Semene Robin

Documentation technique - SAE 23

Ce fichier contient la documentation technique sur les élements que j'ai implémenté sur le site de colis lors de la SAE 23 :

Ce fichier contient la documentation technique des éléments suivants :

Page de login, page de visualisation, page d'ajout et page d'édition

NOTE : En cas de vérification des commits, Camille s'est chargé de merge manuellement le travail fait sur deux branches du dépot GitHub (nous avons eu quelques problèmes lors de la résolution des merge conflicts). Vous pourrez voir mes commits sur la branche 'temp3'. Je vous conseille également de tester mon travail sur cette branche (au détriment de l'aspect visuel) si le code fourni dans la branche main venait à ne pas marcher.

Page de Login

La page de login contient deux zones de texte et un bouton pour soumettre les identifiants. Le fonctionnement de cette page repose sur les lignes de code suivantes, que je vais détailler :

```php
if(isset($_POST['username'])){
    $pdo=new PDO('mysql:host=localhost;charset=utf8;dbname=site','test','xd');
    //$sql = "INSERT INTO users (login,password) VALUES(:username,:password)";
    $sql = "select * from users where username= :username and password= :password ";
    $req = $pdo->prepare($sql);
    $req->bindParam(':username', $_POST['username']);
    $req->bindParam(':password', md5($_POST['password']));
    $req->execute();
    $res = $req->fetch(PDO::FETCH_ASSOC);
    if ($res == NULL){
        echo "<p>Login ou mot de passe incorrect.</p>";
    }
    else{
        session_start();
        $_SESSION['username'] = $_POST['username'];
        echo "<script>location.href = 'index.php';</script>";
    }
}
```

```html
<div id="formulaire">
    <form action="" method="post" name="login">
        <div class="formulaireBlock"><input type="email" name="username" placeholder="Adresse e-mail"></div>
        <div class="formulaireBlock"><input type="password" name="password" placeholder="Mot de passe"></div>
        <div class="formulaireBlock"><button type="submit" value="Connexion" name="submit">Connexion</button></div>
```

La première ligne vérifie si le formulaire à envoyé via la méthode post un élément nommé username, puis il effectue une requète sql pour voir si un utilisateur existe avec l'identifiant et le mot de passe entré dans le formulaire.

La requête SQL ici est préparée, et les paramètres :username et :password sont associés aux variables correspondant à l'identifiant rentré et au mot de passe haché via le système d'encryption MD5.

La raison derrière l'encryption des mots de passe est, dans un cas extrême où quelqu'un viendrait à récupérer toutes les données de la base de données, il se retrouverait avec des mots de passe cryptés et donc inutilisables.

Voici ce à quoi ressemble un mot de passe dans la base de données :

id	login	password
1	michel@gmail.com	098f6bcd4621d373cade4e832627b4f6

Si la base de données trouve un utilisateur, il le renvoie vers home.php avec un cookie 'username' ayant pour valeur le nom d'utilisateur rentré. Sinon il renvoie un message de mot de passe incorrect.

Page de visualisation

La page de visualisation est simple : il s'agit d'une page avec la visualisation sur une carte OpenStreetMap de la commande séléctionnée.

Cette page doit être appelée via un formulaire afin de pouvoir afficher les éléments.

Son code Php est le suivant :

```
if(isset($_POST['idcommande'])){
    $pdo=new
PDO('mysql:host=localhost;charset=utf8;dbname=db_SCHLEGEL_1','22201642'
,'329873');
    $sql = "select idcommande as 'N°commande', nom as Nom, prenom as
Prénom, Adresse, code_postal as 'Code postal', date_commande,
date_livraison, Status from commandes c join clients cli on
cli.idclient = c.idclient where idcommande = :idcommande";
    $req = $pdo->prepare($sql);
    $req->bindParam(':idcommande', $_POST['idcommande']);
    $req->execute();
    $res = $req->fetch(PDO::FETCH_ASSOC);

    $pdo2=new
PDO('mysql:host=localhost;charset=utf8;dbname=db_SCHLEGEL_1','22201642'
,'329873');
    $sql2 = "select longitude, latitude from commandes c join clients
cli on cli.idclient = c.idclient where idcommande = :idcommande";
    $req2 = $pdo2->prepare($sql2);
    $req2->bindParam(':idcommande', $_POST['idcommande']);
    $req2->execute();
```

```
    $res2 = $req2->fetch(PDO::FETCH_ASSOC);
    afficheDataTable($res);
}
else{
    echo "<h2> La commande recherchée n'existe pas ou a été
supprimée</p>";
}
```

Ce code vérifie d'abord si cette page a été appelée avec un formulaire qui envoie une variable POST nommée 'idcommande' vers cette page.

Note : voici un exemple de formulaire simple qui me permet de visualiser la carte de la commande numéro 1 :

```
<form action="visualisation.php" method="post">
    <input type="hidden" value="1" name="idcommande">
    <input type="submit" value="Voir Carte"></input>
</form>
```

Ce type de formulaire est notamment présent dans la page home.php, à coté des valeurs affichées dans le tableau.

Ensuite, il fait deux requètes SQL : Une pour ce qui doît être affiché sur le tableau de valeurs, et une pour récupérer les coordonnées GPS de la commande.

Le tableau est affiché via une fonction définie dans la page, nommée afficheDataTable() puis, les coordonnées GPS obtenues sont utilisées dans le script Javascript permettant d'afficher la carte.

Le script JavaScript est un script simple qui va initialiser dans un span la carte. Il a besoin du script Leaflet pour fonctionner, ainsi que de son CSS pour garantir un bon affichage de la carte.

Le code est le suivant :

```
 var map = null;
function initMap() {
    let mapOptions={center: [<?php echo $res2["latitude"] ?>,<?php echo
$res2["longitude"] ?>], zoom: 11};
    let layerOptions={attribution: '(c) OpenStreetMap France', minZoom:
1, maxZoom: 20};
    map = new L.map('map',mapOptions);
    let layer=new
L.tileLayer('https://{s}.tile.openstreetmap.fr/osmfr/{z}/{x}/{y}.png',l
ayerOptions);
    let marker = new L.Marker([<?php echo $res2["latitude"] ?>,<?php
echo $res2["longitude"] ?>]);
    L.control.scale().addTo(map);
    marker.bindPopup("Point de livraison");
    marker.addTo(map);
```

```
    map.addLayer(layer);
  }

window.onload=function(){
    initMap();
};
```

Le code affiche une carte centrée sur le point de livraison de la commande, et y affiche en plus un marqueur montrant le point exact de la livraison.

Les coordonnées GPS sont récupérées avec les balises `<?php echo $res2["latitude"] ?>` et `<?php echo $res2["longitude"] ?>`

Page d'ajout

La page d'ajout est une page qui contient un formulaire, et du code qui s'active lorsque le formulaire est rempli :

Formulaire :

```
<div id="filtre">
    <form action='' method='post'>
        <input type='text' name='idclient' placeholder='ID client'
required> <br>
        Date de commande :<input type='date' name='date_commande'
required/> <br>
    <input type='submit' value='Ajouter'/> </form></div>
```

Il s'agit d'un formulaire simple, ou il suffit de renseigner l'id du client à livrer et la date de commande. Ces deux éléments sont obligatoires pour l'ajout de commande (car la base de données n'accepte pas une idclient et une date de commande nulle)

Code PHP :

```
if (isset($_POST["date_commande"])) {
        $date1 = strtotime($_POST["date_commande"]);
        $date1 = date('Y-m-d', $date1);
        $pdo = new
PDO('mysql:host=localhost;charset=utf8;dbname=site','test','xd');
        $sql = "INSERT INTO commandes (idclient, date_commande, status)
VALUES (:cli, :date, 'Commandé')";
        $req = $pdo->prepare($sql);
        $req->bindParam(':cli', $_POST['idclient']);
        $req->bindParam(':date', $date1);
        $req->execute();
    }
```

Ce code vérifie si une des variables post du formulaire est initialisée (peu importe la quelle, les deux doivent être remplies de toute façon), puis convertit la date rentrée en une date

compréhensible par la base de données (le format date sous MySQL correspond à une donnée de type année-mois-jour). Il prépare la requète en associant :cli à l'id du client et :date à la date de commande rentrée.

Page d'édition :

La page d'éditon contient un formulaire avec une zone de texte, deux sélécteurs de date et une liste déroulante.

Elle contient également un tableau avec les valeurs de la commande à modifier, affiché exactement de la même façon que celui de la page de visualisation.

Les éléments du formulaire correspondent à des valeurs de la commande.

Le code pour modifier la command est le suivant :

```
if($_POST['date_commande']!=NULL||$_POST['idclient']!=NULL||$_POST['dat
e_livraison']!=NULL||$_POST['status']!=NULL){
        $pdo3 = new
PDO('mysql:host=localhost;charset=utf8;dbname=site','test','xd');
        $sql3 = "update commandes set ";
        if($_POST['idclient']!=NULL){
            $sql3 = $sql3." idclient =".$_POST['idclient'].",";
        }
        if($_POST['date_commande']!=NULL){
            $date1 = strtotime($_POST["date_commande"]);
            $date1 = date('Y-m-d', $date1);
            $sql3 = $sql3." date_commande ='".$date1."'";
        }
        if($_POST['date_livraison']!=NULL){
            $date2 = strtotime($_POST["date_livraison"]);
            $date2 = date('Y-m-d', $date2);
            $sql3 = $sql3." date_livraison ='".$date2."',";
        }
        if($_POST['status']!=NULL){
            $sql3 = $sql3." `status` ='".$_POST['status']."'";
        }
        $sql3 = $sql3." where idcommande =".$_POST['idcommande'];
        $pdoS = new
PDO('mysql:host=localhost;charset=utf8;dbname=site','test','xd');
        $pdoS->query($sql3);
    }
```

Il vérifie d'abord si l'un des POST du formulaire est non-nul, puis vérifie chaque variable POST envoyée et l'ajoute à une requète SQL qui est exécutée en fin de cycle.
La conversion de la date rentrée est la même que celle de la page d'ajout.

La requète SQL n’est pas préparée, celà est dû au fait que le nombre de variables n’est pas fixe et que je n’avais aucune idée de comment faire une requête préparée avec un nombre de variables qui n’est pas fixe.

Une fois la mise à jour terminée, la requète modifiée devrait être visible directement depuis cette même page web.