

PROJET 3 CRUD BACK-END PRODUIT

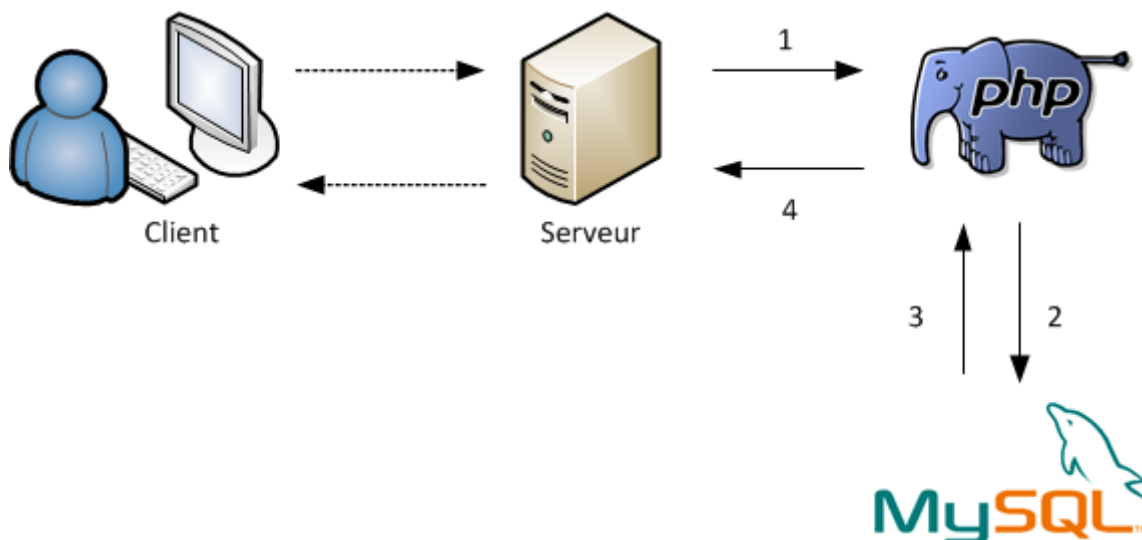
RAPPEL :

- Une base de données est un outil qui stocke vos données de manière organisée et vous permet de les retrouver facilement par la suite.
- On communique avec MySQL grâce au langage SQL. Ce langage est commun à tous les systèmes de gestion de base de données (avec quelques petites différences néanmoins pour certaines fonctionnalités plus avancées).
- PHP fait l'intermédiaire entre vous et MySQL.
- Une base de données contient plusieurs tables.
- Chaque table est un tableau où les colonnes sont appelées « champs » et les lignes « entrées ».

PHPMYADMIN :

- <https://openclassrooms.com/fr/courses/918836-concevez-votre-site-web-avec-php-et-mysql/913893-phpmyadmin>
- phpMyAdmin est un outil qui nous permet de visualiser rapidement l'état de notre base de données et de la modifier, sans avoir à écrire de requêtes SQL.
- On crée généralement un champ nommé id qui sert à numéroter les entrées d'une table. Ce champ doit avoir un index PRIMARY (on dit qu'on crée une *clé primaire*) et l'option AUTO_INCREMENT qui permet de laisser MySQL gérer la numérotation.
- MySQL gère différents types de données pour ses champs, à la manière de PHP. On trouve des types adaptés au stockage de nombres, de textes, de dates, etc.
- phpMyAdmin possède un outil d'importation et d'exportation des tables qui nous permettra notamment d'envoyer notre base de données sur Internet lorsque nous mettrons notre site en ligne.

LE MODELE CONCEPTUEL DE DONNEES :



SQL BRUT:

```
-- phpMyAdmin SQL Dump
-- version 5.0.2
-- https://www.phpmyadmin.net/
--
-- Hôte : 127.0.0.1:3306
-- Généré le : mer. 11 nov. 2020 à 10:05
-- Version du serveur : 5.7.31
-- Version de PHP : 7.3.21

SET SQL_MODE = "NO_AUTO_VALUE_ON_ZERO";
START TRANSACTION;
SET time_zone = "+00:00";

/*!40101 SET @OLD_CHARACTER_SET_CLIENT=@@CHARACTER_SET_CLIENT */;
/*!40101 SET @OLD_CHARACTER_SET_RESULTS=@@CHARACTER_SET_RESULTS */;
/*!40101 SET @OLD_COLLATION_CONNECTION=@@COLLATION_CONNECTION */;
/*!40101 SET NAMES utf8mb4 */;

--
-- Base de données : `multi`
--

--
-- Structure de la table `cd`
--

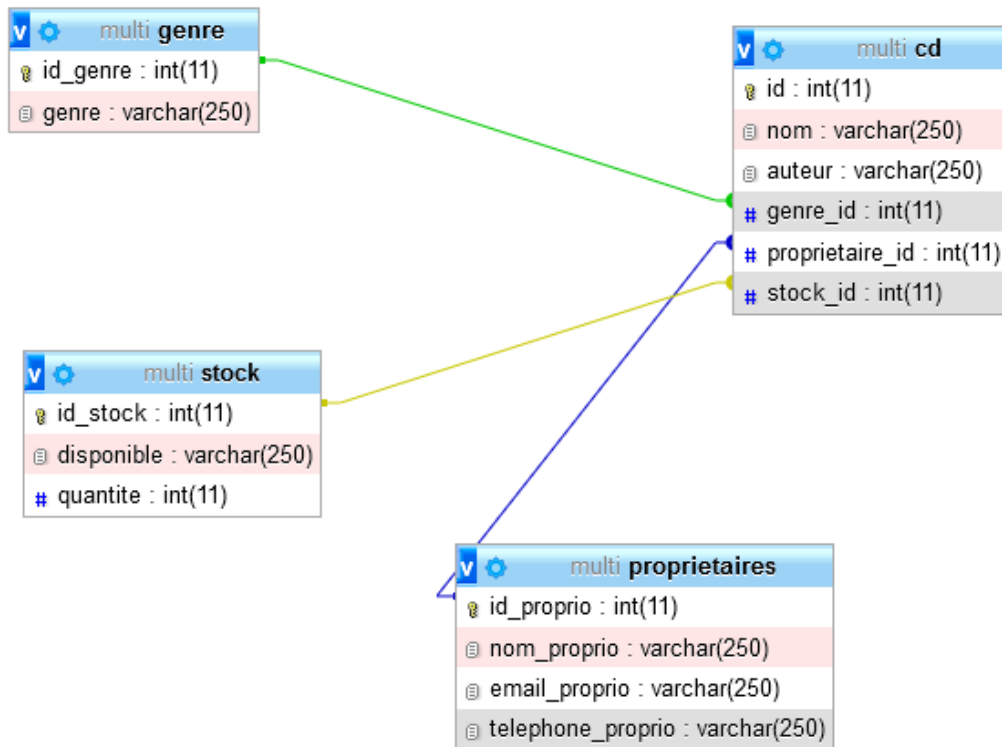
DROP TABLE IF EXISTS `cd`;
CREATE TABLE IF NOT EXISTS `cd` (
  `id` int(11) NOT NULL AUTO_INCREMENT,
  `nom` varchar(250) NOT NULL,
  `auteur` varchar(250) NOT NULL,
  PRIMARY KEY (`id`)
) ENGINE=InnoDB AUTO_INCREMENT=6 DEFAULT CHARSET=latin1;

--
-- Déchargement des données de la table `cd`
--

INSERT INTO `cd` (`id`, `nom`, `auteur`) VALUES
(3, 'Machine Head', 'DeepPurple'),
(4, 'France Afrique', 'Tiken Jha Fakoly'),
(5, 'Premi res R colte', 'Sins milia');
COMMIT;

/*!40101 SET CHARACTER_SET_CLIENT=@OLD_CHARACTER_SET_CLIENT */;
/*!40101 SET CHARACTER_SET_RESULTS=@OLD_CHARACTER_SET_RESULTS */;
/*!40101 SET COLLATION_CONNECTION=@OLD_COLLATION_CONNECTION */;
```

Exemple de MCD avec des clés étrangère (Jointures):



Ressources :

<https://github.com/michelonlineformapro/Projet3-CRUD-BASE-ECOMMERCE>

LES + :

<https://github.com/michelonlineformapro/Projet-3-B-CRUD-Ampoules>

<https://github.com/michelonlineformapro/Projet-2-BIS-SQL-Crud-Sans-Jointure>

<https://github.com/michelonlineformapro/CRUD-Jointures>

<https://github.com/michelonlineformapro/Projet-3-PHP-CRUD-JOINTURES-TWIG>

<https://github.com/michelonlineformapro/Projet-2-PHP--SESSION-NO-DB>

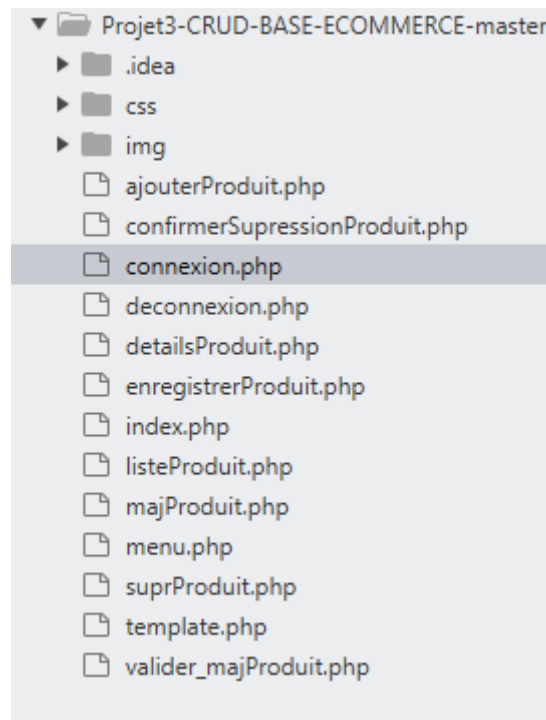
<https://github.com/michelonlineformapro/Projet2-Mini-projets-d-couverte-SQL>

Objectif :

Comprendre et utilisé phpMyAdmin et réalisé le 4 opérations de base d'un site web dynamique CRUD (**create, read, update, delete**) avec PHP.

Le site sera composé de :

- Un page d'accueil **index.php**
- Une page contenant tous les produits (**listeProduits.php**)
- Une page permettant d'ajouté un produits (**ajouteProduit.php**)
- Une page de détail du produits (**detailsproduit.php**)
- Une page de mise à jour d'un produit (**majProduit.php**)
- Une page se suppression d'un produits (**suprProduit.php**)



1 / Première étapes

- Créer une base de données appelée ecommerce

2/ Seconde étapes

- Créer une table « **produits** » avec les champs suivant
- Id_produit (**Int primary key Auto incrément**)
- nom_produit (**varchar 250**)
- description_produit (**text**)
- image_produit (**varchar 250**)
- prix_produit (**float**)

3/ Une connexion a PDO

```
//COONEXION A LE BASE de DONNÉES
//Stock des valeur nom utilisateur phpmyadmin et mot de passe
$user = "root";
$pass = "";
//Essaie de te connecter
try{
    //Stockage et instance de la classe PDO pour connecter php et mysql
    $db = new PDO("mysql:host=localhost;dbname=ecommerce;charset=utf8", $user, $pass);
    //Fonction static de la classe PDO pour debug la connexion en cas d'erreur
    $db->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
    echo "<p class='alert-success p-5'>Salut c bon t connecté a PDO MySQL</p>";
}catch(PDOException $exception){
    die("Erreur de connexion a PDO MySQL :". $exception->getMessage());
}
```

3/BIS Une connexion en dur :

- Créer un système de connexion en dur
- Un formulaire de connexion avec des identifiants dans votre fichier php de traitement

```
<?php
$title = "Ecommerce - ACCEUIL -";
ob_start();
?>

<h1 class="text-danger text-center">BAZAR.COM</h1>
<div class="text-center" id="login-form">
    <form action="connexion.php" method="post">
        <div class="form-group">
            <label for="email">Votre email</label>
            <!-- ici l'attribut name = "" va etre recupérer par la variable super global $_POST['']-->
            <input type="email" id="email" class="form-control" name="email">
        </div>

        <!-- Le champ password-->
        <div class="form-group">
            <label for="password">Votre mot de passe</label>
            <!-- ici l'attribut name = "" va etre recupérer par la variable super global $_POST['']-->
            <input type="password" class="form-control" id="password" name="password">
        </div>

        <!-- le bouton appel <form action="" le fichier connexion.php-->
        <button type="submit" class="btn btn-outline-info mt-5">CONNEXION</button>
    </form>
</div>

<?php
//var_dump($_POST['email']);
//var_dump($_POST['password']);
$content = ob_get_clean();
require "template.php";
```

Traitement :

```

<?php
ob_start();
//Page de traitement de la connexion
//Creer un faux utilisateur
$email_valide = "admin@admin.com";
$mot_de_passe_valide = "admin";

//Verification de l'existence des variables et le champ n'est pas vide
if(isset($_POST['email']) && !empty($_POST['email']) && isset($_POST['password']) && !empty($_POST['password'])){
    //Test de debug
    var_dump($_POST['email']);
    var_dump($_POST['password']);
    //La condition est reussie on assigne l'element posté a la fausse variable
    if($email_valide == $_POST['email'] && $mot_de_passe_valide == $_POST['password']){
        //La connexion est réussie et on affiche la page listeProduit.php
        session_start();
        //On utilise la super glogale $_SESSION[''] qui est egale au element valide ($email_valide et $mot_de_passe_valide)
        $_SESSION['email'] = $_POST['email'];
        $_SESSION['password'] = $_POST['password'];
        //Redirection en php mettre url entière
        header('Location: http://localhost/Ecommerce/listeProduit.php');
    }else{
        //Si un champ est vide ou email mot de passe invalide
        echo "<p class='alert-danger p-5'>Erreur votre email ou mot de passe sont invalide ou des sont vide</p>";
        echo "<a href='index.php' class='btn btn-danger'>Retour</a>";
    }
}else{
    //Si un champ est vide ou email mot de passe invalide
    echo "<p class='alert-warning p-5'>ERREUR : Merci de rentrer un email et mot de passe valide ou de remplir tous les champ</p>";
    echo "<a href='index.php' class='btn btn-danger'>Retour</a>";
}

//Debug de session
//var_dump($_SESSION['email']);
//var_dump($_SESSION['password']);

//Le contenu de template.php definis ce qui se trouve dans le body
//Retourne le contenu du tampon de sortie et termine la session de temporisation.
//Si la temporisation n'est pas activée, alors false sera retourné.
$content = ob_get_clean();
//Rappel du template sur chaque page
require "template.php";

```

BAZAR.COM

Votre email

Votre mot de passe

CONNEXION

4/ Lister tous les produits




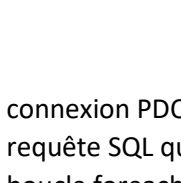
Salut c bon t connecté a PDO MySQL

Votre espace d'administration : Bonjour admin@admin.com

Deconnexion

BAZAR.COM

AJOUTER UN PRODUIT

Nom	Image	Description	Prix	Détails	Mettre à jour	Supprimer
Cendrier		Cendrier de cuba	7845.25 €	Détails du produits	Mettre à jour le produits	Supprimer le produits
chaise		chaise en bois	15 €	Détails du produits	Mettre à jour le produits	Supprimer le produits
Le rouge et le noir		Excelent livre de Senthalllll	1245.35 €	Détails du produits	Mettre à jour le produits	Supprimer le produits
Table		Table avec 4 pieds	250.25 €	Détails du produits	Mettre à jour le produits	Supprimer le produits

- Une connexion PDO
- Une requête SQL qui liste tous les produits
- Une boucle foreach qui affiche tous les produits
- Dans chaque ligne un bouton « détail » « mettre à jour » et « supprimer »

```

<?php
//La requête sql stockée dans une variable
$sql = "SELECT * FROM produits ORDER BY id_produit DESC";
//creation d'une variable qui stocke la connexion a PDO et l'execution de la requête SQL
$resultat = $db->query($sql);

?>

<table class="table table-striped">
<thead>
<tr>
<th>Nom</th>
<th>Image</th>
<th>Description</th>
<th>Prix</th>
<th>Détails</th>
<th>Mettre à jour</th>
<th>Supprimer</th>
</tr>
</thead>

<!--Appel des $row + nom de l'objet de la table produits-->
<tbody>
<?php
//Boucle de lecture (connexion = $db + fonction query() PDO + requête SQL = $row)
foreach($db->query("SELECT * FROM produits ORDER BY id_produit DESC") as $row){
//Creation d'une liste pour chaque elements
?>
<tr>
<td><?php echo $row['nom_produit'] ?></td>
<td>" title="<?=$row['nom_produit'] ?>" /></td>
<td><?php echo $row['description_produit'] ?></td>
<td><?php echo $row['prix_produit'] ?> €</td>
<!-- ici on recupère bien la clé apres le ? $_GET['id_produit'] -> pour les details $_GET['id_maj'] -> la mise a jour -->
<!-- $_GET['id'] -> pour efface une valeur-->
<td><a href="detailsProduit.php?id_produit=<?=$row['id_produit'] ?>" class="btn btn-warning">Détails du produits</a></td>
<td><a href="majProduit.php?id_maj=<?=$row['id_produit'] ?>" class="btn btn-info">Mettre à jour le produits</a></td>
<td><a href="suprProduit.php?id=<?=$row['id_produit'] ?>" class="btn btn-danger">Supprimer le produits</a></td>
</tr>
<?php
}

?>
</tbody>
</table>

```

5/ Ajouter un produit

localhost/Ecommerce/ajouterProduit.php
110 %

Connexion à PDO MySQL

AJOUTER UN PRODUIT

Nom du produit

Description du produit

Image du produit

Prix du produit

Ajouter le produit

- Un formulaire avec les champs (**nom + description + image + prix**)
- Un bouton de soumission
- Une page de traitement (**enregistrerProduit.php**) qui enregistre le produit (**requête insert into**) et qui rediriger vers la page **listeProduit.php**
- Afficher des erreurs si les champs sont vides ou mal rempli (**regex**)

- Un bouton annuler et retour

```
//COONEXION A LE BASE de DONNEES
//Stock des valeur nom utilisateur phpmyadmin et mot de passe
$user = "root";
$password = "";
//essaie de te connecter
try{
    //Stockage et instance de la classe PDO pour connecter php et mysql
    $db = new PDO("mysql:host=localhost;dbname=ecommerce;charset=utf8", $user, $password);
    //Fonction static de la classe PDO pour debug la connexion en cas d'erreur
    $db->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
    echo "<p class='alert-success p-5'>Connexion à PDO MySQL</p>";
}catch(PDOException $exception){
    die("Erreur de connexion à PDO MySQL :". $exception->getMessage());
}

?>

<h1 class="text-center">AJOUTER UN PRODUIT</h1>
<!-- Appel de la page du traitement du formulaire-->
<form action="enregistrerProduit.php" method="post">

    <!--NOM DU PRODUIT-->
    <div class="form-group">
        <!--ICI on recup l'attribut name et sa valeur avec $_POST['nom_produit']-->
        <label for="nom_produit">Nom du produit</label>
        <input type="text" placeholder="Le nom du produits" pattern="^[A-Za-z '-]+$" class="form-control" id="nom_produit" name="nom_produit" required>
    </div>

    <!--DESCRIPTION DU PRODUIT-->
    <div class="form-group">
        <!--ICI on recup l'attribut name et sa valeur avec $_POST['description_produit']-->
        <label for="description_produit">Description du produit</label>
        <textarea rows="5" class="form-control" id="description_produit" name="description_produit" required></textarea>
    </div>

    <!--IMAGE DU PRODUIT-->
    <div class="form-group">
        <label for="image_produit">Image du produit</label>
        <!--ICI on recup l'attribut name et sa valeur avec $_POST['image_produit']-->
        <input type="text" class="form-control" id="image_produit" name="image_produit" required>
    </div>

    <!--PRIX DU PRODUIT-->
    <div class="form-group">
        <label for="prix_produit">Prix du produit</label>
        <!--ICI on recup l'attribut name et sa valeur avec $_POST['prix_produit']-->
        <input type="text" step="4" class="form-control" id="prix_produit" name="prix_produit" required>
    </div>

    <div class="form-group mt-5">
        <!--ICI le type submit appelle l'attribut action= du formulaire-->
        <button type="submit" class="btn btn-outline-success">Ajouter le produit</button>
    </div>

</form>

<?php
/*
//Recupation de input name = nom du produit
$nom_produit = $_POST['nom_produit'];
$description_produit = $_POST['description_produit'];
$image_produit = $_POST['image_produit'];
$prix_produit = $_POST['prix_produit'];
//Afficher les valeurs récupérées du formulaire
var_dump($nom_produit);
var_dump($description_produit);
var_dump($image_produit);
var_dump($prix_produit);
*/

```

Traitement ajouter :

```

//CONNEXION A LE BASE DE DONNEES
//Stock des valeur nom utilisateur phpmyadmin et mot de passe
$user = "root";
$password = "";
//Essaie de te connecter
try{
    //Stockage et instance de la classe PDO pour connecter php et mysql
    $db = new PDO("mysql:host=localhost;dbname=ecommerce;charset=utf8", $user, $password);
    //Fonction static de la classe PDO pour debug la connexion en cas d'erreur
    $db->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
    echo "<p class='alert-success p-5'>Salut c bon t connecté a PDO MySQL</p>";
}catch(PDOException $exception){
    die("Erreur de connexion a PDO MySQL :". $exception->getMessage());
}

//Recupation de input name = nom du produit
//Si la variable existe et si elle n'est pas vide alors
if(isset($_POST['nom_produit']) && !empty($_POST['nom_produit'])){
    $nom_produit = htmlspecialchars(strip_tags($_POST['nom_produit']));
    //on stock $_POST[''] dans une variable et on supprime les balise html avec htmlspecialchars et strip_tags
    //Le premier supprime complètement les balises html et php et javascript, l'autre convertit les caractères spéciaux en entité HTML
    //("<" devient "&lt;" par exemple) mais ton lien le montre bien.
}else{
    //Sinon on affiche une erreur
    echo "<p class='alert-danger'>Erreur, merci de remplir le champ nom du produit</p>";
}

//Recup de la description du produit
if(isset($_POST['description_produit']) && !empty($_POST['description_produit'])){
    $description_produit = htmlspecialchars(strip_tags($_POST['description_produit']));
    var_dump($description_produit);
}else{
    echo "<p class='alert-danger'>Erreur, merci de remplir le champ description du produit</p>";
}

//Recupération de l'url de la photo
if(isset($_POST['image_produit']) && !empty($_POST['image_produit'])){
    $image_produit = htmlspecialchars(strip_tags($_POST['image_produit']));
    var_dump($image_produit);
}else{
    echo "<p class='alert-danger'>Erreur, merci de remplir le champ image du produit</p>";
}

//Recupération du prix
if(isset($_POST['prix_produit']) && !empty($_POST['prix_produit'])){
    $prix_produit = htmlspecialchars(strip_tags($_POST['prix_produit']));
    var_dump($prix_produit);
}else{
    echo "<p class='alert-danger'>Erreur, merci de remplir le champ prix du produit</p>";
}

//On écrit la requête SQL soit insérer dans la table produits les(element) auquel on assigne une valeurs
$sql = "INSERT INTO produits (nom_produit, description_produit, image_produit, prix_produit) VALUES (?,?,?,?)";
//Création d'une requête préparée avec la fonction prepare de PDO qui exécute la requête SQL
$stmt = $db->prepare($sql);
//On lie les éléments les 4 ??? a la variable récupérée dans le formulaire
$stmt->bindParam(1, $nom_produit);
$stmt->bindParam(2, $description_produit);
$stmt->bindParam(3, $image_produit);
$stmt->bindParam(4, $prix_produit);

//Si l'insertion fonctionne
if($stmt->execute(array($nom_produit, $description_produit, $image_produit, $prix_produit)){
    //Message de réussite + bouton de retour à la liste
    echo "<p class='alert-success'>Votre produit à bien été ajouté !</p>";
    echo "<a href='listeProduit.php' class='btn btn-outline-success'>Retour à la liste des produit</a>";
}else{
    echo "<p class='alert-danger'>Erreur: Merci de remplir tous les champs</p>";
}

```

6/ Détails des produits


localhost/Ecommerce/detailsProduit.php?id_produit=4 110 %

Salut c bon t connecté a PDO MySQL

DETAILS DU PRODUITS

[Retour à la liste des produits](#)

RE BONJOUR = admin@admin.com

Id	Nom du produit	Description du produit	Image du produit	Prix du produit
4	Cendrier	Cendrier de cuba		7845.25 €

ICI c id du produit : 4

C:\wamp64\www\Ecommerce\detailsProduit.php:78:string '4' (length=1)

C:\wamp64\www\Ecommerce\detailsProduit.php:80:string '4' (length=1)

- Une page qui affiche un seul produit
- Requête de sélection par id
- Un bouton retour

```

<?php
//démarrer la session pour recup $_SESSION
session_start();
ob_start();
$title = "Ecommerce - DÉTAILS DU PRODUITS -";

//CONNEXION A LE BASE DE DONNÉES
//Stock des valeur nom utilisateur phpmyadmin et mot de passe
$user = "root";
$password = "";
//Essaie de te connecter
try{
    //Stockage et instance de la classe PDO pour connecter php et mysql
    $db = new PDO("mysql:host=localhost;dbname=ecommerce;charset=utf8", $user, $password);
    //Fonction static de la classe PDO pour debug la connexion en cas d'erreur
    $db->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
    echo "<p class='alert-success p-5'>Salut c bon t connecté a PDO MySQL</p>";
}catch(PDOException $exception){
    die("Erreur de connexion a PDO MySQL :". $exception->getMessage());
}
}

<h1>DÉTAILS DU PRODUITS </h1>
<a href="listeProduit.php" class="btn btn-dark mt-5">Retour à la liste des produits</a>

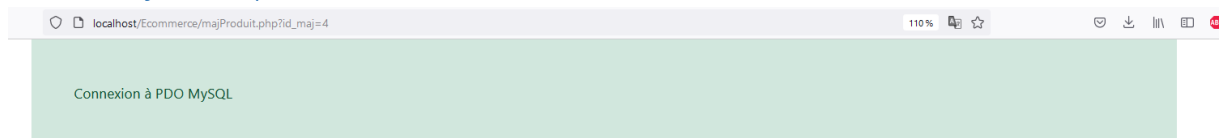
<?php
//Requêtes SQL
$sql = "SELECT * FROM produits WHERE id_produit = ?";
//Stock de la requête dans une variable ($requete) et appel de la connexion puis de la fonction requête préparée
$requete = $db->prepare($sql);
//Objet qui retourne PDO statement etat de la table produits à l'instant
//var_dump($requete);

//Récupération de id <a href=detailsProduit.php?id_produit=<?> $row['id_produit']
//On stocke le resultat de $_GET['id_produit'] = dans une variable
$id = $_GET['id_produit'];
//Passage du ? à la valeur de $_GET['id_produit']
$requete->bindParam(1, $id);
//Exécute la requête
$requete->execute();
//pour afficher les valeurs de la tables produits on doit utilisé la fonction fetch = rechercher
$resultat = $requete->fetch();

//var_dump($resultat);
//Retourne un valeur true (vrai) si des resultat s'affiche
if($resultat){
    ?>
    <h1>RE BONDJOUR = <?> $_SESSION['email'] ?></h1>
    <table class="table">
        <thead>
            <tr>
                <th>Id</th>
                <th>Nom du produit</th>
                <th>Description du produit</th>
                <th>Image du produit</th>
                <th>Prix du produit</th>
            </tr>
        </thead>
        <tbody>
            <tr>
                <td><?> $resultat['id_produit'] ?></td>
                <td><?> $resultat['nom_produit'] ?></td>
                <td><?> $resultat['description_produit'] ?></td>
                <td> $resultat['nom_produit'] ?>" title="<?> $resultat['nom_produit'] ?>"> </td>
                <td><?> $resultat['prix_produit'] ?> €</td>
            </tr>
        </tbody>
    </table>
    <?php
}else{
    echo "<p class='alert-danger p-5'>Erreur : cet ID (produit) n'existe pas</p>";
}

```

7/ Mise à jour du produit



ID du produit à mettre à jour = 4

METTRE UN PRODUIT

Nom du produit

Description du produit

Image du produit

Prix du produit

- Un formulaire de mise à jour du produit
- Le placeholder doit reprendre les données actuelles du produits
- Un bouton valider et retour en cas d'annulation
- Messages d'erreur en cas de champ vide ou invalide
- Un page de traitement (**changerProduit.php**) avec une redirection vers **listeProduit.php**

```
//CONNECTION A LE BASE de DONNEES
//Stock des valeur nom utilisateur phpmyadmin et mot de passe
$user = "root";
$pass = "";
//Essaie de te connecter
try{
    //Stockage et instance de la classe PDO pour connecter php et mysql
    $db = new PDO("mysql:host=localhost;dbname=ecommerce;charset=utf8", $user, $pass);
    //Fonction static de la classe PDO pour debug la connexion en cas d'erreur
    $db->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
    echo "<p class='alert-success p-5'>Connexion à PDO MySQL</p>";
}catch(PDOException $exception){
    die("Erreur de connexion à PDO MySQL :". $exception->getMessage());
}

//Requêtes SQL
$sql = "SELECT * FROM produits WHERE id_produit = ?";
//Stock de la requête dans une variable ($requete) et appel de la connexion puis de la fonction requête préparée
$requete = $db->prepare($sql);
//Objet qui retourne PDO statement etat de la table produits à l'instant
//var_dump($requete);

$mise_a_jour_id = $_GET['id_maj'];
echo "ID du produit à mettre à jour = " . $mise_a_jour_id;
//Passage du ? à la valeur de $_GET['id_produit']
$requete->bindParam(1, $mise_a_jour_id);
//Execute la requête
$requete->execute();
//pour afficher les valeurs de la tables produits on doit utilisé la fonction fetch = rechercher
$resultat = $requete->fetch();

if($resultat){
    <?>
    <h1 class="text-center">METTRE UN PRODUIT</h1>
    <!-- Appel de la page du traitement du formulaire-->
    <form action="valider_majProduit.php?id_maj=<?> $resultat['id_produit'] ?>" method="post">

        <!--NOM DU PRODUIT-->
        <div class="form-group">
            <!--ICI on recup l'attribut name et sa valeur avec $_POST['nom_produit']-->
            <label for="nom_produit">Nom du produit</label>
            <input type="text" value="<?> $resultat['nom_produit'] ?>" class="form-control" id="nom_produit" name="nom_produit" required>
        </div>

        <!--DESCRIPTION DU PRODUIT-->
        <div class="form-group">
            <!--ICI on recup l'attribut name et sa valeur avec $_POST['description_produit']-->
            <label for="description_produit">Description du produit</label>
            <textarea rows="5" class="form-control" id="description_produit" name="description_produit" required><?> $resultat['description_produit'] ?></textarea>
        </div>

        <!--IMAGE DU PRODUIT-->
        <div class="form-group">
            <label for="image_produit">Image du produit</label>
            <!--ICI on recup l'attribut name et sa valeur avec $_POST['image_produit']-->
            <input type="text" value="<?> $resultat['image_produit'] ?>" class="form-control" id="image_produit" name="image_produit" required>
        </div>

        <!--PRIX DU PRODUIT-->
        <div class="form-group">
            <label for="prix_produit">Prix du produit</label>
            <!--ICI on recup l'attribut name et sa valeur avec $_POST['prix_produit']-->
            <input type="text" value="<?> $resultat['prix_produit'] ?>" step="4" class="form-control" id="prix_produit" name="prix_produit" required>
        </div>

        <div class="form-group mt-5">
            <!--ICI le type submit appel le l'attribut action= du formulaire-->
            <button type="submit" class="btn btn-outline-success">Mettre à jour le produit</button>
        </div>
    </form>
}
```

Traitement de la mise à jour :


```

<?php
ob_start();
$title = "Ecommerce - CONFIRMATION DE LA MISE A JOUR DU PRODUITS -";

//COONEXION A LE BASE de DONNÉES
//Stock des valeur nom utilisateur phpmyadmin et mot de passe
$user = "root";
$password = "";
//Essaie de te connecter
try {
    //Stockage et instance de la classe PDO pour connecter php et mysql
    $db = new PDO("mysql:host=localhost;dbname=ecommerce;charset=utf8", $user, $password);
    //Fonction static de la classe PDO pour debug la connexion en cas d'erreur
    $db->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
    echo "<p class='alert-success p-5'>Salut c bon t connecté a PDO MySQL</p>";
} catch (PDOException $exception) {
    die("Erreur de connexion a PDO MySQL :". $exception->getMessage());
}

//Recuperation des valeurs du formulaire avec $_POST[''] qui fait référence a l'attribut name="" dans des formulaires
//Recupation de input name = nom du produit
//Si la variable existe et si elle n'est pas vide alors
if(isset($_POST['nom_produit']) && !empty($_POST['nom_produit'])){
    $nom_produit = htmlspecialchars(strip_tags($_POST['nom_produit']));
    //on stock $_POST[''] dans une variable et on supprime les balise html avec htmlspecialchars et stripslashes
    //Le premier supprime complètement les balises html et php et javascript, l'autre convertit les caractères spéciaux en entité HTML
    //("<" devient "&lt;" par exemple) mais ton lien le montre bien.
}else{
    //Sinon on affiche une erreur
    echo "<p class='alert-danger'>Erreur, merci de remplir le champ nom du produit</p>";
}

//Recup de la description du produit
if(isset($_POST['description_produit']) && !empty($_POST['description_produit'])){
    $description_produit = htmlspecialchars(strip_tags($_POST['description_produit']));
    var_dump($description_produit);
}else{
    echo "<p class='alert-danger'>Erreur, merci de remplir le champ description du produit</p>";
}

//Recupération de l'url de la photo
if(isset($_POST['image_produit']) && !empty($_POST['image_produit'])){
    $image_produit = htmlspecialchars(strip_tags($_POST['image_produit']));
    var_dump($image_produit);
}else{
    echo "<p class='alert-danger'>Erreur, merci de remplir le champ image du produit</p>";
}

//Recuperation du prix
if(isset($_POST['prix_produit']) && !empty($_POST['prix_produit'])){
    $prix_produit = htmlspecialchars(strip_tags($_POST['prix_produit']));
    var_dump($prix_produit);
}else{
    echo "<p class='alert-danger'>Erreur, merci de remplir le champ prix du produit</p>";
}

```

```

//Requête de mise à jour d'un produit (ATTENTION à bien ajouter toutes les propriétés de la table produit
$sql = "UPDATE produits SET nom_produit = ?, description_produit = ?, image_produit = ?, prix_produit = ? WHERE id_produit = ?";
//Stockage de la requête préparée dans une variable
$update = $db->prepare($sql);

//On lie les 4 ? à des nouvelles valeurs
$update->bindParam(1, $nom_produit);
$update->bindParam(2, $description_produit);
$update->bindParam(3, $image_produit);
$update->bindParam(4, $prix_produit);
//Re récupération de id ud produit grâce à la super globale $_GET['id_maj'];
$id_maj = $_GET['id_maj'];
echo "Récupération de id produit depuis un formulaire de mise à jour : " . $id_maj;

$resultatUpdate = $update->execute(array($nom_produit, $description_produit, $image_produit, $prix_produit, $id_maj));
//Si $resultatUpdate == true
if($resultatUpdate){
    //Requêtes SQL
    $sql = "SELECT * FROM produits WHERE id_produit = ?";
    //Stock de la requête dans une variable ($requete) et appel de la connexion puis de la fonction requête préparée
    $requete = $db->prepare($sql);
    //Objet qui retourne PDO statement état de la table produits à l'instant
    $var_dump($requete);

    $id_maj = $_GET['id_maj'];
    echo "ID du produit à mettre à jour = " . $id_maj;
    //Passage du ? à la valeur de $_GET['id_produit']
    $requete->bindParam(1, $id_maj);
    //Exécute la requête
    $requete->execute();
    //pour afficher les valeurs de la table produits on doit utiliser la fonction fetch = rechercher
    $resultat = $requete->fetch();
    ?>
    <div class="alert-success p-5">
        <h1 class="text-center text-warning">Votre produit a bien été modifié </h1>
        <ul class="list-group">
            <li class="list-group-item">ID du produit : <?> $resultat['id_produit'] ></li>
            <li class="list-group-item">Nom du produit<?> $resultat['nom_produit'] ></li>
            <li class="list-group-item">Description du produit<?> $resultat['description_produit'] ></li>
            <li class="list-group-item">Image du produit $resultat['nom_produit'] >" title="<?> $resultat['nom_produit'] >" ></li>
            <li class="list-group-item">Prix du produit<?> $resultat['prix_produit'] > €</li>
        </ul>
        <a href="listeProduit.php" class="btn btn-danger">Retour à la liste des produits</a>
    </div>
</php>
}else{
    echo "<p class='alert-danger p-5'>Une erreur est survenue lors de la mise à jour du produit</p>";
}

//Contenu de template.php défini ce qui se trouve dans le body
//Retourne le contenu du tampon de sortie et termine la session de temporisation.
//Si la temporisation n'est pas activée, alors false sera retourné.
$content = ob_get_clean();
//Rappel du template sur chaque page
require "template.php";

```

8/ Supprimer Produit

```

<?php
//Appel du template

//TEST SANS ob_start()

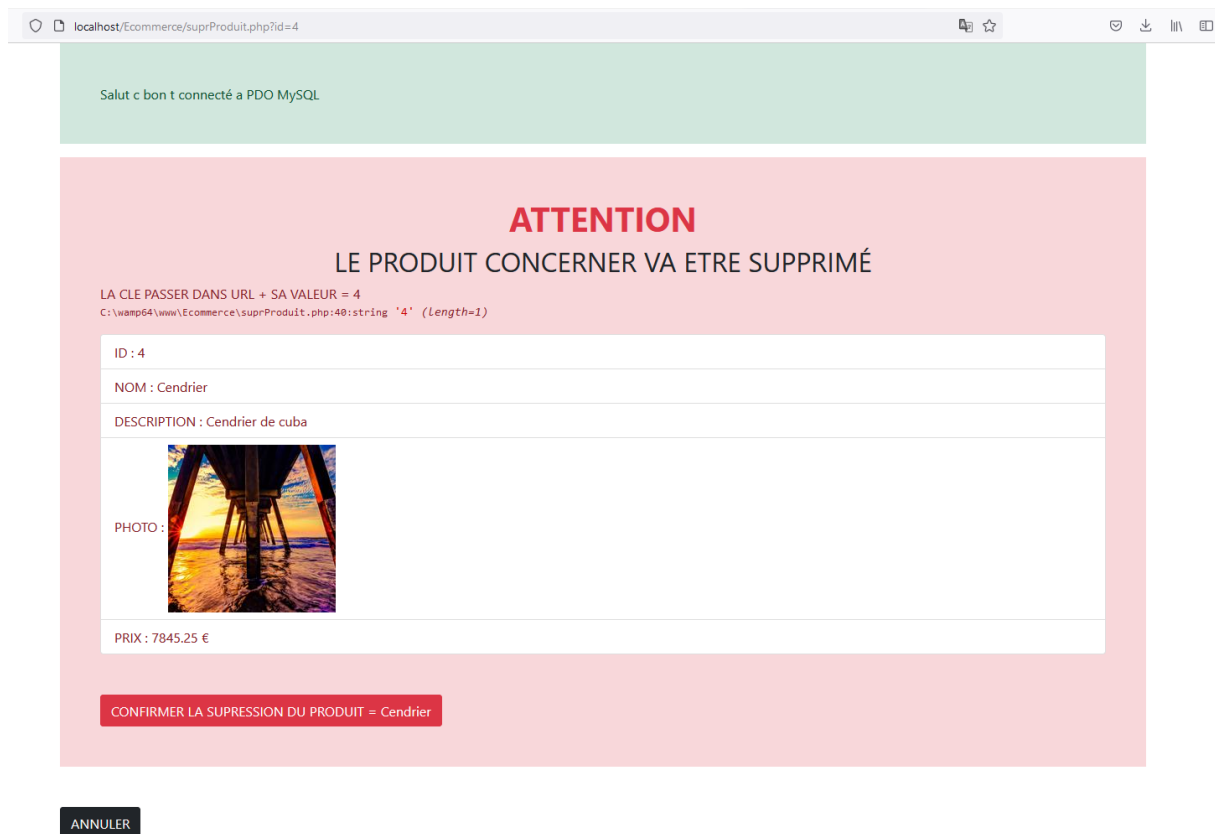
//Connexion à la base de données e-commerce PDO
//Rappel de la connexion PDO
//CONNEXION A LA BASE DE DONNÉES
//Stock des valeurs nom utilisateur phpmyadmin et mot de passe
$user = "root";
$password = "";
//Essaie de te connecter
try{
    //Stockage et instance de la classe PDO pour connecter php et mysql
    $db = new PDO("mysql:host=localhost;dbname=e-commerce;charset=utf8", $user, $password);
    //Fonction static de la classe PDO pour debug la connexion en cas d'erreur
    $db->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
    echo "<p class='alert-success p-5'>Salut c bon t connecté a PDO MySQL</p>";
}catch(PDOException $exception){
    die("Erreur de connexion a PDO MySQL : " . $exception->getMessage());
}

//SUPPRIMER UNE LIGNE DE LA TABLE produits

$sql = "DELETE FROM produits WHERE id_produit = ?";
//Recup l'id passer dans l'url grâce à la super globale $_GET <a href=fichier.php?cle=valeur(dans la table produit (soit id_produit))>
$id = $_GET['id'];
echo "LA CLE PASSER DANS URL + SA VALEUR = " . $id;
var_dump($id);
//Création d'une requête préparée pour lier l'élément ? = $id
$suppression = $db->prepare($sql);
//Bind de $id à ?
$suppression->bindParam(1, $id);
//Exécution de la requête
$suppression->execute();

//Vérification conditionnelle
if($suppression){
    echo "<p class='alert-success p-5'>Le produit a bien été supprimé </p>";
    echo "<a class='btn btn-success' href='listeProduit.php'>Retour à la liste des produits</a>";
}else{
    echo "<p class='alert-danger p-5'>Erreur lors de la suppression du produit</p>";
}

```



- Une page de suppression d'un produit **qui demande une confirmation** (soit un message soit une alert JavaScript)

LES PLUS :

- Créer un Template (ob_start et ob_get_clean)
- Un système de routing
- Mise en page css + animation et event Js
- Système de login – logout
- Ajouter des cookies et doc RGPD (CGV, CGU, etc...)

RAPPEL :

- Comprendre la différence entre site statique et dynamique
- Les sites **statiques** : réalisés en HTML et CSS, leur contenu ne peut être mis à jour que par le webmaster ;
- Les sites **dynamiques** : réalisés avec d'autres outils comme PHP et MySQL en plus de HTML et CSS, ils permettent aux visiteurs de participer à la vie du site, de poster des messages... bref, de rendre le site vivant !
- Le fonctionnement client – server : Les visiteurs du site sont appelés les *clients*. Ils demandent au serveur qui héberge le site de leur transmettre les pages web.
- PHP est un langage exécuté par le serveur. Il permet de personnaliser la page en fonction du visiteur, de traiter ses messages, d'effectuer des calculs, etc. Il génère une page HTML.

- Wamp et Xamp = Apache (Server Web locale de test) + langage PHP (permet au serveur d'exécuter des pages PHP et de retourner de HTML pour le navigateur) + MySQL (SGBDR Système de base de données relationnelle)
- La syntaxe PHP + afficher du texte (echo) + comprendre le debug (var_dump) + afficher des commentaires
- Comprendre les variables scalaires (\$) + affecter des valeurs aux variables + afficher et concaténer des variables + faire des calculs simples
- Comprendre et appliquer des conditions (if...elseif...else, switch case et les ternaire)
- Comprendre et créer des boucles (while et for)
- Comprendre et créer des tableaux (les 2 types : numérotés et associatif)
- Parcourir et rechercher dans des tableaux
- Comprendre et créer des fonctions
- Les fonctions prête à l'emploi PHP
- Inclure des portions pages
- Les bonnes pratiques (nom claires, indentation, commentaires + standard)

ETAPE 1 :

- Transmettre et récupérer des données avec l'URL (Variables Super Globale \$_GET[""] et
- Les failles et la confiance des données reçues (manipulation URL, faille XSS, etc...)
- Tester la présence et le contrôle des valeurs des paramètres passe dans url
- Une URL représente l'adresse d'une page web (commençant généralement par http://).
- Lorsqu'on fait un lien vers une page, il est possible d'ajouter des paramètres sous la forme `bonjour.php?nom=Dupont&prenom=Jean` , qui seront transmis à la page.
- La page `bonjour.php` , dans l'exemple précédent, recevra ces paramètres dans un array nommé `$_GET` :
 - o `$_GET['nom']` aura pour valeur « Dupont » ;
 - o `$_GET['prenom']` aura pour valeur « Jean ».
- Cette technique est très pratique pour transmettre des valeurs à une page, mais il faut prendre garde au fait que le visiteur peut les modifier très facilement. Il ne faut donc pas faire aveuglément confiance à ces informations, et tester prudemment leur valeur avant de les utiliser.
- La fonction `isset()` permet de vérifier si une variable est définie ou non.
- Le transtypage est une technique qui permet de convertir une variable dans le type de donnée souhaité. Cela permet de s'assurer par exemple qu'une variable est bien un int (nombre entier).

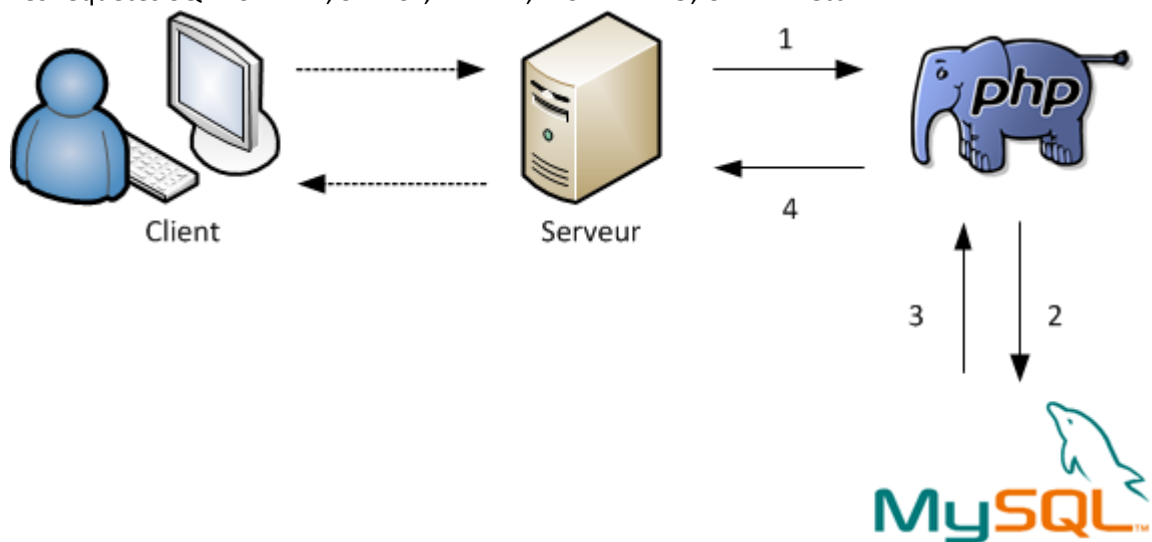
ETAPE 2 : Transmettre des données avec des formulaires :

- Les formulaires de base, les attributs action et méthodes (POST et GET)
- get : les données transiteront par l'URL, comme on l'a appris précédemment. On pourra les récupérer grâce à l'array `$_GET` . Cette méthode est assez peu utilisée car on ne peut pas envoyer beaucoup d'informations dans l'URL (je vous disais dans le chapitre précédent qu'il était préférable de ne pas dépasser 256 caractères).
- post : les données ne transiteront pas par l'URL, l'utilisateur ne les verra donc pas passer dans la barre d'adresse. Cette méthode permet d'envoyer autant de données que l'on veut, ce qui fait qu'on la privilégie le plus souvent. Néanmoins, les données ne sont pas plus sécurisées qu'avec la méthode GET , et il faudra toujours vérifier si tous les paramètres sont bien présents et valides, comme on l'a fait dans le chapitre précédent. **On ne doit pas plus faire confiance aux formulaires qu'aux URL.**

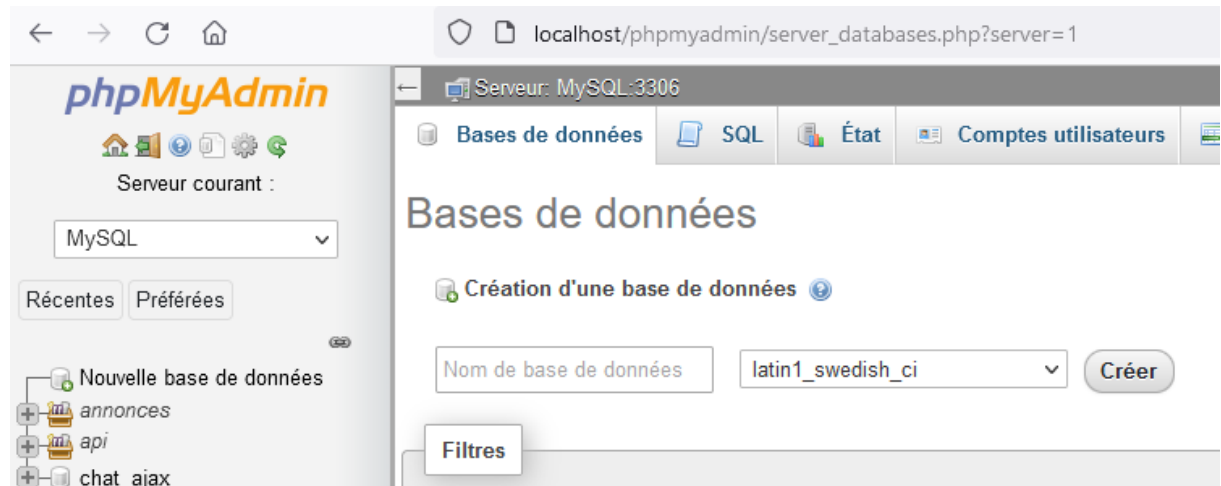
- Action = la cible <form action="" methode=""></form>
- La balise input : attribut type + name
- Interaction entre la variable super globale \$_POST[""] et l'attribut name="" des formulaire
- La faille XSS : htmlspecialchars, htmlentities et trim
- L'envoi de fichier via input type='file'
- Les formulaires sont le moyen le plus pratique pour le visiteur de transmettre des informations à votre site. PHP est capable de récupérer les données saisies par vos visiteurs et de les traiter.
- Les données envoyées via un formulaire se retrouvent dans un array \$_POST .
- De la même manière que pour les URL, il ne faut pas donner sa confiance absolue aux données que vous envoie l'utilisateur. Il pourrait très bien ne pas remplir tous les champs, voire trafiquer le code HTML de la page pour supprimer ou ajouter des champs. Traitez les données avec vigilance.
- Que ce soit pour des données issues de l'URL (\$_GET) ou d'un formulaire (\$_POST), il faut s'assurer qu'aucun texte qui vous est envoyé ne contient du HTML si celui-ci est destiné à être affiché sur une page. Sinon, vous ouvrez une faille appelée XSS qui peut être néfaste pour la sécurité de votre site.
- Pour éviter la faille XSS, il suffit d'appliquer la fonction htmlspecialchars sur tous les textes envoyés par vos visiteurs que vous afficherez.
- Les formulaires permettent d'envoyer des fichiers. On retrouve les informations sur les fichiers envoyés dans un array \$_FILES . Leur traitement est cependant plus complexe.

ETAPE 3 : Base de données - Tables - SQL - UML et phpMyAdmin

- Les différents SGBDR : (MySQL, MariaDB, PostgreSQL, SQLite, Oracle, Microsoft SQL Server, etc...)
- Les requêtes SQL : CREATE, SELECT, DELETE, INSERT INTO, UPDATE etc...



Création et structure d'une base de donnée via phpMyAdmin :

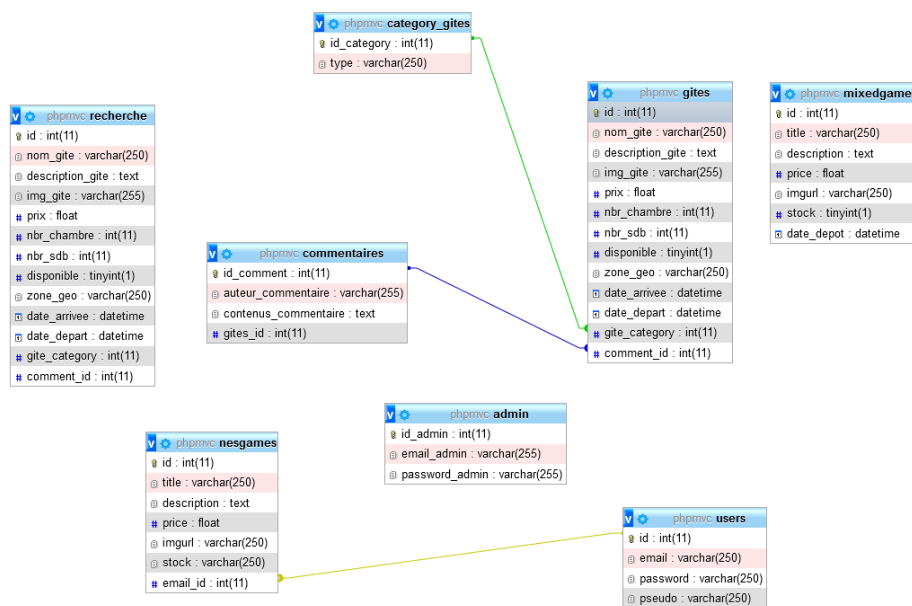


- Une base de données est un outil qui stocke vos données de manière organisée et vous permet de les retrouver facilement par la suite.
- On communique avec MySQL grâce au langage SQL. Ce langage est commun à tous les systèmes de gestion de base de données (avec quelques petites différences néanmoins pour certaines fonctionnalités plus avancées).
- PHP fait l'intermédiaire entre vous et MySQL.
- Une base de données contient plusieurs tables.
- Chaque table est un tableau où les colonnes sont appelées « champs » et les lignes « entrées ».

LES OPERATIONS AVEC PHPMYADMIN

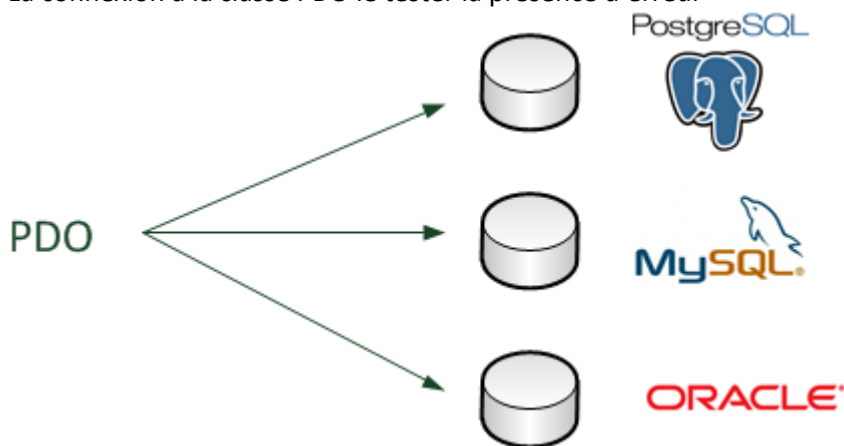
- phpMyAdmin est un outil qui nous permet de visualiser rapidement l'état de notre base de données et de la modifier, sans avoir à écrire de requêtes SQL.
- On crée généralement un champ nommé `id` qui sert à numéroté les entrées d'une table. Ce champ doit avoir un index PRIMARY (on dit qu'on crée une *clé primaire*) et l'option `AUTO_INCREMENT` qui permet de laisser MySQL gérer la numérotation.
- MySQL gère différents types de données pour ses champs, à la manière de PHP. On trouve des types adaptés au stockage de nombres, de textes, de dates, etc.
- phpMyAdmin possède un outil d'importation et d'exportation des tables qui nous permettra notamment d'envoyer notre base de données sur Internet lorsque nous mettrons notre site en ligne.

UML et les MCD



AFFICHER DES DONNÉES AVEC PHP

- La connexion à la classe PDO le tester la présence d'erreur



- Récupérer des données avec PDO query et SQL SELECT
- Boucle while ou foreach PHP et fetch (aller chercher) pour afficher les données
- Le filtrage de données à l'aide des critères de sélection (WHERE, ORDER BY, LIMIT, etc...)
- Construire des requêtes en fonction des variables et lutter contre les injection SQL (PHP et les requêtes préparées)
- Traquer les erreurs
- `$bdd = new PDO('mysql:host=localhost;dbname=test;charset=utf8', 'root', '', array(PDO::ATTR_ERRMODE => PDO::ERRMODE_EXCEPTION));`
- Pour dialoguer avec MySQL depuis PHP, on fait appel à l'extension PDO de PHP.
- Avant de dialoguer avec MySQL, il faut s'y connecter. On a besoin de l'adresse IP de la machine où se trouve MySQL, du nom de la base de données ainsi que d'un login et d'un mot de passe.

- Les requêtes SQL commençant par SELECT permettent de récupérer des informations dans une base de données.
- Il faut faire une boucle en PHP pour récupérer ligne par ligne les données renvoyées par MySQL.
- Le langage SQL propose de nombreux outils pour préciser nos requêtes, à l'aide notamment des mots-clés WHERE (filtre), ORDER BY (tri) et LIMIT (limitation du nombre de résultats).
- Pour construire une requête en fonction de la valeur d'une variable, on passe par un système de requête préparée qui permet d'éviter les dangereuses failles d'injection SQL.

ETAPE 4 : Ecrire des données :

- On utilise différents mots-clés en fonction du type de modification que l'on souhaite effectuer :
 - o INSERT INTO : ajout d'une entrée ;
 - o UPDATE : modification d'une ou plusieurs entrées ;
 - o DELETE : suppression d'une ou plusieurs entrées.
- Comme pour la sélection de données, on utilise les requêtes préparées pour personnaliser nos requêtes en fonction de variables.
- Lorsqu'on utilise UPDATE ou DELETE , il faut penser à filtrer avec un WHERE , sinon toute la table sera affectée par l'opération !
- PHP indique si une erreur avec MySQL est intervenue.

ETAPE 5 : Les fonctions SQL :

- MySQL permet d'exécuter certaines fonctions lui-même, sans avoir à passer par PHP. Ces fonctions modifient les données renvoyées.
- Il existe deux types de fonctions :
 - o Les **fonctions scalaires** : elles agissent sur chaque entrée récupérée. Elles permettent par exemple de convertir tout le contenu d'un champ en majuscules ou d'arrondir chacune des valeurs ;
 - o Les **fonctions d'agrégat** : elles effectuent des calculs sur plusieurs entrées pour retourner une et une seule valeur. Par exemple : calcul de la moyenne, somme des valeurs, comptage du nombre d'entrées...
- On peut donner un autre nom aux champs modifiés par les fonctions, en créant des alias à l'aide du mot-clé AS .
- Lorsqu'on utilise une fonction d'agrégat, il est possible de regrouper des données avec GROUP BY .
- Après un groupement de données, on peut filtrer le résultat avec HAVING . Il ne faut pas le confondre avec WHERE qui filtre avant le groupement des données.

ETAPE 6 : Les Dates en SQL :

- MySQL propose plusieurs types de champs pour stocker des dates.
- Les deux types les plus couramment utilisés sont :
 - o DATE : stocke une date au format AAAA-MM-JJ ;
 - o DATETIME : stocke une date et une heure au format AAAA-MM-JJ HH:MM:SS.
- On peut trier et filtrer des champs contenant des dates comme s'il s'agissait de nombres.

- Il existe de nombreuses fonctions SQL dédiées au traitement des dates. La plus connue est NOW() qui renvoie la date et l'heure actuelles.

ETAPE 7 : Les jointures entre des tables en SQL :

- Les bases de données permettent d'associer plusieurs tables entre elles.
- Une table peut contenir les id d'une autre table, ce qui permet de faire la liaison entre les deux. Par exemple, la table des jeux vidéo contient pour chaque jeu l'id de son propriétaire. Le nom et les coordonnées du propriétaire sont alors stockés dans une table à part.
- Pour rassembler les informations au moment de la requête, on effectue des **jointures**.
- On peut faire des jointures avec le mot-clé WHERE , mais il est recommandé d'utiliser JOIN qui offre plus de possibilités et qui est plus adapté.
- On distingue les jointures internes, qui retournent des données uniquement s'il y a une correspondance entre les deux tables, et les jointures externes qui retournent toutes les données, même s'il n'y a pas de correspondance.

ETAPE 8 : Créer des images avec PHP

- PHP permet de faire bien plus que générer des pages web HTML. En utilisant des extensions, comme la bibliothèque GD, on peut par exemple générer des images.
- Pour qu'une page PHP renvoie une image au lieu d'une page web, il faut modifier l'en-tête HTTP à l'aide de la fonction header() qui indiquera alors au navigateur du visiteur l'arrivée d'une image.
- Il est possible d'enregistrer l'image sur le disque dur si on le souhaite, ce qui évite d'avoir à la générer à chaque fois qu'on appelle la page PHP.
- On peut créer des images avec GD à partir d'une image vide ou d'une image déjà existante.
- GD propose des fonctions d'écriture de texte dans une image et de dessin de formes basiques.
- Des fonctions plus avancées de GD permettent de fusionner des images ou d'en redimensionner.

ETAPE 9 : Les RegEx

- Les expressions régulières sont des outils de recherche et de remplacement de texte très avancés qui permettent d'effectuer des recherches très précises, pour vérifier par exemple que le texte saisi par l'utilisateur correspond bien à la forme d'une adresse e-mail ou d'un numéro de téléphone.
- La fonction preg_match vérifie si un texte correspond à la forme décrite par une expression régulière.
- Une expression régulière est délimitée par un symbole (par exemple le dièse #).
- Les classes de caractères permettent d'autoriser un grand nombre de symboles (lettres et chiffres) selon un intervalle.
- Les quantificateurs permettent d'exiger la répétition d'une chaîne de texte un certain nombre de fois.
- Certains caractères sont spéciaux au sein d'une expression régulière : on parle de *métacaractères*. Si on souhaite les rechercher dans une chaîne, il faut les échapper en plaçant un symbole antislash devant. Par exemple : \[.
- Il existe des classes abrégées, c'est-à-dire des classes toutes prêtes, comme par exemple \d qui revient à écrire [0-9] .
- La fonction preg_replace permet d'effectuer des remplacements dans une chaîne de texte.

- Dans le cas d'un remplacement, les parenthèses au sein d'une expression régulière permettent de capturer une portion de texte pour la réutiliser dans une autre chaîne.

ETAPE 10 : Introduction au Routing et a une connexion sécurisée

Ressources :

- <https://openclassrooms.com/fr/courses/918836-concevez-votre-site-web-avec-php-et-mysql/4237646-decouvrez-le-fonctionnement-dun-site-ecrit-en-php>