Architecture applicative de l'application Web

Remarques préalables

Nous ne présentons pas ici les avantages de la structuration du code relevant de l'architecture Modèle-Vue-Contrôleur ; de nombreux documents se penchent sur la question.
Des frameworks (Zend, Symphony, PhpCake) fournissent les classes mettant en œuvre cette technologie.

Un développement guidé par les cas d'utilisation

C'est le propre de l'architecture MVC ; le système (l'application) doit répondre aux sollicitations de l'utilisateur. Les cas d'utilisation sont les moyens textuels de décrire ces sollicitations et les réponses.

Prenons l'exemple du cas d'utilisation suivant :

PROJET : Application web de gestion des frais	**Description cas d'utilisation**

Nom cas d'utilisation : Renseigner fiche de frais

Acteur déclencheur : Visiteur médical

Pré conditions : Visiteur médical authentifié

Post conditions : néant

Scénario nominal :

1. ***L'utilisateur demande à saisir un ou plusieurs frais pour le mois courant***.
2. Le système retourne les frais actuellement saisis – éléments forfaitisés et hors forfait - pour le mois courant.
3. ***L'utilisateur modifie une ou des valeurs des frais au forfait et demande la validation***.
4. Le système enregistre cette ou ces modifications et retourne ces valeurs à jour.
5. ***L'utilisateur ajoute un nouveau frais hors forfait en renseignant les différents champs – date d'engagement, libellé, montant - et valide.***
6. Le système enregistre la ligne de frais hors forfait.

Exceptions :

2.a- C'est la première saisie pour le mois courant. Si ce n'est pas encore fait, le système clôt la fiche du mois précédent et crée une nouvelle fiche de frais avec des valeurs initialisées à 0. Retour à 3.
4.a. Une valeur modifiée n'est pas numérique : le système indique 'Valeur numérique attendue '. Retour à 3.
6.a Un des champs n'est pas renseigné : le système indique : 'Le champ date (ou libellé ou montant) doit être renseigné'.
6.b La date d'engagement des frais hors forfait est invalide : le système indique 'La date d'engagement doit être valide'. Retour à 5.
6.c La date d'engagement des frais hors forfait date de plus d'un an. Le système indique 'La date d'engagement doit se situer dans l'année écoulée'. Retour à 5.
7***. L'utilisateur sélectionne un frais hors forfait pour suppression***.
8. Le système enregistre cette suppression après une demande de confirmation.

Contraintes :

L'utilisateur sollicite à 4 reprises le système (points 1, 3, 5 et 7 en ***italique gras***). Le contrôleur (fichier spécifique) doit donc répondre à ces 4 sollicitations :

```php
$action = $_REQUEST['action'];
switch($action){
    case 'saisirFrais':{
    case 'validerMajFraisForfait':{
    case 'validerCreationFrais':{
    case 'supprimerFrais':{
}
```

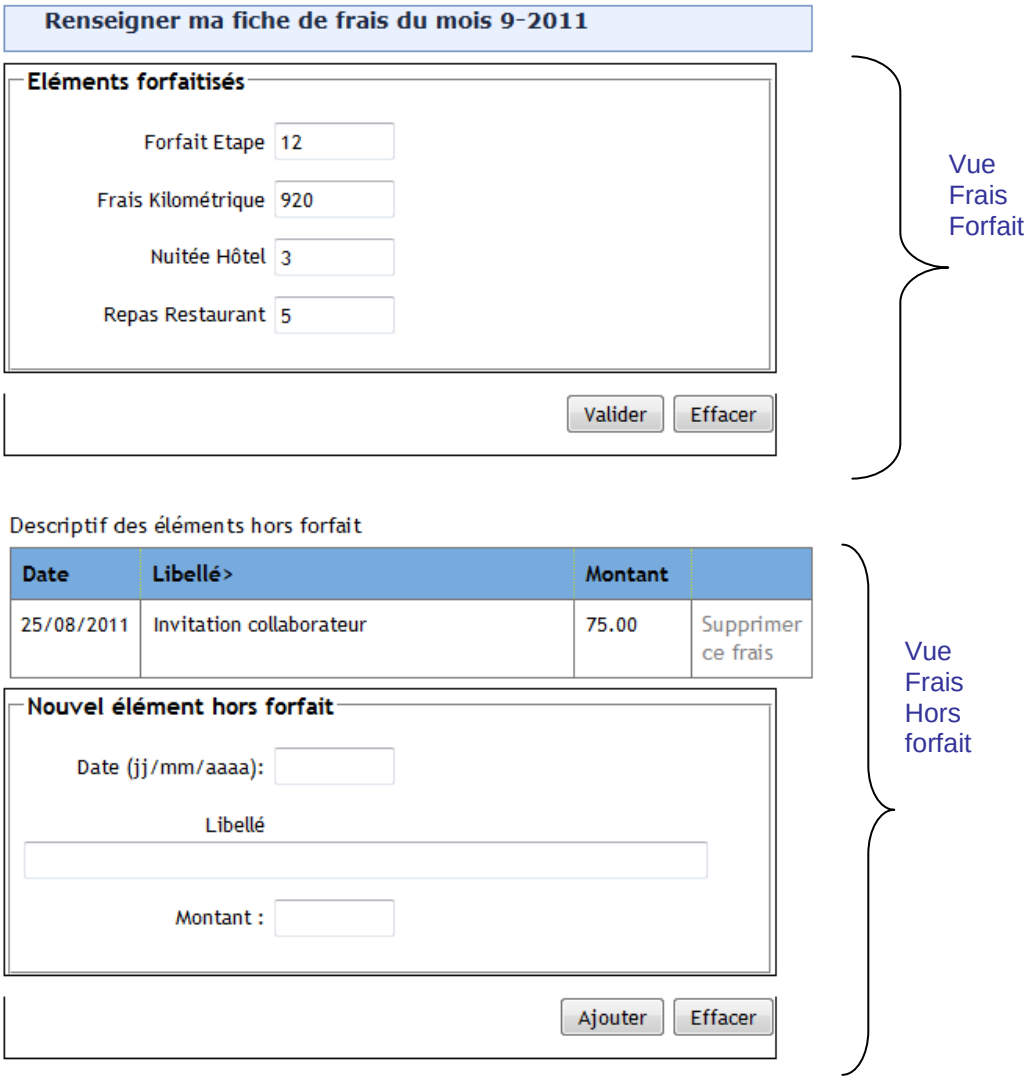
Remarque : le code des *cases* a été plié ici pour se concentrer sur l'essentiel.

Pour chacune des sollicitations, le système réagit et agit en conséquence, par exemple pour la demande de saisie des frais :

```php
7   $action = $_REQUEST['action'];
8   switch($action){
9       case 'saisirFrais':{
10          if($pdo->estPremierFraisMois($idVisiteur,$mois)){
11              $pdo->creeNouvellesLignesFrais($idVisiteur,$mois);
12          }
13          break;
14      }
15      case 'validerMajFraisForfait':{
26      case 'validerCreationFrais':{
39      case 'supprimerFrais':{
44  }
45  $lesFraisHorsForfait = $pdo->getLesFraisHorsForfait($idVisiteur,$mois);
46  $lesFraisForfait= $pdo->getLesFraisForfait($idVisiteur,$mois);
47  include("vues/v_listeFraisForfait.php");
48  include("vues/v_listeFraisHorsForfait.php");
49
50  ?>
```

Le système teste (ligne 10) si c'est la première fois que l'utilisateur accède à cette demande de saisie de frais –cf extension 2.a- et va chercher en base (lignes 45-46) les données concernant les frais forfaitisés et non forfaitisés afin d'afficher les deux vues demandées (lignes 47-48). Ici, ces affichages sont communs aux autres *cases*.

Les deux vues affichées sont ici :



Nous avons fait le choix de présenter deux vues distinctes –nous aurions pu bien sûr mettre ce code dans un seul fichier- pour éventuellement réutiliser une de ces vue dans un autre cas d'utilisation.

Dans cette architecture, l'affichage des vues est provoqué par un ordre **include** (ou require) *nomVue*.

Pour respecter l'indépendance des couches (vue, modèle), le modèle (fichier php) retourne des tableaux :

```
public function getLesFraisForfait($idVisiteur,$mois)
{
    $req = "select fraisforfait.id as idfrais, fraisforfait.libelle as libelle,
    lignefraisforfait.quantite as quantite from lignefraisforfait,
    fraisforfait where fraisforfait.id = lignefraisforfait.idfraisforfait
    and lignefraisforfait.idvisiteur ='$idVisiteur' and lignefraisforfait.mois='$mois'
    order by lignefraisforfait.idfraisforfait";
    $res = PdoGsb::$monPdo->query($req);
    $lesLignes = $res->fetchAll();
    return $lesLignes;
}
```

La vue construit le code HTML à partir du tableau retourné :

```
<fieldset>
  <legend>Eléments forfaitisés
  </legend>
  <?php
      foreach ($lesFraisForfait as $unFrais)
      {
          $idFrais = $unFrais['idfrais'];
          $libelle = $unFrais['libelle'];
          $quantite = $unFrais['quantite'];
  ?>
          <p>
              <label for="idFrais"><?php echo $libelle ?></label>
              <input type="text" id="idFrais" name="lesFrais[<?php echo $idFrais?>]" size="10" maxlength="5" value="<?php echo $quantite?>'
          </p>

  <?php
      }
```

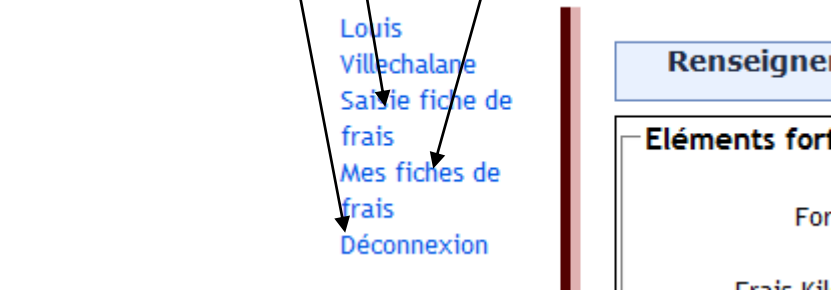
Fonctionnement de l'application

C'est la page index qui sert d'aiguilleur principal et oriente vers un contrôleur de cas d'utilisation : :

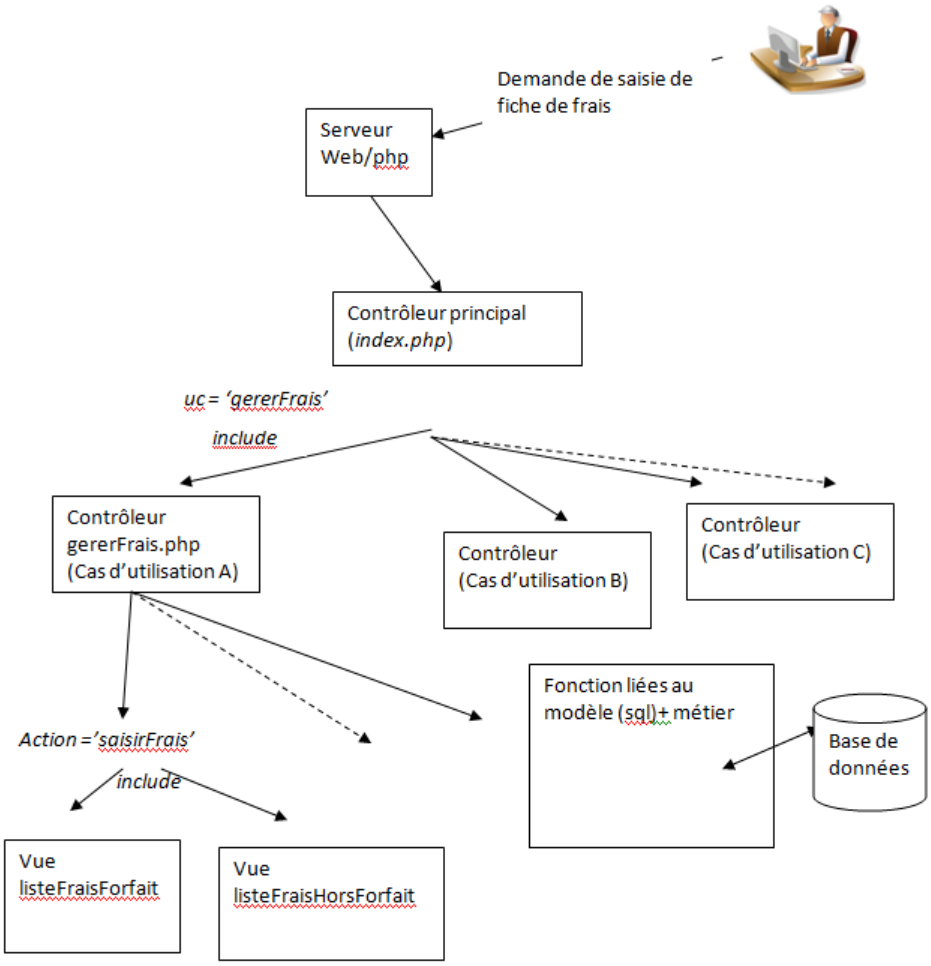
```
require_once("include/fct.inc.php");
require_once ("include/class.pdogsb.inc.php");
include("vues/v_entete.php") ;
session_start();
$pdo = PdoGsb::getPdoGsb();
$estConnecte = estConnecte();
if(!isset($_REQUEST['uc']) || !$estConnecte){
     $_REQUEST['uc'] = 'connexion';
}
$uc = $_REQUEST['uc'];
switch($uc){
    case 'connexion':{
        include("controleurs/c_connexion.php");break;
    }
    case 'gererFrais' :{
        include("controleurs/c_gererFrais.php");break;
    }
    case 'etatFrais' :{
        include("controleurs/c_etatFrais.php");break;
    }
}
include("vues/v_pied.php") ;
?>
```

Ceci est à associé à ce que l'utilisateur sélectionne dans le sommaire :

```
</div>
  <ul id="menuList">
      <li >
          <?php echo $_SESSION['prenom']."  ".$_SESSION['nom']  ?>
      </li>
      <li class="smenu">
        <a href="index.php?uc=gererFrais&action=saisirFrais" title="Saisie fiche de frais ">Saisie fiche de frais</a>
      </li>
      <li class="smenu">
        <a href="index.php?uc=etatFrais&action=selectionnerMois" title="Consultation de mes fiches de frais">Mes fiches de frais</a>
      </li>
 <li class="smenu">
        <a href="index.php?uc=connexion&action=deconnexion" title="Se déconnecter">Déconnexion</a>
      </li>
    </ul>
```

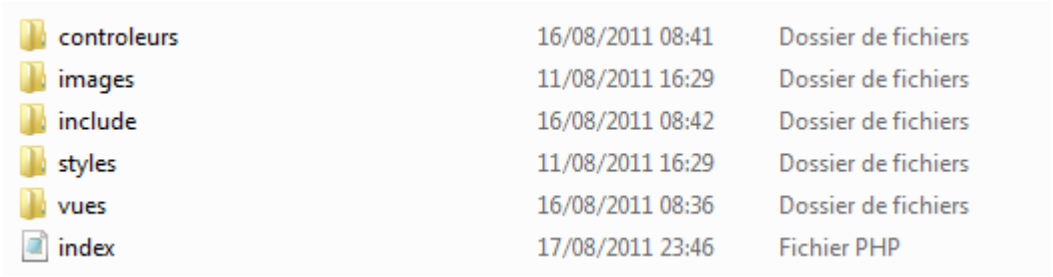


On peut résumer cette cinématique par un schéma :



Ainsi chaque page reçue est construite à partir de l'index comme une succession de fichiers *include* selon le cas d'utilisation. L*'action* demandée entraine un traitement, à partir de la base de données et des règles métier (responsabilité de la couche Modèle) et expose les vues associées.

L'arborescence du site reflète cette architecture :



Le répertoire *include* contient les fichiers utiles au *modèle* : accès à la base, fonctions métier, gestion des erreurs.

Utilisation d'une bibliothèque d'accès aux données

Php contient en standard une classe PDO, d'accès aux données ; elle a l'avantage d'être générique-indépendante du SGBD- et propose des services performants qui évitent la plupart du temps de faire soi-même les boucles de parcours des jeux d'enregistrements :

```
/**
 * Retourne tous les id de la table FraisForfait
 * @return un tableau associatif
*/
    public function getLesIdFrais(){
        $req = "select fraisforfait.id as idfrais from fraisforfait order by fraisforfait.id";
        $res = PdoGsb::$monPdo->query($req);
        $lesLignes = $res->fetchAll();
        return $lesLignes;
    }
/**
```

Cette classe n'est pas associée au modèle MVC, mais elle allège grandement la gestion des données.

Dans l'application, la classe PDO est encapsulée dans une classe PdoGsb