3.1.1. TP7 : Analyse d'un système de gestion de commandes

A l'aide du logiciel AnalyseSI vous réaliserez l'étude d'un système de gestion de commandes pour un magasin. Bien entendu, vous prendrez *InnoDB* comme moteur de la BDD MySql. Puis vous le coderez en PHP.

Objectifs :

- Réaliser le modèle MCD à l'aide du logiciel AnalyseSI ;
- Importer le modèle généré dans MySql ;
- Entrer des valeurs fictives afin d'alimenter la base ;
- Définir les contraintes sur les clés étrangères entre les tables et les tester ;
- Vous créerez un *dump* de la BDD.
- Enfin, en vous aidant du code *pdo-master* vous réaliserez la création automatique de la BDD, puis les formulaires : d'ajout de client, de produit et de commande.

Déroulement.

- Travailler par groupe de 2 étudiants (maxi 3 seulement s'il reste un(e) étudiant(e) seul(e)) ;
- Durée de 2 heures ;
- Consigner les résultats dans un compte-rendu clair et propre (éventuellement un document partagé).

Vous pouvez donc vous partager le travail mais ...

Vous devez avoir tout compris de l'intégralité du TD, la suite (examen final + note de TP) dépend de votre compréhension globale.

Les autres TD ... Et les TP ...

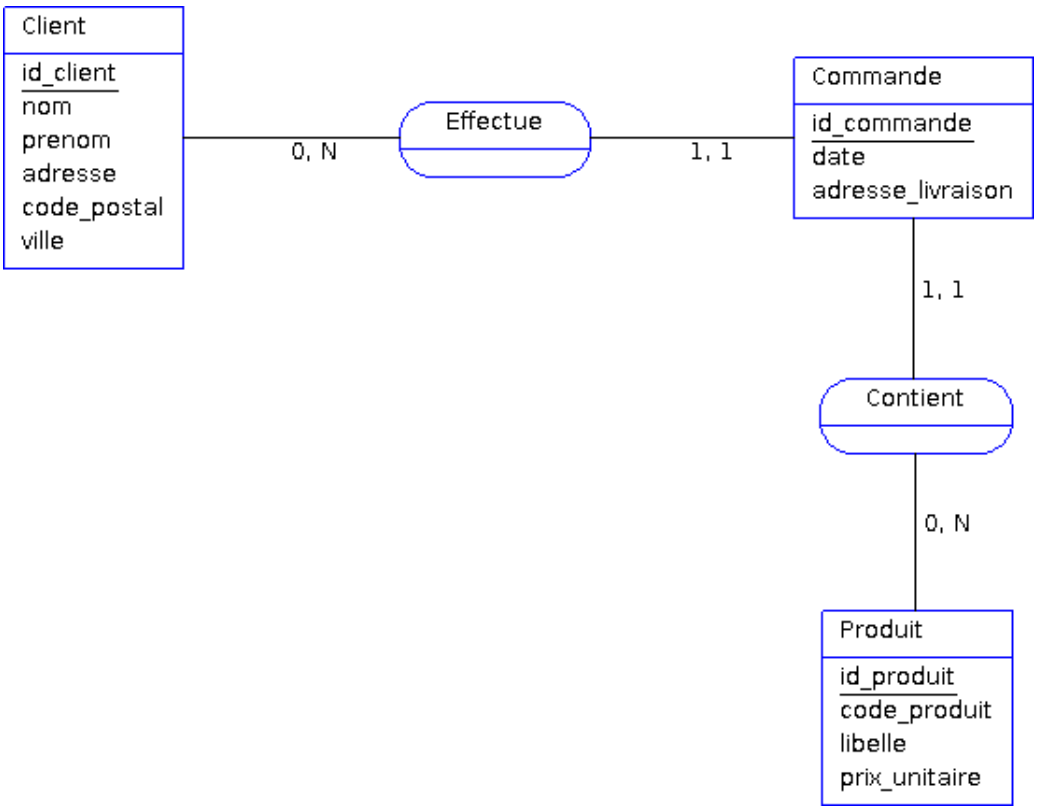
Les deux autres TD dépendent du TD1, ils sont la suite ... Les TP c'est différent, c'est pas le même problème qui est posé, mais tout le code que vous aurez fourni en TD vous sera utile.

3.1.1.1. Etude du système de gestion de commandes (MCD)

A la fin de cette étude vous devez fournir un *dump* de la BDD (en utilisant phpmyadmin) qui sera l'entrée de la réalisation qui suit.

Les règles de gestion :

- Le magasin vend des produits à des clients ;
- Les produits possèdent une référence (un code), un libellé et un prix unitaire ;
- Les clients ont une identité (nom, prénom, adresse ...) ;
- Les clients passent des commandes de produits. On mémorise la date de la commande (0 à N commandes) ;
- Pour chaque commande, le client précise une adresse de livraison ;
- La commande concerne un produit ;
- Un produit est concerné par 0 à N commandes.



Les contraintes sur les clés étrangères :

- Si je supprime un client (de la base) la commande qu'il a effectuée est aussi supprimée ;
- Si je mets à jour le champ `id_client` de la table client, le champ correspondant de la table commande est mis à jour automatiquement ;
- Si je supprime un produit (de la base) la commande qu'il a effectuée est aussi supprimée.
- Si je mets à jour le champ `id_produit` de la table produit, le champ correspondant de la table commande est mis à jour automatiquement ;

3.1.1.2. Réalisation de l'application PHP

Je vous conseille vivement de suivre la démarche proposée ci-dessous. Vous pouvez récupérer le code de *pdo-master* ici : GITHUB [https://github.com/taniarascia/pdo/tree/feba19a43e63c617c95a1f6e68825f6e3086336c] ou sur Moodle.

Démarche à suivre :

- Vous avez un script dans la machine virtuelle permettant de créer un compte de base de données, utilisez le !
- Avec phpmyadmin créer une table fictive (avec des champs ressemblant à ce qui est demandé ...) faites en un *dump* (ceci permet de travailler sans avoir encore la première partie du TD de fonctionnelle) ;
- Tester la création automatique de la BDD avec le code proposé dans *pdo-master* ;
- Pour votre compréhension lisez les exmplications données sur *pdo-master* sur le site [https://www.taniarascia.com/create-a-simple-database-app-connecting-to-mysql-with-php/] ;
- Réaliser les formulaires d'ajout de client, de produit et de commande en vous basant sur le code existant. Ci-dessous, les copies d'écrans de la réalisation des différents formulaires.

Figure 3.1. Formulaire principal

Figure 3.2. Formulaire d'ajout d'un client

Figure 3.3. Formulaire d'ajout d'un produit

Figure 3.4. Formulaire permettant de commander

3.1.2. TP8 : Incorporer un framework CSS & javascript au système de gestion de commandes

Maintenant que vous avez modifié *pdo-master* vous allez améliorer le rendu de celui-ci.

Objectifs :

- Installer Foundation (ZURB Foundation [https://foundation.zurb.com/]);
- Copier l'ancien *magasin* dans le *nouveau* ;
- Coder toutes les interfaces et formulaires en utisant le fichier `index.html` ;
- Modifier les fichiers `header.php` et `footer.php` afin qu'ils tiennent compte de Foundation ;
- Modifier alors toutes les interfaces.

Déroulement.

- Travailler par groupe de 2 étudiants (maxi 3 seulement s'il reste un(e) étudiant(e) seul(e)) ;
- Durée de 2 heures ;
- Consigner les résultats dans un compte-rendu clair et propre (éventuellement un document partagé).

Vous pouvez donc vous partager le travail mais ...

Vous devez avoir tout compris de l'intégralité du TD, la suite (examen final + note de TP) dépend de votre compréhension globale.

Les autres TD ... Et les TP ...

Les deux autres TD dépendent du TD1, ils sont la suite ... Les TP c'est différent, c'est pas le même problème qui est posé, mais tout le code que vous aurez fourni en TD vous sera utile.

3.1.2.1. Installation de Foundation

Installer Foundation n'est pas forcément une chose aisée, car il y a un grand nombre de méthode d'installation. Nous nous choisirons d'installer le script **foundation** permettant d'utiliser facilement par la suite le compilateur **sass**.

Quelques indications pour l'installation :

- Vérifier que la commande **`foundation new`** est installé, si oui vous passez directement à la suite ;
- Installer **`npm`** puis **`foundation`** :

```
[root@machina]# apt install npm
[root@machina]# npm i browser-sync --save
[root@machina]# npm rebuild node-sass
[root@machina]# npm install -g bower # pour pouvoir utiliser après : foundation new
 newqqc
[root@machina]# npm install --global foundation-cli
```

`foundation` n'est pas installé via apt

Mais via **npm** ...

3.1.2.2. Configuration de Foundation

Pour bien comprendre comment fonctionne Foundation, je vous conseille de regarder la vidéo sur le système de grille (C.F. XY Grid [https://foundation.zurb.com/sites/docs/xy-grid.html]). Ici vous devez utiliser le dernier système de grille disponible (inutile d'aller chercher à utiliser un autre système qui finira par devenir obsolète). Le système de grille est surement la chose la plus importante à comprendre afin d'être à l'aise avec les copiés/collès que vous finirez par faire inévitablement.

Démarche à suivre :

- Maintenant, je vous conseille de modifier le nom du répertoire de travail *de votre magasin* en `magasin_v0` au même niveau grâce à la commande **`foundation new`** vous allez créer un autre répertoire `magasin_v1`, celui-ci contiendra tous les fichiers (**`foundation`** + `magasin_v0`). Vous devez avoir cette arborescence :

```
[user@machina]$ pwd
/home/eleve/public_html/magasin     # ceci est un exemple !
[user@machina]$ ls
magasin_v0
[user@machina]$ foundation new

foundation new
? What are you building today? A website (Foundation for Sites)
? What's the project called? (no spaces) magasin_v1
? Which template would you like to use? Basic Template: includes a Sass compiler

            .
           /|     ,
      , /|/  \/| /|       Thanks for using ZURB Foundation for Sites!
     /|/       |/ |       ----------------------------------------
 |___|            |___|   Let's set up a new project.
 \___|  ^^   ^^   |___/   It shouldn't take more than a minute.
     | -[O]--[O]- |

     |    ___,    |
     |    ...     |
      \__________/


Downloading the project template...

Done downloading!


Installing dependencies...

You're all set!


 . New project folder created.
 . Node modules installed.
 . Bower components installed.

Now run foundation watch while inside the magasin_v1 folder.

[user@machina]$ ls
magasin_v0
magasin_v1
[user@machina]$ cd magasin_v1
```

- Comparez les noms des fichiers et répertoires puis renommer éventuellement certains fichiers afin que la recopie se passe bien ;
- Par exemple faites ceci :

```
[user@machina]$ pwd
/home/eleve/public_html/magasin/magasin_v1
[user@machina]$ cp ../magasin_v0/* .
```

- Il est préférable d'utiliser la page `index.html` de Foundation afin de réaliser toutes les interfaces, formulaires et page d'accueil. Car en l'utilisant le navigateur est mise à jour automatiquement.

- Modifier dans un premier temps les fichiers `header.php` et `footer.php` afin qu'ils tiennent compte de Foundation. Pour cela, prenez exemple en regardant le fichier `index.html` ;

- Modifier les formulaires d'ajout de client, de produit et de commande en vous basant sur le code existant. Ci-dessous, les copies d'écrans de la réalisation des différents formulaires.

Figure 3.5. Formulaire principal

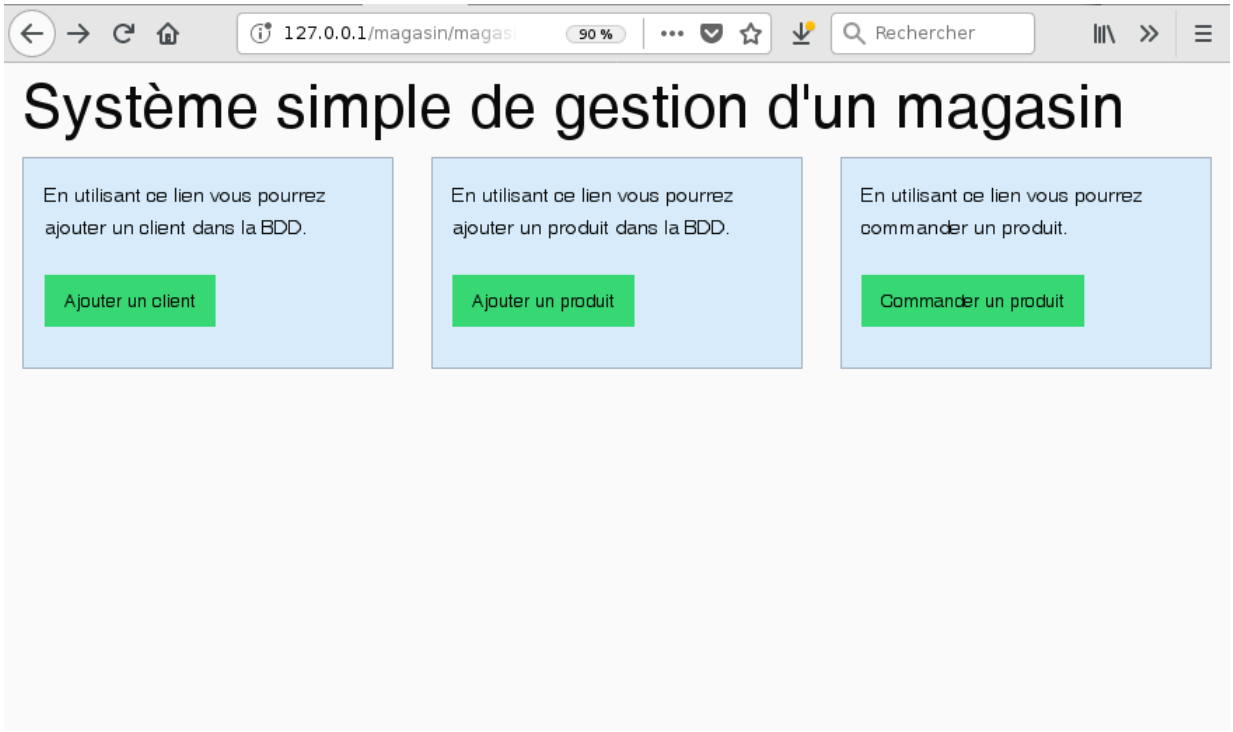


Figure 3.6. Formulaire d'ajout d'un client

Figure 3.7. Formulaire d'ajout d'un produit

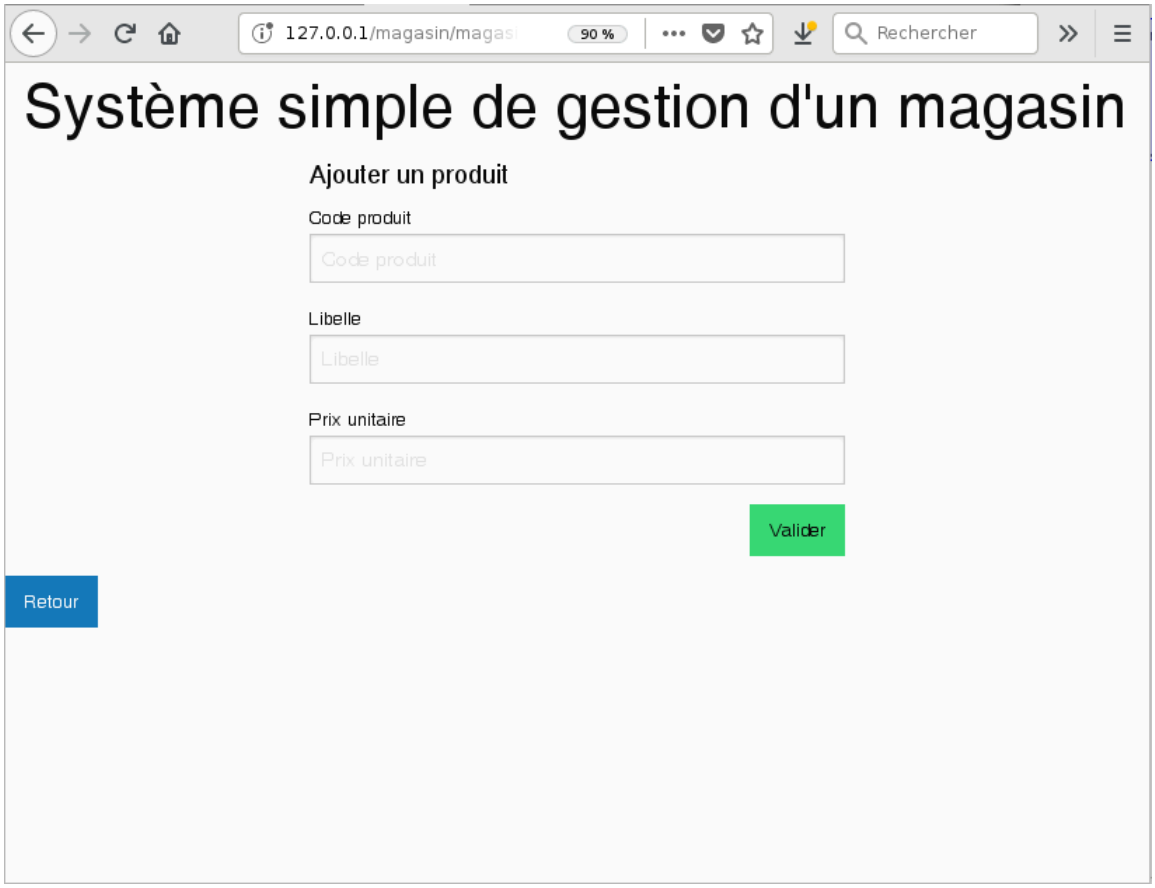
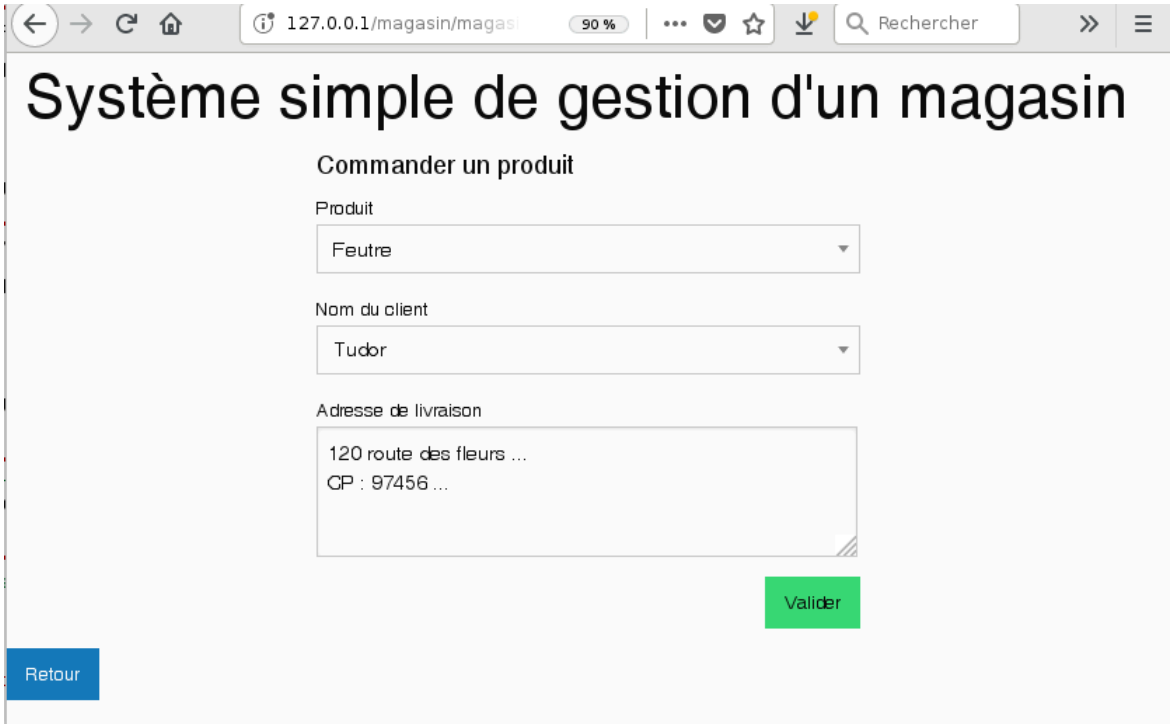


Figure 3.8. Formulaire permettant de commander



3.1.3. TP9 : Amélioration et finalisation du système de gestion de commandes

Oui, mais avons nous tout codé ? Avons nous terminé ? Est ce que tout cela est d'une qualité suffisante pour une utilisation professionnelle ? La réponse est non ! Dans ce cas, commencez à créer une copie de votre travail `magasin_v1` ...

Objectifs :

- Dans un premier temps, modifier le code PHP, car il y a des choses qui font très mal aux yeux ... Des balises ouvrantes, suivies de balises fermantes, puis à suivre une autre ouverture de balise. Il faut limiter ce genre de pratique (C.F. document de cours) ;
- Après avoir nettoyé le code, il nous manque quelque chose : une interface de suppression des clients, produits ...
- Retrouver l'adresse de livraison d'une commande, il manque cette interface, en choisissant le client, une liste déroulante montrant tous les produits commandés par ce client est mise à jour automatiquement. En validant le formulaire, l'adresse de livraison s'affiche. Cela semble simple, mais la seconde liste déroulante est liée à la première, dans ce cas il faut utiliser une requête Ajax !
- On a pas du tout modifié le thème par défaut de Foundation et franchement c'est pas terrible !
- Utilisation d'un moteur de template comme Smarty (https://www.smarty.net/docsv2/fr/ [https://www.smarty.net/docsv2/fr/]. **C'est pas une obligation**, mais si vous voulez tenter l'aventure, je valide à 100% ! J'en tiendrais compte inévitablement d'une manière ou d'une autre dans la note finale.

Déroulement.

- Travailler par groupe de 2 étudiants (maxi 3 seulement s'il reste un(e) étudiant(e) seul(e)) ;
- Durée de 2 heures ;
- Consigner les résultats dans un compte-rendu clair et propre (éventuellement un document partagé).

Vous pouvez donc vous partager le travail mais ...

Vous devez avoir tout compris de l'intégralité du TD, la suite (examen final + note de TP) dépend de votre compréhension globale.

Les autres TD ... Et les TP ...

Les deux autres TD dépendent du TD1, ils sont la suite ... Les TP c'est différent, c'est pas le même problème qui est posé, mais tout le code que vous aurez fourni en TD vous sera utile.

3.1.3.1. Amélioration du code PHP

Voici un exemple de ce qu'il faut faire ... En gros, on a travaillé jusqu'à présent en mode brouillon, il faut nettoyer le code, le rendre de fait plus lisible. Donc vous devez le faire pour tout le code du projet !

Figure 3.9. Une partie du fichier `ajouter_produit.php` modifié

```
 1
 2
 3 if (isset($_POST['submit'])) {
 4   if (!hash_equals($_SESSION['csrf'], $_POST['csrf'])) die();
 5
 6   try  {
 7     $connection = new PDO($dsn, $username, $password, $options);
 8
 9     $nv_prod = array(
10       "code_produit" => $_POST['code_produit'],
11       "libelle"  => $_POST['libelle'],
12       "prix_unitaire"     => $_POST['prix_unitaire']
13     );
14
15     $sql = sprintf(
16       "INSERT INTO %s (%s) values (%s)",
17       "Produit",
18       implode(", ", array_keys($nv_prod)),
19       ":" . implode(", :", array_keys($nv_prod))
20     );
21
22     $statement = $connection->prepare($sql);
23     $statement->execute($nv_prod);
24   } catch(PDOException $error) {
25       echo $sql . "<br>" . $error->getMessage();
26   }
27 }
28 require "templates/header.php";
29
30     if (isset($_POST['submit']) && $statement) :
31         echo "<blockquote>" . escape($_POST['code_produit']) . " successfully added.</
blockquote>";
32     endif; ?>
33
34
```

Suppression de certaines balises ouvrantes / fermantes

L'idée quand il y en a trop, c'est d'utiliser un `echo` avec des concaténations.

Dans le cas d'utilisation d'un moteur de template

C'est plus facile, car tout le code html est dans un template, des fonctions PHP permettent d'écrire dans le template. C'est une architecture plus propre, et c'est d'ailleur pour cela que les moteurs de templates ont été développés. De plus, on peut dans ce cas avoir une architecture MVC.

3.1.3.2. Utilisation d'Ajax

Retrouver l'adresse de livraison d'une commande. Deux listes liées entre elles. C'est un cas classique ou Ajax semble inévitable. vous nommerez le fichier `adresse_livraison.php`, une bonne idée serait d'utiliser comme base de travail le fichier `commander.php` qui possède déjà des listes déroulantes.

Lisez donc l'article suivant ... Web 2.0, allez plus loin avec AJAX et XMLHttpRequest [https://siddh.developpez.com/articles/ajax/]

Figure 3.10. Formulaire principal

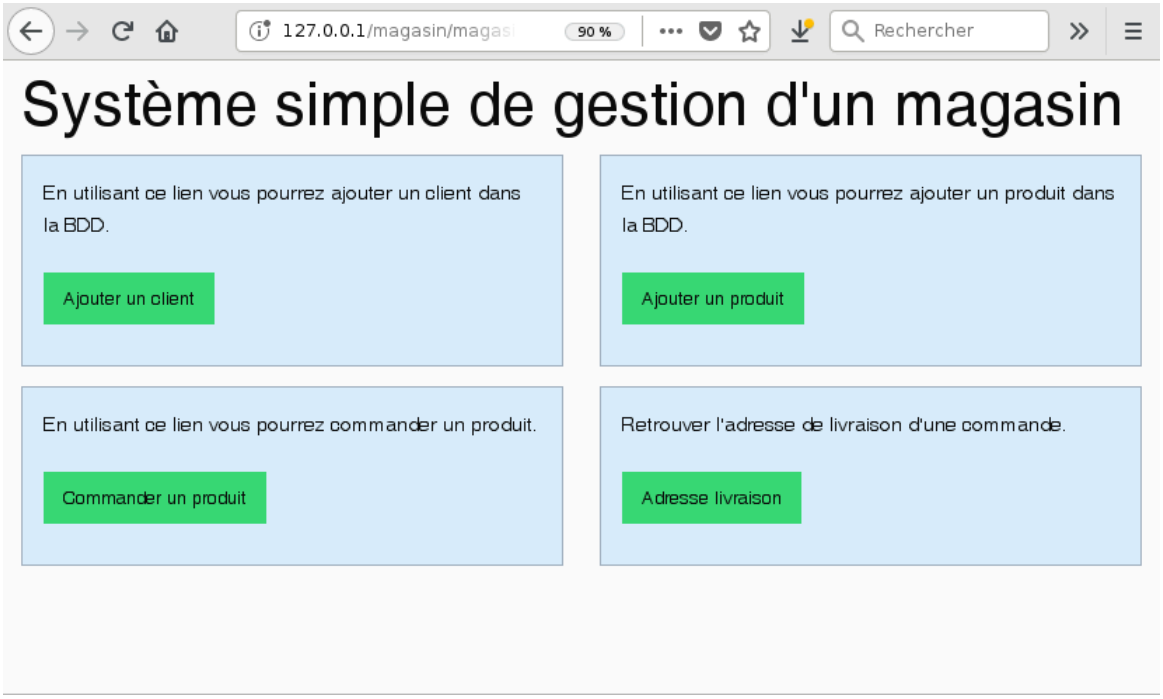


Figure 3.11. Adresse livraison 1/2

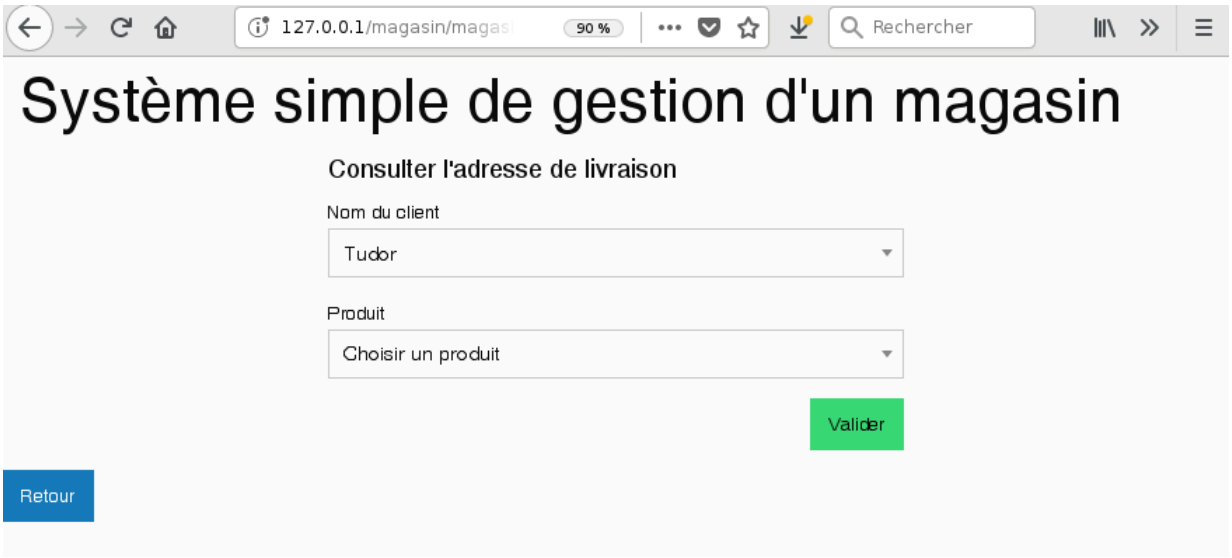


Figure 3.12. Adresse livraison 2/2

C'est pas optimisé ...

C'est simplement histoire de le mettre en oeuvre, pas d'optimisation en effet, mais l'essentiel y est !

Démarche à suivre (3 fichiers à coder) :

- Lire l'article joint, puis réfléchissez à la méthode la plus rapide pour coder ce qui est demandé ! Oui, la plus rapide !
- Vous aurez besoin d'une requête SQL, vous permettant pour un client donné de récupérer la liste de ce qu'il a commandé.

Figure 3.13. La requête SQL (en cadeau)

```
   // Requête pour récupérer tous les 'id_produit' et leur 'libelle'
   $sql_1 = "SELECT P.`id_produit`, P.`libelle` FROM `Commande` AS C, `Produit` AS P WHERE
C.`id_client` = ". $_POST['id_client'] . " AND P.`id_produit` = C.`id_produit` ";
```

- Premier fichier : modifier le fichier `header.php` afin qu'il tienne compte du code Ajax ;
- Deuxième fichier : créer un fichier (`ajaxadresse.php`) réalisant la requête produisant le code du `select`. Donc requête SQL.
- Troisième fichier : copier le fichier `commander.php` en `adresse_livraison.php` puis modifier le ...
- Résumons : vous avez une documentation à lire, vous devez donc adapter ce code, ci-dessous une très grande partie du travail vous est donnée. Donc, il vous reste en tout et pour tout le début du fichier `adresse_livraison.php` à coder.

Est ce possible ?

Oui à condition de bien comprendre le tutoriel, et de bien comprendre le code fourni. C'est un travail de déduction, puis de codage de la partie manquante.

Figure 3.14. Une partie du fichier header.php (code Ajax)

```
 1
 2
 3 <script type='text/javascript'>
 4 function getXhr(){
 5     var xhr = null;
 6     if(window.XMLHttpRequest)       // Firefox et autres
 7         xhr = new XMLHttpRequest();
 8     else if(window.ActiveXObject){  // Internet Explorer
 9         try {
10             xhr = new ActiveXObject("Msxml2.XMLHTTP");
11         } catch (e) {
12             xhr = new ActiveXObject("Microsoft.XMLHTTP");
13         }
14     }
15     else { // XMLHttpRequest non supporté par le navigateur
16         alert("Votre navigateur ne supporte pas les objets XMLHTTPRequest...");
17         xhr = false;
18         }
19         return xhr;
20     }
21
22     /**
23     * Méthode qui sera appelée sur le click du bouton
24     */
25     function go(){
26         var xhr = getXhr();
27         // On défini ce qu'on va faire quand on aura la réponse
28         xhr.onreadystatechange = function(){
29             // On ne fait quelque chose que si on a tout reçu et que le serveur est ok
30             if(xhr.readyState == 4 && xhr.status == 200){
31                 leselect = xhr.responseText;
32                 // On se sert de innerHTML pour rajouter les options a la liste
33                 document.getElementById('id_produit').innerHTML = leselect;
34             }
35         }
36
37         // Ici on va voir comment faire du post
38         xhr.open("POST","ajaxadresse.php",true);
39         // ne pas oublier ça pour le post
40         xhr.setRequestHeader('Content-Type','application/x-www-form-urlencoded');
41         // ne pas oublier de poster les arguments
42         // ici, l'id du client
43         sel = document.getElementById('id_client');
44         id_client = sel.options[sel.selectedIndex].value;
45         xhr.send("id_client="+id_client);
46     }
47 </script>
48
49
```

Figure 3.15. Fichier `ajaxadresse.php`

```
 1
 2
 3 <?php
 4 require "../config.php";
 5 require "../common.php";
 6
 7 try  {
 8         $connection = new PDO($dsn, $username, $password, $options);
 9
10         // Requête pour récupérer tous les 'id_produit' et leur 'libelle'
11         $sql_1 = "SELECT P.`id_produit`, P.`libelle` FROM `Commande` AS C, `Produit` AS P
 WHERE C.`id_client` = ". $_POST['id_client'] . " AND P.`id_produit` = C.`id_produit` ";
12
13         $statement_1 = $connection->prepare($sql_1);
14         $statement_1->execute();
15         $result_1 = $statement_1->fetchAll();
16
17     } catch(PDOException $error) {
18         echo $sql_1 . "<br>" . $error->getMessage();
19 }
20
21 echo "<select name='id_produit'>";
22     if ($result_1 && $statement_1->rowCount() > 0)
23         foreach ($result_1 as $row_1) :
24             echo "<option value='". escape($row_1['id_produit']). "'>".
 escape($row_1['libelle']) ."</option>";
25         endforeach;
26 echo "</select>";
27
28 ?>
29
30
```

Figure 3.16. Une partie du fichier `adresse_livraison.php`

```
 1
 2
 3 <div class="grid-x grid-padding-x">
 4             <div class="large-3 medium-3 small-12 cell"></div>
 5             <div class="large-6 medium-6 small-12 cell">
 6                 <h5>Consulter l'adresse de livraison</h5>
 7                 <form method="post">
 8                     <input name="csrf" type="hidden" value="<?php echo
 escape($_SESSION['csrf']); ?>">
 9                     <div class="grid-x grid-padding-x">
10                         <div class="large-12 cell">
11                             <label>Nom du client</label>
12                             <select name='id_client' id='id_client' onchange='go()'>
13                                 <?php
14                                 if ($result_0 && $statement_0->rowCount() > 0)
15                                     foreach ($result_0 as $row_0) :
16                                         echo "<option value='".
 escape($row_0['id_client']). "'>". escape($row_0['nom']) ."</option>";
17                                     endforeach;
18                                 ?>
19                             </select>
20                         </div>
21                         <div class="large-12 cell">
22                             <label>Produit</label>
23                             <div id='id_produit' style='display:inline'>
24                                 <select name='id_produit'>
25                                     <option value='-1'>Choisir un produit</option>
26                                 </select>
27                             </div>
28                         </div>
29
30                         <div class="large-12 cell">
31                             <div class="float-
right"><input type="submit" name="submit" value="Valider" class="success button"/></div>
32                         </div>
33                     </div>
34                 </form>
35             </div>
36             <div class="large-3 medium-3 small-12 cell"></div>
37     </div>
38
39     <div class="grid-x grid-padding-x">
40             <div class="large-12 medium-12 small-12 cell"></div>
41                 <a href="index.php" class="Alert button">Retour</a>
42             </div>
43     </div>
44
45
```

3.1.3.3. Ce qui reste à faire !

Si vous avez envie de réussir à atteindre un certain niveau :

- Développer les interfaces de suppression des clients, produits ...
- On a pas du tout modifié le thème par défaut de Foundation et franchement c'est pas terrible ! Alors prenez le temps de lire les docs sur le site officiel de Foundation. Puis commencer par exemple à changer les couleurs en modifiant / rajoutant du code *SCSS* dans le fichier `_settings.scss` !

Références