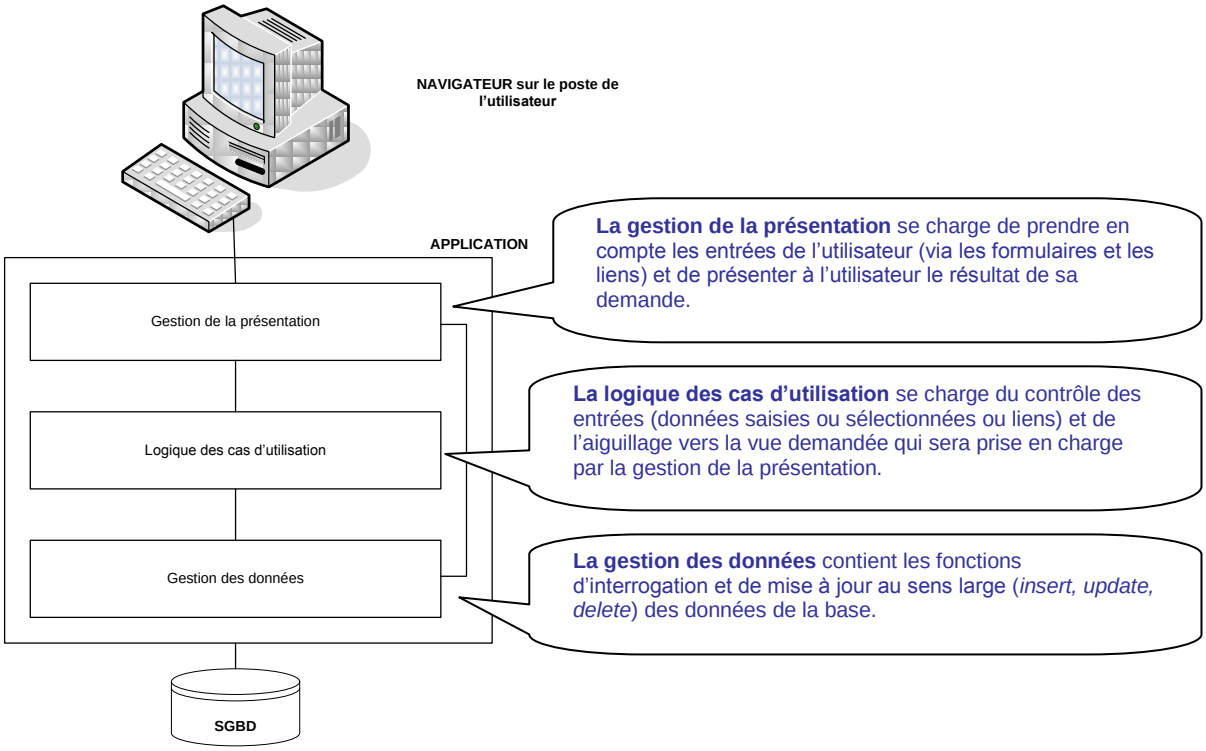
Architecture applicative de l'application festival

Principes d'organisation de l'application PHP festival

Le code PHP a été organisé de façon à respecter les contraintes suivantes :

- Structurer les traitements en s'appuyant sur la description du scénario de chaque cas d'utilisation :
  - un traitement est composé d'étapes, chaque étape correspond à la description d'une étape du scénario typique prise en charge par le système ;
  - la gestion des erreurs décrite dans les cas particuliers est prise en charge grâce à la mise à disposition d'une bibliothèque de fonctions ad hoc.
- Structurer l'application de manière à séparer les responsabilités des différentes couches : traitement (couche contrôleur), génération du code HTML (couche vue), accès aux données (fonctions d'accès à la base de données).
- Réaliser un codage pédagogique (pas de préoccupation d'optimisation du code).
- Utiliser le langage PHP de base sans avoir recours à un quelconque framework tierce.
- Séparer la présentation des informations de leur description : avoir recours à des feuilles de style CSS pour la mise en forme afin que le langage HTML ne soit utilisé que pour décrire les informations.

Présentation générale de l'architecture applicative

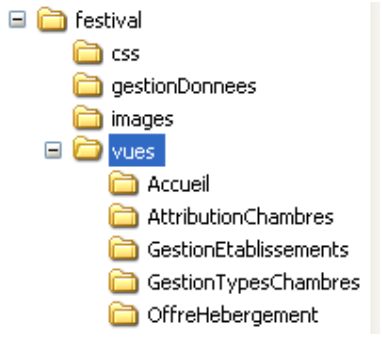


Prise en charge des différentes couches

- La gestion de la présentation est contenue dans des vues ; une **vue** est une page affichée à l'utilisateur. Dans l'application, une vue est obtenue via un fichier PHP préfixé par *v*. Les fichiers « vue » sont stockés dans le répertoire *vues* de l'application.
- La logique des cas d'utilisation est gérée par des contrôleurs ; un **contrôleur** prend en charge la logique d'un cas d'utilisation ou d'une famille de cas d'utilisation. Dans l'application, un contrôleur est implanté dans un fichier PHP préfixé par *c*. Les fichiers « contrôleur » sont stockés à la racine du répertoire d'installation de l'application.
- La gestion des données est prise en charge par des appels de fonctions dans les pages « vues » ou les pages « contrôleurs ». Ces fonctions sont définies dans trois bibliothèques stockées dans le répertoire *gestionDonnees* de l'application.

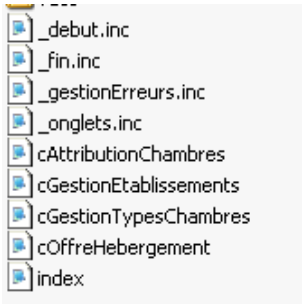
Organisation du stockage des fichiers de l'application

Les fichiers de l'application sont rangés dans différents répertoires



Le répertoire `css` contient les deux feuilles de style : pour les pages et les onglets.
Le répertoire `gestionDonnées` contient les bibliothèques de fonctions d'accès aux données. Il s'agit de fichiers d'inclusion.
Le répertoire `images` contient une image utilisée dans l'entête des tableaux.
Le répertoire `vues` contient cinq sous-répertoires correspondant respectivement à l'accueil et à chacun des quatre cas d'utilisation. Le répertoire Accueil contient la vue présentant l'application et chacun des autres répertoires contient les fichiers « vue » utilisés lors de la réalisation du cas d'utilisation.

À la racine de l'application se trouvent les fichiers « contrôleur » et certains fichiers d'inclusion.



Le fichier index est le fichier de démarrage de l'application.

Les quatre fichiers préfixés `c` sont les contrôleurs responsables de la réalisation des quatre cas d'utilisation.

Les fichiers d'inclusion :
- `_debut.inc` est utilisé dans chaque vue pour construire le titre, le menu de la page ;
- `_fin.inc` est utilisé dans chaque vue pour fermer les balises de la page et afficher les éventuelles erreurs ;
- `_gestionErreurs.inc` est une bibliothèque de fonctions utilisées pour la gestion des erreurs ;
- `_onglets.inc` contient une fonction utilisée pour la construction de chaque onglet.

Conventions de nommage

Item	Règle de nommage[1]
Variables	
tableau associatif	accès par le nom
jeu d'enregistrements	*$rst* suivi du rôle
résultat du jeu d'enregistrements	tableau *$lg* suivi du rôle
chaîne contenant une requête SQL	*$req*
autre variable	pas de règle (excepté bien sûr le respect de la norme « Camel »)
Fonctions	
fonction retournant une requête SQL	*obtenirReq* suivi du rôle (exemple : obtenirReqIdNomEtablissements est le nom de la fonction qui retourne la requête permettant d'obtenir les id et noms des établissements)
fonction retournant une ligne ou une valeur	*obtenir* suivi du rôle (exemple : obtenirDetailEtablissement est le nom de la fonction qui retourne la ligne correspondant à l'établissement demandé)
fonction de suppression ou modification ou création	verbe *créer* ou *modifier* ou *supprimer* suivi du nom de la table. Si la fonction prend en charge plusieurs actions, on accole les noms des actions (exemple : creerModifierEtablissement)
fonction de test	son nom sera formé de *estUn* ou *verifier* ou *existe* suivi du rôle
Fichiers	
contrôleur de cas d'utilisation	*c* suivi du nom du cas d'utilisation et suffixé php (exemple : cGestionEtablissements.php)
vue	*v* suivi du rôle de la vue et préfixé php (exemple : vObtenirDetail.php)
fichier inclus (excepté les vues)	son nom commence par _ et est suffixé .inc.php

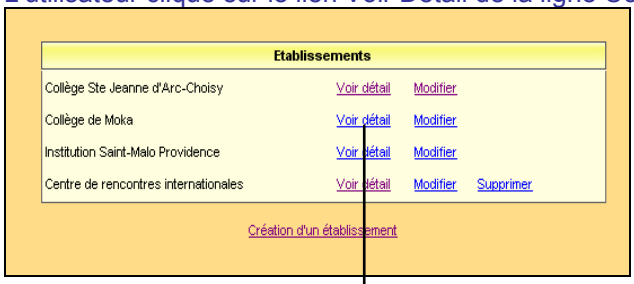
Concernant le schéma de la base de données les règles d'écriture suivantes ont été appliquées :
- pas de blanc ni de caractère accentué dans les noms de table ou d'attribut ;
- chaque nom de table commence par une majuscule et est suivi de minuscules. Si elle est composée de deux mots, ils sont collés 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.

Illustration du schéma d'interaction entre les différentes couches

L'illustration s'appuie sur la situation suivante : l'utilisateur a devant lui la liste des établissements (vue *vObtenirEtablissements.php*) et il demande le détail du collège de Moka (vue *vObtenirDetailEtablissement.php*).

[1] Tous les noms respectent la règle « Camel » qui est notamment utilisée pour la programmation en langage Java.

Couche « vue » : rendu de la page *vObtenirEtablissements.php* :
L'utilisateur clique sur le lien Voir Détail de la ligne Collège Moka :



Le fait de cliquer sur le lien nommé Voir détail permet d'activer ce lien qui appelle le contrôleur avec en paramètres l'étape du scénario et l'identifiant de l'établissement.

http://127.0.0.1/festival/cGestionEtablissements.php?action=detailEtab&id=0350785N

Couche « contrôleur » : extrait du contrôleur cGestionEtablissements.php :

```
// Aiguillage selon l'étape
switch ($action)
{
   case "initial" :
      $vue="vues/GestionEtablissements/vObtenirEtablissements.php";
      break;
   case "detailEtab":
      $id=$_REQUEST["id"];  // On récupère l'identifiant passé en paramètre
      include("vues/GestionEtablissements/vObtenirDetailEtablissement.php");
      break;
   …
}
```

Couche « gestion des données »

Appel d'une fonction d'accès à la base de données.

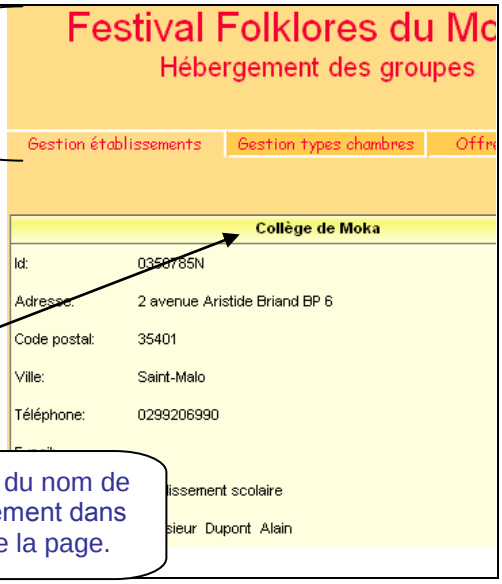
```
function obtenirDetailEtablissement($id)
{
   global $oCnx;
   $sReq="select * from Etablissement
where id='" .$id . "'";
   $rstEtab=mysql_query($sReq,$oCnx);
   return mysql_fetch_array($rstEtab);
}
```

Couche « vue »
Extrait du fichier vObtenirDetailEtablissement.php :

```
<?php
include("_debut.inc.php");  // Affiche le titre
et les onglets

// OBTENIR LE DÉTAIL DE L'ÉTABLISSEMENT
SÉLECTIONNÉ
$lgEtab=obtenirDetailEtablissement($id);
$nom=$lgEtab["nom"];
?>
<br>
<table width="60%" cellspacing="0"
cellpadding="0" class="tabNonQuadrille">

   <tr class="enTeteTabNonQuad">
      <td colspan="3"><strong><?php echo $nom
?></strong></td>
   </tr>
   <tr class="ligneTabNonQuad">
      <td  width="20%"> Id: </td>
      <td><?php echo $id ; ?></td>
   </tr>…
```



Affichage du nom de l'établissement dans l'entête de la page.

L'entête de chaque page est affichée grâce à l'exécution du code contenu dans le fichier `_debut.inc`. Dans ce fichier se trouve la déclaration des deux feuilles de style utilisées `cssGeneral.css` pour les pages et `cssOnglets.css` pour les onglets. Ensuite est inclus le fichier

`_onglets.inc` contenant la fonction de construction d'onglet. Enfin les cinq onglets sont construits : ils correspondent à l'accueil et aux quatre cas d'utilisation et pointent sur les fichiers contrôleur correspondants.

Détail des choix opérés pour l'organisation de l'application

Gestion de la présentation

On isole chaque vue dans un fichier PHP.
Chaque vue est constituée de la même façon :
. un début : titre et menu sous forme d'onglets,
. un corps,
. une fin.

Squelette d'une page Vue

```
<?php
// CODE POUR GENERER LE DEBUT : TITRE ET MENU PRESENTE SOUS FORME D'ONGLETS
// REFERENCE AUX FEUILLES DE STYLE
include("_debut.inc.php");

// OBTENTION DES DONNEES NECESSAIRES A LA CONSTRUCTION DE LA PAGE
$variable=….

// GENERATION DU CODE HTML


// RENVOI VERS LE CONTROLEUR
<a href="cNomControleur.php">Retour</a>

// CODE POUR GENERER LA FIN : FERMETURE DES BALISES HTML ET GESTION DES ERREURS
include("_fin.inc.php");
?>
```

Les pages « vue » sont utilisées soit pour permettre la saisie d'informations par l'utilisateur (gestion des entrées) soit pour afficher des données à l'utilisateur (gestion de l'affichage).

Gestion des entrées

Les entrées sont représentées par des pages HTML construites dynamiquement par le code PHP des pages « vue » comportant pour l'essentiel des formulaires et/ou de simples liens hypertexte.
Les informations portées par les pages HTML sont structurées en HTML relativement dépouillé. Les règles de mise en forme sont placées dans des fichiers CSS (une feuille de styles générale nommée *cssGeneral.css* et une feuille de styles relative aux onglets nommée *cssOnglets.css*).
Les formulaires et liens hypertexte présentés aux utilisateurs pointent vers un fichier php contrôleur servant de point d'entrée de cas d'utilisation de l'application. Ils fournissent donc aux contrôleurs des informations sur le type d'action à réaliser comme « detail… », « demanderModifier… », « demanderCreer… »..., ainsi que d'autres données nécessaires à la réalisation de ces opérations.

Exemple d'entrée de l'utilisateur : il clique sur le lien *Voir Détail* de la ligne Collège de Moka de la vue *vObtenirEtablissements.php*. Cela appelle le contrôleur *cGestionEtablissements.php* en lui transmettant l'action '*detailEtab'* et l'identifiant du collège Moka (voir illustration en page 5).

Gestion de l'affichage

La vue à afficher est déterminée au sein du contrôleur grâce à la variable *$action*.
Contrairement aux documents php contrôleurs de cas d'utilisation, les documents de type « vue » produisent du contenu HTML à l'aide des fonctions **echo** ou **print**.

Par exemple, si l'action choisie par l'utilisateur est d'obtenir le détail d'un établissement, la variable *$action* contiendra '*detailEtab'* et l'action dans le code du contrôleur sera '*detailEtab'* et la vue affichée sera issue de *vObtenirDetailEtablissement.php*.

Logique des cas d'utilisation

On isole chaque famille de cas d'utilisation dans un fichier PHP nommé « contrôleur du cas d'utilisation ». Par exemple, pour la famille de cas d'utilisation « Gestion établissements », le fichier *cGestionEtablissements.php* se charge du contrôle des entrées et de l'aiguillage vers la vue demandée.

Squelette d'un fichier Contrôleur

```
<?php
// INCLUSION DES BIBLIOTHEQUES DE FONCTIONS NECESSAIRES :
// GESTION DES ERREURS ET GESTION DES DONNEES
include("_gestionErreurs.inc.php");
include("gestionDonnees/_connexion.inc.php");
include("gestionDonnees/_gestionBaseFonctionsCommunes.inc.php");

// TRAITEMENT DES DIFFERENTES ETAPES DE DEROULEMENT DU CAS D'UTILISATION

// 1ère étape (donc pas d'action choisie)
if (! isset($_REQUEST["action"]))
{
   $_REQUEST["action"]="initial";
}
// Récupération de l'action demandée par l'utilisateur
$action=$_REQUEST["action"];

// Aiguillage selon l'étape
switch ($action)
{
   case "initial" :
      // Inclusion puis exécution du code de la vue correspondant à l'étape
      // initiale
      include("EmplacementVue\vNomVueInitial.php");
      break;

   case "uneDeuxièmeEtape":
      // Récupération des données nécessaires
      // Puis exécution du code de la vue par inclusion de son fichier
      . . .
      break;
   . . .
  // Définition éventuelle de fonctions utilisées dans le contrôleur
?>
```

Gestion des données

On isole dans un fichier PHP *_connexion.inc.php* la connexion au serveur MySQL et l'ouverture de la base festival.
On déporte la construction des requêtes d'interrogation SQL, la sélection des données (dans certains cas) et la mise à jour au sens large (*delete, update, insert*) de la base dans des fonctions localisées dans deux fichiers :

- le fichier *_fonctionsGestionBaseCommunes.inc.PHP* pour les fonctions utilisées dans tous les cas d'utilisation,
- le fichier *_fonctionsGestionAttributions.inc.php* pour les fonctions spécifiques au cas d'utilisation 'Gérer les attributions'.

Pour être plus précis, les interrogations de la base retournant une seule ligne seront entièrement prises en charge dans une fonction déportée ; cette fonction retourne alors le résultat dans un tableau (si plusieurs colonnes ont été demandées) ou dans une variable élémentaire.
Les interrogations de la base pouvant retourner plus d'un enregistrement sont traitées ainsi :

- constitution de la requête dans une fonction,
- exécution de la requête et traitement du jeu d'enregistrements dans le fichier PHP constituant la vue.

Les fonctions de gestion de données comportent éventuellement des paramètres pour la construction de requêtes dynamiques. En outre si la fonction prend en charge plusieurs actions, par exemple pour les fonctions prenant en charge la création et la modification, on aura un paramètre $mode valant 'C' ou 'M'.

Gestion des erreurs

Par convention dans cette application, lorsqu'une erreur est détectée, un message d'erreur approprié est construit et est fourni au système de gestion des erreurs. Ceci est simplifié par l'utilisation de la fonction *ajouterErreur*.

Dans le contrôleur *cGestionEtablissements.php*, si l'action est '*validerCreerEtab'* ou *'validerModifierEtab'* :

```
switch ($action)
{
   …
   case "validerCreerEtab":case "validerModifierEtab":
      $id=$_REQUEST["id"];
      …
      if ($action == "validerCreerEtab")
      {
         verifierDonneesEtabC($id, $nom, $adresseRue, $codePostal, $ville, $tel,
         $nomResponsable);
         if (nbErreurs()==0)
         {
            creerModifierEtablissement('C', $id, $nom, $adresseRue, $codePostal,
                                       $ville, $tel, $adresseElectronique, $type,
                                       $civiliteResponsable, $nomResponsable,
            include("vues/GestionEtablissements/vObtenirEtablissements.php");
         }
         else
         {
            include("vues/GestionEtablissements/vCreerModifierEtablissement.php");
         }
      …
```

```
function verifierDonneesEtabC($id, $nom, $adresseRue, $codePostal, $ville, $tel,
                              $nomResponsable)
{
   if ($id=="" || $nom=="" || $adresseRue=="" || $codePostal=="" ||
       $ville=="" || $tel=="" || $nomResponsable=="")
   {
```

```
        ajouterErreur("Chaque champ suivi du caractère * est obligatoire");
    }
    if($id!="")
    {
        // Si l'id est constitué d'autres caractères que de lettres non accentuées
        // et de chiffres, une erreur est générée
        if (!estChiffresOuEtLettres($id))
        {
            ajouterErreur
            ("L'identifiant doit comporter uniquement des lettres non accentuées et
            des chiffres");
        }
        else
        {
            if (estUnIdEtablissement($id))
            {
                ajouterErreur("L'établissement $id existe déjà");
            }
        }
    }

    if ($nom!="" && estUnNomEtablissement('C', $id, $nom))
    {
        ajouterErreur("L'établissement $nom existe déjà");
    }
    if ($codePostal!="" && !estUnCp($codePostal))
    {
        ajouterErreur("Le code postal doit comporter 5 chiffres");
    }
}
```

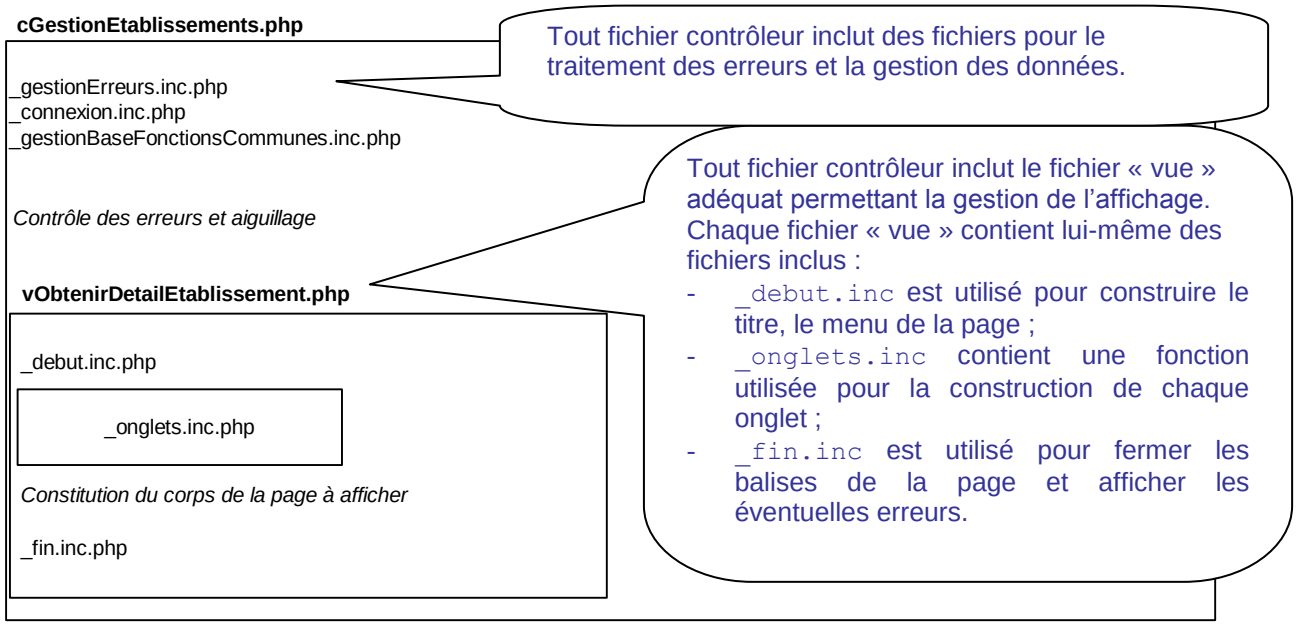
On constate que la fonction de vérification *verifierDonneesEtabC* fait appel à *ajouterErreur* dès qu'une erreur survient et que l'appel à *verifierDonneesEtabC* est suivi par le test du nombre d'erreurs (appel à la fonction *nbErreurs*).

Organisation du code de l'application

Schéma d'imbrication des différents fichiers

Quand l'utilisateur fait appel à un service de l'application, le navigateur envoie une requête au contrôleur du cas d'utilisation. Le serveur d'application charge alors la page PHP du contrôleur.

L'illustration porte sur le fichier contrôleur *cGestionEtablissements.php* et la vue *vObtenirDetailEtablissement.php,* mais la structure est identique pour chaque fichier contrôleur.



Le contrôleur *cGestionEtablissements.php* permet l'exécution du fichier *vObtenirDetailEtablissement.php* et c'est uniquement ce fichier « vue » qui produit l'affichage (avec les fichiers qui sont inclus dans le fichier « vue » tel le fichier *_debut.inc.php*).

Interaction entre les fichiers de l'application

Nom du fichier contrôleur	Chemin des fichiers utilitaires inclus	Chemin des fichiers « vue » inclus	Nom des fichiers utilitaires inclus dans les fichiers « vue »
index.php Page d'accueil de l'application	_gestionErreurs.inc.php	vues/Accueil/vAccueil.php	_debut.inc.php incluant - _onglets.inc.php - La déclaration d'utilisation des feuilles de style css/cssGeneral.css et css/cssOnglets.css. _fin.inc.php
cGestionEtablissements.php Gestion des établissements	_gestionErreurs.inc.php	vues/GestionEtablissements/vObtenirEtablissements.php	
	_gestionDonnees/_connexion.inc.php	vues/GestionEtablissements/vObtenirDetailEtablissement.php	
	_gestionDonnees/_gestionBaseFonctionsCommunes.inc.php	vues/GestionEtablissements/vSupprimerEtablissement.php	
		vues/GestionEtablissements/vCreerModifierEtablissement.php	
cGestionTypesChambres.php Gestion des types de chambres	_gestionErreurs.inc.php	vues/GestionTypesChambres/vObtenirTypesChambres.php	
	_gestionDonnees/_connexion.inc.php	vues/GestionTypesChambres/vSupprimerTypeChambre.php	
	_gestionDonnees/_gestionBaseFonctionsCommunes.inc.php	vues/GestionTypesChambres/vCreerModifierTypeChambre.php	
cOffreHebergement.php Gestion des offres d'hébergement	_gestionErreurs.inc.php	vues/OffreHebergement/vConsulterOffreHebergement.php	
	_gestionDonnees/_connexion.inc.php	vues/OffreHebergement/vModifierOffreHebergement.php	
	_gestionDonnees/_gestionBaseFonctionsCommunes.inc.php		
cAttributionChambres.php Attribution des chambres	_gestionErreurs.inc.php	vues/AttributionChambres/vConsulterAttributionChambres.php	
	_gestionDonnees/_connexion.inc.php	vues/AttributionChambres/vModifierAttributionChambres.php	
	_gestionDonnees/_gestionBaseFonctionsCommunes.inc.php	vues/AttributionChambres/vDonnerNbChambresAttributionChambres.php	
	gestionDonnees/_gestionBaseFonctionsGestionAttributions.inc.php		

Remarque : des fonctions d'accès aux données spécifiques au cas d'utilisation « Attribution des chambres » ont été isolées dans le fichier d'inclusion *_gestionBaseFonctionsGestionAttributions.inc.php*.