GUENOUN Ava
PIQUERAS Maud
GUICHARD Ingrid

EPF
ÉCOLE D'INGÉNIEUR-E-S
Sceaux I Troyes I Montpellier

3A

Projet BioCal

GUIDE DE FONCTIONNEMENT PHP

Encadrants : Mr GADEMER Antoine
Mme ZITZMAN Cathel
Mme MOUTON Julia

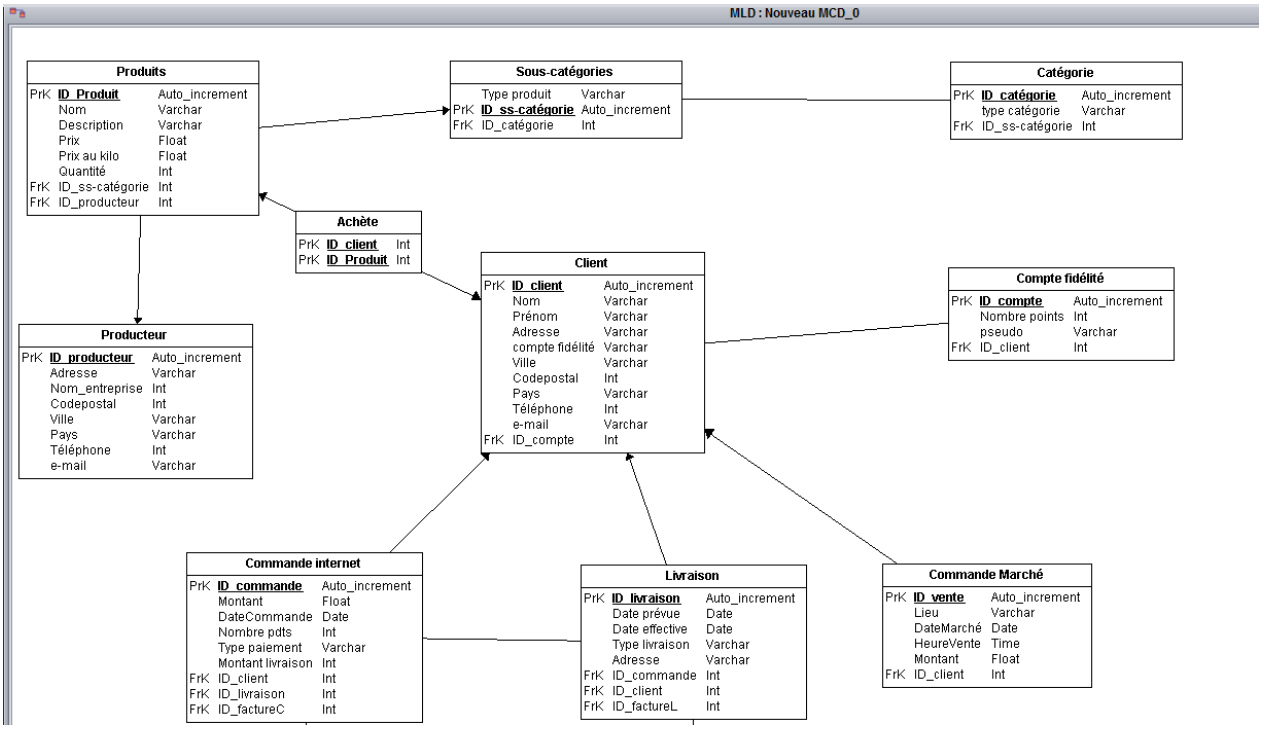
Table des matières

INTRODUCTION

Le projet Biocal réside en la création d'un site internet permettant aux fournisseurs de faciliter la visualisation de leur marché. En effet, grâce à cet outil, ils pourront avoir accès rapidement et simplement à leurs stocks, aux fiches produits, aux fiches clients ainsi qu'aux fiches des autres fournisseurs. Ils pourront aussi avoir des vues sur l'état de livraison de leurs produits chez les consommateurs ainsi que d'autres fonctionnalités.

I – Création de la Base de données.

Pour pouvoir créer cet outil, et enregistrer les données, il faut tout d'abord créer une base de données. Nous avons choisi de créer cette dernière sur XAMPP en suivant le MLD suivant :



Ce modèle à été établi à partir des besoins majeurs imposés par l'entreprise BioCal. Nous avons donc les tables suivantes :

Chaque table possède un id auto-incrémenté qui permettra un rangement plus ordonné des données et qui vont permettre d'éviter les confusions si deux clients ont le même nom et prénom par exemple.

Chaque table est donc vide au départ et va pouvoir être implémentée au fur et à mesure que les premières commandes se font. Ainsi, l'utilisateur aura le pouvoir d'ajouter un client dans la table « clients », un fournisseur dans la table « agriculteur » ou encore des produits dans la table « produits ». le reste des tables concernant les commandes, les livraisons etc. elles se remplissent automatiquement lorsque des commandes sont passées et varient selon l'état d'avancement de la commande et de sa livraison.

Pour implémenter manuellement ces tables sans passer directement par les fonctionnalités de XAMPP nous allons les implémenter grâce à des formulaires. Ainsi, quand l'utilisateur voudra rajouter un client, un producteur ou encore un produit, chaque mise à jour se fera dans leur table respective selon plusieurs critères comme le nom ou le prénom par exemple.

II- Conception et agencement

Notre outil va donc se décomposer en plusieurs pages pour être le plus intuitif, attrayant et facile d'utilisation possible.

Il se compose donc de 7 pages qui tentent de répondre au mieux au cahier des charges.

Nous avons donc les pages suivantes :

- Page d'accueil
- Bilan des ventes
- Générer fiches
- Gestion des stocks
- Suivi des commandes
- Mise à jour

Ces pages sont donc conçues à partir de plusieurs langages qui ont chacun leurs intérêts :

- PHP : c'est le langage qui nous permettra de mettre en place un lien entre notre interface html et notre base de données.
- html : c'est le langage qui nous permettra d'avoir une plateforme interactive avec l'utilisateur.
- CSS : c'est le langage qui associé avec le langage php à travers un code source va nous permettre d'agencer esthétiquement nos pages.
- Mysql : c'est le langage qui nous permet d'écrire des requêtes vers la base de données à laquelle on est connectés.

III - Détail des pages

Tout d'abord, dans un souci d'esthétisme, nous sommes partis de la même base pour toutes nos pages. En effet dans « head » nous avons relié nos pages à notre code source en css pour une certaine homogénéité du site.

Pour ce faire, nous utilisions la commande suivante :

```
<head>
<title>BIOCAL</title>
<meta charset="utf-8">
<link rel="stylesheet" type="text/css" href="sourcecode.css">
</head>
```

De plus, ici nous pouvons voir que nous utilisons la balise `<meta charset="utf-8">` qui désigne un encodage dans les spécifications html pour spécifier un jeu de caractères particulier ici utf-8 qui est le plus adapté.

Aussi, l'un des points communs entre toutes nos pages qui nous a aussi permit des les relier entre elles, est le menu en tête de page que nous avons inséré dans une balise « body ».


```
<body>
<ul class="menu">
<li class="menu"><a class="active" href="index.php">Accueil</a></li>
<li class="menu"><a href="bilans.php">Bilan Ventes</a></li>
<li class="menu"><a href="fiches.php">Générer Fiches</a></li>
<li class="menu"><a href="stock.php">Gestion Stocks</a></li>
<li class="menu"><a href="commandes.php">Suivi Commande</a></li>
<li class="menu"><a href="ajouter.php">Mise à jour</a></li>
</ul>
```

Pour ce faire, nous avons donc utilisé une fonction permettant de créer une liste non numérotée. Ici, nous comprenons donc que nous sommes sur la page d'accueil car le bouton accueil est activé comme nous le montre le code correspondant.

Ensuite, l'un des points les plus importants est la connexion de nos différentes pages avec la BDD. Celle-ci est réalisée par la commande suivante :

```php
<?php

/* En général on configure les paramètres de configuration dans des variables partagées
 * entre toutes les pages avec la commande include() (plus facile quand on veut changer
 * les paramètres du site)
 */
    $hostname = "localhost";
    $database = "biocal_vf";
    $username = "root";
    $password = "";

    $pdo_options[PDO::ATTR_ERRMODE] = PDO::ERRMODE_EXCEPTION;
    try {
    $bdd = new PDO('mysql:host=' . $hostname . ';dbname=' . $database.';charset=UTF8', $username, $password, $pdo_options); // Connexion à la BDD
    echo "Si vous voyez ce message, c'est que la connexion à la base de donnée s'est bien passée.";
    } catch(PDOException $ex) { // On attrape les Exceptions (si quelquechose s'est mal passé).
        echo "Si vous voyez ce message, c'est qu'une erreur s'est produite
        (XAMPP n'est pas lancé, le nom de la base de donnée n'est pas bon, vous avez mis un mot de passe sur le compte root, etc.<br/>";
        echo "<div style='color:red;'>".$ex->getMessage()."</div>";
    }

?>
```

Cette commande est donc à ajouter sur toutes nos pages avant d'effectuer toutes requêtes avec la BDD. Nous comprenons donc que notre BDD se nomme « Biocal_vf », que notre nom d'utilisateur est root.

Notre site doit aussi être capable de gérer les erreurs simplement pour que notre site ne s'arrête pas brusquement si telle ou telle partie du site web n'est pas accessible à un moment T mais qu'à la place, il continue à faire ce qui lui est demandé. Pour ce faire, nous avons donc utilisé pour toutes nos requêtes une commande try/catch qui nous permet donc de gérer les erreurs et de nous afficher des messages d'erreurs eu lieu de crasher le système :

```
try {
    $response = $statement->execute(); // Execution de la requête a proprement parlé.

    // ATTENTION, lorsqu'on utilise un $statement, il faut appliquer le fetchAll
        sur l'objet $statement et non $response !
    $output = $statement->fetchAll(PDO::FETCH_CLASS);
        // L'option PDO::FETCH_CLASS récupère les données sous forme de tableau associatif
    echo "Si vous voyez ce message, c'est que tout s'est bien passé.<br/>";
} catch (PDOException $ex) {
    echo "Quelque chose s'est mal passé. Avez-vous bien envoyé tous les champs ?";
    echo "<div style='color:red;'>" . $ex->getMessage() . "</div>";
}
```

Pour finir, pour l'affichage de nos données issues de la BDD après chaque requête, nous avons choisit d'afficher les informations sous forme de tableau :

- Style du tableau :

```
<style>
    table {
        border-collapse: collapse;  /* Pour fusionner les bordures des cases*/

    }
        th {
                background-color: #F66913; /* Fond de la case*/
                border: 5px solid #F66913; /* Bordure des cases*/
                padding: 8px; /* Pour laisser "respirer" un peu le texte*/
        }

        td{
                border: 5px solid #F66913; /* Bordure des cases*/
                padding: 8px; /* Pour laisser "respirer" un peu le texte*/
        }
</style>
```

  Cette commande n'est à afficher qu'une fois dans chaque page même si plusieurs requêtes sont faites.
- Création du tableau :

```
if (isset($output)) { // On reprend la variable $output chargé à "l'exercice"
        précédent, on vérifie donc qu'elle existe bien avec isset()
    echo "<table>"; // Début du tableau

    for ($ligne = 0; $ligne < count($output); $ligne++) { // Pour chaque ligne
        echo "<tr>"; // début de ligne

        if ($ligne == 0) { // Pour la première ligne, on commence par générer la ligne de titre
            foreach ($output[$ligne] as $index => $valeur) { // Pour chaque colonne
                echo "<th>"; // début de colonne

                echo $index; // Affichage du nom de la colonne (qui sert d'index)

                echo "</th>"; // fin de colonne
            }
            echo "</tr><tr>"; // Nouvelle ligne (pour les données)
        }

        foreach ($output[$ligne] as $index => $valeur) { // Pour chaque colonne
            echo "<td>"; // début de colonne

            echo $valeur; // Affichage de la valeur de la case.

            echo "</td>"; // fin de colonne
        }

        echo "</tr>"; // fin de ligne
    }

    echo "</table>"; // fin du tableau
}
```

III- conception des pages

Nous allons maintenant détailler les points importants sur chaque page.

A- Partie CSS (code source)

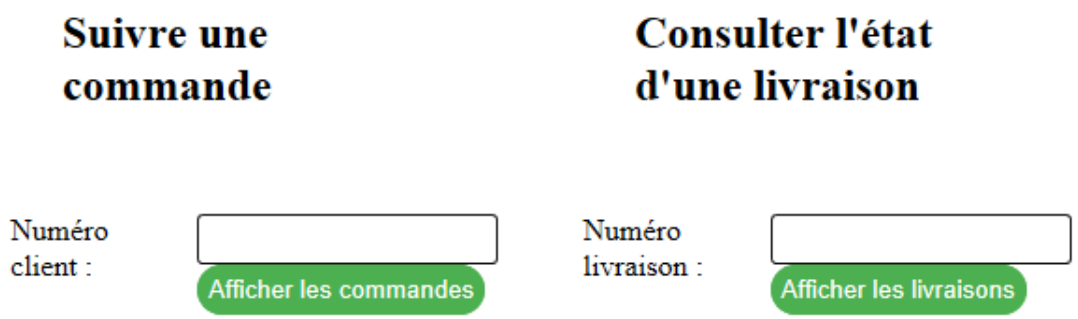
Tout d'abord, il en convient de détailler notre page CSS base de l'esthétisme de toutes nos pages :

-
```
#container {
        background-color: #4CAF50;
        padding: 10px;
        width: 90%;
        height: auto;
}
```

Tout d'abord nous avons donc une division container dans laquelle nous allons pouvoir placer d'autres divisions pour les aligner horizontalement par exemple. Cette division va donc représenter 90% de l'espace parent et délimiter les sous divisions qu'on va ajouter.

```
.formulaire {
        width: 30% ;
        float: left;
        height: 120px;
}
```

Ici, nous avons donc une première sous division. Celle-ci nous a permis de pouvoir aligner les différents formulaires comme ci-dessous :




```
.inputbasic {
        border: 1px solid #111;
        border-radius: 3px;
        padding: 5px;
        transition: all 0.5s ease;

}

.inputbasic:hover{
        background-color: lightgrey;
        color: #FFFFFF;
}
```

```
.bouton {
        background-color: #4CAF50;
        color: #FFFFFF;
        border: 1px solid #4CAF50;
        padding: 5px;
        transition: all 0.5s ease;
        border-radius: 100px;

}
.bouton:hover {
        background-color: #FFFFFF;
        color: #4CAF50;
        border: 1px solid #4CAF50;
}
```

Ces différentes commandes servent à rendre nos formulaires plus attrayant en y ajoutant des couleurs, en arrondissant nos boîtes, et en ajoutant des effets comme avec la commande « hover » qui permet d'avoir des animations au moment du survol de nos éléments comme un changement de couleur ou encore une surbrillance.

La partie CSS comporte encore d'autres parties plus basiques à incrémenter selon ses choix.

B- Page d'accueil (index.php)

Cette page comporte donc un menu comme explicité plus haut un texte de présentation et une image.

```
<p>Bienvenue sur le sie internet de BioCal. Cet outil vous permettra de garder un oeil sur vos commandes,
        vos stocks et de pouvoir facilement mettre vos listes produits et clients. </p>

<img    src="images/BioCal.png"
        alt="Le bio Made in France"
```

La commande <p> représente le début d'un paragraphe et permet de sauter deux lignes. Pour l'image, nous l'avons enregistré dans un dossier « images » associé aux autres pages comme le montre le chemin dans le code et nous y avons ajouté un texte alternatif au cas où il y aurait une erreur de chargement.

C- Page « Bilan Ventes » (bilans.php)

Dans cette page, nous avons utilisé les divisions ainsi que le menu expliqués plus haut.

Pour pouvoir consulter une catégorie ou encore consulter les ventes par années, nous avons créer une requête de type « SELECT » :

```
// On utilise l'opérateur LIKE qui tolère la présence de JOKER (%)
$sql = "SELECT sum(achats.Quantité*produits.Prix) AS 'Montant total des ventes de la categorie' FROM achats
INNER JOIN produits ON produits.ID_Produit=achats.ID_Produit WHERE produits.categories_produit LIKE :Categorie ;";
```

Ici nous avons utilisé des un « INNER JOIN » pour pouvoir créer des liens entre deux tables et chercher des informations dans des dernières.

Nous avons aussi créé ici un menu déroulant proposant des choix de catégories :

```
<form action="bilans.php" method="POST">
<form class="form_style">

    Catégorie :  <select name="Categorie">
            <option value="">--Choisissez une catégorie --</option>
        <option value="cremerie">Crèmerie</option>
        <option value="epicerie salee">Epicerie salée</option>
        <option value="epicerie sucree">Epicerie sucrée</option>
                <option value="fruits et legumes">Fruits et légumes</option>
                <option value="viande et poisson">Viandes et poissons</option>
     </select>

        <input type="hidden" name="action" value="filterShowData">
        <input type="submit" value="Afficher le bilan" class="bouton">
        </form>
</form>
```

Pour les ventes par année, nous avons utilisé un formulaire qui récupère une donnée comme explicité plus haut. De plus pour l'affichage de nos données nous avons choisit d'utiliser des tableaux aussi expliqué plus haut.

D- Page « Générer fiches » (fiches.php)

Pour cette page, nous avons utilisé des techniques similaires aux autres pages expliquées auparavant, comme le formulaire où l'utilisateur va entrer la données qu'il recherche manuellement. Cette donnée va ensuite être traitée grâce à une requête de type « SELECT » pour enfin être retournée à l'utilisateur sous forme encore une fois de tableau.

E- Page « Gestion stocks » (stock.php)

Cette page moins interactive que les autres va se mettre à jour automatiquement dès que la page sera rechargée. En effet, ici l'utilisateur n'a pas de choix à faire quant aux données qu'il veut voir apparaître. La page va donc se mettre à jour en renvoyant la requête mysql à la BDD dès que cette dernière est rechargée. Cette page composée de deux parties nous informe tout d'abord avec un premier tableau des produits dont le seuil critique est atteint. Cette commande se fait donc grâce à un « SELECT » avec des conditions comme suit :

```
$sql = "SELECT `ID_Produit`, `Nom`, `Description`, `Quantité`, `categories_produit`,
`sous_categories_produit` FROM `produits` WHERE `Quantité` < 5";
```

Le second tableau lui affiche tout simplement la totalité des stocks.

F- Page « Suivi Commande » (commandes.php)

Cette page permet à l'utilisateur de pouvoir suivre sa commande c'est-à-dire voir à qui elle appartient de quels produits elle est composée, etc. Soit de visualiser la table commande de la BDD à travers un tableau.

Elle permet aussi de pouvoir consulter l'état d'une livraison c'est-à-dire si le produit est en cours de livraison ou s'il sera en retard et donc non livré dans les temps.

En majeure partie cette page recouvre donc des techniques déjà détaillées soit le formulaire encore une fois qui renvoi donc à une requête mysql de type SELECT et qui nous affiche donc le résultat dans un tableau. Et un pop-up qui s'affiche pour alerter l'utilisateur du numéro de livraison en retard grâce au code suivant :

```
<?php

    $sql = "SELECT * from livraisons WHERE Date_prévue < NOW() AND Date_effective
=0;";

    try {
        $response = $bdd->query($sql); // Execution de la requête a proprement
parlé.
        $output = $response->fetchAll(PDO::FETCH_ASSOC); // Récupération des
données.
        //print_r($output);
        //echo "<br/>Si vous voyez ce message, c'est que tout s'est bien
passé.<br/>";
    } catch (PDOException $ex) {
        echo "Quelque chose s'est mal passé. Est-ce que la table existe bien ?";
        echo "<div style='color:red;'>" . $ex->getMessage() . "</div>";
    }
        if(count($output)>0) {

echo'<script type="text/javascript">
alert("Attention commandes en retard :\n';
foreach($output as $livraison) {

        echo 'Livraison '.$livraison["ID_Livraison"].' en retard ! Date prévue :
'.$livraison["Date_prévue"].'\n';
        }
echo '");
</script>';


        }
?>
```

G- Page « Mise à jour » (ajouter.php)

Dans cette page, nous avons aligner trois formulaires qui servent à ajouter des lignes dans les tables suivantes : « clients », « producteurs » et « produits ».

Pour ce faire, nous avons utilisé la requête « INSERT INTO » comme suit :

```
$sql = "INSERT INTO `clients`(Nom, Prénom, Adresse, Code_postal, Ville, Pays,
Telephone, email, compte_fidelite) VALUES (:nameFilter, :surnameFilter, :adressFilter,
:codeFilter, :villeFilter, :paysFilter, :telFilter, :emailFilter, :compteFilter);";
$statement = $bdd->prepare($sql, array(PDO::ATTR_CURSOR => PDO::CURSOR_FWDONLY));
```

Le code est similaire pour les autres tables.