Exercice pratique – PHP5 MVC

Création d'une application Web de gestion de cinémas (style Allociné) selon l'architecture MVC

I. MVC

A. Rappels des notions

MVC est un principe d'architecture qui consiste à séparer une application en trois types de composants :

- Les modèles
- Les vues
- Les contrôleurs

1. Les modèles

Ce sont des représentations des objets stockés en Base de Données. Ils contiennent aussi toutes les règles métiers de l'application.

2. Les vues

Elles constituent l'interface de l'application. Ce sont elles qui ordonnent les informations des modèles à des fins de présentation pour l'utilisateur.

3. Les contrôleurs

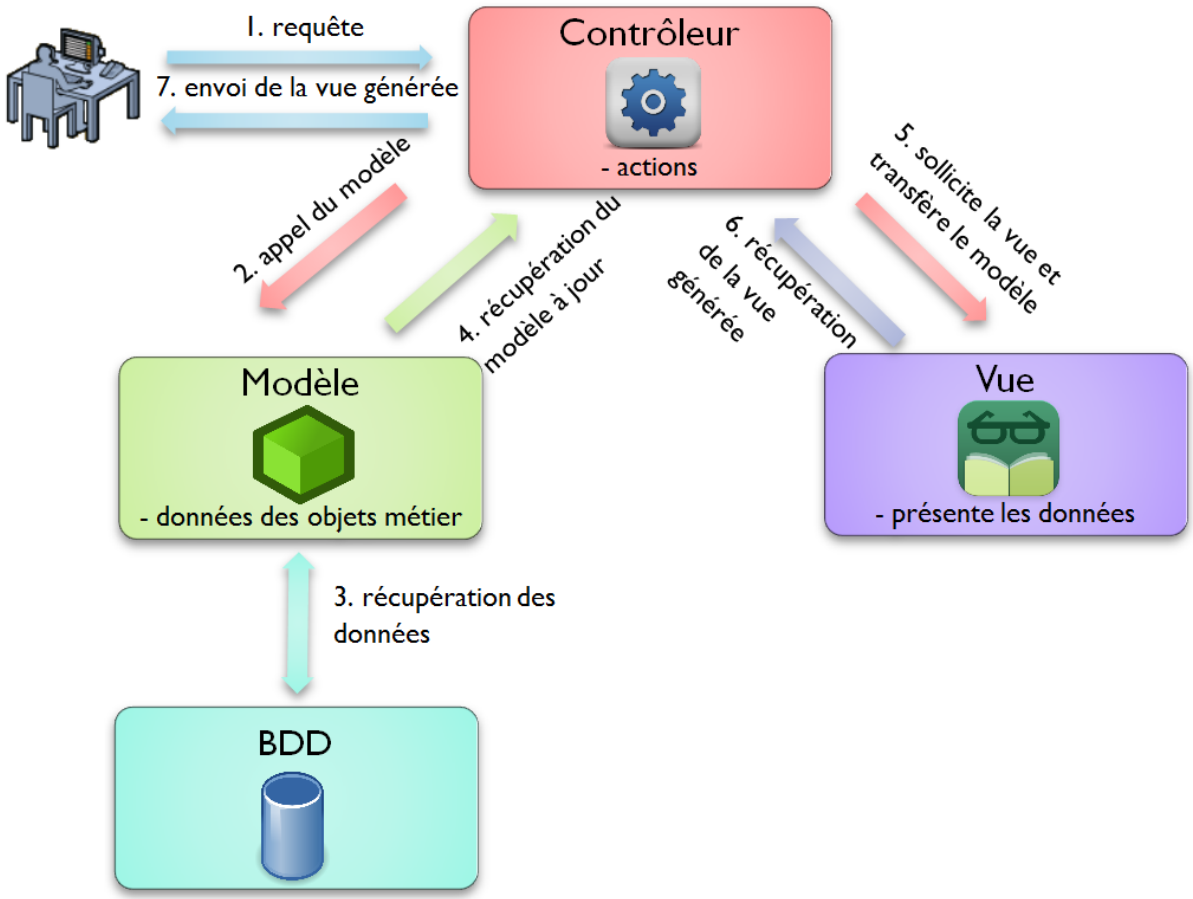
Ce sont les aiguilleurs de l'application. Ils s'occupent des interactions avec l'utilisateur, communiquent avec les modèles et les vues. Ils sont une sorte de passe-plat.

Ainsi, les trois logiques :

- Logique métier
- Logique d'interface
- Logique d'interactions

Sont bien séparées.

B. Schéma du principe Modèle-Vue-Contrôleur



II. Cahier des charges

Le cahier des charges initial reste le même.

A. Rappel architecture applicative

Le client demande deux versions de l'application :

- La première version a été livrée
- **La deuxième version (version 1.0) disposera des mêmes fonctionnalités mais implémentera en plus l'architecture MVC à livrer au client le 9 Décembre 2016 à 16h00**

B. Rappel périmètre des versions

Ci-dessous la description des fonctionnalités à développer en fonction du périmètre de chacune des versions.

1. Version 1.0

a) 1ère et dernière itération

Migration de l'application selon l'architecture MVC.

(1) Espace personnel

- Inscription d'un utilisateur

- Authentification d'un utilisateur (login)
- Logout d'un utilisateur
- Création d'une liste de films préférés pour un utilisateur donné
- Modification (mise à jour / suppression) d'une préférence de film pour un utilisateur donné

(2) Gestion des cinémas

- Consulter la liste des cinémas
- Consulter la liste des films
- Consulter la liste des séances pour un film donné
- Consulter la liste des séances pour un cinéma donné
- Ajouter/Modifier/Supprimer un cinéma
- Ajouter/Modifier/Supprimer un film
- Ajouter/Modifier/Supprimer une séance pour un film et un cinéma donnés

III. Démarche

A. Premier découpage

1. Préparation

Créer deux dossiers à la racine de votre projet :

- views
- models

Le dossier views contiendra les vues de notre projet (où comment présenter nos données).

Le dossier models contiendra les modèles de notre projet (où comment récupérer/stocker nos données).

2. Séparation de l'affichage

a) Création de la vue

Créer un fichier viewHome.php dans le répertoire views.

Couper-coller le code HTML (tout ce qui est contenu entre les balises `<html></html>`) depuis index.php vers viewHome.php.

b) Inclure la vue

Dans index.php, ajouter à la fin du fichier l'instruction PHP `require 'views/viewHome.php'` qui inclut la vue dynamiquement.

3. Séparer le traitement des données

a) Préparatifs

Modifier la visibilité de `$logger` à `protected` dans la classe includes/DBFunctions.php.

Egalement des fonctions `extraireNxN` et `extraire1xN`. La classe `DBFunctions` passe abstraite (elle ne sera jamais instanciée).

b) Création du modèle

Créer la classe models/Utilisateur.php qui **hérite** de DBFunctions et qui servira de modèle (reflet de l'objet métier utilisateur).

Couper-coller les fonctions :

- verifyUserCredentials
- testPasswords
- getUserIDByEmailAddress
- getCompleteUsernameByEmailAddress
- createUser

Depuis DBFunctions.php dans Utilisateur.php.

Pour l'instant, la classe ne contiendra que les méthodes liées à cet objet.

c) Appel du modèle

Dans includes/fctManager.php, ajouter la ligne suivante :

```
$utilisateursMgr = new Utilisateur($logger);
```

d) Finalisation

Dans index.php, des modifications sont à effectuer. Lesquelles ?

4. Résumé

En créant deux fichiers supplémentaires :

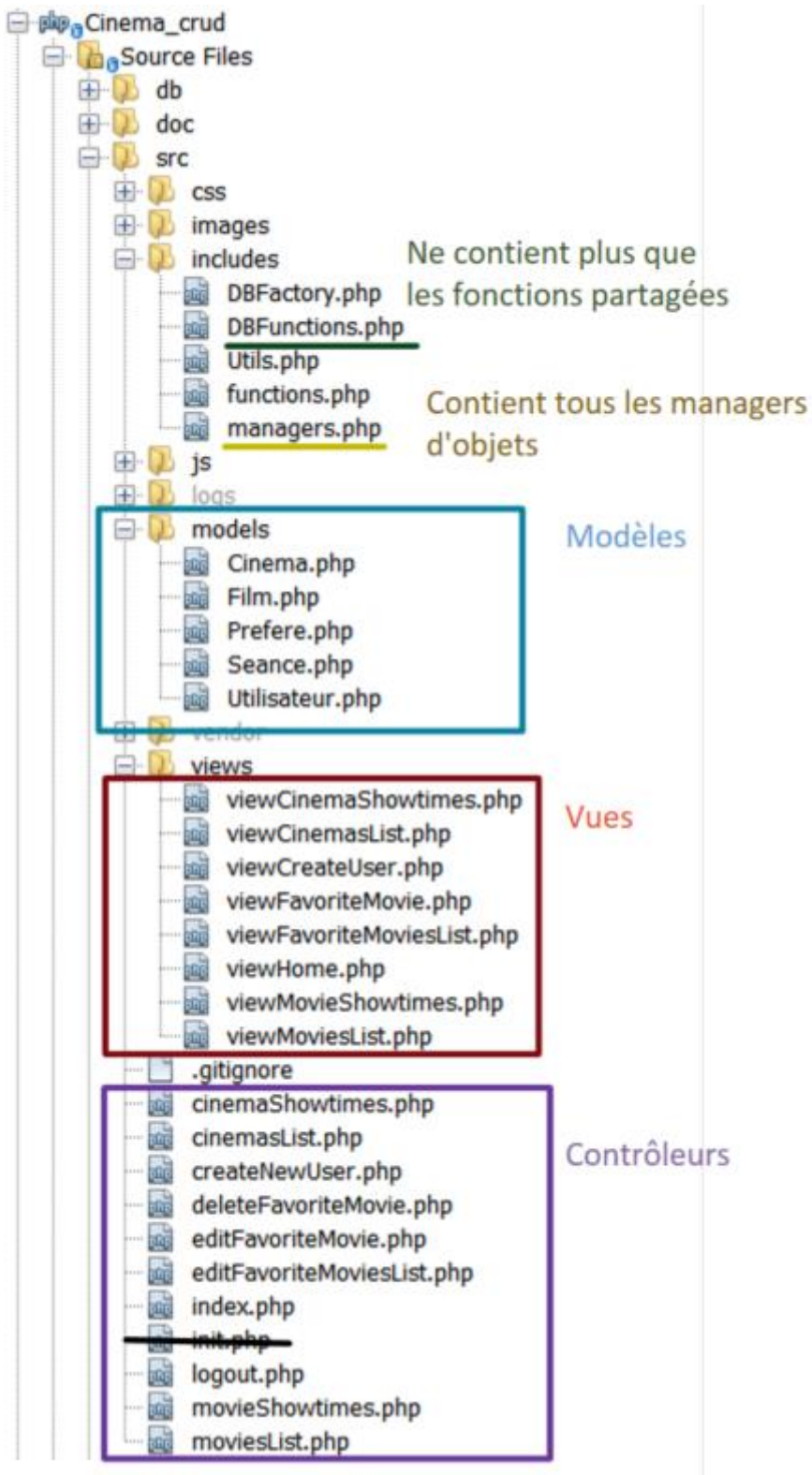
- views/viewHome.php
- models/Utilisateur.php

À partir d'index.php, nous avons séparé l'interface (vue) du métier (modèle) et des interactions (contrôleur). index.php joue le rôle du contrôleur **ultra-simplifié**.

5. Application

Appliquer ce même raisonnement pour toutes les autres pages.

Vous devriez obtenir une arborescence finale conforme à ceci :



Branche Git correspondante : mvc-iteration-01

B. Création d'un véritable contrôleur principal

Nous allons créer un contrôleur appelé contrôleur principal (ou contrôleur frontal), *front controller in English*. Ou plus exactement, utiliser le fichier index.php comme contrôleur frontal.

Ce contrôleur frontal est le point d'entrée unique du site. C'est lui qui réceptionne les requêtes de l'utilisateur et qui lance les traitements adéquats.

Il utilise les services d'autres **contrôleurs** pour réaliser les **actions demandées** et renvoyer les résultats sous forme de **vues**.

1. Déterminer les actions de notre application

La première étape consiste à déterminer quelles sont les actions que l'utilisateur peut réaliser. Commençons par traiter deux actions utilisateurs :

- Affichage de la liste des cinémas
- Affichage de la page d'accueil (action par défaut)

Cela donne :

Action	Traitement afférent
cinemasList	Afficher la liste des cinémas
Ø	Afficher la page d'accueil

Ces actions seront transmises par la méthode GET au fichier index.php.

2. Création de notre premier contrôleur 'serviteur'

a) Préparation

(1) Managers

Afin de faciliter le partage des managers entre les contrôleurs, modifions ainsi le fichier includes/managers.php :

```
$managers = ['utilisateursMgr'=> new Utilisateur($logger),
    'cinemasMgr'=> new Cinema($logger),
    'seancesMgr'=> new Seance($logger),
    'preferesMgr'=> new Prefere($logger),
    'filmsMgr'=> new Film($logger)];
```

b) Contrôleur 'serviteur'

Dans un répertoire /controllers, créer un fichier controleur.php qui contient une fonction par route :

- Fonction home($managers) : affiche la page d'accueil (formulaires d'authentification, boutons, etc.)
- Fonction cinemasList($managers) : affiche la liste des cinémas

(1) Contenu des fonctions

La fonction home($managers) contient tout le code précédemment contenu dans index.php excepté les require du début.

La fonction cinemasList($managers) contient tout le code précédemment contenu dans cinemasList.php excepté les require du début. Ainsi, le fichier cinemasList.php peut être supprimé.

Nous n'incluons pas les require puisqu'ils sont présents ailleurs.

3. Contrôleur frontal (index.php)

Modifions index.php comme ci-dessous :

```
<?php

require_once __DIR__ . '/vendor/autoload.php';

// init. des managers
require_once __DIR__ . './includes/managers.php';

// initialisation de l'application
require_once __DIR__ . './init.php';

// appel au contrôleur serviteur
require __DIR__ . './controllers/controleur.php';

// on "sainifie" les entrées
$sanitizedEntries = filter_input_array(INPUT_GET,
        ['action' => FILTER_SANITIZE_STRING]);
if ($sanitizedEntries && $sanitizedEntries['action'] !== '') {
    // si l'action demandée est la liste des cinémas
    if ($sanitizedEntries['action'] == "cinemasList") {
        // Activation de la route cinemasList
        cinemasList($managers);
    } else {
        // Activation de la route par défaut (page d'accueil)
        home($managers);
    }
} else {
    // Activation de la route par défaut (page d'accueil)
    home($managers);
}
```

Observer que l'appel au contrôleur 'serviteur' se fait bien en fonction de la valeur de l'action demandée (ou route).

4. Modifications restantes

Modification	Avant	Après

Formulaire lié au bouton "Consulter la liste des cinémas" (**viewHome.php**)	`<form name="cinemasList" action="cinemasList.php">`
	`  <input type="submit" value="Consulter la liste des cinémas"/>`
	`</form>`
		`<form name="cinemasList" method="GET" action="index.php">`
		`  <input name="action" type="hidden" value="cinemasList"/>`
		`  <input type="submit" value="Consulter la liste des cinémas"/>`
		`</form>`

Formulaire lié au bouton "Retour à	`<form action="cinemasList.php">` `  <input type="submit" value="Retour à la liste des cinémas"/>`	`<form name="cinemasList" method="GET" action="index.php">`

la liste des cinémas" (**viewCinemaShowtimes.php**)	</form> <input name="action" type="hidden" value="cinemasList"/> <input type="submit" value="Retour à la liste des cinémas"/> </form>

5. Résumé

Nous venons de :

- répertorier une partie des actions utilisateur
- convertir notre index.php en contrôleur frontal
- créer un contrôleur 'serviteur' chargé d'effectuer les traitements en fonction de l'action demandée
- modifier les liens impactés par ce changement d'architecture

6. Application

Effectuer le même travail pour toutes les actions restantes non traitées (création d'un utilisateur, affichage des films, etc.).

Branche Git correspondante : mvc-iteration-02