# Modèle MVC avec l'application *LeCadeau*

***LeCadeau*** est une application sera visible sur un site du commerce, vas développer en PHP par votre groupe. L'application gère un panier électronique de petites cadeaux (3 euros max ;)) de différents types.
Je m'attacherai à proposer une implémentation de la technologie MVC sur ce contexte simple.

## 1) Cas d'utilisation

Je n'aborderai que trois cas d'utilisation, deux du *front office* (acteur l'internaute) et un du back office (acteur l'administrateur) :

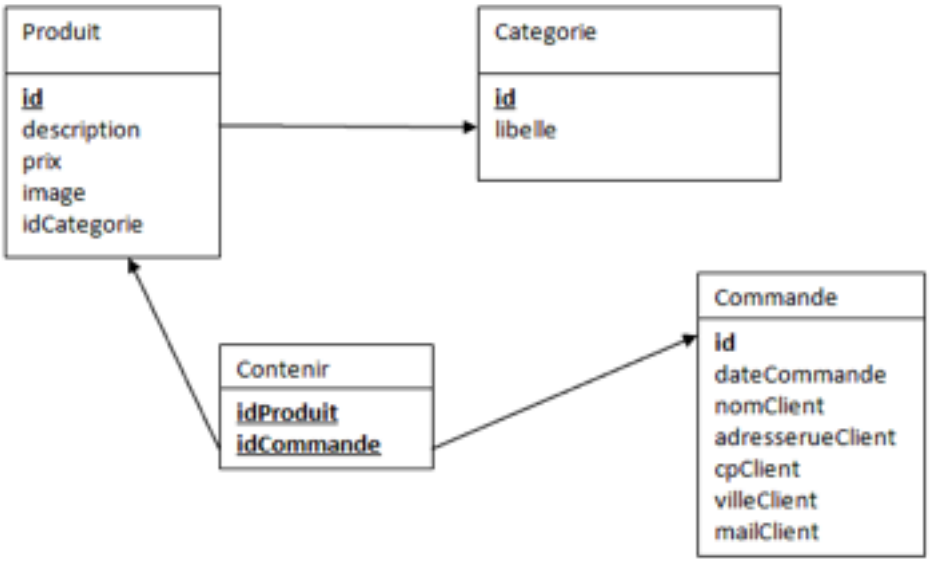
**Internante** —-> Consulter les cadeaux
— —> Gérer son panier

**Administrateur — —>** Gérer les cadeaux (Ajouter, modifier, supprimer)

**Note:** 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.

## 2) Modèle de données

La base de données est sous MySql,
Le script de la base (tables et occurrences) est à faire !



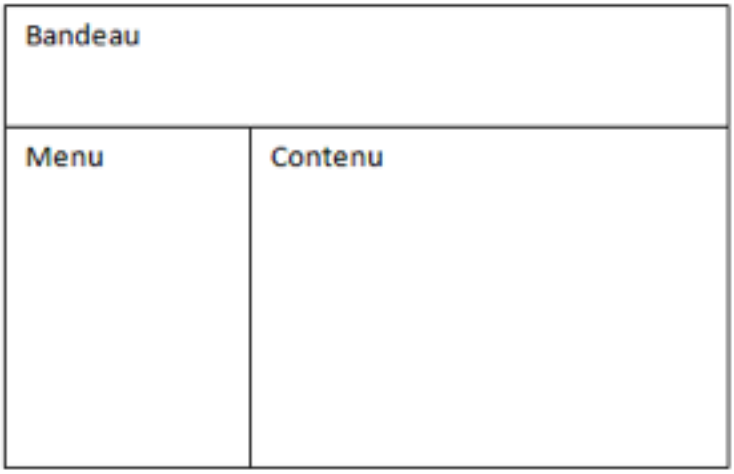

## 3) Le modèle MVC

Ce modèle de développement distingue 3 fonctionnalités :

**3.a La vue (V)** représente ce qui est exposé à l'utilisateur, en général il s'agit de HTML statique ou généré par du php ; il y a deux sortes de vue :

+ Les pages d'information navigables grâce à des liens
+ Les formulaires de saisies d'informations ; ces formulaires peuvent être présentées à plusieurs reprises pour confirmation ou signalement d'erreurs.

Une vue propose une partie commune à chaque page et des zônes liées à la demande de l'utilisateur :




En général le bandeau est du code statique et les zônes Menu et Contenu sont générés dynamiquement. Ainsi, chaque vue contiendra des *include* sur les fichiers correspondants ; on peut envisager que dans certaines vues les parties Menu et Contenu soient dans un même fichier car logiquement liés (ce sera le cas dans notre développement).

La mise en forme (disposition) est gérée grâce aux balises div et une feuille de style.

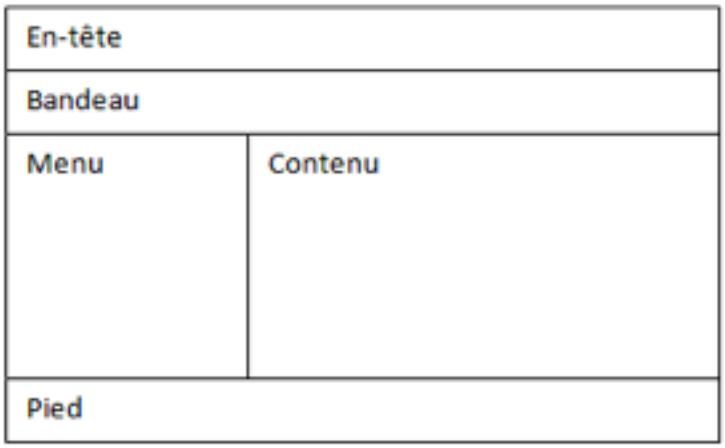
Ainsi une vue peut être construite avec plusieurs *sous-vues*, chaque sous-vue est un fichier php contenant presque exclusivement du html (généré dynamiquement ou non).

Par la suite je distingurerais la notion de **vue externe** de la notion de *vue* :

- La **vue externe** est la page que voit l'internaute
- Cette *vue externe* est construite à partir de vues (sous-vues ou zones) : chaque vue (sous-vues) est dans un fichier distinct.

Remarque : par ailleurs dans notre application *LaCadeau*, la vue externe contiendra au moins 5 vues:

- Afin de distinguer l'en-tête (partie *head*, *meta*) du bandeau, il y aura une vue en-tête.
- Une vue bandeau
- Une vue menu
- Une vue contenu
- Une vue pied (balises fermantes HTML)




### 3.b Le contrôleur

Ce sont les contrôleurs qui vont être à l'écoute des requêtes de l'utilisateur et fournir ainsi la *vue externe* correspondante (constituée de sous-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 une variable qui sera nommée ***$action***, *provenant d'une requête POST ou GET*. Ainsi le contrôleur se présentera souvent sous le format suivant :

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

Je ne distinguerais pas les variables POST ou GET, elles sont présentes dans le tableau **$_REQUEST**.
On pourra trouver qu'il y a parfois redondance d'instructions dans les contrôleurs ; ceci est justifié par un désir de clarté dans la présentation des responsabilités des options des contrôleurs.
On trouvera un fichier (préfixé par c_) contrôleur par cas d'utilisation ainsi qu'un *contrôleur principal* qui oriente vers les différents contrôleurs.

### 3.c Le modèle

C'est la couche (bibliothèque de fonctions) qui accède à la base de données. Ainsi ses fonctions retournent soit une donnée calculée, soit une ligne, soit un tableau ; par exemple pour obtenir toutes les catégories de produits :

```php
function getLesCategories()
{
     $connexion = connexion();
     $req="select * from categorie";
    $rsCategorie = mysql_query($req, $connexion);
    $lgCategorie = mysql_fetch_array($rsCategorie);
     // BOUCLE SUR LES CATEGORIES
    $lesCategories=array();
    while ($lgCategorie != FALSE)
    {
        $lesCategories[]=$lgCategorie;
        $lgCategorie = mysql_fetch_array($rsCategorie);
    }
     mysql_close();
     return $lesCategories;
```

Cette fonction **retourne un tableau** à deux dimensions.
La vue qui va utiliser cette fonction va parcourir la tableau et insérer du code html :

```php
<?php
foreach( $lesCategories as $uneCategorie)
{
    $idCategorie = $uneCategorie['id'];
    $libCategorie = $uneCategorie['libelle'];
    $url ="<a href=index.php?uc=voirProduits&categorie=$idCategorie&action=voirProduits>$libCategorie </a>";
    echo "<tr><td>".$url."</td></tr>";
}
?>
```

Cette technologie **charge en mémoire un tableau de données** (couche modèle), le contrôleur va dispatcher ce tableau à la vue qui parcours le tableau pour placer la couche de présentation (en html). Ce fonctionnement a bien sûr un coût en terme d'utilisation de ressources machine ; c'est le prix à payer pour un développement plus cohérent. Par contre ceci allège la base de données. Les framework (zend, Symphonie) qui mettent en oeuvre cette techno de manière plus industrielle utilisent des mécanismes de cache (sur disque) ou chargements partiels paramétrables.
Dans la couche modèle peuvent figurer aussi des fonctions *métiers*, règles de gestion du contexte.

**4) Mise en oeuvre dans l'application LeCadeau.**

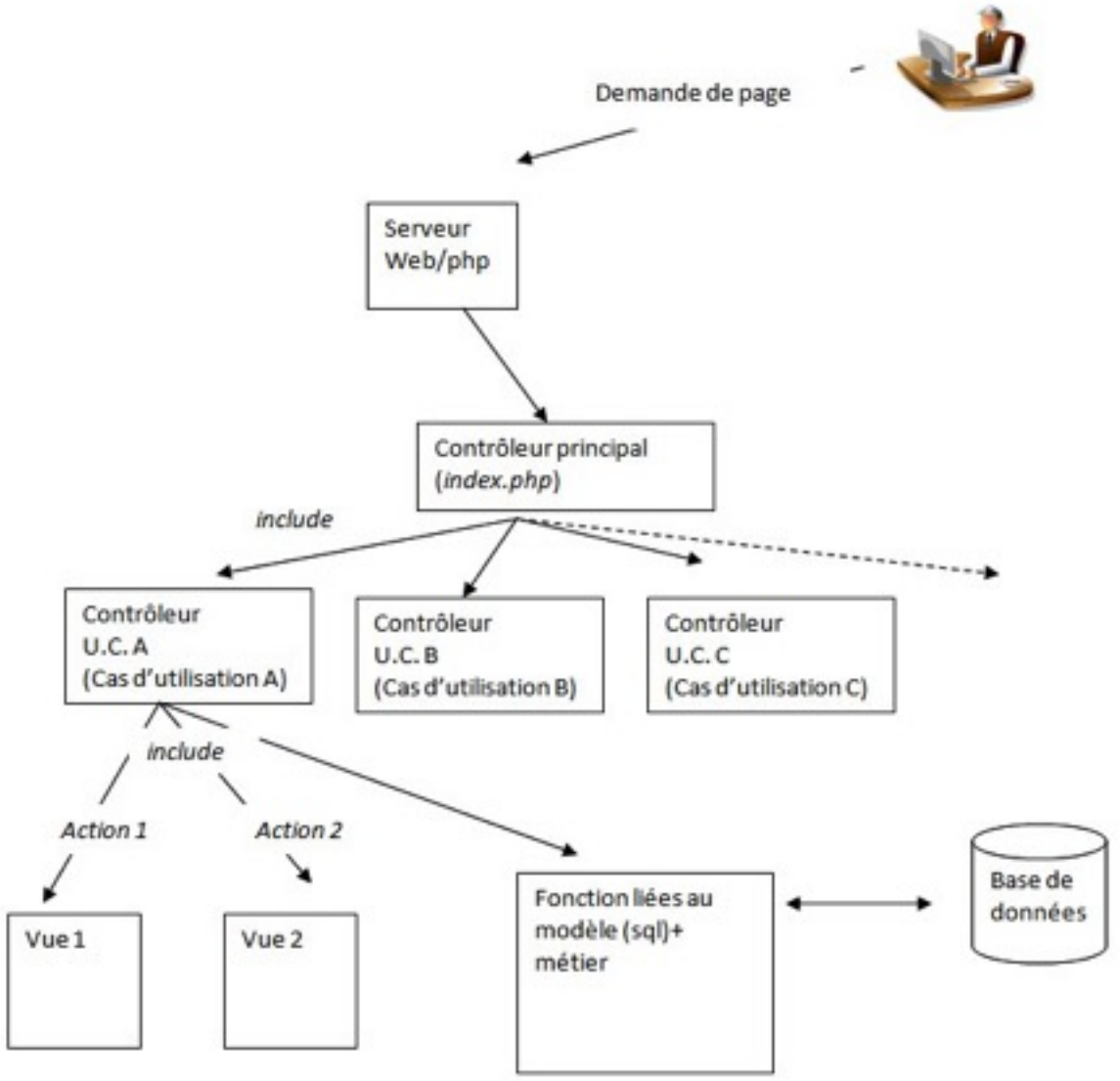
La page index joue le rôle de contrôleur principal ; c'est à lui de dispatcher vers les trois cas d'utilisation.

**On peut shématiser le fonctionnement du chargement des pages (include) :**

L'utilisateur ne chargera que la page index dans laquelle seront inclus les contrôleurs et vues corrsepondant à l'action demandée.

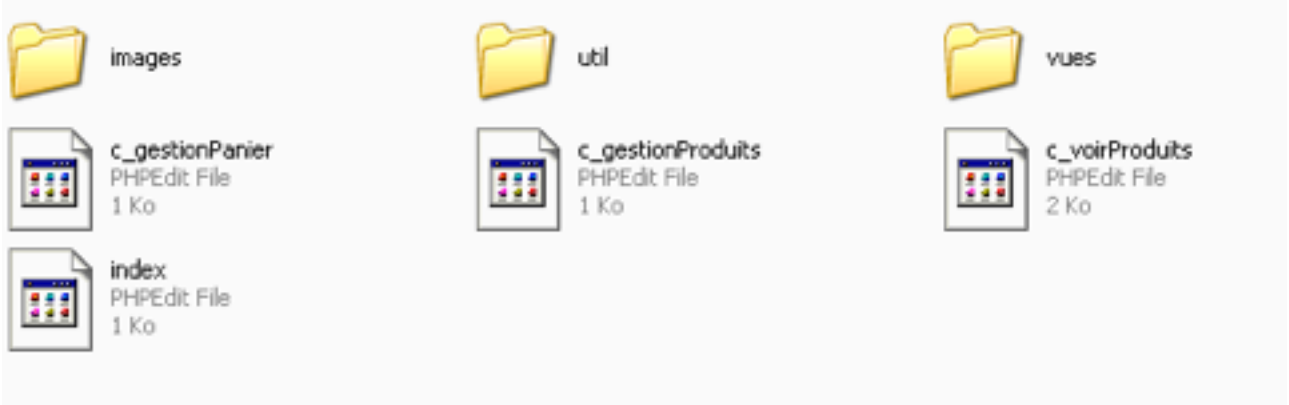
**4.c Gestion des erreurs**

Les erreurs liées à des saisies incorrectes sont signalées dans une sous-vue distincte (*v_erreurs.php*) ;




### 4.d Organisation des fichiers

Voici une organisation possible :




Les contrôleurs (c_...) sont dans le répertoire principale, les vues (v_..) et sous-vues sont dans un répertoire particulier, le répertoire util contient les fichiers du modèle.

### 5) Démarche conseillée

**5.1 Définir les cas d'utilisation.**

D'abord en décomposant l'application en petites fonctionalités (gérer un produit, éditer un état, passer une commande, etc...). Trouver les acteurs, dessiner le diagramme des cas d'utilisation.

**5.2 Préciser chaque cas d'utilisation**

Indiquer textuellement les interactions entre l'utilisateur et le système ; par exemple le premier cas d'utilisation (consultation des produits ) pourrait se présenter ainsi :
Nom du cas : consulter les articles
Acteur : un internaute

Scénario normal (le plus fréquent)

1. **L'internaute demande à consulter les articles**
2. Le système retourne la liste des catégories
3. **L'internaute sélectionne une catégorie**
4. Le système retourne la liste des articles de la catégorie choisie

Scénario particulier

5. **L'internaute demande à déposer un article dans son panier**
6. le système ajoute l'article au panier

On remarque que l'on retrouvera bien les trois *actions* (en gras) présentes dans le contrôleur du cas d'utilisation (cf plus haut)
De même pour la gestion du panier
Nom du cas : gérer le panier
Acteur : l'internaute
Scénario normal

1. **L'internaute demande à voir son panier**
2. Le système retourne la liste des articles du panier

Scénario étendu

3. **L'internaute demande la suppression d'un article**
4. Le système retire l'article du panier
5. **L'internaute demande à passer commande**
6. Le système retourne un formulaire pour saisies
7. **L'internaute remplit le formulaire et l'envoie**
8. Le système enregistre la commande

Scénario particulier

8.1 Le système constate une erreur de saisie ; il en informe l'internaute, retour à 6

Nous retrouvons les 4 actions et la gestion d'erreur alternative dans le contrôleur présenté plus haut.
Cette phase de description textuelle est donc très importante.

**5.3 Imaginer les *vues externes***

Il s'agit ici de dessiner comment se présenteront chaque page, telle que la voie l'internaute. Ceci permet de mettre en évidence les sous-vues qui peuvent être partagées par plusieurs cas d'utilisation, cf plus haut pour le bandeau principal ou l'affichage de la sous-vue des erreurs.

**5.4 Construire l'architecture physique de l'application.**

Créer les répertoires, les contrôleurs vides de code, le contrôleur principal, etc...

**5.5 Ecrire le contrôleur principal**

C'est le fichier index.php avec le switch pointant sur chaque cas d'utilisation

**5.6 Coder chaque cas d'utilisation**

**tester.**

6.1 ) Modifier la vue externe des produits. Il s'agit d'ajouter le prix du produit dans les détails présentés.

**6.2 ) Le cas d'utilisation de Back Office (gestion des produits) n'est pas traité. Développer ce cas en s'appuyant sur le cas d'utilisation présenté plus bas et en suivant la démarche proposée.**

Nom du cas : gérer les articles

Acteur : l'administrateur
Scénario normal (le plus fréquent)
1. L'administrateur demande à se connecter
2. Le système demande le nom de user et le mot de passe.
3. L'internaute saisit le nom de user et le mot de passe
4. Le système retourne la liste des catégories
5. L'administrateur sélectionne une catégorie
6. Le système retourne la liste des articles de la catégorie (id et description) avec pour chaque article la possibilité de modifier ou supprimer. Une option d'ajout de produit pour cette catégorie est proposée
7. L'administrateur sélectionne modifier
8. Le système retourne les infos de l'article, description et prix.
9. L'administrateur modifie les infos et envoie le formulaire
10. le système enregistre la modification
Extensions
11 L'administrateur sélectionne supprimer
12 Le système demande si l'article doit bien être supprimé
13 L'administrateur confirme ou pas
14 Le système supprime ou non

15 L'administrateur demande à créer un nouveau produit
16 Le système retourne un formulaire de saisies
17 L'administrateur remplit le formulaire et envoie le formulaire
18 Le système enregistre les informations
Scénarios particuliers
4.1 Les infos de connexion sont invalides. Retour à 2 (avec un message d'erreur)
10.1 Les infos reçues ne sont pas valides. Retour à 8 (avec un message d'erreur)
18.1 des informations ne sont pas valides ; retour à 15 (avec un message d'erreur)
x.1 A tout moment, l'administrateur demande à retourner dans l'administration
x.2 Le système ne lui retourne pas de formulaire de connexion mais le catalogue des catégories.
- Dessiner rapidement les vues externes.
- Coder les pages

Remarque : une table **Administrateur** vous pouvez les créer avec deux lignes :

```
INSERT INTO Administrateur VALUES ('1','toto','toto');
INSERT INTO Administrateur VALUES ('2','titi','titi');
```