



BTS SIO – Option SLAM

Documentation d'épreuve

Rédacteur	Version	Date	Nb pages
Johann DIETRICH	1.3	15/02/2023	22

**Mise en place d'un système pour les
payements en ligne, dans le cas d'une
boutique d'E-commerce**

SOMMAIRE

1	CAHIER DES CHARGES	3
1.1	Introduction.....	3
1.2	Expression fonctionnelle du besoin	4
1.3	Contraintes.....	5
1.4	Gestion des droits d'accès.....	6
2	DESCRIPTION DES ENVIRONNEMENTS	7
3	METHODOLOGIE.....	8
3.1	Méthodologie et versioning.....	8
3.2	Gestion des tests de la solution	8
3.3	Gestion de projet	9
4	MISE EN OEUVRE	10
4.1	Connexion a des comptes	10
4.2	Page de création de compte.....	13
4.3	Modifications de la page d'affichage et création du panier.....	15
4.4	Utilisation de l'API STRIPE.....	19
5	GESTION DE LA MAINTENANCE (CORRECTIVE / EVOLUTIVE).....	21
5.1	Mise à jour de la documentation du SI.....	21
5.2	Evaluation de la qualité de la solution.....	21
6	BILAN DU PROJET.....	22
6.1	Validation des exigences point par point.....	22
6.2	Axes d'amélioration	22
6.3	Compétences acquises	22

1 Cahier des charges

1.1 Introduction

Type de mission
Mission effectuée en : a l'école, puis a été retravaillé durant un temps personnel.
Contexte
Création d'un petit site de vente d'articles en PHP
Demande du client
Le client souhaite pouvoir mettre en vente et vendre ses articles grâce a un site d'E-commerce
Budget disponible
Aucun réel budget disponible puisqu'il s'agit d'une simulation d'un vrai cas
Outils disponibles
Stripe API , PHP, Mysql

1.2 Expression fonctionnelle du besoin

Liste des fonctionnalités attendues :

Front office

- Page de panier

Permettant à chaque utilisateur de voir ses articles

- Page de connexion
- Page d'insertion d'informations de carte de crédit

Back Office

1.3 Contraintes

Générales

Projet total réalisé au cours de 1 mois et demi en classe dans le cadre d'apprentissage de PHP et MySQL

Juridiques

Techniquement, les information enregistrée dans la base de données devraient être protégés par les lois RGPD mais vu qu'il s'agit d'un projet en local, il n'y en a pas besoin.

Techniques

PHP, mySQL

Ergonomique

Il faut que le site soit compréhensible, aucune autre demande a été faite.

1.4 Gestion des droits d'accès

Administrateur
Dois pouvoir gérer la base de données en cas de problèmes

Utilisateurs
Peuvent mettre des articles dans leur panier et payer

2 Description des environnements

Environnement de développement

Environnement purement en local avec un serveur hosté par VScode Live

3 Méthodologie

3.1 Méthodologie et versioning

Utilisation de GitHub pour le Versionning.

3.2 Gestion des tests de la solution

expliquer comment vous gérez les tests

unitaires

Ajout d'un compte utilisateur, ajout de cet utilisateur d'un article au panier, confirmation du panier par cet utilisateur, puis insertion des données demandées afin de payer pour l'article.

3.3 Gestion de projet

3.3.1 Budget

Aucun réel budget nécessaire vu que l'API est gratuite d'utilisation

4 Mise en oeuvre

Avant de commencer, il est important a noter que l'on va reprendre des éléments créés précédemment dans une autre mission E4 afin de simplifier la tâche. L'on va donc reprendre la partie de présentation d'articles de la mission E4 « E4-Mise en place d'un système d'affichage, d'ajout, de suppression et de modification d'articles de vente », c'est-à-dire, ceci :

Nom	Référence	Prix
marteau de menuisier bois verni	81968453	8.90 €
marteau massette fibre de verre	80166978	21.90 €
maillet de menuisier bois	82039106	14.90 €
marteau arrache-clou	81968500	12.90 €
tournevis électricien plat	74936295	1.95 €
tournevis électricien isolé plat	67337361	5.10 €

Notre objectif sera de créer donc un panier atteignable via un compte utilisateur, et ensuite de pouvoir donc payer de manière réaliste.

4.1 Connexion a des comptes

Pour commencer, notre première étape sera de créer des comptes utilisateur qui seront donc capables de se servir du panier. Pour cela il nous faut créer une structure de base de données comme ceci :

id	prenom	nom	email	date_naissance	adresse	ville_id	parrain_id	mot_de_passe	administrateur
1	Melville	DESFORGES	Melville.DESFORGES@laposte.fr	1965-02-09	96 boulevard d'Alsace	1	NULL	a	1

On n'utilisera uniquement l'adresse mail et le mot de passe pour se connecter mais les autres informations sont la purement par réalisme des informations normalement demandées d'un site d'e-commerce.

Maintenant, il faut nous créer la page de connexion pour les utilisateur, celle-ci pouvant être faite très facilement avec Bootstrap pour la forme et simplement avec des requêtes SQL pour le fonctionnement. Ceci donne donc :

Se connecter

Adresse mail

Adresse mail

Mot de passe

Mot de passe

Se connecter

Pas encore de compte ?

```

<center> <!-- formulaire de connection -->
    <div class="form_signup form-login">
        <div class="formtitle">Se connecter</div>
        <br><br>
        <form action="connection_util.php" name="form" id="form" method="post">
            <div class="form-group">
                <label for="username">Adresse mail</label>
                <input type="text" class="form-control" id="username"
name="username" placeholder="Adresse mail" required>
            </div>
            <div class="form-group">
                <label for="password">Mot de passe</label>
                <input type="password" class="form-control" id="password" name="password"
placeholder="Mot de passe" required>
            </div>
            <button type="submit" class="btn btn-primary">Se connecter</button>
        </form>
    </div>
    <div><a href="creation_compte.php">Pas encore de compte ?</a></div>
</center>

```

Un formulaire vu et revu pour simplement insérer des données, pour appeler une fonction qui permet de vérifier et de se connecter en SQL après. Avec comme petite particularité d'avoir ajouté un bouton redirigeant vers notre futur page de création de compte.

```

<?php
session_start();
include_once "connection.php";
include_once "fonctions.php";

if(!empty($_POST)){ /* vérifie si le formulaire a été envoyé */
    connection_util($conn);
}
?>

```

La page appelée appelle donc une autre fonction, tout en démarrant la connexion a la BDD grâce au fichier « connection.php » qui contient les identifiants pour se connecter celle-ci.

Notre fonction de connection_uti faisant cela :

```
if (password_verify($_POST["password"], $passwd['mot_de_passe']) ){
    $stmt = $conn->prepare("SELECT * FROM clients WHERE email = :username");
    /* permet de chercher la colone"droits" determinant si un utilisateur est administrateur
    ou non */
    $stmt->execute(['username' => $_POST["username"]]);
    $user = $stmt->fetch();
    $droits = $user['administrateur']; //vérifiie dans la BDD si
l'utilisateur a des droits administrateurs
    $_SESSION['id'] = $user['id'];
    $_SESSION["username"] = $_POST["username"]; /* Crée les variables de
sessions permettant donc de confirmer la connection et plus */
    $_SESSION["droits"] = $droits;
    $_SESSION['conn'] = "Bienvenue ". $_POST["username"]."!";
    //creation de logs
    header("location:articles.php");
    exit;
}
else
{
    echo "Mauvais mot de passe ou nom d'utilisateur";
}
```

Celle-ci vérifie donc le mot de passe (hashé bien évidemment) vérifie si le nom d'utilisateur existe bien, et si c'est le cas, des valeurs du compte sont associées a des variables session permettant d'être gardées pour de futur utilisations.

4.2 Page de création de compte

Page qui va également être vu très rapidement car elle n'apporte pas beaucoup de fonctionnalités intéressantes.

Celle-ci sera constitué d'un formulaire mis en forme par Bootstrap et utilisera une fonction PHP pour insérer les informations du compte dans la BDD. Le tout ressemblant à ça :

Créer un compte

Prenom

nom

Nom

nom

Adresse Mail

email

Date de naissance

1998-02-25

Adresse

adresse

Ville

VÉLIZY-VILLACOUBLAY

Parrain

prenom_du_parrain nom_du_parrain

Mot de passe

Mot de passe

Créer le compte !

```
<div class="form_signup form-login">
    <div class="formtitle">Créer un compte</div>
    <br><br>
    <form action="crea_util.php" name="form" id="form" method="post">
    <div class="form-group">
        <label for="prenom">Prenom</label>
        <input type="text" class="form-control" id="prenom"
name="prenom" placeholder="prenom">
    </div>
<label>Ville </label><br>
    <select name="ville" class="form-select choisir_ville" aria-label="Default
select example">
    <?php
        $data = $conn->query("SELECT * FROM villes")->fetchAll();
        foreach ($data as $row)
        {
            echo "<option value=$row[id]> $row[nom] </option>";
        }
    <?>
    </select>
</div>
```

Le formulaire étant comme celui de connexion a la seule exception qu'on a créé un petit sélecteur, permettant a l'utilisateur de choisir sa ville parmi une liste prédéfinie dans la BDD.

Et voici la fonction utilisée pour créer un compte, qui est comprise d'une vérification pour qu'aucune partie du formulaire est vide, de vérification d'un parrain (s'il existe) et finalement de l'insertion des données dans la BDD.

```
if ($_POST['prenom'] == "" OR $_POST['nom'] == "" OR $_POST['email'] == "" OR
$_POST['naiss'] == "" OR $_POST['adresse'] == "" OR $_POST['ville'] == "" OR
$_POST['password'] == "" ){
    $resultat['message'] = "Veuillez remplir les champs obligatoires";
    exit;
}
else {
    $id_de_ville = $_POST['ville'];

    if($_POST['parrain'] == ""){ //vérifie si le parrain existe, et si c'est le cas, le
lie a l'utilisateur
        $id_de_parrain = NULL;
    }
    $data_util = [
        'prenom' =>$_POST['prenom'],
        'nom' =>$_POST['nom'],
        'email' =>$_POST['email'],
        'date_naissance' =>$_POST['naiss'],
        'adresse' =>$_POST['adresse'],
        'ville_id' =>$id_de_ville,
        'parrain_id' =>$id_de_parrain,
        'mot_de_passe' =>password_hash($_POST['password'],PASSWORD_DEFAULT),
        'administrateur' => NULL
    ];
    $sql = "INSERT INTO clients (prenom, nom, email, date_naissance, adresse,
ville_id, parrain_id, mot_de_passe, administrateur) VALUES (:prenom, :nom, :email,
:date_naissance, :adresse, :ville_id, :parrain_id, :mot_de_passe, :administrateur)";
//Insertion des données
    $conn->prepare($sql)->execute($data_util);
    header("location:panier.php");
    exit;
}
```

4.3 Modifications de la page d'affichage et création du panier

La page d'affichage d'article doit être légèrement modifiée afin d'avoir la capacité d'ajouter un objet dans le panier de manière pratique. Ceci est fait de cette façon :

```
<table class="table" id="articles">
  <thead>
    <tr>
      <th scope="col">Nom</th> <!-- défini la forme du tableau -->
      <th scope="col">Référence</th>
      <th scope="col">Prix</th>
      <?php if (isset($_SESSION['id'])) {
        echo "<th scope='col'>Ajouter au panier</th>";
      } ?>
    </tr>
  </thead>

  <tbody>
    <?php foreach ($data as $article) {
      echo "<tr>";
      echo "<th> $article[nom]</th>";
      echo "<th> $article[reference]</th>";
      echo "<th> $article[prix_ht] € </th>";
      if (isset($_SESSION['id'])) {
        echo "<th><form action='insertion_panier.php' class='panier' method='post'>
          <button name='valpanier' value='$article[id]'>Ajouter</button></form></th>";
      }
      echo "</tr>";
    }
  </tbody>
</table>
```

Nom	Référence	Prix	Ajouter au panier
marteau de menuisier bois verni	81968453	8.90 €	<input type="button" value="Ajouter"/>

Juste une petite condition permettant la création d'un bouton d'ajout au panier si le site remarque que l'utilisateur est connecté (et qu'il a une ID de session). Celui-ci permettant de rediriger vers la page de panier et de garder les données de l'article sélectionné.

Avant de créer le panier il nous faut créer la structure de la table qui sera utilisée pour celui-ci.

id	client_id	article_id	quantite
1	1	1	1

Celle-ci mettant donc en lien le client, l'article et la quantité de celui-ci, qui vont être utilisés sur la page de panier.

Pour le bon fonctionnement du panier il nous faut, un tableau qui grandit pour chaque article inséré par l'utilisateur, un bouton capable de modifier la quantité ou de supprimer l'article du panier et un bouton pour valider la commande affichant le total. Tout ceci est fait ci-dessous

```
<table class="table" id="articles">
    <thead>
        <tr>
            <th scope="col">Nom</th>
            <th scope="col">Prix</th>
            <th scope="col">Quantité</th>
            <th scope="col">Modifier</th>
        </tr>
    </thead>
    <tbody>
        <?php
            $prix_total = 0;
            $quantite_total = 0;
            $num_article_actuel = 0; //compteur pour les id des articles
            //selectionne l'id des articles dans le panier
            $obtention_id_article = $conn->prepare("SELECT * FROM panier WHERE client_id = ?");
            $obtention_id_article->execute([$SESSION['id']]); //lie le panier avec la session de l'utilisateur
            $id_articles = $obtention_id_article->fetchAll();
            foreach ($id_articles as $obtention_id_article) {
                //utilise les id récupérés pour chercher dans la table "articles" les données de chacun de ceux-ci
                $obtention_infos_article = $conn->prepare("SELECT * FROM articles WHERE id = ?");
                $obtention_infos_article->execute([$id_articles[$num_article_actuel]['article_id']]);
                $infos_article = $obtention_infos_article->fetchAll();
                $num_article_actuel +=1;
                foreach ($infos_article as $obtention_infoss_article) { //affiche une ligne par article avec le nom, le prix et la quantité
                    $quantite_total += $obtention_id_article['quantite'];
                    $prix_total+=$obtention_infoss_article['prix_ht']*$obtention_id_article['quantite'];
                }
            }
        </tbody>
    </table>
```

Cette partie du code permettant l'affichage simple grâce a une requête SQL de tout les articles qui sont lié a l'utilisateur.

Nom	Prix	Quantité
marteau de menuisier bois verni	8.90 €	1

```

        echo "<tr>";
        echo "<th> $obtention_infoss_article[nom]</th>";
        echo "<th> $obtention_infoss_article[prix_ht] € </th>";
        echo "<th> $obtention_id_article[quantite] </th>";
        echo "<th> <div class='modifier_article_css'>
                <form action='supression_panier.php' class='panier'
method='post'>
                    <button name='id_article'
value='$obtention_infoss_article[id]'>Supprimer</button></form>

                    <form action='enlev_quant_panier.php' class='panier'
method='post'>
                    <button name='id_article'
value='$obtention_infoss_article[id]'>-</button></form>

                    <form action='ajout_quant_panier.php' class='panier'
method='post'>
                    <button name='id_article'
value='$obtention_infoss_article[id]'>+</button></form>

                </div>
                </th>";
        echo "</tr>";
    }
}
?>

```

Cette partie s'occupe donc d'ajouter les boutons fonctionnels permettant d'ajouter, supprimer ou enlever un article.

Modifier

Et finalement, la dernière partie permettant la validation et l'affichage du prix total de tous les articles sélectionnés.

```
<?php      echo "<div class='partie_panier'><div class='prix_total'> Le prix total
serait de : " . $prix_total. ' euros <br> Quantité : ' . $quantite_total. '</div>';
      echo "<form action='valider_panier.php' class='panier' method='post'>
      <button name='id_util' class='btn btn-
primary' value=\".$_SESSION['id'].\">Valider le panier</button></form></div></div>";

      if (isset($_SESSION['alert'])) {
          echo "<div class='alert alert-danger'
role='alert'>".$_SESSION[alert]</div>";
      }
      unset($_SESSION['alert']);

      if (isset($_SESSION['info_panier'])) {
          echo "<div class='alert alert-success'
role='alert'>".$_SESSION[info_panier]</div>";
      }
      unset($_SESSION['info_panier']);
  }

?>
```

Le prix total serait de : 8.9 euros
Quantité : 1

Valider le panier

Il est à noter que chaque bouton fait parti d'un petit formulaire existant uniquement pour appeler une fonction permettant d'effectuer le besoin demandé.

4.4 Utilisation de l'API STRIPE

Celle-ci est indispensable pour effectuer quelconque paiement en ligne pour notre site.

Tout d'abord, il faut installer la bibliothèque Stripe en utilisant Composer en le mettant dans un header de préférence:

```
composer require stripe/stripe-php
```

Après cela, il faudra créer un compte sur le site de STRIPE, ceci afin d'obtenir une clé API permettant d'utiliser celle-ci.

```
require_once('vendor/autoload.php');

\Stripe\Stripe::setApiKey('sk_test_XXXXXXXXXXXXXXXXXXXX');
```

Une fois que l'on a obtenu celle-ci on peut mettre en place ce bout de code pour 'autoriser' l'API de fonctionner.

La prochaine partie est assez simple, il suffit de créer un formulaire et la fonction dès que le formulaire est envoyé permettant de donner les informations à l'API et de faire le retrait d'argent.

```
<form action="charge.php" method="POST">
  <input type="text" name="name" placeholder="Nom complet">
  <input type="email" name="email" placeholder="Adresse email">
  <input type="text" name="card_number" placeholder="Numéro de carte de crédit">
  <input type="text" name="exp_month" placeholder="Mois d'expiration (MM)">
  <input type="text" name="exp_year" placeholder="Année d'expiration (YYYY)">
  <input type="text" name="cvc" placeholder="Code de sécurité (CVC)">
  <input type="submit" value="Payer">
</form>
```

Nom complet	Adresse email	Numéro de carte de crédit	Mois d'expiration (MM)	Année d'expiration (YYYY)	Code de sécurité (CVC)	Payer
-------------	---------------	---------------------------	------------------------	---------------------------	------------------------	-------

Cette partie permettant donc d'appeler la page charge.php qui s'occupe de faire passer le paiement.

```
<?php
require_once('vendor/autoload.php');

\Stripe\Stripe::setApiKey('sk_test_XXXXXXXXXXXXXXXXXXXX');

$token = $_POST['stripeToken'];
$amount = $_POST['amount'];

try {
    $charge = \Stripe\Charge::create([
        'amount' => $amount,
        'currency' => 'eur',
        'source' => $token,
        'description' => 'Paiement pour un produit'
    ]);
    $_SESSION['info_panier'] = "Commande effectuée avec succès ! ";
    $sql = "DELETE FROM panier WHERE client_id=?"; //supprime la panier quand tout est fini
    $stmt= $conn->prepare($sql);
```

```
$stmt->execute([$id_utilisateur]);  
header("location:panier.php");  
exit;  
  
} catch (\Stripe\Exception\CardException $e) {  
    $_SESSION['info_panier'] = "La commande a échoué, veuillez recommencer";  
    header("location:panier.php");  
    exit;  
}  
?>
```

Elle même qui donc récupère les informations, les envoie à l'API pour récupérer le paiement puis envoie un message de session pour dire si celui-ci a bien fonctionné ou s'il a échoué. Tout en supprimant le panier si l'achat a bien été effectué.

5 Gestion de la maintenance (corrective / évolutive)

5.1 Mise à jour de la documentation du SI

Recueillir, analyser et mettre à jour les informations sur une version d'une solution

Normalement pas de mise à jour à faire, la documentation reste stable, le seul point potentiellement changeant serait l'utilisation de l'API, si celle-ci est décidée qu'elle change de manière de fonctionner, de nouvelles documentations et de la re vérification serait d'ordre.

5.2 Evaluation de la qualité de la solution

Évaluer la qualité d'une solution

Elle effectue ce qui est demandé, c'est-à-dire avoir des utilisateurs pouvant choisir des articles et pouvant les acheter à l'aide de leur carte de crédit.

6 Bilan du projet

6.1 Validation des exigences point par point

Création d'un panier

Création de comptes utilisateurs afin d'utiliser le panier

Capaciter de vendre des articles mis dans le panier

6.2 Axes d'amélioration

Potentiellement ajouter plus de méthodes d'achat, tel que PayPal.

6.3 Compétences acquises

Apprentissage d'utilisation d'une API

Différente fonctionnalités liés a la création de comptes

Utilisation d'une base de données afin de lier plusieurs tables ensemble