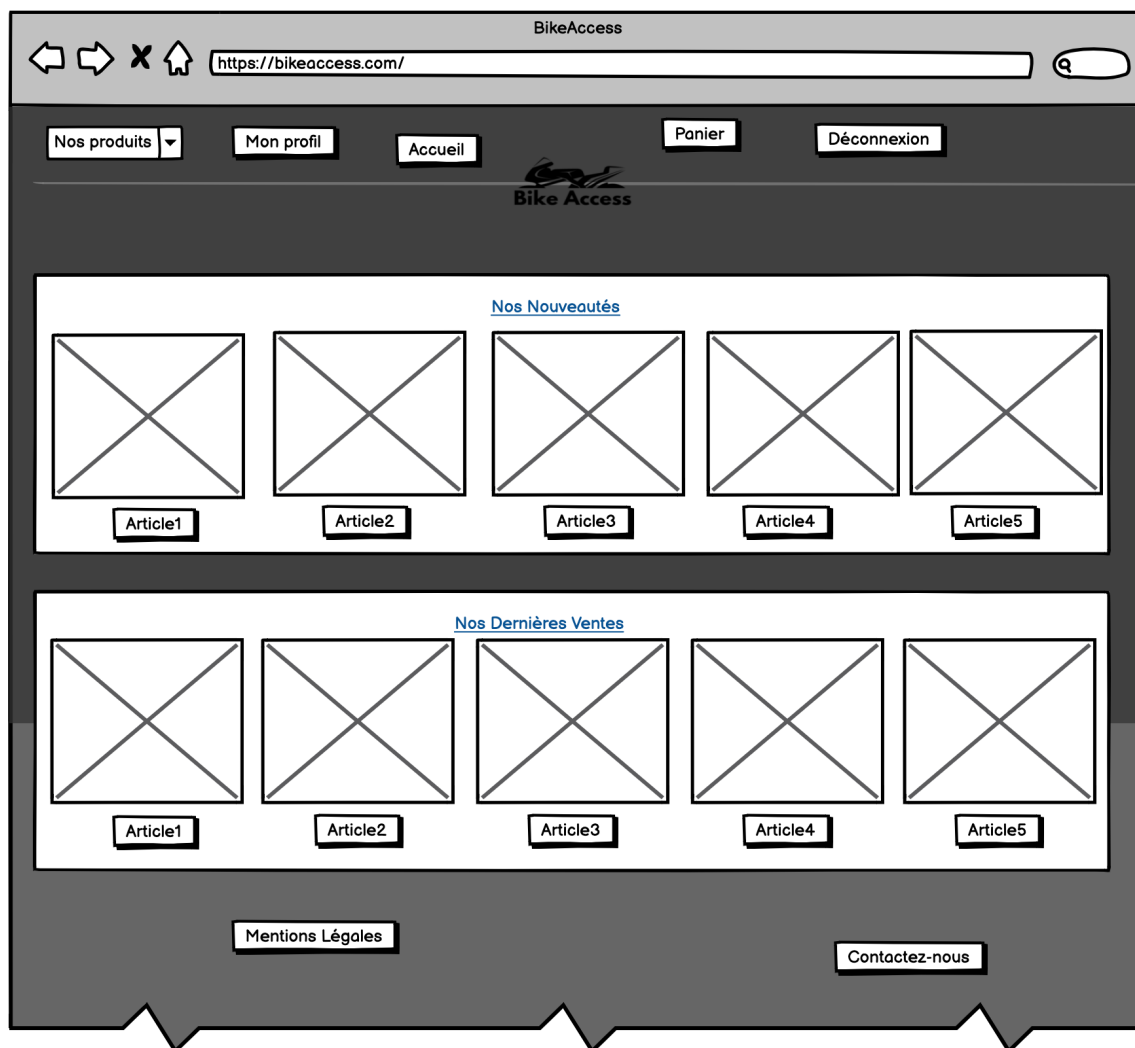
Concernant la police d'écriture, nous avons choisi la police officielle de Moto GP afin de faire un rappel à l'univers moto

Concernant les logos, je m'en suis servi pour habiller les pages du site et faciliter la navigation en associant des repères visuels à certaines actions pour plus de convivialité lors du parcours des utilisateurs.



Concernant le maquettage :

- Zoning
- Wireframe via Balsamiq
- Enfin le Prototype.



le wireframe ci-dessus

Spécificités techniques : Backend

Organisation et background de développement

Technologies :

Le stack choisi pour le développement de ce projet est WAMP, acronyme référénçant la suite logicielle que voilà.

- **W**indows, en tant que système d'exploitation
- **A**pache, en tant que logiciel de serveur web
- **M**ySQL, en tant que système de gestion de bases de données
- **P**HP, en tant que langage de script permettant la communication avec le serveur web et la génération de pages dynamiques.

Workflow :

Pour la gestion du projet et la répartition du travail d'équipe en trinôme, nous avons aidé d'une suite logicielle complète permettant la disponibilité complète du projet ainsi que des tenants et aboutissants à tous les collaborateurs.

En premier lieu, la suite google :

- le Chat pour la communication instantanée et permanente
- le Drive pour le partage de documents ou ressources textuelles et graphiques.
- GMail pour des check-up hebdomadaires et mensuels réguliers.

mais aussi Git Kraken (logiciel desktop) couplé à GitHub

Le travail sur GitHub se fait autour de deux branches principales:

- la branche "Master"
- la branche "Dev".
- une branche par fonctionnalité en cours de développement.

Ensuite un merge de ces branches de fonctionnalités vers la branche développement pour que l'équipe entière vérifie le code.

Nous avons aussi utilisé l'outil collaboratif Trello.

Il repose sur une organisation des projets en planches listant des cartes, chacune représentant des tâches.

Modélisation de la base de données

Méthode MERISE

MERISE (**M**éthode d'**E**tude et de **R**éalisation Informatique pour les **S**ystèmes d'**E**ntreprise) :

C'est une méthode qui est née au cours des années 1970 en France.

Elle permet la modélisation et la conception de Systèmes Informatiques

Parmi les ressources informatiques de ces S.I., il y a en particulier les Fichiers de Données, Bases De Données et Système de Gestion de Bases de Données (S.G.B.D.).

C'est sur les modèles que comportent la méthode MERISE que nous avons développé la Base De Données de BikeAccess.

Modèle conceptuel de données (MCD)

Le Modèle Conceptuel de Données (MCD) est un modèle simplifié de la base de données, dans lequel les tables sont représentés au stade d'entités (ou "**table**" en MCD).

Entre les différentes entités se forment des liaisons que l'on nomme "**association**", qui est le verbe d'action qui décrit l'opération entre les deux entités, donc nous distinguons principalement les entités et les association, d'où l'autre appellation du M.C.D, le "**schéma Entité/Association**".

La conception de la BDD implique la prévision totale des besoins utilisateurs et administrateurs, actuels ou futurs.

De cette analyse découlent les **règles de gestion** des données à conserver.

Vient ensuite la définition du **Dictionnaire de données**, c'est -à -dire la sélection des données à conserver en BDD, et dont certains attributs se retrouveront dans le MCD.

On peut donc passer à la définition des entités et leurs associations entre elles, c'est à dire les **liens** et **cardinalités** entre les entités.

Les **cardinalités** possibles sont :

- **0,1**: au minimum 0, au maximum 1 seule valeur (CIF) ;

- **1,1** : au minimum 1, au maximum 1 seule valeur (CIF) ; par exemple, une commande passée peut ne l'avoir été que par au minimum 1 utilisateur et au maximum 1 seul
- **0,n** : au minimum 0, au maximum plusieurs valeurs ;
- **1,n** : au minimum 1, au maximum plusieurs valeurs. Par exemple, une catégorie pour exister doit contenir au moins 1 article mais peut aussi en contenir une infinité.

Les cardinalités sont importantes car elles traduisent les règles de gestion, elles doivent être validées avec l'utilisateur final.

Modèle Logique de données (MLD)

Le modèle logique de données (MLD), est une étape intermédiaire entre le modèle conceptuel de données et le modèle physique de données.

Le passage d'un MCD en MLD s'effectue selon quelques règles de conversion précise :

Une **entité** du M.C.D devient une **table** :

Dans un SGBD de type relationnel, une table est une structure tabulaire :

- chaque ligne correspond aux données d'un objet enregistré
- Chaque colonne correspond à une propriété de cet objet.

Ces colonnes font notamment référence aux caractéristiques définies dans le **dictionnaire de données** du M.C.D.

- Les **identifiants respectifs** de chaque entité deviennent des **clefs primaires**.
Les **clefs primaires** permettent d'identifier de façon unique chaque enregistrement dans une table et ne peuvent pas avoir de valeur nulle.
- toutes les autres propriétés définies dans le MCD deviennent des **attributs**.

Les cardinalités de type "0:n" / "1:n" sont représentées à travers le référencement de la clef primaire de la table qui la possède, en clef étrangère au sein de la table auquel elle est liée.

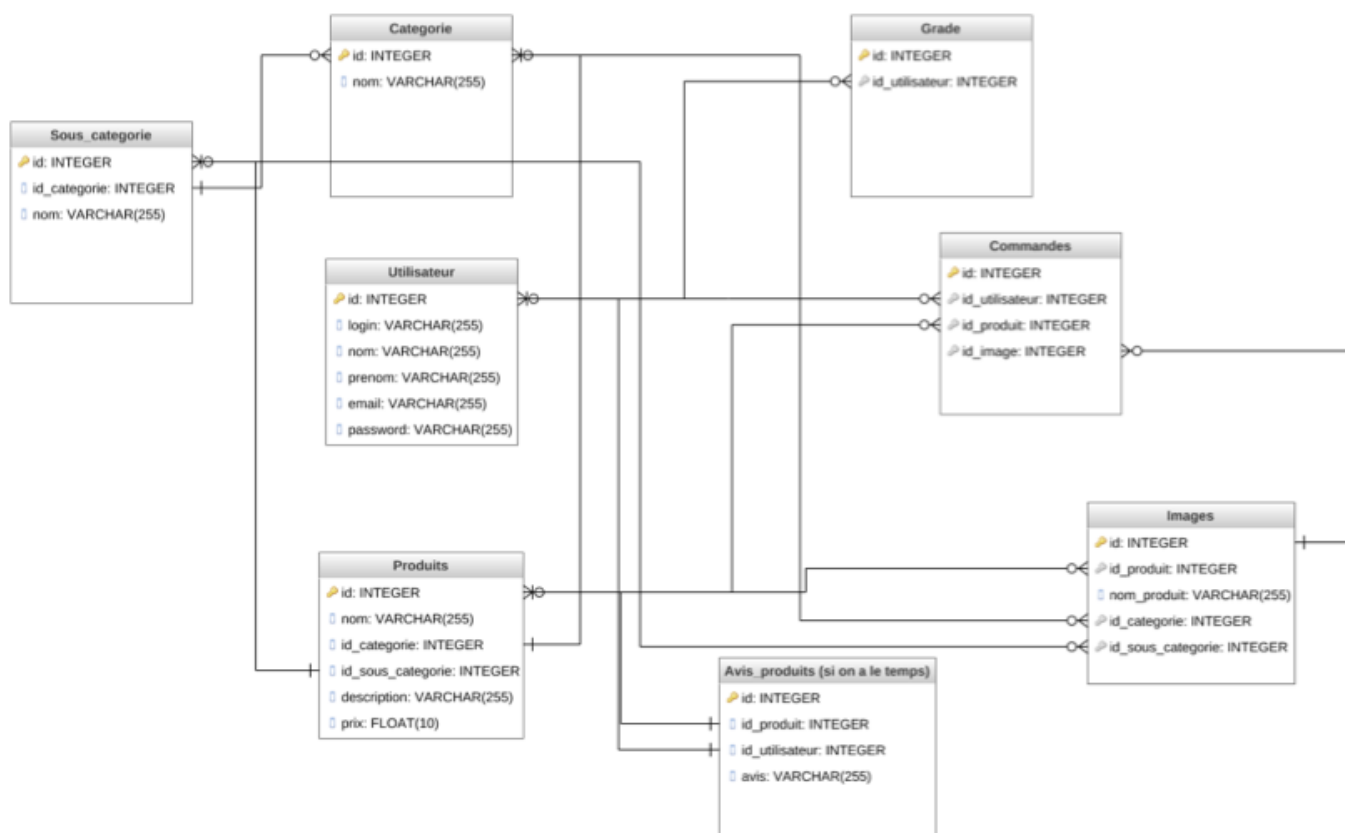
Si deux tables liées possèdent une cardinalité de type "0:n / 1:n", la relation sera traduite par la création d'une table de jonction, dont la clef primaire de chaque table deviendra une clef étrangère au sein de la **table de jonction**.

Modèle physique de données (MPD)

Le MPD correspond à l'implémentation des modèles générés précédemment dans le Système de Gestion de Base de Données (SGBD) utilisé pour le projet.

C'est en l'occurrence **MySQL** couplé au moteur de stockage **InnoDB**, dont la particularité est la gestion de la contrainte des clefs étrangères en BDD.

Voici une capture d'écran du M.P.D du projet "Boutique en Ligne"



Extraits de code significatif

Toutes les requêtes en base de données sont gérées par 5 classes différentes : **"userpdo"**, **"produit"**, **"panier"**, **"catégorie"**, **"commentaire"**. Ces classes possèdent des méthodes permettant d'exécuter des requêtes préparées.

L'utilisation de requêtes préparées, rendues possibles grâce à l'extension **PDO(PHP Data Objects)**, présente plusieurs avantages par rapport à l'exécution directe de **requêtes SQL**.

- Une seule préparation est nécessaire afin de les réutiliser à souhait, avec des paramètres identiques ou bien différents.
- Il existe un avantage majeur en termes de sécurité pour prévenir des injections SQL.

Nos requêtes étant pré-formatées, elles empêchent le passage de code malicieux en paramètre ; nous n'avons donc pas besoin de protéger nos paramètres ou valeurs manuellement.

Création d'une classe

Je commence par créer ma classe et y ajouter les attributs désirés selon les modèles précédemment cités.

```
// PARTIE UTILISATEUR

class userpdo
{
    private $id;
    public $login;
    public $nom;
    public $prenom;
    public $email;
    public $password;
    public $password2;
    public $grade;
```

Un **attribut** est une **variable** qui est partagée par toutes les instances de la **classe** : on peut y stocker par exemples les paramètres de la connexion à la base de données pour ne pas avoir à les réécrire à chaque fois que l'on y fait appel.

Inclusion des fonctions :

Préalablement à l'appel aux fonctions sur une page, il faut que je j'inclus mon fichier contenant mes classes dans le code de la page où je souhaite que mes fonctions soient disponibles.

Je dois ensuite instancier la classe qui contient les fonctions auxquelles je dois faire appel :

```
<?php
session_start();
include("../function/fonctions.php");
$panier = new panier();
$panier->dbconnect();
?>
```

Connexion à la BDD:

4 paramètres sont attendus :

- le nom du serveur, ici localhost lors du développement local
- le nom de la BDD, ici 'boutique'
- l'identifiant
- le mot de passe

Concernant la connexion à la base de données, elle attend quatre paramètres, le nom du serveur, la en l'occurrence localhost vu que je travaille en local, le nom de ma base de données qui est "boutique", l'identifiant et le mot de passe.

ensuite je le stocke dans un attribut pour pouvoir faire appel dans toutes mes autres fonctions.

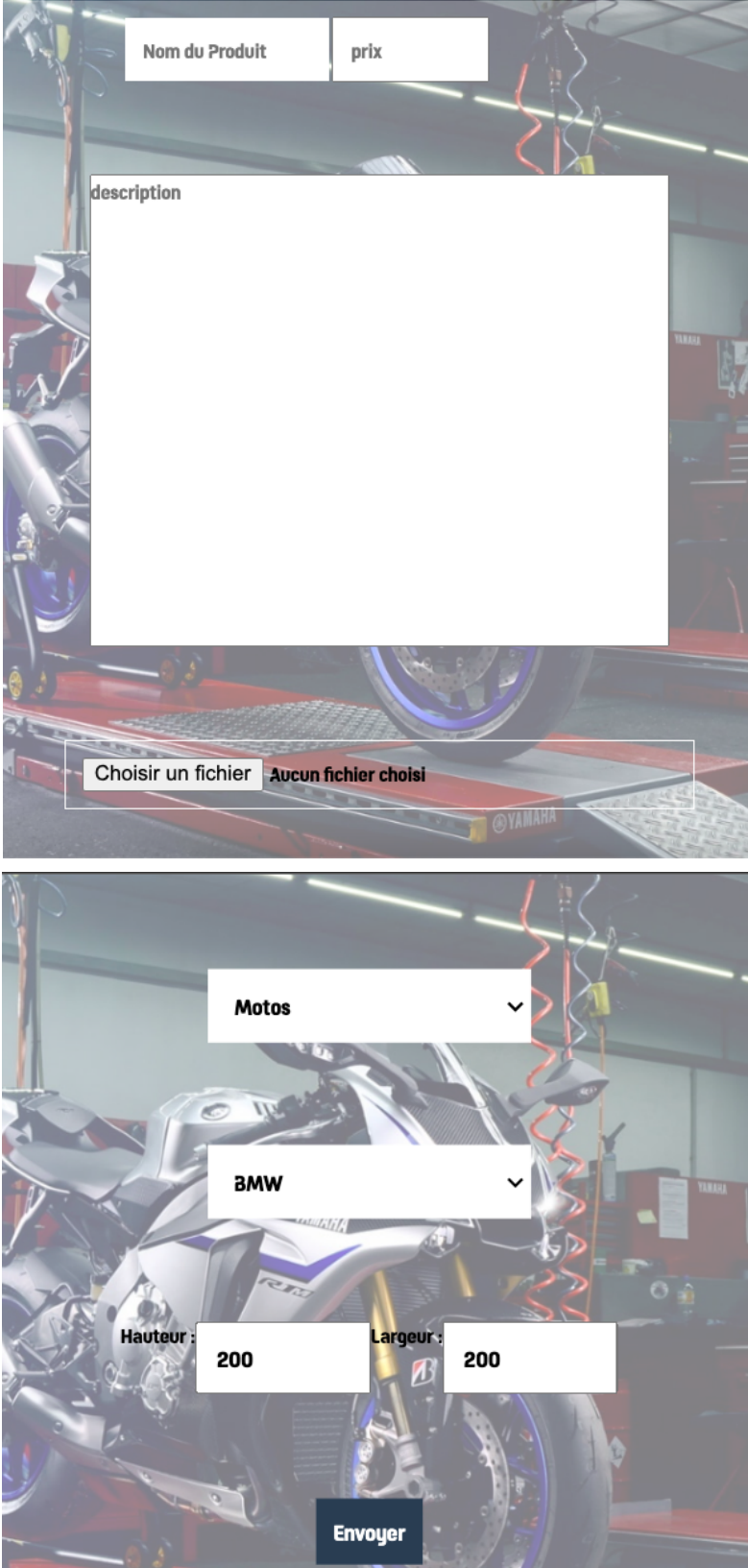
```
public function dbconnect()
{
    $db = new PDO("mysql:host=localhost; dbname=boutique", 'root', '');
    $this->_db = $db;
}
```

L'espace administrateur:

Les quatre principales opérations pour le stockage d'informations en base de données sont la création, l'affichage, la mise à jour et la suppression de données; correspondant à l'acronyme anglophone **C.R.U.D** (Create, Read, Update, Delete).

Create:

La requête SQL "INSERT", pour insérer des éléments en base de données. Nous nous en sommes servi pour insérer en base de données les articles que nous souhaitions ajouter au catalogue. Je récupère les données en méthode 'POST' dans le formulaire correspondant :



Nom du Produit

prix

description

Choisir un fichier

Aucun fichier choisi

Motos

BMW

Hauteur : 200

Largeur : 200

Envoyer

En voici une partie du code HTML :

```

?>
<div class="adminform">
  <form action="administration.php" method="post" enctype="multipart/form-data">
    <div class="form">
      <input name="nom" required type="text" placeholder="Nom du Produit">
      <input name="prix" required type="number" min="0.00" max="10000.00" step="0.01" placeholder="prix"
    </div>
    <textarea name="description" required placeholder="description"></textarea>
    <input required type="file" name="fileToUpload" id="fileToUpload">
    <select name="categorie">
      <?php
        for($i=0; $i < sizeof($stab); $i++)
        {
          ?>
          <option value="<?php echo $stab[$i][0];?>"><?php echo $stab[$i][1];?></option>
          <?php
          }
          ?>
        </select>
      <select name="sous_categorie">
        <?php
          for($i=0; $i < sizeof($stab1); $i++)
          {
            ?>
            <option value="<?php echo $stab1[$i][0];?> "><?php echo $stab1[$i][1];?></option>
            <?php
            }
            ?>
          }
        </select>
      </div>
    </form>
  </div>

```

La fonction permettant l'ajout effectif d'articles en base de données a été nommée **"insert_produits"**

Là voici :

```

public function insert_produits($nom, $categorie, $sous_categorie, $description, $prix, $image, $hauteur, $largeur)
{
  $existname=$this->connectdb()->query("SELECT *FROM produits WHERE nom = '$nom'");
  $value = $existname->rowCount();

  if($value ==0)
  {
    $desc=str_replace ( '"', '""', $description);
    $insert_produits=$this->connectdb()->query("INSERT INTO produits VALUES(NULL, '$nom','$categorie','$sous_categorie','$desc','$prix')");
    $id_produit=$this->connectdb()->query("SELECT id FROM `produits` ORDER by id DESC");
    $id=$id_produit->fetch();

    $numid=$id['id'];
    $insert_images=$this->connectdb()->query("INSERT INTO images VALUES(NULL, '$numid','$image','$hauteur','$largeur')");
    $exist="Produit bien rajouté";
  }
  else
  {
    $exist="Produit déjà existant";
  }

  return $exist;
}

```


- Je déclare la fonction en public afin d'y avoir accès en dehors de la classe, là où je vais effectivement instancier la fonction,
- Je vérifie si le nom choisi pour l'article existe déjà en base de données ; s'il n'existe pas d'occurrence, la variable \$value est égale à zéro et la fonction continue ; S'il existe déjà, le programme affiche "produit déjà existant"
- Je lance la requête **SQL "INSERT"**, je précise la table "produits" ainsi que la colonne dans laquelle chaque valeur doit être insérée.
- Je stocke la requête dans une variable qui devient un paramètre de la fonction **PHP "QUERY"**
- C'est cette fonction QUERY qui effectue l'insertion des data en BDD
- Le message de confirmation "produit bien ajouté" apparaît à l'écran.
- Il faut ensuite appeler la fonction dans la page où on souhaite l'utiliser.

Dans l'exemple qui va suivre, la fonction se lance à la condition que le bouton d'ajout soit enclenché :

```
if(isset($_POST['ajout_produit']))
{
    ?>
    <div class="uploadad">
        <?php include("upload.php");?>
    </div>
    <?php

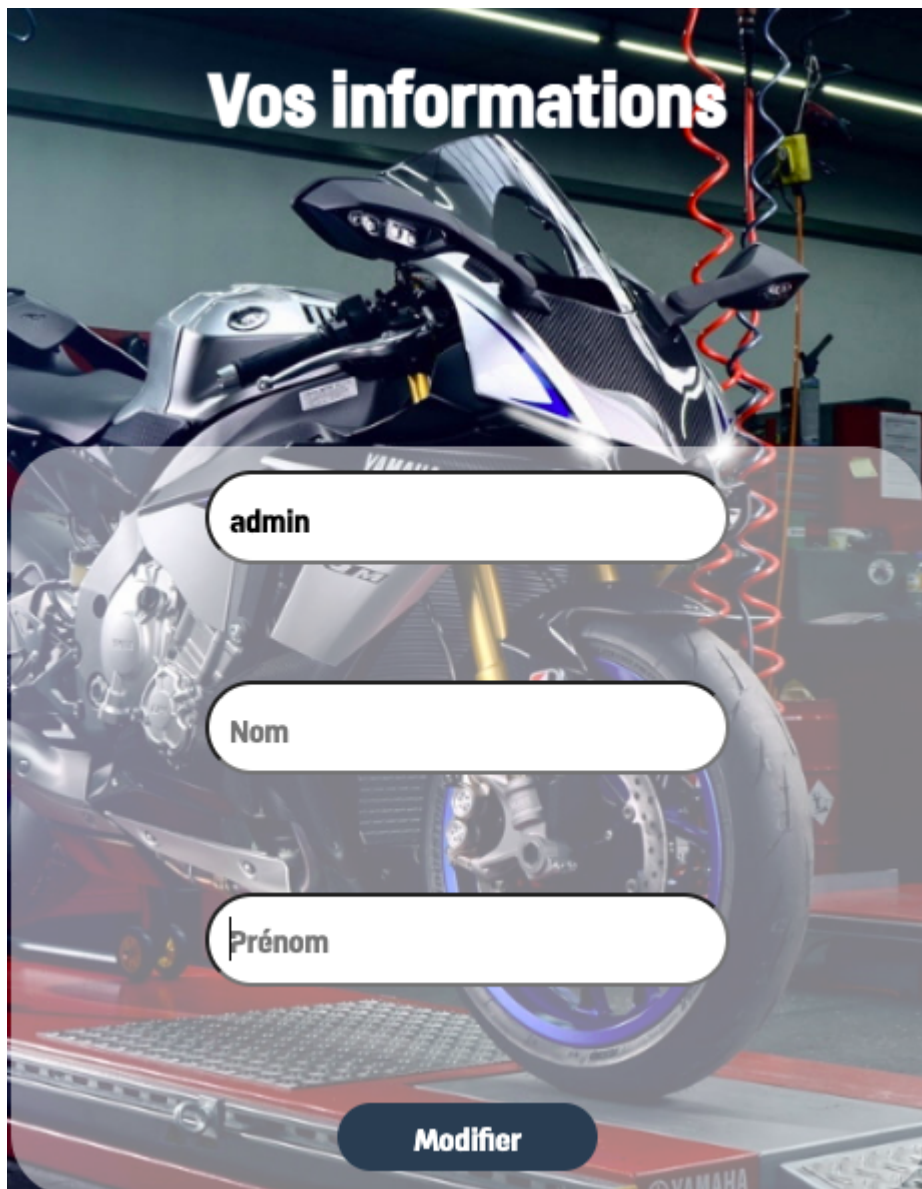
    $chemin="../img/".$name;
    $_POST['image']= $chemin;

    $produit ->insert_produits($_POST['nom'], $_POST['categorie'], $_POST['sous_categorie'],
```

Update :

L'update est gérée par la requête SQL du même nom, à savoir "UPDATE". J'ai utilisé cette fonction dans ma boutique notamment pour permettre à l'utilisateur de modifier son identité, à savoir nom & prénom.

Voici mon formulaire.



Vos informations

admin

Nom

Prénom

Modifier

et voici la fonction correspondante :

```
public function modifinfo($nom, $prenom)
{
    if(!empty($nom) && !empty($prenom)){
        $db = $this->_db;
        $id = $_SESSION['id'];
        $requete = $db->prepare("UPDATE utilisateurs SET nom='$nom' , prenom='$prenom' WHERE id=$id");
        $requete->execute();
        $_SESSION['nom'] = $nom;
        $_SESSION['prenom'] = $prenom;
        $msg = 'Informations mise a jours ! ';
    }else{
        $msg = 'Veuillez remplir les champs';
    }
}

echo $msg;
```

les étapes sont les suivantes :

- Je récupère l'attribut qui contient la connexion à la BDD pour que je puisse communiquer avec et lui donner des instructions,
- Je récupère l'id de l'utilisateur connecté qui est stocké dans une variable dite **Superglobale** ce qui signifie qu'elles sont disponibles quel que soit le contexte du script, en l'occurrence c'est la variable **"\$_SESSION"** qui est utilisée pour stocker des **tableaux associatifs**.
- je rédige ma requête préparé, comportant instruction de mettre à jour le nom et le prénom, par quelles valeurs les remplacer et quel élément cibler via l'id
- Je l'exécute et je remplace les valeurs stockées dans ma variable superglobale par les nouvelles données que l'utilisateur a entrées dans le formulaire, puis un message de confirmation.
- l'utilisateur pourra ensuite voir sur son profil que les informations ont bien été modifiées.

Delete:

L'opération Delete est gérée par la fonction SQL du même nom.

Nous avons notamment utilisé la fonction Delete pour l'annulation d'une commande par le client, ce qui procède à une suppression des données de la commande en BDD.

La fonction est la suivante : "delete_commande"

```
public function delete_commande($id)
{
    $delete=$this->connectdb()->query("DELETE FROM commande WHERE id='$id' ORDER by id ASC");
}
```

Après exécution, la commande correspondant à l'id sélectionnée par l'utilisateur va être supprimée de son dashboard de commande.

Produit acheté	Nom du produit	Quantité	Prix	Date d'achat	Adresse de livraison	Annuler ma commande
	BMW 1250 GS	1	835 €	Friday 06 March 2020 à 18:03:39	43000 Avenue DesFleurs	Délai dépassé (24h)
	BMW S 1000 XR	1	750 €	Sunday 08 March 2020 à 18:15:30	43000 Avenue DesFleurs	Délai dépassé (24h)
	Casque Moto BMW System 7 Carbon	1	550 €	Monday 09 March 2020 à 19:43:44	une adresse de livraison par défaut	Délai dépassé (24h)

Prix total : 2135 €

Intégration

L'intégration de la maquette a été faite en utilisant le langage de balisage HTML. C'est un langage limité à la création de représentations structurées des pages.

l'utilisation du langage CSS est donc nécessaire à la concrétisation du style graphique en accord avec la charte graphique pré établie.

Je le fais par le biais de feuilles de style personnalisées propres à chaque page.

C'est souvent le cas sur les projets qui sont plus que des sites OnePage.

Cette technique a l'avantage d'offrir une maintenance plus aisée et agréable du code, car les modifications apportées impactent seulement la page liée à la feuille de style concernée.

Pour faire correspondre du code CSS à du code HTML, il faut lui attribuer une **"Class"**

Cette "Class" CSS permet de viser des balises HTML, c'est-à-dire d'apporter ses modifications graphiques seulement sur les extraits de HTML concernés et ciblés.

Voici le **lien** : le chemin du fichier CSS inscrit dans le fichier HTML dans lequel je désire ajouter du style :

cela se fait avec une balise **<link>** à l'intérieur de la balise **<head>**

```
<link rel="stylesheet" href="../../style/css/boutique.css">
```

Après avoir fait cela, je donne la "CLASS" à n'importe quelle balise HTML et lui fait correspondre du style dans le fichier CSS correspondant.

```
<div class='titrecauserouge'>  
    Ajouter un article  
</div>
```

```
.titrecaserouge {  
  color: ■ #F7EBE8;  
  font-size: 20PX;  
  font-family: 'Poppins', sans-serif;  
  font-weight: bolder;  
}
```

Responsive design

L'adaptation d'un site web à tout type de support est capital tant les moyens d'accès au web sont aujourd'hui divers et variés.

Ces derniers regroupent principalement les ordinateurs, tablettes et smartphones.

Nous avons utilisé des **Media Queries**, un module CSS3 permettant d'adapter le contenu d'une page web aux caractéristiques de l'appareil de l'utilisateur.

En appelant les classes que je veux cibler et en définissant les largeurs d'écran maximal ou minimal, je peux facilement adapter le contenu affiché en changeant sa taille, masquer ou afficher du contenu ciblé. En voici un exemple, dont le seuil d'adaptation se situe à 860 pixels de largeur.

```

/* //////////////////////////////////////// RESPONSIVE PAGE CONNEXION INSCRIPTION */
@media screen and (max-width: 860px) {
  .caseinscription {
    display: flex;
    align-items: center;
    justify-content: center;
  }
  .caseconnexion {
    display: flex;
    align-items: center;
    justify-content: center;
  }
  .buttonconnexion {
    margin-bottom: 100px;
  }
  .caseconnexioninscription {
    flex-direction: column;
  }
  .connexionsepareinscription {
    display: none;
  }
}

```

Sécurité

Une veille concernant les failles de sécurité web a été faite durant la réalisation du site.

J'ai parcouru le web afin de recouper des informations sur les failles web les plus connues.

Voici une liste non-exhaustive des sources utilisées :

<https://www.cert.ssi.gouv.fr/>

<https://vigilance.fr/?langue=1>

<https://security.stackexchange.com>

<https://owasp.org/>

Faillle INCLUDE

La faille INCLUDE exploite une mauvaise sécurisation de la fonction Include.

La fonction PHP include permet généralement de n'avoir qu'une seule page mysql.php, et toutes les pages du site peuvent l'inclure via cette fonction.

Cette faille consiste généralement en l'utilisation de la fonction Include afin d'exécuter du code PHP malveillant se situant dans une page tierce, et permettant un accès à la base de données du site .

Il existe deux types de failles include :

A distance : C'est l'inclusion via une URL distante d'un code PHP malveillant.

En local : utilisation de la fonction include afin de naviguer l'arborescence du serveur. Il y a possibilité de trouver par exemple le fichier contenant les mots de passe en crypté, qui seront aisément décryptés en local par les hackers à l'aide de logiciels adéquats.

Pour se protéger de cette faille, rien de plus simple que de la tester ! Il vous suffit d'inclure une page qui n'existe pas manuellement dans chacune des pages disponibles de votre site.

Si l'URL de celle-ci est vulnérable, un message d'erreur vous sera transmis venant de PHP.

Injections SQL

Les attaques par injection de commandes SQL exploitent les failles de sécurité d'une application qui interagit avec des bases de données.

L'attaque SQL consiste à modifier une requête SQL en cours par l'injection d'un morceau de requête non prévu, souvent par le biais d'un formulaire non sécurisé.

Le hacker peut ainsi accéder à la base de données, mais aussi modifier le contenu et donc compromettre la sécurité du système.

Cette manipulation peut donner accès à des données sensibles telles que les login, mots de passe ou adresses e-mail.

Faillle UPLOAD

La faille UPLOAD est une vulnérabilité inhérente à l'autorisation d'upload de fichiers sur un site web.

Le principe de l'attaque est très simple : le pirate essaie d'uploader un fichier qui contient du code malveillant.

Si la faille est ouverte, alors le fichier finira par atterrir sur le serveur.

Il suffit ensuite au pirate d'appeler son fichier pour que celui-ci s'exécute.

Bien entendu une telle attaque peut avoir de graves conséquences comme par exemple:

- l'espionnage des fichiers et dossiers du site.
- l'accès aux fichiers systèmes et fichiers confidentiels.
- la destruction ou altération de données existantes sur le serveur.
- la perte de contrôle du serveur.

Les moyens de s'en prémunir sont les suivants :

- Interdire l'exécution de code depuis le dossier dans lequel sont stockés les fichiers uploadés sur votre site, une zone de quarantaine.
- Vérifier et autoriser l'extension des fichiers que vous tolérez via une liste blanche, en vérifiant rigoureusement le type **MIME** (Multipurpose Internet Mail Extensions) de chaque fichier uploadé sur le serveur.

Faillle XSS

Une faille **XSS** (pour Cross Scripting) consiste en l'injection de code qui pourra être interprété directement par le navigateur Web.

Ce dernier ne fera ainsi aucune différence entre le code du site et celui injecté par le pirate.

Les conséquences peuvent être :

- Des redirections malveillantes vers un autre site, de type phishing.
- Le vol de cookies.
- Une modification du code de votre page.

Dans une optique de protection, la meilleure solution est le remplacement des caractères pouvant être compris par le navigateur comme des balises par leur entité HTML.

En procédant ainsi, le navigateur affichera mot à mot le caractère et ne cherchera plus à l'interpréter.

En PHP, vous pouvez utiliser notamment les fonctions `htmlspecialchars` ou `htmlspecialchars_decode` qui effectuent ce remplacement.

ATTAQUE PAR FORCE BRUTE

Cette méthode consiste à deviner le mot de passe ou la clé cryptographique correspondant à ce mot de passe, puis ensuite la décrypter.

Il est donc capital, à l'heure du tout numérique, d'utiliser des mots de passe forts (c'est-à-dire complexes et difficilement déchiffrables) afin de complexifier la tâche des pirates informatiques utilisant les attaques par force brute.

Voici 3 consignes afin de renforcer un mot de passe :

- Mélanger l'utilisation de lettres minuscules et majuscules, de chiffres et de caractères spéciaux (notez qu'il existe des générateurs automatiques de mots de passe sur internet)
- Renouveler fréquemment ses mots de passe, même sans alerte.
- Utiliser des mots de passe différents pour chaque site, chaque service, sur chaque appareil.

Conclusion

Ce projet est l'aboutissement du lot de connaissances acquises au cours de mon année scolaire passée à LaPlateforme_.

Plus que de simples connaissances brutes, ce sont aussi des méthodes de travail et de fonctionnement que j'ai acquies, que ce soit pour le travail individuel ou en équipe, que ce soit dans l'interaction avec mes pairs ou les instructeurs dans la résolution de problèmes et dans la poursuite de recherches, en différents langages de programmation, mais aussi en différentes langues au fil des pérégrinations sur le Web.

J'y ai aussi appris une certaine humilité nécessaire à la progression, afin de ne pas persévérer dans des chemins stériles et savoir me réorienter à temps afin de toujours rentabiliser mon temps et mon énergie.

J'y ai surtout appris, comme cette présentation le démontre, la gestion d'un projet de développement web, de sa conception jusqu'à la production de sa mouture finale.

J'ai participé à la conception du projet, à son maquettage, autant dans sa partie Front-End que Back-End puis j'ai utilisé les hard skills acquis à LaPlateforme_ afin de mettre en œuvre techniquement le projet.

J'ai donc désormais, je l'espère, une certaine maîtrise des différentes technologies étudiées au cours de l'année comme l'HTML, le CSS, le SQL, le PHP ainsi que le Javascript.