

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 C/	AHIER 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 DE	ESCRIPTION DES ENVIRONNEMENTS	7
3 MI	ETHODOLOGIE	8
3.1	Méthodologie et versioning	8
3.2	Gestion des tests de la solution	8
3.3	Gestion de projet	9
4 MI	ISE 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 GI	ESTION 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 BI	LAN DU PROJET	22
6.1	Validation des exigences point par point	22
6.2	Axes d'amélioration	22
63	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

l ic	oto.	doc	fonc	tionn	alitác	attendues
L١٤	ste	ues	ionc	uonn	ames	allendues

F	ro	nt	off	ica
	u		OH	

- 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 purement en local avec un serveur hosté par VScode Live

Environnement de développement		

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.

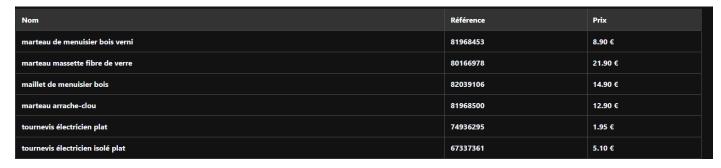
3.3 Gestion de projet

3.3.1 Budget

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

4 Mise en oeuvre

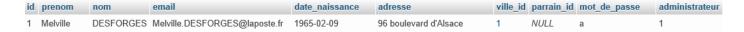
Avant de commencer, il est important a noter que l'on va reprendre des éléments crées précédemment dans une autre mission E4 afin de simplifier la tâche. L'on va donc reprendre la partie de présentation d'articles de la mission E4 « E4-Mise en place d'un système d'affichage, d'ajout, de suppression et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



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

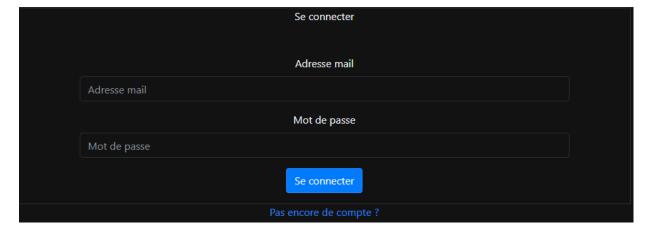
4.1 Connexion a des comptes

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 :



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 :



```
<center> <!-- formulaire de connection -->
        <div class="form signup form-login">
            <div class="formtitle">Se connecter</div>
            <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"</pre>
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"</pre>
placeholder="Mot de passe" required>
            </div>
            <button type="submit" class="btn btn-primary">Se connecter
            </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($_POST)){    /* vérifie si le formulaire a été envoyé */
     connection_uti($conn);
  }
}</pre>
```

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 :

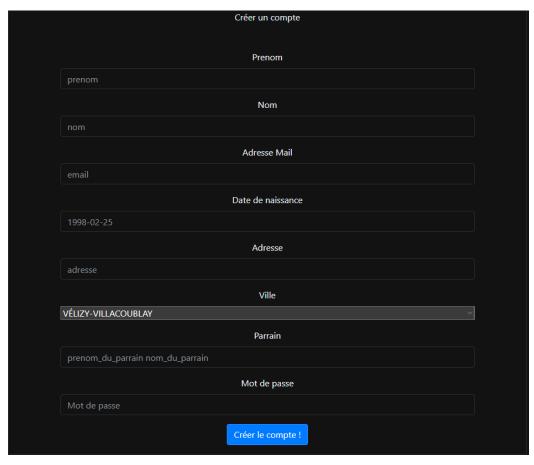
```
if (password_verify($_POST["password"], $passwd['mot_de_passe']) ){
                $stmt = $conn->prepare("SELECT * FROM clients WHERE email = :username");
/* permet de chercher la colone"droits" determinant si un utilisateur est administrateur
ou non */
                $stmt->execute(['username' => $_POST["username"]]);
                     $user = $stmt->fetch();
                     $droits = $user['administrateur']; //vérifiie dans la BDD si
                     $ 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 :



```
<div class="form_signup form-login">
            <div class="formtitle">Créer un compte</div>
            <br><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"</pre>
name="prenom" placeholder="prenom" >
            </div>
<label>Ville </label><br>
            <select name="ville" class="form-select choisir_ville" aria-label="Default"</pre>
select example">
                $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";
   else {
        $id_de_ville = $_POST['ville'];
        if($_POST['parrain']==""){  //vérifie si le parrain exite, 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 donnés
        $conn->prepare($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 :

```
Nom <!-- défini la forme du tableau -->
        Référence
        Prix
   <?php if (isset($_SESSION['id'])){</pre>
         echo "Ajouter au panier";
      </thead>
      foreach ($data as $article) {
        echo "";
        echo " $article[nom]";
        echo " $article[reference]";
        echo " $article[prix_ht] € ";
        if (isset($_SESSION['id'])){
         echo "<form action='insertion_panier.php' class='panier' method='post'>
         <button name='valpanier' value='$article[id]'>Ajouter</button></form>";
        echo "";
      }
                                   Référence
                                               Prix
                                                        Ajouter au panier
marteau de menuisier bois verni
                                   81968453
                                               8.90 €
                                                        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 quantite1 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

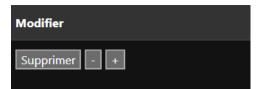
```
<thead>
          Nom
              Prix
              Quantité
              Modifier
          </thead>
       <?php
          $prix total = 0;
          $quantite total = 0;
          $num_article_actuel = 0; //compteur pour les id des articles
          //selectionne l'id des articles dans le 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
              $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_articl
e['quantite'];
```

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 "";
                    echo " $obtention_infoss_article[nom]";
                    echo " $obtention_infoss_article[prix_ht] € ";
                    echo " $obtention_id_article[quantite] ";
                    echo " <div class='modifier_article_css'>
                            <form action='supression_panier.php' class='panier'</pre>
method='post'>
                            <button name='id_article'</pre>
value='$obtention_infoss_article[id]'>Supprimer</button></form>
                            <form action='enlev_quant_panier.php' class='panier'</pre>
method='post'>
                            <button name='id_article'</pre>
value='$obtention_infoss_article[id]'>-</button></form>
                            <form action='ajout_quant_panier.php' class='panier'</pre>
method='post'>
                            <button name='id_article'</pre>
value='$obtention_infoss_article[id]'>+</button></form>
                            </div>
                            ";
                    echo "";
                }
            }
```

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



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

```
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-</pre>
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 a 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 celleci.

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 a l'API et de faire le retrait d'argent.

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

```
$stmt->execute([$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 a 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 a jour a faire, la documentation reste stable, le seul point potentiellement changeant serait l'utilisation de l'API, si celle-ci est décidé 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 utilisateur pouvant choisir des articles et pouvant les acheter a 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