SLAM

SLAM5 Conception et adaptation de solutions applicatives

PROJET EN PHP NETBOUQUET
PRESENTATION ET CAHIER DES CHARGES

Objectif : **Développer une application Web**
- Modèle relationnelle et base de données
- Utiliser le modèle d'architecture MVC
- Utilisation de la classe objet PHP PDO pour l'accès aux données de la base de données MySQL
- Sécurisation avec des variables de sessions, et autres techniques
- Documenter une solution logicielle
- Gestion de version

CONTEXTE

La société *NetBouquet* est spécialisée dans la vente de bouquets aux entreprises. Elle vous demande de créer un site Web pour compléter le service actuel des commandes qui ne se fait pour l'instant que par téléphone, fax, ou mail. Les entreprises clientes seront obligatoirement abonnées auprès de NetBouquet pour pouvoir commander par internet.

CAHIER DES CHARGES

Définition des besoins

- **Accessibilité/Sécurité**
  - **Une partie du site est public**
    - Consultation de la liste des produits
    - Voir la présentation de l'entreprise
    - Un formulaire d'identification (connexion) permet d'accéder à la partie sécurisée du site
  - **La partie sécurisée est accessible à 2 types d'utilisateurs authentifiés avec des droits différents :**
    - Si ***admin*** :
      - Il peut créer, supprimer, modifier des bouquets
      - Extensions possibles : consulter la liste des clients, suivre les commandes
    - Si ***client*** :
      - Il peut consulter des bouquets, passer des commandes en ligne, consulter ses anciennes commandes.

Contraintes

- **Architecture**

L'application respectera l'architecture des scripts fournis et du modèle MVC.

- **Ergonomie**

Un fichier CSS permet d'uniformiser la présentation. L'application se conforme aux recommandations du W3C concernant les feuilles de style. Des améliorations pourront être proposées car cette phase n'a pas été terminée

- **Codage**

Le document "ApplisWeb-NormesDevelpt" présente des règles de bonnes pratiques de développement à utiliser pour encadrer le développement d'applications en PHP et en faciliter la maintenance.
Les éléments à produite devront respecter le nommage des fichiers, variables et paramètres, ainsi que les codes couleurs et la disposition des éléments déjà fournis.

- **Environnement**

Le langage de script côté serveur est le PHP, avec une base de données MySql

- **Modules**

L'application présente deux modules :
- Commande pour les clients : FRONT OFFICE

- Gestion des produits : BACK OFFICE

DESCRIPTION DU DOMAINE DE GESTION

- **Les visites sur le site (partie publique)**

Ils pourront lire les renseignements sur l'entreprise, et voir les produits proposés.

- **La gestion des produits**

L'administrateur pourra voir la liste des produits. Des liens lui permettront de gérer les produits : les modifier, les supprimer ou en ajouter. Pour cela, il faudra qu'il se connecte.

- **Les commandes (panier)**

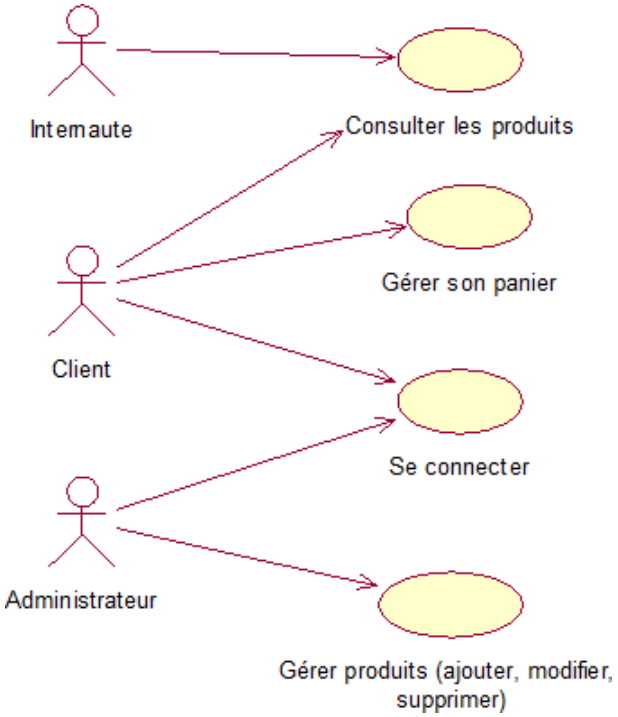
Les clients qui possèdent un login et mot de passe pourront passer des commandes, et voir l'historique de ses commandes.

Spécifications fonctionnelles de l'application de NetBouquet

- **Diagramme des cas d'utilisation**

La définition des cas d'utilisation est une étape très importante, c'est à partir de ce découpage que s'organisera l'application ; il ne faut absolument pas négliger cette étape.

Nous aborderons 4 cas d'utilisation, un pour les internautes, deux du front office (acteur les clients) et un du back office (acteur l'administrateur) :



De nouveaux besoins pourront être proposés par la suite : pour l'administrateur, gérer les comptes des clients, pour les internautes, la demande de création de compte.

SLAM

SLAM5 Conception et adaptation de solutions applicatives

PROJET EN PHP NETBOUQUET
LA BASE DE DONNEES

Compétences à rajouter à votre portefeuille :

REGLE DE NOMMAGE COTE BASE DE DONNEES

Concernant le schéma de la base de données les règles d'écriture suivantes ont été appliquées :
- pas de blanc ni de caractère accentué dans les **noms de table ou d'attribut** ;
- **chaque nom de table** commence par une majuscule et est suivi de minuscules. Si elle est composée de deux mots, ils sont collés et distingués par une majuscule ;
- **chaque nom d'attribut** est écrit en minuscule. S'il est composé de deux mots, ils sont collés et distingués par une majuscule. Le nom choisi pour l'attribut représente le rôle de son domaine dans la table ;
- **une clef étrangère** porte un nom significatif de son rôle dans la table.

CREATION DE LA BASE DE DONNEES SOUS MYSQL bouquetnet

Un script est fourni pour la création des tables et d'occurrences. La base sera susceptible d'évoluer en fonction de nouveaux besoins.

SLAM5 Conception et adaptation de solutions applicatives

SLAM PROJET EN PHP NETBOUQUET
LE MODELE VUE CONTROLEUR

MODELE VUE CONTROLEUR

Le MVC, tout comme l'orientation objet du code, semble être devenu un standard dans le développement d'applications web, avec la réputation d'être une bonne pratique de conception. De nombreux Frameworks (bibliothèques de classe) utilisent aujourd'hui MVC, car le but principal de ce motif est de séparer les couches logiques d'une application.

La séparation en plusieurs couches permet à différentes équipes de travailler chacune sur une couche, indépendamment des autres. Et c'est aussi un gros avantage lorsqu'une couche doit évoluer, sans que les autres n'en aient besoin.

Ce modèle de développement distingue 3 fonctionnalités :

- **La vue (V)**

Représente ce qui est exposé à l'utilisateur. Il s'agit de HTML statique ou généré par du php ;

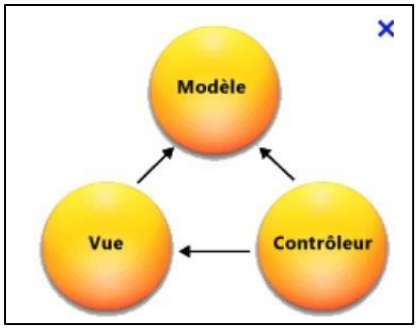
Une page web côté client sera en général constituée de plusieurs vues. Une vue peut-être commune à plusieurs pages web.

Dans notre application, les pages web seront constituées de plusieurs vues, qui seront appelées grâce à des « include » sur les fichiers correspondants.

➢ Le contrôleur (C)

Ce sont les contrôleurs qui vont être à l'écoute des requêtes de l'utilisateur et fournir ainsi la page web correspondante (constituée de vues). Pour cela, il faudra à tout moment connaître **l'état de l'application** c'est à dire le contexte de la demande : *"la page demandée fait suite à quelle **action** précise de l'utilisateur ?*" C'est au contrôleur de connaître l'état applicatif en testant les variables nommées ***$uc*** et ***$action***, *provenant d'une requête POST ou GET.* Ainsi le contrôleur se présentera souvent sous le format suivant :

```
$uc = $_REQUEST['uc'];
switch($uc)
{
   case 'ceci' :
      include("vues/v_ceci.php");
      include("vues/v_encorececi.php");
    case 'cela" :
      include("controleur/c_cela.php");
}
```



➢ Le modèle (M)

C'est la couche qui accède à la base de données. C'est donc là que nous trouverons nos requêtes et l'exécution de celle-ci. Mais nous allons créer des fonctions qui dans la plupart du temps vont retourner des tableaux (le résultat, c'est à dire les occurrences répondant à la requête). On utilisera PDO et on créera une CLASSE.

ORGANISATION DES FICHIERS

« ***controleurs*** » contiendra tous les contrôleurs.

- « ***images*** » toutes les images
- « ***util*** » contiendra
  - Le fichier « **class.pdobouquet.inc.php** » qui contiendra notre classe et les méthodes d'accès aux données grâce à la classe PDO
  - Le fichier « **fct.inc.php** » qui contiendra les fonctions métiers
  - Le fichier « **cssGeneral.css** » contiendra le style
- « ***vues*** » contiendra toutes les vues
- A la ***racine***, le contrôleur principal est placé « **index.php** »

CAS D'UTILISATION POUR LES VISITES SUR LE SITE

- **Les visites sur le site (partie publique)**

PROJET : Application web commande et gestion de produit	**Description cas d'utilisation**

Nom cas d'utilisation : Visite de la boutique

Acteur déclencheur : Internaute, client

Pré conditions : Néant, connexion ou création de compte

Scénario nominal :

- **1- L'internaute demande l'url du site. Le système affiche la page d'accueil qui présente l'entreprise**
- **2- L'internaute demande à voir les produits. Le système affiche la liste des catégories de produits et par défaut tous les produits**
- **3- L'internaute sélectionne une catégorie d'article. Le système affiche la liste des produits de la catégorie choisie**
- **4- L'internaute demande à se connecter. Le système affiche le formulaire de connexion**
- **5- L'internaute n'a pas de compte et veut en créer un. Le système affiche le formulaire de création de compte**
- **6- Le client se connecte ou créé un compte client. Le système affiche la page d'accueil et ajoute les onglets 'Mon panier', 'Mes commandes'**
- **7- Le client demande d'ajouter un article à son panier, l'article est ajouté**
- **8- Le client veut voir son panier. Le système affiche le panier.**
- **9- Le client demande d'augmenter la quantité d'un article. Le système augmente la quantité de l'article**
- **10- Le client demande de diminuer la quantité d'un article. Le système diminue la quantité de l'article**
- **11- Le client demande de supprimer un article du panier. Le système supprime l'article du panier**
- **12- Le client veut passer commande. Le système affiche la commande et demande la validation**
- **13- Le client valide la commande. La commande est enregistrée**
- **14- Le client souhaite voir ses commandes. Le système affiche l'historique des commandes avec les produits contenus dans chaque commande**
- **15- Le client se déconnecte. Le client estdéconnecté et le système affiche la page d'accueil.**

```
<?php
//pour récupérer l'action qui vient des liens, ou si n'existe pas on met l'acceuil
if(!isset($_REQUEST['action'])){
    $_REQUEST['action'] = 'accueil';
}
 $action = $_REQUEST['action'];

        switch($action)  {
                case 'accueil' :
                {
                        include("vues/v_entreprise.php");
                        break;
                }

                case 'voirProduits' :
                {
                        break;
                }
        }
?>
```

Nous avons donc ici **20 actions** de l'utilisateur. Il faudra donc des contrôleurs qui dirigeront sur l'action souhaitée.

Action	**Contrôleur**	**Cas**	**Vue**
Action 1	index.php	accueil	v_entreprise.php
Action 2	c_voirProduits.php	voirCategories.php	v_: categories / listeProduits.php
Action 3	c_voirProduits.php	voirProduits	v_categories/listeProduits.php
Action 4	c_connexion.php	connexion	v_formConnexion.php
Action 5	c_connexion.php	inscription	v_inscription.php
Action 6	c_connexion.php	verifConnexion	v_entreprise.php
Action 7	c_voirProduits.php	ajouterAuPanier	v_: message / categories / listeProduits.php
Action 8	c_gestionPanier.php	voirPanier	v_panier.php + option: v_message.php
Action 9	c_gestionPanier.php	augmenterQte	v_panier.php
Action 10	c_gestionPanier.php	diminuerQte	v_panier.php
Action 11	c_gestionPanier.php	supprimerUnProduit	v_panier.php + option: message.php
Action 12	c_gestionPanier.php	passerCommande	v_commande.php ou v_message.php
Action 13	c_gestionPanier.php	confirmerCommande	v_message.php
Action 14	c_commandes.php	historiqueCommandes	v_historiqueCommandes ou v_message.php
Action 15	c_deconnexion.php		v_entreprise.php

L'ACCES AUX DONNEES

TRAVAIL A FAIRE

➢ Création des répertoires

Dans le répertoire www, créer le dossier « NetBouquet ». Y créer les 4 répertoiressuivants :

a. **controleurs** : le fichier des styles css
b. **images** : les images
c. **util** : le fichier modèle qui regroupent les fonctions et le fichier des styles CSS
d. **vues** : toutes nos vues

```
<?php
//pour récupérer l'action qui vient des liens, ou si n'existe pas on met l'acceuil
if(!isset($_REQUEST['action'])){
    $_REQUEST['action'] = 'accueil';
}
 $action = $_REQUEST['action'];

        switch($action)  {
                case 'accueil' :
                {
                        include("vues/v_entreprise.php");
                        break;
                }

                case 'voirProduits' :
                {
                        break;
                }
        }
?>
```

- **Les fichiers de vues**

Récupérer les fichiers suivants et les placer dans le répertoire « vues »

e. v_entreprise.php
f. v_entete.php
g. v_formConnexion.php

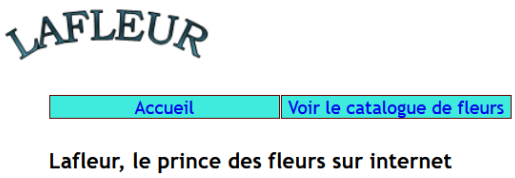
- **Créer un nouveau projet avec un IDE évolué.** Ouvrer les 3 fichiers de vues et les étudier.

- **La page web index.php**

2. **Créer la page index.php : c'est la page web que verra l'utilisateur lorsqu'il arrive sur le site. Cette page est notre CONTROLEUR PRINCIPAL.**

3. **Le résultat doit être celui-ci**

4. **Avec les vues données, compléter « index.php » afin de construire cette interface :**



J'ai pensé que ça pouvait être mieux que la vue entreprise soit la vue d'accueil plutôt que d'avoir une page d'accueil vide. De plus, j'ai créé un onglet 'Se connecter'.

- **Le formulaire de connexion** **v_formConnexion.php**

```
<?php
//pour récupérer l'action qui vient des liens, ou si n'existe pas on met l'acceuil
if(!isset($_REQUEST['action'])){
    $_REQUEST['action'] = 'accueil';
}
 $action = $_REQUEST['action'];

        switch($action)  {
                case 'accueil' :
                {
                        include("vues/v_entreprise.php");
                        break;
                }

                case 'voirProduits' :
                {
                        break;
                }
        }
?>
```

- **L'entête** **v_entete .php**

- **La vue d'accueil** **v_entreprise.php**

- **Le contrôleur de la partie publique** **c_voirProduits.php**

5. Dans le répertoire « controleurs » créer le fichier avec le code suivant.

➢ **La vue des informations de l'entreprise** **v_entreprise.html**

6. Créer le fichier et y placer le code :

```
<div id="accueil">
L'entreprise a été fondée en 2002.<br>
Nous vous proposons un largue choix de produits pour agrémenter vos bureaux en toutes occasions. <br>
<b>Nos coordonnées : </b><br>
Adresse : …<br>
Tél : …<br>
Mail : …<br>
</div>
```

LES STYLES

1. Rajouter le fichier cssGeneral.css dans le répertoire «util». L'ouvrir pour l'étudier

REGLE DE NOMMAGE

Les éléments à fournir doivent respecter le nommage des fichiers, variables, paramètres.

Item	Règle de nommage
Tableau associatif retourné par une méthode	$lesLignes
Message d'erreur à afficher	avec la vue v_erreurs.php

L'ACCES AUX DONNEES

➢ Le modèle — class.pdobouquet.inc.php

1. **Récupérer le fichier de la classe d'accès aux données, à placer dans le répertoire « util ».**
   Cette classe « PdoBouquet » encapsule la classe PDO d'accès aux données. On y placera uniquement les méthodes qui permettront d'envoyer les requêtes : Select, Insert, Update, Delete.

2. **Récupérer le fichier de la classe d'accès aux données, à placer dans le répertoire « util ».**
   L'ouvrir et l'étudier. Vérifier le nom de la base de données.

```
private static $bdd='dbname=lafleur';
```

Rappel : le constructeur est privé. Il est encapsulé dans une méthode qu'il faudra appeler pour créer un objet.

```
public static function getPdoBouquet(){
    if(PdoLafleur::$monPdoLafleur==null){
        PdoLafleur::$monPdoLafleur= new PdoLafleur();
    }
    return PdoLafleur::$monPdoLafleur;
}
```

Appel du constructeur

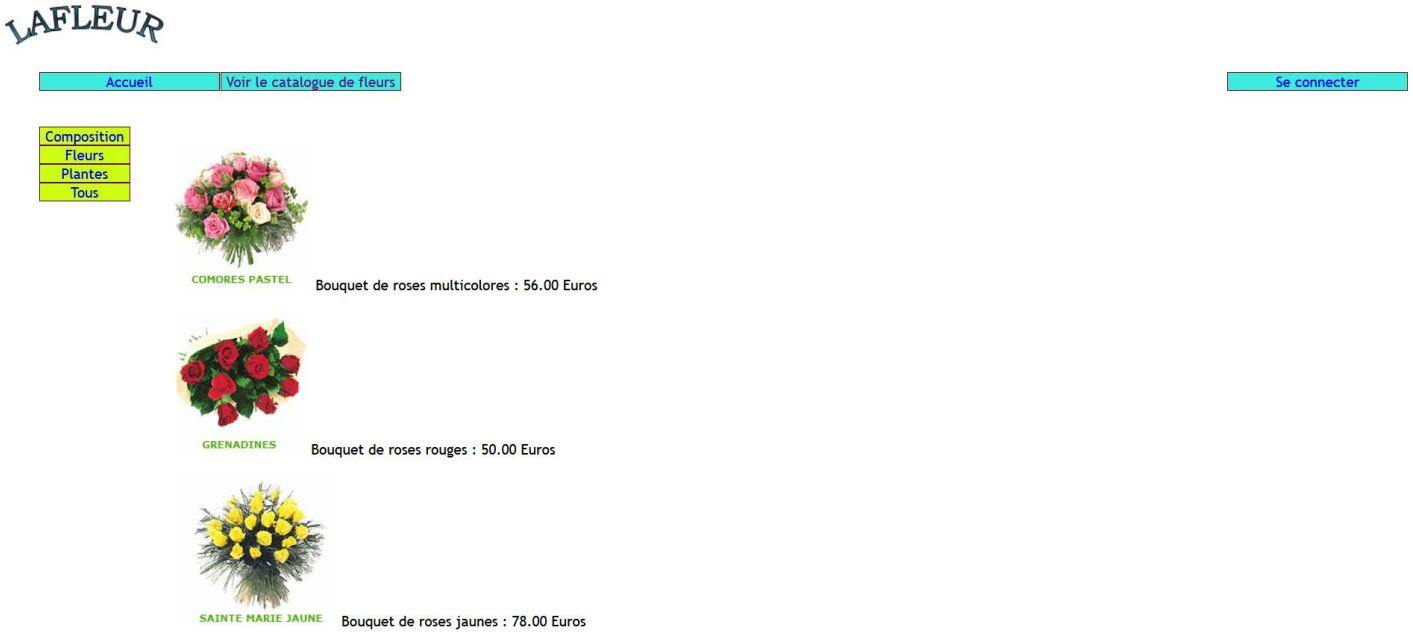
Ne sera mis qu'ici, une fois pour tout

3. **Dans l'index, décommenter les lignes :**
   a. **require_once ("include/class.pdoLafleur.inc.php");**
   b. **$pdo = PdoLafleur::getPdoLafleur();**

La seule instance de la classe, préfixera toutes les méthodes d'accès aux données

➢ Afficher les catégories / les produits c_voirProduits.php case : voirToutProduits

L'utilisateur pourra demander :
- soit tous les produits : lorsque l'utilisateur arrive sur la page, ou qu'il sélectionne TOUS
- **soit les produits de la catégorie sélectionnée** dans la liste déroulante.



4. **Dans la classe « class.pdoLafleur.inc.php », créer une méthode qui retourne un tableau associatif des catégories (id et libellé)**

   Les méthodes par la suite respecterons ces commentaires pour la doc et le nom de variable $lesLignes

5. **Dans le contrôleur « c_consulter.php », cas « voirProduit », appeler la méthode et la vue à créer.**

6. **Créer la vue « v_listeCategories.php » qui affiche la liste déroulante.** Il faut ajouter une option « **Tous** »

7. **Dans la classe « class.pdobouquet.inc.php », créer une méthode qui retourne un tableau associatif de tous les produits (tous les champs).**

8. **Créer la vue « v_listeProduits.php » qui affiche les produits.**

9. **Modifier le contrôleur concerné afin d'afficher les produits sous la liste déroulante.**

➢ Afficher les produits d'une catégorie c_consulter.php case : voirProduits

1. **Dans la classe « class.pdoLafleur.inc.php », créer une méthode « getLesProduitsCategorie($Categorie) » qui retourne un tableau associatif des produits de l'id catégorie passé en paramètre.** Tester la requête dans Mysql avant.

2. **Dans le contrôleur, SI la variable $_POST["Categorie"] existe, il faut récupérer la valeur de la catégorie sélectionnée. Attention, il faut faire un test : Si la valeur est « Tous » ou pas. Et appeler la méthode correspondante.**
   - SI $_POST["idCategorie"] n'existe pas, appeler getTousLesProduits().
   - SI $_POST["idCategorie"] existe, c'est que l'utilisateur à sélectionné un produit dans la liste déroulante. Appeler getLesProduitsCategorie($idCategorie)

PROJET NETBOUQUET MVC
PARTIE 2 : LA CONNEXION

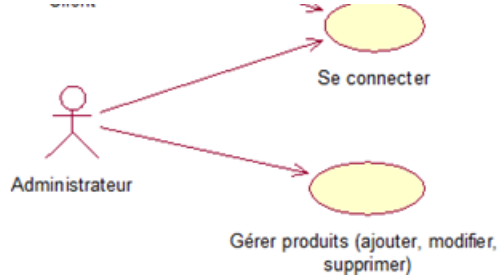
Dans la base de données, soit l'utilisateur a un STATUT « administrateur », soit « client'

1) L'administrateur

L'administrateur gère les produits dans la base de données. Il peut :
- **ajouter des produits**
- **supprimer des produits**
- **modifier des produits**
- **se déconnecter**

COMME POUR LA PARTIE PUBLIQUE DU SITE, TOUS LES LIENS ET LES ACTIONS DES FORMULAIRES SONT REDIRIGES VERS LA PAGE INDEX, avec les variables « UC » et « ACTION ».

CAS D'UTILISATION GERER LES PRODUITS



PROJET : Application web commande et gestion de produit	**Description cas d'utilisation**

Nom cas d'utilisation : Gérer produit
Acteur déclencheur : Administrateur
Pré conditions : Néant
Post conditions : L'internaute pourra demander un identifiant pour se connecter
Scénario nominal : • **1- L'internaute demande l'url du site. Le système affiche la page d'accueil qui présente l'entreprise** • **2- L'internaute demande à se connecter. Le système affiche le formulaire de connexion** • **3- Un administrateur se connecte. Le système affiche la page d'accueil et ajoute l'onglet 'Ajouter un produit'** • **4- L'administrateur souhaite ajouter un produit. Le système affiche le formulaire de création de produite** • **5- L'administrateur ajoute un produit. Le système l'enregistre et peut l'afficher** • **6- L'administrateur supprime un produit. Le système le supprimer** • **7- L'administrateur modifie un produit. Le système le modifie** • **8- L'administrateur se déconnecte. L'administrateur estdéconnecté et le système affiche la page d'accueil.**

Nous avons donc ici **beaucoup d'actions** de l'administrateur. Le contrôleur prévoira cela.

ORGANISATION DES FICHIERS

1. Organisation des répertoires :

Les fichiers se termineront par .admin.php
Les méthodes d'accès aux données seront toujours dans le fichier « class.pdobouquet.inc.php »

Il y aura du copier-coller avec la partie publique, le développement sera rapide pour certains cas.
Commençons par voir comment se passe la connexion.

➢ **Le formulaire** **v_formConnexion.html**

<form action="index.php?uc=connexion&action=valideConnexion" method="post">

2. Lorsque l'on soumet le formulaire, les variables $login et $mdp sont envoyées vers l'index en méthode POST, avec en plus la variable « uc » qui vaut « **connexion** » qui est passée par l'url de l'action du formulaire.

- Quel est le nom du contrôleur dans le cas où $uc='connexion' ? _____
  - **Rajouter l'action dans la vue « v_formConnexion.html ».**

➢ Le contrôleur pour la connexion c_connexion.php

Le contrôleur va récupérer la variable $action provenant du formulaire de connexion.
Selon la valeur, le traitement diffère :

→ Si c'est le cas « valideConnexion » :

- Il faut récupérer le login et le mdp saisis, dans des variables locales
- Appeler une méthode « **$pdo->getInfosUtilisateur($login,$mdp)** » qui retourne un tableau associatif composé du nom et du statut de l'utilisateur (voir plus loin).
  - Si le tableau est vide :
    - Il faut ajouter un message d'erreur grâce à la fonction **ajouterErreur**("Le message")
    - Puis inclure les vues du formulaire de connexion, du sommaire d'index et des erreurs.
  - Si le tableau n'est pas vide :
    - Il faut récupérer le nom et le statut contenus dans le tableau
    - Appeler une fonction « **connecter**($login,$nom,$statut) » qui permet d'enregistrer les 3 paramètres dans des variables de session.
      - Si le statut est « admin » :on inclut le contrôleur «c_gererProduits.php»
      - Si le statut est « client » : sera étudiée par la suite.

➢ Les variables de session

La fonction **connecter($login,$nom,$statut)** permet d'enregistrer les variables de sessions les 3 variables dont nous allons avoir besoin par la suite dans différentes pages : le login, le nom de l'utilisateur ainsi que son statut.

➢ Le contrôleur de la gestion des produits c_gestionProduits.php

Créer la page contrôleur de gestion des produits. Le principe est le même que le contrôleur « c_consulter.php ».
Il va vérifier qu'elle action veut l'utilisateur :

- Afficher tous les produits, ou les produits d'une catégorie
- Supprimer un produit
- Modifier un produit
  - Ajouter un produit