RAPPORT DE PROJET

Présenté par
Sébastien Givone

Numéro d'auditeur
RHA027140

Unité
NFA021

Formation
Certificat Professionnel Webmaster

Année :
2015 / 2016

Nom du projet :
Cuisine du CNAM

I - Objectif du projet

Le projet a avant tout un but pédagogique. En effet, chaque unité composant la formation permettant l'obtention du Certificat Professionnel Webmaster, m'a permit d'apprendre différents langages de programmation :
- SQL
- HTML / CSS
- Javascript
- PHP

Le projet consiste en la création d'un site web dynamique complet en utilisant les différents langages appris.

II – Présentation du site

1 – Résumé

Le site est un site de recette de cuisine, permettant aux visiteurs de consulter des recettes que les membres du site auront publiés. Il pourra chercher une recette spécifique en filtrant les recettes par leur catégories et leur sous catégories.

Un visiteur pourra devenir membre du site en s'y inscrivant ; à partir de là il pourra à son tour publier des recettes. Il pourra aussi modifier ou effacer son compte et ses recettes. Enfin, il pourra noter et commenter une recette qui n'est pas la sienne.

Enfin, un des membres du site est l'administrateur. Celui-ci pourra en plus modifier ou effacer les comptes des utilisateurs. Il pourra aussi ajouter ou supprimer des catégories ou des sous catégories.

2 – Général

Le site est composé de plusieurs pages dont certaines ne sont accessibles qu'aux membres authentifiés ou qu'à l'administrateur.

Les visiteurs auront accès aux pages suivantes :

- *Page d'accueil*
- *Lister les recettes*
- *Consulter une recettes*
- *Créer un compte*

Les membres authentifiés auront en plus accès aux page suivantes :

- *Consulter son compte*
- *Modifier son compte*
- *Créer une recettes*
- *Modifier une recette*
- *Lister ses recettes*

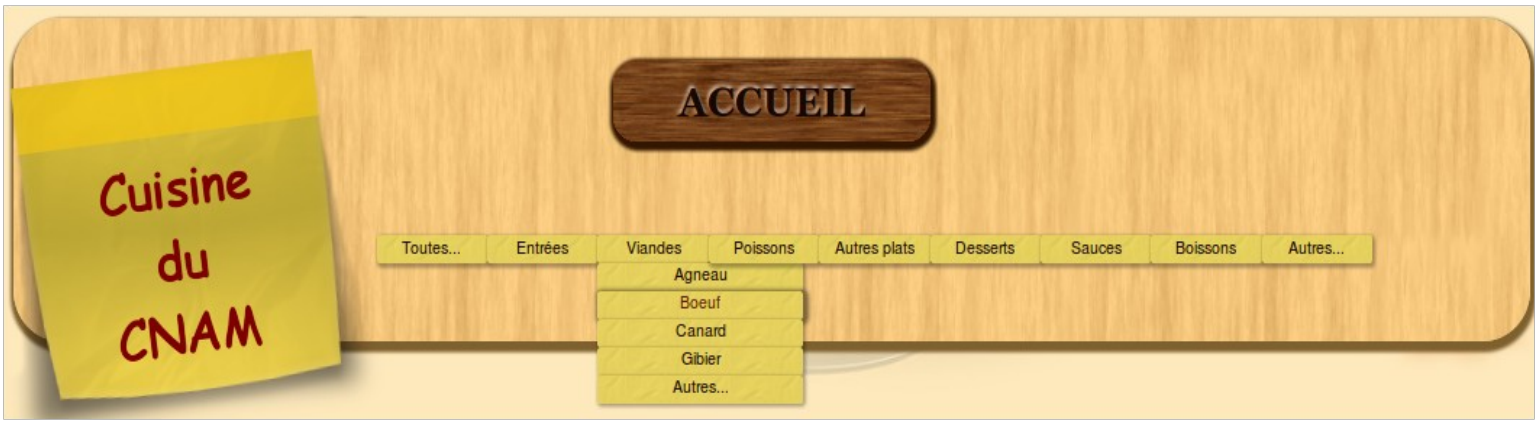
Enfin l'administrateur aura aussi accès aux pages suivantes :

- *Gestion des catégories*
- *Lister les membres*
- *Consulter le compte d'un membre*
- *Modifier le compte d'un membre*

Toutes les pages à l’exception des pages de création et de modification des recettes et des comptes ainsi que les pages d'erreur, sont composées d'un header en haut comportant un logo cliquable envoyant à la page d'accueil et d'un menu des catégorie, d'un footer en bas comportant un lien revoyant à la page d'accueil, d'un panneau d'identification sur la gauche du site prenant un quart de la largeur et enfin le contenue à proprement parlé à droite entre le header et le footer et prenant

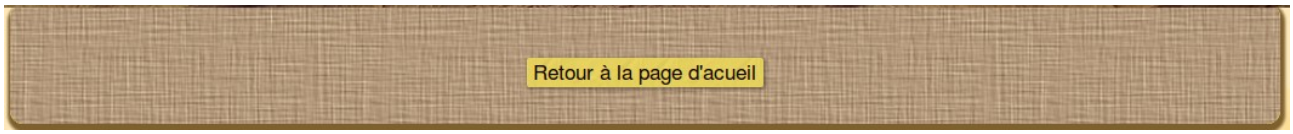
les trois quart de la largeur.

Le header :

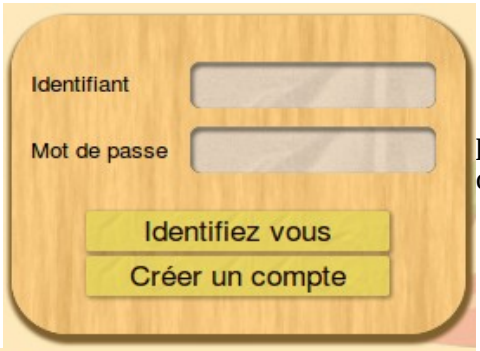


Le menu des catégories est composé d'une liste de lien correspondant à chaque catégories ; lorsque le curseur de la souris survol un de ces liens une liste déroulante de ses sous catégorie apparaît. Lorsqu'un lien de catégorie est cliqué le site envoie le visiteur sur la page de listage des recettes avec une liste de recette correspondant à la catégorie choisie : lorsque un lien de sous catégorie est cliqué alors la liste des recettes est alors trié selon la sous catégorie choisie. Lorsque l'on clique sur le lien « Toutes... », toutes les recette du site sont listées.

Le footer :



Le panneau d'authentification :



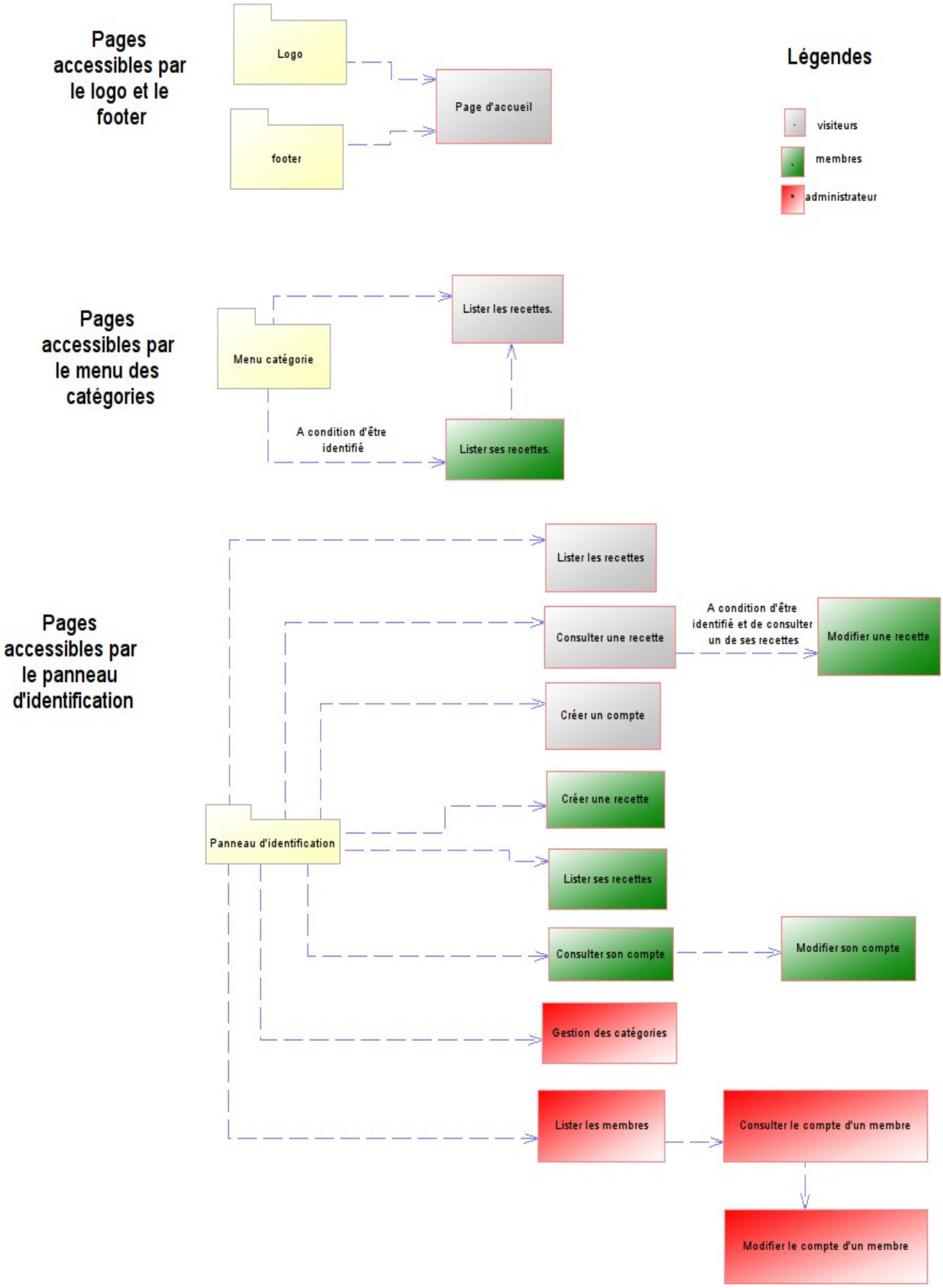
Un visiteur non authentifié verra ce panneau grâce auquel il pourra s'identifier s'il possède un compte, ou se créer un compte.

Un membre identifié verra ce panneau grâce auquel il pourra se déconnecté (pour revenir au panneau précédent), accéder aux données de son compte, créer une recette ou lister ses recette.

Lorsque l'administrateur sera identifié, il aura accès à ce panneau qui est presque identique au panneau des membres identifiés mais il pourra aussi accéder à la liste des membres du site et aussi à la gestion des catégories.

Voici à quoi ressemble une page typique du site :

Voici le diagramme de conception du site :



3 – Détail de chaque page :

Page d'accueil :

La page d'accueil en plus de posséder les header, footer et panneau d'identification classiques comporte un panneau présentant le début d'une recette au hasard parmi toutes les recettes du site. Cliquer sur le titre de la recette nous amène à la page de consultation de la recette.

Lister les recettes :

Cette page en plus de posséder les header, footer et panneau d'identification classiques comporte trois panneaux. Le premier indique sur quels critères sont listé les recettes. Le second comporte une liste des recettes dons la catégorie ou la sous catégorie à été sélectionné ; il suffit de cliquer sur une des recette pour aller à sa page de consultation. Le second panneau présente une recette au hasard parmi toutes les recettes dont la catégorie ou la sous catégorie à été sélectionnée ; cliquer sur le titre de la recette au hasard nous amène à la page de consultation de la recette.

Consulter une recette :

Cette page en plus de posséder les header, footer et panneau d'identification classiques comporte deux panneaux. Le premier panneau présente la recette dans tous ses détails : Sa photo si elle existe, un commentaire général s'il existe, des détails comme le nom de l'auteur, le nombre de personnes, les temps de préparation et de cuisson, la catégorie, la sous catégorie et les catégories de difficulté et de prix ; ensuite vient la liste des ingrédients, puis la listes des étape de préparation et enfin un conseil pour la dégustation s'il existe.
Le second panneau concerne les commentaires : si l'utilisateur est identifié et qu'il ne consulte pas une de ses recettes, alors il peut donner une note à la recette entre 0 et 5 et mettre un commentaire ; si il a déjà mis une note et un commentaire, il peut les modifier. Il y a ensuite la liste des commentaires avec les nom des auteurs et leur notes.

Créer un compte :

Cette page ne comprend ni menu des catégories ni panneau d'identification.
Voici à quoi la page ressemble :

Lorsque un membre est identifié, il peu accéder à d'autres pages :
Consulter son compte :

Cette page en plus de posséder les header, footer et panneau d'identification classiques comporte un panneau qui contient toutes les données du membre identifié : son identifiant (mais pas son mot de passe), son nom, son prénom, son adresse email si elle a été renseignée, et la photo si elle existe. De ce panneau il y a un lien qui amène à la page de modification du compte et un autre lien qui permet de détruire le compte et il y a une case à cocher ou non si le membre en détruisant son compte souhaite détruire toute ses recettes ou les cède à l'administrateur.

Modifier son compte :

Cette page ressemble sensiblement à la page de création de compte si ce n'ai qu'elle possède un champs supplémentaire pour donner son ancien mot de passe pour pouvoir changer de mot de passe. Sinon, tous les champs sont évidement remplis avec les données du compte.

Créer une recettes :

Cette page ne comprend ni menu des catégories ni panneau d'identification.
Elle possède un panneau central dans lequel il y a plusieurs champs à renseigner : le champs texte du titre, un bouton pour ajouter une photo (non obligatoire), une aire de texte pour mettre un commentaire général (non obligatoire), un champs de nombre pour indiquer le nombre de personne, des champs de nombres pour indiquer les temps de préparation et de cuisson en minute, une liste déroulante pour sélectionné une catégorie, une liste déroulante pour sélectionner une sous catégorie, deux listes déroulante pour sélectionner les catégories de prix et de difficulté.
Il y a ensuite deux zone de champs de texte pour les ingrédients et les étape de préparation :

Les boutons « + » permettent d'ajouter un ingrédient ou une étape. Les boutons « - » permettent de supprimer un champs. il doit toujours y avoir au moins un champs d'ingrédient et un champs d'étape, c'est pourquoi dans ce cas le bouton « - » n'apparaîtra pas. Enfin les boutons « flèche haut »et « flèche bas » permettent de permuter un champs avec son voisin du dessus ou du dessous.
Enfin après les ingrédients et la préparation, il y a encore un champs de texte dans lequel on mettre des conseil de dégustation de la recette.

Modifier une recettes :

Lorsqu'un membre consulte sa propre recette (dans la page d'affichage de recette), il a la possibilité de la modifier ; dans ce cas là on accède à la page de modification de recette qui ressemble sensiblement à la page de création de recettes à la différence que les champs sont remplis.

Lister ses recettes :

Cette page ressemble sensiblement à la page de listage général des recettes ; elle

fonctionnent de manière identique (notamment le menu des catégories) mais ne liste que les recettes du membre. Il peut revenir à tout instant à la page de listage de toutes les recettes grâce à un lien.

Lorsque l'administrateur est identifié, il peu accéder à d'autres pages :
Gestion des catégories :

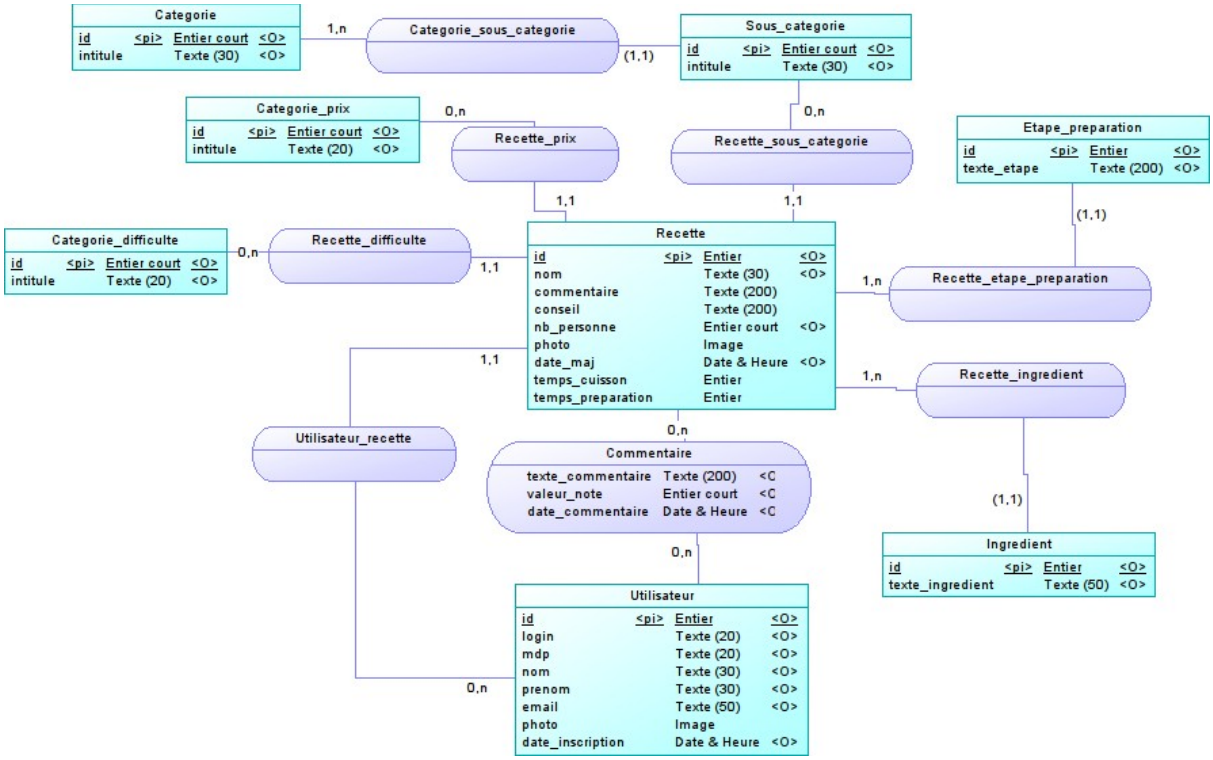
La page ne contient ni menu des catégorie, ni panneau d'identification. Elle comprend une liste de champs de textes dans lesquels sont inscrits les intitulés des différentes catégories, sous catégories, catégorie de prix et catégorie de difficulté. Chaque champs de texte est suivit d'un bouton de modification pour modifier l'intitulé de la catégorie et un bouton de suppression pour supprimer celle ci. Chaque type de catégorie possède forcément une catégorie par défaut qui n'est pas supprimable. Lorsqu'un catégorie est supprimer, les recettes de cette catégorie sont automatiquement mises dans la catégorie par défaut. Enfin il y a champs de texte à la fin de chaque type de catégories et bouton d'ajout pour pouvoir ajouter uhe catégorie.

Lister les membres :

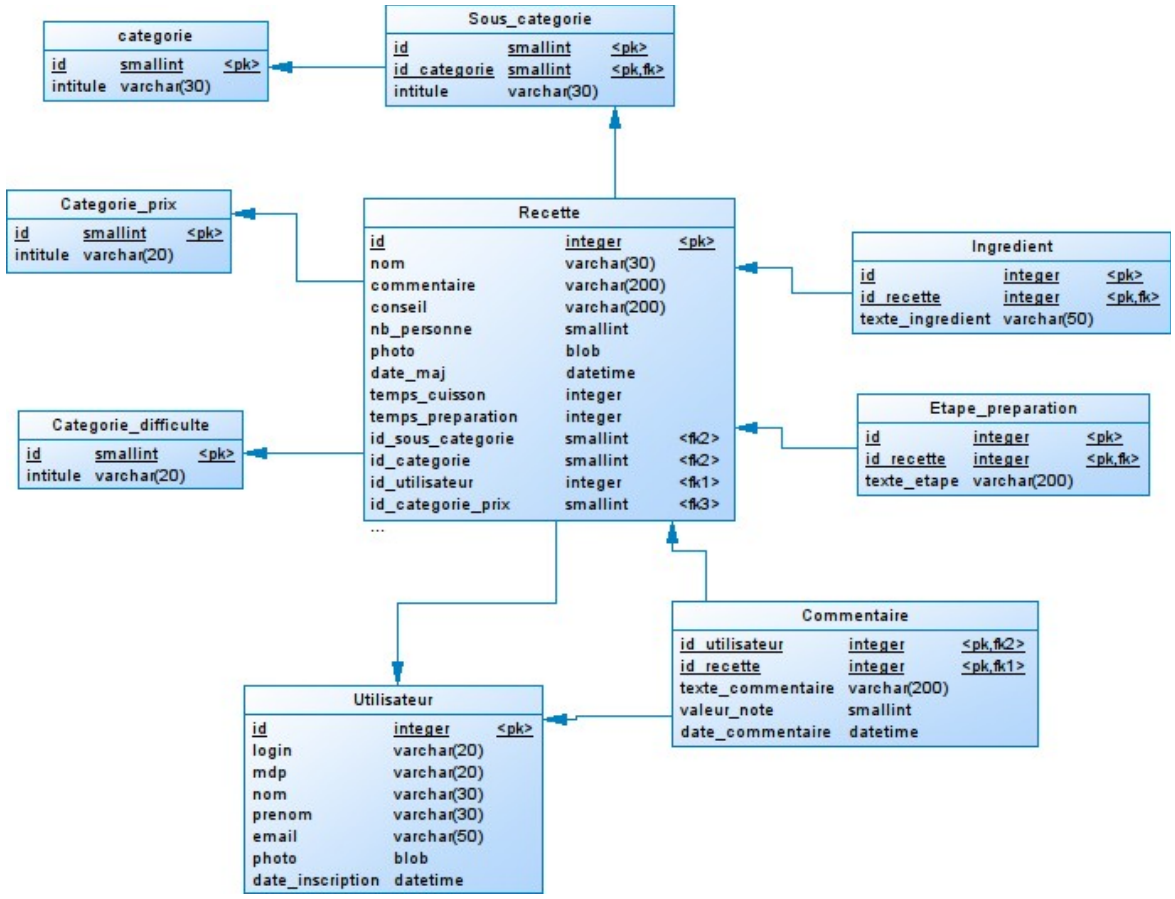
La page ne contient ni menu des catégorie, ni panneau d'identification. Le panneau entre le header et le footer contient la liste des utilisateurs. En cliquant sur l'un d'eux, on arrive sur la page de consultation du compte de l'utilisateur. A partir de cette page il est donc possible de modifier ou de détruire le compte avec les recettes ou en transférant celles ci à l'administrateur.

III – Base de donnée

1 – MCD



2 – MLD



3 – explication

La base de donnée a été créer avec le langage SQL pour le SGBD MySQL.
Elle est composée de 9 tables :

- *Utilisateur*
- *Recette*
- *Ingredient*
- *Etape_preparation*
- *Categorie*
- *Sous_categorie*
- *Categorie_prix*
- *Categorie_difficulte*
- *Commentaire*

Comme on peut le voir une recette possède un nombre variable d'ingrédients et d'étapes de préparation ; c'est pourquoi ceux ci sont écrits dans des tables à part. Par ailleurs des tables pour les catégories ont été créées pour y écrire leur intitulé. Enfin la table Commentaire dépend à la fois de la table utilisateur et de la table Recette car il existe un unique commentaire pour une recette et un utilisateur donné.
A noter que Les photos des utilisateur et des recettes sont enregistré dans un répertoire spécifique et elles sont liés à leur propriétaire par leur nom écrit dans la table adéquat.

IV – Programmation

1 – Côté client

Du côté client, chaque page est écrite dans le langage HTML5 et est mise en page grâce à une feuille de style commune écrite dans le langage CSS3. Certain mécanisme de navigation dans les pages (notamment le menu des catégories) nécessite l'utilisation d'un fichier de scriptes écrits dans le langage Javascript. Pour faciliter l'écriture de ces scriptes, le framework jQuery à été utilisé. Chaque page est accessible grâce à un lien dans un formulaire de cette forme :

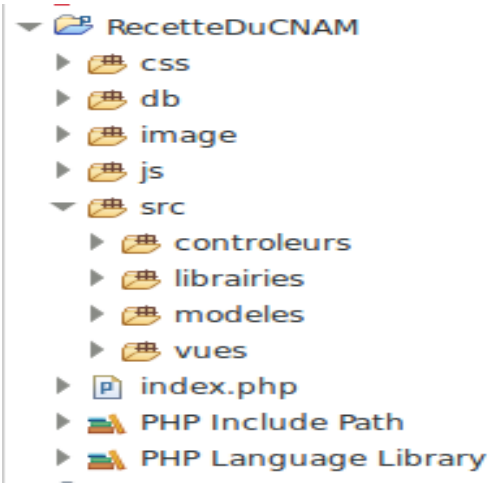
```
<form method="post" action="" name="retour_accueil">
	<input type="hidden" name="page" id="page" value="accueil" />
	...
	<!-- d'autres inputs →
	...
	<a href="javascript:document.retour_accueil.submit()">Retour à l'accueil</a>
</form>
```

Dans cet exemple le lien permet un retour à la page d'accueil. Le premier input est toujours de type hidden et à pour ses attributs « id » et « name » le nom « page ». L'URL d'une page est donc toujours la racine du répertoire dans lequel est installé le site. La valeur de l'input « page » détermine la page qui va être afficher.

2 – côté serveur

Du côté serveur, chaque page HTML du site est construite grâce au langage PHP. L'application est structurée en suivant le patron d'architecture MVC (Modèle-vue-contôleur). Comme vu précédemment, chaque appelle de page amène à la racine du site. En fait le fichier index.php installé à la racine du site est exécuté.

Voici la structure du projet (fait avec l'IDE eclipse) :



Le dossier « css » comporte la feuille de style, le dossier « db » contient les données nécessaire à la connexion à la base de données et le chemin des photos des utilisateurs et des recettes, le dossier « image » contient toutes les images nécessaires à l'embellissement du site et est lié à la feuille de style, le dossier « js » comprend le fichier des scriptes Javascript et enfin le dossier « src » contient l'essentiel du site programmé en PHP. Le sous-dossier « controleurs » de « src » comprend tous ce qui est relatif aux contrôleurs, le sous-dossier « modeles » tous ce qui est relatifs aux modèles, le sous-dossier « vues » tous ce qui est relatifs aux vues et le sous dossier « librairies » tous ce qui est commun à la fois aux contrôleurs, aux modèles et aux vues.

Voici le contenue de la page index.php :

```php
<?php
	// include
	include_once (dirname(__FILE__) . '/db/db_define.inc.php');
	include_once (dirname(__FILE__) . '/src/librairies/define/Define.inc.php');
	include_once (dirname(__FILE__) . '/src/librairies/exception/RequeteException.class.php');
	include_once (dirname(__FILE__) . '/src/librairies/fb/fb.php');


	try { try
	{
		// choix de la page
		if (isset($_REQUEST['page']))
			$donneesControleur['page'] = $_REQUEST['page'];
		else
			$donneesControleur['page'] = 'accueil';

		switch ($donneesControleur['page'])
		{
			// accueil
			case 'accueil' :
				include_once (dirname(__FILE__) .'/src/controleurs/Accueil.inc.php');
				break;
			// creer un compte
			case 'creer_compte' :
				include_once (dirname(__FILE__) .'/src/controleurs/Creer_compte.inc.php');
				break;
			// afficher un compte
			case 'afficher_compte' :
				include_once (dirname(__FILE__) .'/src/controleurs/Afficher_compte.inc.php');
				break;
			// gerer un compte
			case 'modifier_compte' :
				include_once (dirname(__FILE__) .'/src/controleurs/Modifier_compte.inc.php');
				break;
			// creer une recette
			case 'creer_recette' :
				include_once (dirname(__FILE__) .'/src/controleurs/Creer_recette.inc.php');
				break;
			// afficher une recette
			case 'modifier_recette' :
				include_once (dirname(__FILE__) .'/src/controleurs/Modifier_recette.inc.php');
				break;
			// afficher une recette
			case 'afficher_recette' :
				include_once (dirname(__FILE__) .'/src/controleurs/Afficher_recette.inc.php');
				break;
			// lister une recette
			case 'lister_recette' :
				include_once (dirname(__FILE__) .'/src/controleurs/Lister_recette.inc.php');
				break;
			// lister une recette d'utilisateur
			case 'lister_recette_utilisateur' :
				include_once (dirname(__FILE__) .'/src/controleurs/Lister_recette_utilisateur.inc.php');
				break;
			// lister les utilisateurs
			case 'lister_utilisateurs' :
				include_once (dirname(__FILE__) .'/src/controleurs/Lister_utilisateurs.inc.php');
				break;
			// gerer les categories
			case 'gerer_categorie' :
				include_once (dirname(__FILE__) .'/src/controleurs/Gerer_categories.inc.php');
				break;
			// image
			case 'image' :
				include_once (dirname(__FILE__) .'/src/controleurs/Image.inc.php');
				break;
			// le nom de la page n'existe pas
			default :
				throw new PageInconnueExcep($donneesControleur['page']);
				break;
		}
	}


	// exceptions
	catch (PDOException $e)
	{
		throw new ErreurInterneExcep($e->getMessage());
	}}
	catch (AppException $e)
	{
		$excepMessage = $e->getMessage();
```

```
                include_once (dirname(__FILE__) .'/src/controleurs/Exception.inc.php');
                die();
        }
```

C'est une sorte de tableau de bord qui va exécuter le contenue du code du contrôleur de la page associé à la valeur de la requête de nom « page ». Cette requête correspond au premier input de type hidden de chaque formulaire comme vu précédemment. Si une exception est lancé au cours de l'exécution, une page de type Exception est exécuté comme on peut le voir dans le dernier « catch » de la page.

Le contrôleur :

Le contrôleur est donc exécuté à partir du fichier index.php grâce à la directive PHP include_once. Ainsi, à chaque page est associé un fichier de contrôleur. Dans les contrôleurs sont récupérés les données envoyés par les formulaires. C'est dans les contrôleurs qu'est vérifié la validité des données de formulaires ; si celles-ci ne sont pas valides alors une exception est levée (voir index.php pour le traitement des exceptions).
Voici un exemple de contrôleur (le contrôleur de la page d'accueil) :

```php
<?php
        // includes
        include_once (dirname(__FILE__) . '/commun/LoginControleur.class.php');
        include_once (dirname(__FILE__) . '/commun/RecetteControleur.class.php');


        // donnees du controleur
        // login
        $donneesControleur['identification'] = LoginControleur::Identification();
        $donneesControleur['deconnexion'] = LoginControleur::Deconnexion();
        // destruction recette
        $donneesControleur['detruire_recette'] = RecetteControleur::DestructionRecette();
        // destruction compte
        $donneesControleur['detruire_compte'] = LoginControleur::DestructionCompte();


        // modele
        include_once (dirname(__FILE__) . '/../modeles/Accueil.inc.php');

        // vue
        include_once (dirname(__FILE__) . '/../vues/Accueil.inc.php');
```

Comme on peut le voir, chaque donnée du contrôleur est stockée dans la variable « $donneesControleur » et est fournie par des méthodes statiques de classes utilisés en commun par plusieurs contrôleur. Ces classes sont :
- LoginControleur qui fourni les données des formulaires d'identification et de création de compte.
- RecetteControleur qui fourni les données des formulaires de création et de modification des recettes.
- CritereControleur qui fourni les données des formulaires du menu des catégories.
- CommentaireControleur qui fourni les données des formulaires de la création et la modification des commentaires.

A la fin du fichier du contrôleur sont appelés successivement le modèle de la page et la vue de la page.
Voici la méthode statique « Identification » de la classe « LoginControleur » :

```php
public static function Identification()
{
        $donnees['identifie'] = FALSE;
        if (isset($_REQUEST['login']) && isset($_REQUEST['mdp']))
        {
                $donnees['login'] = $_REQUEST['login'];
```

```
                $donnees['mdp'] = $_REQUEST['mdp'];
                if (is_null($donnees['login']) || !is_string($donnees['login']) || empty($donnees['login']) ||
                    is_null($donnees['mdp']) || !is_string($donnees['mdp']) || empty($donnees['mdp']))
                        throw new LoginIncorrectExcep();
                $donnees['identifie'] = TRUE;
        }
        return $donnees;
}
```

Le modèle :

Le modèle est exécuté à partir du contrôleur associé grâce à la directive PHP include_once (comme vu précédemment). Dans les modèles sont récupérés ou modifiés les données de la base de données et de la session. C'est aussi ici que sont enregistrés les photos des utilisateurs et des recettes. C'est dans les modèle qu'est vérifié la validité des données du modèle (comme la validité du mot de passe d'un utilisateur, le nombre de caractères maximum d'une donnée texte ou la taille maximal d'une image) ; si celles-ci ne sont pas valides alors une exception est levée (voir index.php pour le traitement des exceptions).
Voici un exemple de modèle (le modèle de la page d'accueil) :

```
<?php
        // include
        include_once (dirname(__FILE__) . '/../librairies/dao/Connexion.class.php');
        include_once (dirname(__FILE__) . '/commun/LoginModele.class.php');
        include_once (dirname(__FILE__) . '/commun/CritereModele.class.php');
        include_once (dirname(__FILE__) . '/commun/RecetteModele.class.php');


        // connexion a la base de donnee
        $conn = new Connexion(SERVEUR, USER, PWD);
        ModeleBase::SetConnexion($conn);


        // donnees du modele
        // destruction compte
        $donneesModele['detruire_compte'] = FALSE;
        if ($donneesControleur['detruire_compte']['existe'])
        {
                $donneesModele['detruire_compte_succes'] =
LoginModele::DestructionCompte($donneesControleur['detruire_compte']['garder_recette']);
                $donneesModele['detruire_compte'] = TRUE;
        }
        // login
        if ($donneesControleur['deconnexion']) $donneesModele['identification'] = LoginModele::Deconnexion();
        else $donneesModele['identification'] =
LoginModele::Identification($donneesControleur['identification']);
        // categories / sous categories
        $donneesModele['categories'] = CritereModele::Categories();
        // destruction recette
        $donneesModele['detruire_recette'] = FALSE;
        if ($donneesControleur['detruire_recette'])
        {
                $donneesModele['detruire_recette_succes'] = RecetteModele::DestructionRecette();
                $donneesModele['detruire_recette'] = TRUE;
        }
        // random recette
        $donneesModele['critere'] = CritereModele::CritereRechercheToutes();
        $donneesModele['recette_random'] = RecetteModele::RandomRecette($donneesModele['critere']);
```

Comme on peut le voir, chaque donnée du modèle est stockée dans la variable « $donneesModele» et est fournie par des méthodes statiques de classes utilisés en commun par plusieurs modèles et modulée par la variable « $donneesControleur ». Ces classes sont :
- LoginModele qui fourni, créé, modifie et détruit les données lié aux utilisateurs.
- RecetteModele qui fourni, créé, modifie et détruit les données lié aux Recettes.
- CritereModele qui fourni, créé, modifie et détruit les données lié aux Catégories.
- CommentaireModele qui fourni, créé, modifie et détruit les données lié aux Commentaires.
- ImageModele qui affiche, créé, modifie et détruit les images.

Voici les méthodes statiques « Identification » et « IdentificationSession » de la classe « LoginModele» :

```php
// identification
public static function Identification($ident)
{
	parent::StartSession();
	if ($ident['identifie'])
	{
		$login = trim($ident['login']);
		$mdp = trim($ident['mdp']);
		$donnees['identifie'] = TRUE;
		$daoUt = new DAOUtilisateur(parent::GetConnexion());
		$donnees['utilisateur'] = $daoUt->RetrieveByLogin($login);
		if (is_null($donnees['utilisateur']) || (strcmp($mdp, $donnees['utilisateur']['mdp']) != 0))
			throw new LoginIncorrectExcep();
		$_SESSION['id_utilisateur'] = $donnees['utilisateur']['id'];
		return $donnees;
	}
	return self::IdentificationSession();
}

// identification par session
public static function IdentificationSession()
{
	parent::StartSession();
	if (isset($_SESSION['id_utilisateur']))
	{
		$daoUt = new DAOUtilisateur(parent::GetConnexion());
		$donnees['utilisateur'] = $daoUt->RetrieveById($_SESSION['id_utilisateur']);
		$donnees['identifie'] = TRUE;
		return $donnees;
	}
	$donnees['identifie'] = FALSE;
	return $donnees;
}
```

Comme on peut le voir dans cet exemple, lorsqu'un utilisateur s'identifie la première fois, son id est sauvegarder dans les données de session ($_SESSION['id_utilisateur']) ce qui fait que les prochaines fois il sera automatiquement identifié ; il n'aura donc plus besoin de fournir l'identifiant et le mot de passe.
Le modèle se connecte à la base de données par une instance de la classe Connexion :

```php
<?php
	// assure la connexion avec une base de donnee
	class Connexion
	{
		// attributs
		private $dbh;
		private $serveur;
		private $user;
		private $pwd;

		// constructeur
		public function __construct($serveur, $user, $pwd)
		{
			$this->SetConnexion($serveur, $user, $pwd);
		}

		// destructeur
		public function __destruct()
		{
			$dbh = null;
		}

		// creation d'une nouvelle connexion
		public function SetConnexion($serveur, $user, $pwd)
		{
			$this->serveur = $serveur;
			$this->user = $user;
			$this->pwd = $pwd;
			$this->dbh = new PDO($this->serveur, $this->user, $this->pwd);
			$this->dbh->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
		}

		// getters des attributs
		public function GetDB()
		{
			return $this->dbh;
		}

		public function GetServeur()
		{
			return $this->serveur;
		}

		public function GetUser()
		{
			return $this->user;
		}

		public function GetPwd()
		{
			return $this->pwd;
		}

	}
```

Les classes communes des modèles lisent, créeent, modifient et détruisent les données de la base de données par le biais de classes DAO (Data Access Object). A chaque table de la base de données est associé une classe DAO qui s'occupe des opération de CRUD (create, read, update, delete) sur la table.
Voici la classe de base des DAO suivi du DAO de la table utilisateur (quelque peu écourté dans cet exemple) :

Classe DAO :

```php
<?php
	include_once(dirname(__FILE__) . '/Connexion.class.php');

	// class de base des DAO (Data base Access Object) permettant de récupérer les donnees sur une base de donnee
	abstract class DAO
	{
		// attributs
		protected $connexion;

		// constructeur
		public function __construct(Connexion $connexion)
		{
			$this->connexion = $connexion;
		}

		// getter
		public function GetConnexion()
		{
			return $this->connexion;
		}

		// prepare une requette sql
		protected function Prepare($sql)
		{
			$resultat = $this->connexion->GetDB()->prepare($sql);
			$resultat->setFetchMode(PDO::FETCH_ASSOC);
			return $resultat;
		}

		// renvoie generique
		protected function RetrieveAllGen($sql)
		{
			// execute la requette
			$resultat = $this->Prepare($sql);
			$resultat->execute();
			return $this->RetrieveAllGenEx($resultat);
		}

		protected function RetrieveAllGenEx($resultat)
		{
			// creation de la liste
			$listElt = array();
			if (!is_null($resultat))
			{
				foreach ($resultat as $resElt)
					$listElt[] = $resElt;
			}

			// renvoie la liste
			return $listElt;
		}

		protected function RetrieveGen($sql)
		{
			// execute la requette
			$resultat = $this->Prepare($sql);
			$resultat->execute();
			return $this->RetrieveGenEx($resultat);
		}

		protected function RetrieveGenEx($resultat)
		{
			// renvoie le premier element
			if (!is_null($resultat))
			{
				foreach ($resultat as $resElt)
					return $resElt;
			}
			return null;
		}

	}
```

Classe DAOUtilisateur :

```php
<?php
	include_once (dirname(__FILE__) . '/DAO.class.php');
	include_once (dirname(__FILE__) . '/DAORecette.class.php');
	include_once (dirname(__FILE__) . '/DAOCommentaire.class.php');

	final class DAOUtilisateur extends DAO
	{
```

```php
		// constructeur
		public function __construct($connexion)
		{
			parent::__construct($connexion);
		}

		// CRUD
		// retrieve
		public function RetrieveAll()
		{
			$sql =
<<<SQL
			SELECT id, login, mdp, nom, prenom, email, photo, date_inscription, admin
			FROM v_utilisateur
SQL;
			return $this->RetrieveAllGen($sql);
		}

		public function RetrieveById($id)
		{
			$sql =
<<<SQL
			SELECT id, login, mdp, nom, prenom, email, photo, date_inscription, admin
			FROM v_utilisateur
			WHERE id = :id
SQL;
			$resultat = $this->Prepare($sql);
			$resultat->bindParam('id', $id, PDO::PARAM_INT);
			$resultat->execute();
			return $this->RetrieveGenEx($resultat);
		}


		// create
		public function Create($ut)
		{
			// creation d'un utilisateur
			$sql =
<<<SQL
			INSERT INTO v_utilisateur (login, mdp, nom, prenom, email, date_inscription)
			VALUES (:login, :mdp, :nom, :prenom, :email, :date_inscription)
SQL;
			$resultat = $this->Prepare($sql);
			self::BindUtilisateurBase($resultat, $ut, FALSE);
			$resultat->execute();
			return $this->connexion->GetDB()->lastInsertId();
		}

		// update
		public function Update($ut)
		{
			// modification d'un utilisateur
			$sql =
<<<SQL
			UPDATE v_utilisateur SET login = :login, mdp = :mdp, nom = :nom, prenom = :prenom, email = :email
			WHERE v_utilisateur.id = :id
SQL;
			$resultat = $this->Prepare($sql);
			self::BindUtilisateurBase($resultat, $ut, TRUE);
			$resultat->bindParam('id', $ut['id'], PDO::PARAM_INT);
			$resultat->execute();
			return $ut['id'];
		}

		// delete
		public function Delete($ut)
		{
			// destruction des recettes
			$daoRec = new DAORecette($this->connexion);
			$listRec = $daoRec->RetrieveIdByUtilisateur($ut['id']);
			foreach ($listRec as $rec)
				$daoRec->Delete($rec);

			// destruction des commentaires
			$daoCommentaire = new DAOCommentaire($this->connexion);
			$daoCommentaire->DeleteByUtilisateur($ut['id']);

			// destruction de la photo
			$tmpUt = $this->RetrievePhoto($ut);
			if (isset($tmpUt['photo']))
			{
				$filename = IMAGE_CHEMIN . $tmpUt['photo'];
				if (file_exists($filename))
					unlink($filename);
			}

			// execute la requette
			$sql =
<<<SQL
			DELETE FROM v_utilisateur
			WHERE v_utilisateur.id = :idUt
SQL;
			$resultat = $this->Prepare($sql);
			$resultat->bindParam('idUt', $ut['id'], PDO::PARAM_INT);
			return $resultat->execute();
		}


		// fonctions privees
		private static function BindUtilisateurBase(&$resultat, $ut, $maj)
		{
			$resultat->bindParam('login', $ut['login'], PDO::PARAM_STR);
			$resultat->bindParam('mdp', $ut['mdp'], PDO::PARAM_STR);
			$resultat->bindParam('nom', $ut['nom'], PDO::PARAM_STR);
			$resultat->bindParam('prenom', $ut['prenom'], PDO::PARAM_STR);
```

```
                        $resultat->bindParam('email', $ut['email'], isset($ut['email']) ? PDO::PARAM_STR : PDO::PARAM_NULL);
                        if (!$maj)
                                $resultat->bindParam('date_inscription', $ut['date_inscription'], PDO::PARAM_STR);
                }

        }
```

Comme on peut le voir, c'est dans les instances de DAOs que sont exécutés les requêtes SQL de lecture et écriture de la base de données.

La vue :

La vue est exécuté à partir du contrôleur associé grâce à la directive PHP include_once (comme vu précédemment). La vue est exécuté après le modèle. Dans les vues sont récupérés les données affichables à partir des données de contrôleurs ($donneesControleur) et surtout des données de modèles ($donneesModele). La vue se charge ensuite de construire la page HTML5 en utilisant les données affichables récupérées. Pour construire une page sont utilisées des méthodes statique de la classe HtmlStruct :

```
<?php
        class HtmlStruct
        {
                // balises principales
                public static function DebutHtml($titre, array $styles, array $scripts)
                {
                        ?>
<!DOCTYPE html>
<html>
        <head>
                <meta charset="UTF-8">
                <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
                <title><?php echo $titre; ?></title>

                <!-- style -->
                <?php foreach ($styles as $css) { ?>
                        <link type="text/css" rel="stylesheet" href="<?php echo $css; ?>" />
                <?php } ?>

                <!-- scripts -->
                <script src="https://ajax.googleapis.com/ajax/libs/jquery/1.12.0/jquery.min.js"></script>
                <?php foreach ($scripts as $js) { ?>
                        <script type="text/javascript" src="<?php echo $js; ?>"></script>
                <?php } ?>
        </head>

        <body>
                <section class="wrapper">
                        <?php
                }

                public static function DebutHeader($class = null)
                {
                        echo '<header' . self::GetAttrStr('class', $class) . '>';
                }
                public static function FinHeader($clear = false)
                {
                        echo '</header>' . ($clear ? '<br clear="all" />' : '') .'';
                }
                public static function DebutSection($class = null)
                {
                        echo '<section' . self::GetAttrStr('class', $class) . '>';
                }
                public static function FinSection($clear = false)
                {
                        echo '</section>' . ($clear ? '<br clear="all" />' : '') .'';
                }
                public static function DebutFooter($class = null)
                {
                        echo '<footer' . self::GetAttrStr('class', $class) . '>';
                }
                public static function FinFooter($clear = false)
                {
                        echo '</footer>' . ($clear ? '<br clear="all" />' : '') .'';
                }

                public static function DebutArticle($class = null)
                {
                        echo '<article' . self::GetAttrStr('class', $class) . '>';
                }
                public static function FinArticle($clear = false)
                {
                        echo '</article>' . ($clear ? '<br clear="all" />' : '') .'';
                }
                public static function DebutNav($class = null)
                {
                        echo '<nav' . self::GetAttrStr('class', $class) . '>';
                }
                public static function FinNav($clear = false)
                {
                        echo '</nav>' . ($clear ? '<br clear="all" />' : '') .'';
```

```
                }
                public static function DebutDiv($class = null)
                {
                        echo '<div' . self::GetAttrStr('class', $class) . '>';
                }
                public static function FinDiv($clear = false)
                {
                        echo '</div>' . ($clear ? '<br clear="all" />' : '') .'';
                }
                public static function Titre($size, $titre, $class = null)
                {
                        echo '<h' . $size . self::GetAttrStr('class', $class) . '>' . $titre . '</h' . $size . '>';
                }

                public static function FinHtml()
                {
                        ?>
                        </section>
        </body>
</html>
                        <?php
                }

                // fonctions utilitaires
                private static function GetAttrStr($attrName, $attr)
                {
                        return is_null($attr) ? '' : ' ' . $attrName . '="' . $attr . '"';
                }

        }

?>
```

Voici un exemple de vue utilisant la classe HtmlStruct (la vue de la page d'accueil) :

```
<?php
        include_once (dirname(__FILE__) . '/../librairies/html/HtmlStruct.class.php');

        // donnees de la vue
        // menu critere
        $dVueMenuCritere['lister_recette_utilisateur'] = FALSE;
        $dVueMenuCritere['categories'] = $donneesModele['categories'];
        // login
        $dVueLogin['identifie'] = $donneesModele['identification']['identifie'];
        if ($dVueLogin['identifie']) $dVueLogin['utilisateur'] = $donneesModele['identification']['utilisateur'];
        $dVueLogin['lister_recette_utilisateur'] = FALSE;
        $dVueLogin['afficher_compte'] = FALSE;
        $dVueLogin['page'] = $donneesControleur['page'];
        // random recette
        $dVueRandomRecette['recette_random'] = $donneesModele['recette_random'];
        // message
        $dVueMessage['destruction_recette'] = $donneesModele['detruire_recette'];
        if ($donneesModele['detruire_recette']) $dVueMessage['destruction_recette_succes'] =
$donneesModele['detruire_recette_succes'];
        $dVueMessage['destruction_compte'] = $donneesModele['detruire_compte'];
        if ($donneesModele['detruire_compte']) $dVueMessage['destruction_compte_succes'] =
$donneesModele['detruire_compte_succes'];

        // html
        HtmlStruct::DebutHtml('Accueil', array('css/styles.css'), array('js/script.js'));


                // header
                HtmlStruct::DebutHeader();
                        // logo
                        HtmlStruct::DebutNav('section_logo');
                                include (dirname(__FILE__) . '/commun/Logo.inc.php');
                        HtmlStruct::FinNav();

                        // en tete
                        HtmlStruct::DebutSection('section_header');
                                // titre
                                HtmlStruct::Titre(1, 'ACCUEIL', 'accueil_titre');

                                // menu
                                HtmlStruct::DebutNav('menu_critere');
                                        include (dirname(__FILE__) . '/commun/MenuCritere.inc.php');
                                HtmlStruct::FinNav();
                        HtmlStruct::FinSection(true);
                HtmlStruct::FinHeader();


                // section
                HtmlStruct::DebutSection();
                        // login
                        HtmlStruct::DebutSection('section_login');
                                include (dirname(__FILE__) . '/commun/Login.inc.php');
                        HtmlStruct::FinSection();

                        // section
                        HtmlStruct::DebutSection('section_principale');
                                // message de destruction de recette
                                if ($dVueMessage['destruction_recette'])
                                {
                                        HtmlStruct::DebutArticle('section_message');
                                                include (dirname(__FILE__) .
'/commun/MessageDestructionRecette.inc.php');
                                        HtmlStruct::FinArticle();
                                }
```

```
                                        // message de destruction de compte
                                        if ($dVueMessage['destruction_compte'])
                                        {
                                                HtmlStruct::DebutArticle('section_message');
                                                        include (dirname(__FILE__) . '/commun/MessageDestructionCompte.inc.php');
                                                HtmlStruct::FinArticle();
                                        }

                                        // recette au hasard
                                        HtmlStruct::DebutArticle('random_recette');
                                                include (dirname(__FILE__) . '/commun/RandomRecette.inc.php');
                                        HtmlStruct::FinArticle();
                                HtmlStruct::FinSection(true);
                        HtmlStruct::FinSection();


                        // footer
                        HtmlStruct::DebutFooter();
                                // lien recette
                                HtmlStruct::DebutNav('footer_nav');
                                        include (dirname(__FILE__) . '/commun/LienRandomRecette.inc.php');
                                HtmlStruct::FinNav(true);
                        HtmlStruct::FinFooter();


                // fin html
                HtmlStruct::FinHtml();
```

Comme dans toutes les vues, on peut voir qu'il y a deux parties.
Il y a d'abord une création des données d'affichages (les variables commençant par « dVue ») à partir des données du modèle ($donneesModele) et parfois des données du contrôleurs ($donneesControleur).
Ensuite, la structure de la page est générée grâce à la classe HtmlStruct. Dans les blocs sont ensuite inclus les parties visibles de la page (avec la directive php « include »). On peut voir par exemple que le fichier « commun/Login.inc.php » est inclus dans le bloc « section_login ».

Voici le code de « commun/Login.inc.php » :

```
<?php if ($dVueLogin['identifie']) { ?>

<!-- titre -->
<h3>Bienvenue<br /><?php if ($dVueLogin['utilisateur']['admin'] == 1) echo 'Ô '; ?><span><?php echo $dVueLogin['utilisateur']
['prenom'].' '.$dVueLogin['utilisateur']['nom']; ?></span></h3>

<!-- liens utilisateurs -->
<!-- deconnexion -->
<form method="post" action="" name="deconnexion">
          <?php if ($dVueLogin['lister_recette_utilisateur']) { ?>
          <input type="hidden" name="page" id="page" value="lister_recette" />
          <?php } elseif ($dVueLogin['afficher_compte']) { ?>
          <input type="hidden" name="page" id="page" value="accueil" />
          <?php } else { ?>
          <input type="hidden" name="page" id="page" value="<?php echo $dVueLogin['page']; ?>" />
          <?php } ?>
          <input type="hidden" name="deconnexion" id="deconnexion" value="true" />
          <a href="javascript:document.deconnexion.submit()">Déconnexion</a>
</form>

<!-- compte -->
<form method="post" action="" name="afficher_compte">
          <input type="hidden" name="page" id="page" value="afficher_compte" />
          <a href="javascript:document.afficher_compte.submit()">Mon compte</a>
</form>

<!-- creer une recette -->
<form method="post" action="" name="creer_recette">
          <input type="hidden" name="page" id="page" value="creer_recette" />
          <a href="javascript:document.creer_recette.submit()">Créer une recette</a>
</form>

<!-- lister les recettes -->
<form method="post" action="" name="lister_recette_utilisateur">
          <input type="hidden" name="page" id="page" value="lister_recette_utilisateur" />
          <a href="javascript:document.lister_recette_utilisateur.submit()">Lister mes recettes</a>
</form>

<!-- zone administrateur -->
<?php if ($dVueLogin['utilisateur']['admin'] != 0) { ?>
<!-- Lister les membres -->
<form method="post" action="" name="lister_utilisateurs">
          <input type="hidden" name="page" id="page" value="lister_utilisateurs" />
          <a href="javascript:document.lister_utilisateurs.submit()">Lister les membres</a>
</form>

<!-- gerer les categories -->
<form method="post" action="" name="gerer_categorie">
          <input type="hidden" name="page" id="page" value="gerer_categorie" />
          <a href="javascript:document.gerer_categorie.submit()">Gérer les catégories</a>
</form>
<?php } ?>

<?php } else { ?>
```

```php
<!-- menu de login -->
<form method="post" action="" name="identifier_login">
          <input type="hidden" name="page" id="page" value="<?php echo $dVueLogin['page']; ?>" />
          <label for="login">Identifiant  </label><input type="text" name="login" id="login" /><br />
          <label for="mdp">Mot de passe  </label><input type="password" name="mdp" id="mdp" /><br />
          <a href="javascript:document.identifier_login.submit()">Identifiez vous</a>
</form>

<!-- creer compte -->
<form method="post" action="" name="creer_compte">
          <input type="hidden" name="page" id="page" value="creer_compte" />
          <a href="javascript:document.creer_compte.submit();">Créer un compte</a>
</form>

<?php } ?>
```

On peut voir dans cet exemple que en fonction des valeurs contenus dans « dVueLogin » la vue du panneau d'identification est différente si l'utilisateur est identifié ou pas.

3 – Dernières précisions

La session :

Les données de session sont utilisées pour enregistrer l'id de l'utilisateur lorsque celui-ci s'identifie la première fois. Les données de création de recettes ou de comptes entré par l'utilisateur sont aussi enregistrés en session avant la vérification de validité. Ainsi, si les données ne sont pas valides (trop de caractères dans le prénom par exemple), alors les données seront remises dans les champs de texte et l'utilisateur n'aura pas à tout remplir à nouveau mais seulement modifier les valeurs non valides. Enfin, lorsque l'utilisateur parcoure les recettes d'une catégorie, seul les dix premières recettes de la liste sont affichées, les autres sont accessibles par des boutons « suivant » et « précédent » permettant de naviguer dans la liste des recettes. La position de navigation est enregistré en session pour pouvoir se rappeler de l'état de navigation.

Les images :

L'affichage d'une photo d'utilisateur ou de recette se gère de la même façon que la construction d'une page. Donc l'affichage d'une image comporte un contrôleur accessibles à partir de index.php comme pour les pages du site.

Voici un exemple de code HTML de l'affichage d'une image :

```php
<img alt="" src="?page=image&image=<?php echo $dVueAfficherRecette['recette']['photo']; ?>" />
```

Le contrôleur :

```php
<?php
          // constroleur
          if (isset($_REQUEST['image']) && !is_null($_REQUEST['image']))
                    $donneesControleur['image'] = $_REQUEST['image'];
          else
                    $donneesControleur['image'] = '';


          // modele
          if (!empty($donneesControleur['image']))
          {
                    $donneesModele['image_name'] = 'upload/image/' . $donneesControleur['image'];
                    $finfo = new finfo(FILEINFO_MIME_TYPE);
                    $ftype = $finfo->file($donneesModele['image_name']);
                    $eltInfo = explode('/', $ftype);
                    if (isset($eltInfo[1])) $donneesModele['image_type'] = $eltInfo[1];
                    $donneesModele['image_content'] = file_get_contents($donneesModele['image_name']);
          }


          // vue
          if (isset($donneesModele))
          {
                    if (isset($donneesModele['image_type']))
                    {
```

```
                        // header
                        header('Content-type:image/' . $donneesModele['image_type']);

                        // affichage de l'image
                        echo $donneesModele['image_content'];
                }
        }
```

On peut voir que le modèle et la vue sont codé directement dans le contrôleur car ils sont suffisamment simple.

Les exceptions :

Lorsqu'une erreur survient, une exception dérivé de AppException est levée et est attrapée dans le « catch » à la fin du fichier index.php. A noter qu'une exception de type PDOException (exception lié à la base de donnée) est attrapée une première fois puis transformée en AppException avant d'être levée à nouveau.

Voici la classe AppException :

```
class AppException extends Exception
        {
                // constructeur
                public function __construct($message)
                {
                        parent::__construct($message);
                }

                // message
                protected static function CreateMessage($page, $type, $mess, $lien)
                {
                        $message = '<h3>' . $type . '</h3>'."\n";
                        $message .= '<p>' . $mess . '</p>'."\n";
                        $message .= '<form method="post" action="" name="retour_' . $page . '">'."\n";
                        $message .= "\t".'<input type="hidden" name="page" id="page" value="' . $page . '" />'."\n";
                        $message .= "\t".'<a href="javascript:document.retour_' . $page . '.submit();">' . $lien . '</a>'."\n";
                        $message .= '</form>';
                        return $message;
                }
        }
```

Le message de l'exception est ensuite présenté sous forme d'une page HTML construite sur le modèle MVC comme les autres pages.