# CONSIGNES DE RÉALISATION, SÉRIE 5 : ADMINISTRATION DU SITE

Cette étape vérifie principalement la maîtrise des concepts et techniques que vous avez déjà appliqués.

Pour compléter le système, il vous reste à permettre à l'administrateur de gérer sa BD à distance. Bien sûr, on pourrait lui faire utiliser **phpMyAdmin** et **FTP** mais on doit lui permettre de faire son travail de façon ergonomique et en supposant qu'il n'a pas de connaissances informatiques appropriées.

Vous devez lui offrir les fonctionnalités suivantes:

- Gestion des catégories
- Gestion des produits
- Production de rapports de gestion.

On profite du fait qu'on commence une nouvelle partie pour ajouter un niveau de sécurité sur vos pages. Comme toute cette partie du site doit n'être accessible qu'à l'administrateur, dans chaque page autre que la première, vous commencerez le code ainsi (ou quelque chose d'équivalent… n'utilisez pas tous la même variable de session!):

```
if (!isset($_SESSION['authenthAdmin']))
{
        header("location:index.php");
        exit();
}
```

Même si vous n'offrez les liens vers les pages protégées qu'après l'authentification de l'administrateur, il pourrait bien se mettre un signet (ou favori ou marque-page) sur une de ces pages et tenter par la suite d'y accéder directement. Cette instruction fait en sorte que toute entrée non authentifiée se fait automatiquement rediriger vers la page d'accueil. Une variable de session d'authentification — par exemple **$_SESSION['authenthAdmin'] —** différente de celle du site des clients évitera des problèmes car si un client qui est authentifié tapait directement l'adresse d'une page d'administrateur, il y aurait accès.

## PARTIE 1: LA PAGE D'ACCUEIL DE L'ADMINISTRATEUR

L'administrateur doit obligatoirement s'authentifier pour procéder. La page d'accueil présentera donc une saisie de son nom et son mot de passe. La validation du contenu se fera dans la même page, côté serveur.

Dans votre BD, créez un usager **Admin —** dans la vraie vie il faudrait choisir un nom un peu plus difficile à deviner — et donnez-lui un mot de passe. Une fois authentifié, vous lui donnez accès aux fonctionnalités de l'administrateur soit en activant des hyperliens, soit en lui offrant une page qui les contient. Pour ajouter de la sécurité à votre BD vous encoderez le mot de passe — ne faites cette opération **que pour** l'administrateur même si on devrait l'implanter en général.

Assurez-vous que la longueur du champ « mot de passe » dans la BD est de longueur **32**. Dans la table des clients, insérez un nouvel enregistrement avec **Admin** comme nom d'usager. Pour son mot de passe, dans la colonne **Fonction**, choisissez **MD5** et saisissez un mot de passe que vous allez vous rappeler parce que ne le reverrez plus jamais en clair. Les mots de passe des autres utilisateurs/clients resteront lisibles.


Par la suite, à l'authentification, vous vérifierez que **md5(*mot de passe saisi) == $tClient['MotPasse']*** ou, en SQL,

…**WHERE motPasse = '".md5($_POST['motPasse'])."' ";** c'est-à-dire que vous encoderez ce que l'usager a tapé et vous le comparerez avec le mot de passe contenu dans la BD. S'il y a égalité, l'administrateur sera authentifié.

## PARTIE 2: LA GESTION DES CATÉGORIES

Pour des raisons évidentes d'intégrité des données, on ne doit pas permettre de supprimer des catégories qui seraient reliées à des données. Les seules fonctionnalités à implanter sont donc l'ajout et la modification.

Le modèle de traitement présenté par le site de démonstration est un exemple dont le fonctionnement sera décrit plus bas mais vous pouvez effectuer ces opérations autrement. L'important est que vous permettiez la modification des catégories ainsi que l'ajout dans une interface simple.

Commencez par vous assurer que votre clé de la table des catégories est « *auto-increment* », également avant de faire l'insertion d'une nouvelle catégorie, assurez-vous que le libellé de la catégorie n'existe pas encore en BD et qu'il contient entre 2 et 20 caractères.

Dans mon exemple — affichez le code source de ma page pour visualiser — , toutes les catégories sont affichées dans des champs de saisie d'un formulaire. Pour ne mettre à jour que celles qui changent, la stratégie employée est la suivante: dans le formulaire, on écrit, pour chaque catégorie, un champ caché dont le nom est **catc*XX*** où ***XX*** est le numéro de la catégorie. Ce champ contient la valeur de la catégorie avant tout changement. Un autre champ, visible celui-là, porte le nom **cat*XX*** et est celui qui permet de modifier le texte de la catégorie.


Au clic du bouton de soumission, le contenu du formulaire est analysé dans une boucle **foreach()** et si le contenu d'un champ **catc*XX*** est différent de celui du **cat*XX*** correspondant, la catégorie dont la clé est ***XX*** est mise à jour par un énoncé **UPDATE**. La clé à

utiliser dans la partie **WHERE** de cet énoncé vient simplement de la partie ***XX*** du nom des champs.

La fonction php **substr()**, pourra vous être utile dans votre recherche des valeurs correspondantes.

Pour le champ d'ajout, s'il contient quelque chose (**if (!empty($_POST['nouvelle']))…** par exemple), vous créez par un **INSERT** cette nouvelle catégorie. **Attention**, le test **if (isset($_POST['nouvelle']))** ne détermine pas s'il y a un contenu ou non dans ce cas car la variable existe bel et bien même si elle est vide. Le test **!empty()** est plus approprié.

**Attention :** le bouton Soumettre fera les deux actions : **mises à jour** des catégories et **insertion** de nouvelles catégories

## PARTIE 3: LA GESTION DES PRODUITS SANS LES PHOTOS

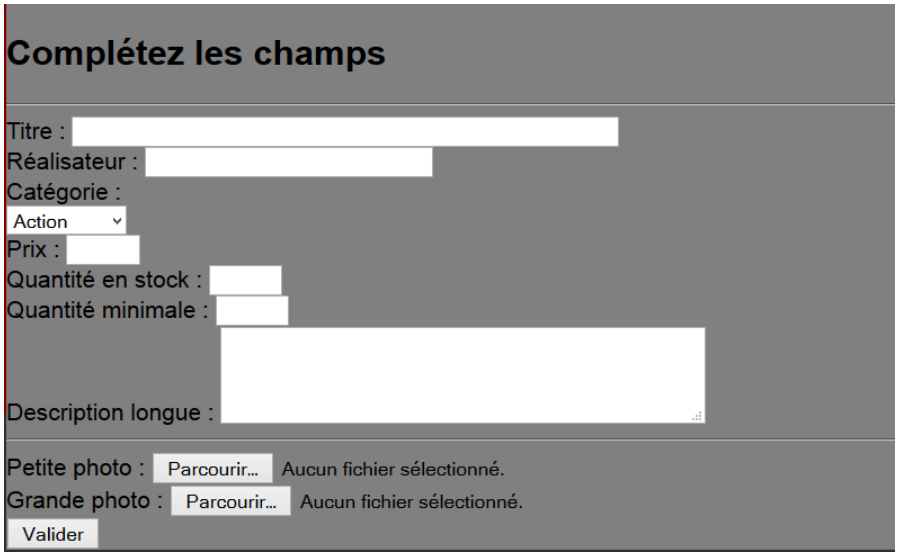
Vous utilisez la méthode présentée dans le site de démonstration pour créer ou ajouter un nouveau produit à votre base de données. Vous implantez la même méthode de programmation que la page d'inscription du client, soit la validation côté serveur implantée dans une seule page qui gère à la fois l'affichage et la validation.

Dans le site de démonstration, en cliquant sur le lien **Produits** à gauche de la page, on obtient la page suivante:.


Si vous cliquez sur "Ajouter un film", le formulaire plus bas apparait:

Proposez quelque chose d'équivalent pour ces pages qui permet de modifier et d'ajouter un produit pour votre site

Utilisez les expressions régulières pour valider le prix — l'utilisateur doit saisir deux chiffres après le point — ainsi que les quantités qui ne doivent




contenir que des chiffres. Le nom du produit (titre ici) doit être contenir entre 2 et 60 caractères. La description ne doit pas dépasser 255 caractères.

## PARTIE 4: LA GESTION DES PHOTOS

L'ajout de photos doit se faire sur la même page que la modification des produits. Vous devez suivre les spécifications avec attention pour que le tout fonctionne parfaitement. Pour éviter de devoir toujours saisir des données, codez une page de tests et l'intégrer au code normal une fois que tout est correct.

L'énoncé **<form>** , lorsqu'il contient un transfert de fichiers doit être modifié en lui ajoutant un nouvel élément :

***<form action='…' method='post'*** **enctype='multipart/form-data'>** (la méthode POST est obligatoire… le transfert d'une image ne pourrait se faire dans le GET)

Dans ce formulaire, outre les champs habituels, vous devez en ajouter un ainsi:

**<input type='hidden' name='MAX_FILE_SIZE' value='100000'>** qui spécifiera la grandeur maximale des fichiers transmis. Qu'il y ait un ou plusieurs fichiers, on ne met qu'une seule fois ce champ caché. **Attention**: le nom doit être exactement celui donné plus haut.

Pour transférer un fichier, utilisez **<input type='file' name=*'petitePhoto'*>** par exemple. Ceci génère automatiquement un bouton **Parcourir** et tout ce qui vient avec. Le transfert s'effectue en deux étapes:

étape 1: le fichier passe du POST vers un fichier temporaire du serveur

étape 2 : Vous pouvez le copier dans votre répertoire en précisant le nom que vous voulez lui donner.

Dans la partie de votre script qui traite le résultat du POST, vous traiterez donc également les fichiers d'images.

Notez que ce traitement doit être fait dans le même programme que la validation côté serveur parce que les variables **$_FILES**, tout comme les variables **$_POST** n'ont qu'une durée de vie d'un seul « transfert ». Quand une page s'appelle elle-même dans une validation côté serveur, les fichiers doivent être traités dans cette même page.

Voici ce dont vous avez besoin :

- **if (!empty($_FILES['*petitePhoto*'][ 'name']))** vous dira si un fichier a été transmis;

- **$_FILES['*petitePhoto*'][ 'error']** devrait être à zéro si le transfert vers le fichier temporaire a bien fonctionné. Les codes d'erreur sont documentés sur le WEB;
- **$_FILES['*petitePhoto*']['tmp_name']** vous donne le nom du fichier temporaire que vous voudrez ensuite copier vers votre espace;
- **$_FILES['*petitePhoto*']['size']** vous donne la grandeur de votre fichier, si requis.

Dans le traitement du formulaire, la fonction **is_uploaded_file($_FILES['*petitePhoto*']['tmp_name'])** devrait d'abord être exécutée pour prévenir des erreurs.

Si elle retourne vraie, utilisez la fonction **move_uploaded_file($_FILES['*petitePhoto*']['tmp_name'], $destination)** pour compléter l'opération. Cette dernière retourne TRUE si elle a fonctionné correctement et FALSE dans le cas contraire.

À vous de construire la valeur de la variable **$destination** pour qu'elle désigne le nom définitif et complet du fichier. N'oubliez pas de prévoir la petite et la grande photo.

## PARTIE 5: LES RAPPORTS

Cette partie ne donnera que deux affichages:

- Liste des produits dont la quantité en stock est inférieure à la quantité minimale;
- Liste des commandes passées, triées en ordre chronologique inverse (plus récent en premier).

**Note** : si votre partie **Commande** du TP4 ne fonctionne pas bien, créez tout de même les tables **Commandes** et **CommandesDetails** et mettez-y des données « à la main » pour pouvoir faire cette partie de l'interface administrateur.