Département d'Informatique
Université de Fribourg, Suisse
http://diuf.unifr.ch

Aperçu d'un site d'achats de places de cinéma

Un prototype écrit avec PHP/MySQL

Marie Zbinden

No étudiant : 13-201-066

Travail de séminaire en Informatique de Gestion

Encadré par :
Prof. Dr. Jacques Pasquier – Rocha

Fribourg, Mai 2015

Table des matières

Liste des figures

Liste des codes

1

Introduction

De nos jours, l'informatique est omniprésente. En effet, ces dernières années, la croissance a été telle que nos modes de vie, de consommation et de travail se sont vus chamboulés. Dans de nombreux secteurs de l'économie, par exemple pour la gestion des stocks ou des commandes, l'informatisation est devenue un standard.

Dès lors, que cela soit les entreprises ou les prestataires de loisirs, tel que les exploitants de cinéma, soucieux d'être au plus proche besoins de leurs clients sont de plus en plus nombreux à proposer leur service sur internet. C'est pourquoi il me paraît important d'être apte à l'utilisation de certains éléments de programmation pour se lancer dans le monde professionnel actuel. Ainsi, le but de ce travail est de montrer un aperçu de la création d'un site de vente de places de cinéma en ligne, en appliquant et explorant les bases de la gestion d'une base de données et du développement d'un site web dynamique.

Afin d'y procéder, deux technologies largement répandues et libres d'accès sur le web seront utilisées : MySQL et PHP.

Après cette brève introduction, le présent document se poursuit par un chapitre 2, où est présenté de façon générale, ce qu'un véritable site de vente de places de cinéma devrait fournir comme prestations. Puis, le chapitre 3 décrira la modélisation, la structure et la composition de la base de données crée lors de ce travail. Finalement, le dernier chapitre exposera une possibilité d'interface web partielle.

2

Généralités

En réalité, un site de vente de places de cinéma est bien plus complexe qu'on ne le pense. En effet, c'est lors de ce travail que je me suis aperçue de la complexité d'un tel environnement. Un site de ce type ne doit pas que servir à acheter des places mais doit considérer 3 points de vue.

2.1 Les 3 points de vue

Comme énoncé précédemment, un site du type, vente de places de cinéma doit prendre en considération 3 points de vue. Nous allons étudier plus précisément chacun d'entre eux.

2.1.1 Le point de vue du client

Un utilisateur de ce type de site désire consulter les films à l'affiche, éventuellement en voir la bande-annonce et/ou se renseigner sur les acteurs de ce dernier. De plus, il souhaite bien souvent aussi s'informer des séances prévues, de leur date, de leur langue et d'autres caractéristiques que vous pourrez trouver dans le chapitre 3, décrivant la base de données modélisée, au paragraphe décrivant la table *seance*. Une fois son choix fait, le client désirera acheter des places en fonction de la séance choisie. Parfois, une inscription au site ou s'il en est déjà membre, une connexion, lui sera demandée. Souvent, l'utilisateur aura pu sélectionner la séance mais aussi via une représentation graphique de la salle où sera diffusé le film, choisir sa voire ses places désirée(s) en fonction des places disponibles restantes. L'interface devra lui permettre un payement en toute sécurité et lui envoyer un email de confirmation de sa réservation. Finalement, un utilisateur régulier et/ou membre voudra peut-être de temps à autre consulter l'ensemble de ses réservations effectuées. Déjà, de ce point de vue, nombreux paramètres doivent être pris en considération lors de la création du site web. Mais ce n'est pas tout, voyons le point de vue du gérant.

2.1.2 Le point de vue du gérant

Que cela soit du point de vue du gérant d'un multiplexe, regroupant un certain nombre de salles de cinéma ou de l'exploitant d'une salle de l'un de ces complexes, tous deux désirent optimiser leur rendement et rentabiliser leurs infrastructures. Ainsi, depuis leur site, lié à une base de données, décrite dans le chapitre 3, ils doivent par exemple pouvoir consulter le nombre de réservations effectuées durant une année ou encore déterminer quel film a été le plus vu. Dès lors, ces informations qui peuvent très bien se traduire sous forme de graphique, leur serviront à plus au moins long terme dans leur gestion. Il reste un dernier point de vue à considérer, celui de l'opérateur.

2.1.3 Le point de vue de l'opérateur

Dans les tâches d'un opérateur, on trouve par exemple la programmation des séances ou encore la gestion des places disponibles ou non. Par le site du cinéma, ce dernier souhaitera actualiser les films à l'affiche et donc ajouter des séances. L'opérateur devra informer au mieux les utilisateurs de l'interface sur l'emplacement des salles, sur les films actuels et leur permettre de choisir en fonction des disponibilités leurs séances et leurs places.

Après ces quelques lignes, on peut déduire qu'un site de vente de places de cinéma n'est pas une interface des plus simples à construire. En effet, il faut prendre en considérations 3 points de vue et mettre en place de nombreux paramètres. Pour ce faire, il est nécessaire d'avoir de très bonnes connaissances et de l'expérience en création de sites. Il aurait été bien ambitieux de vouloir implémenter un site d'une telle ampleur lors de ce travail de séminaire de bachelor. Cependant, avec les connaissances acquises durant mes deux premiers semestres, je suis apte à vous exposer un aperçu de la création d'un tel site. Ceci fera l'objet des deux prochains chapitres de ce document.

3
La base de données

3.1 La base de données MySQL

Tout site du type vente de places de cinéma, décrit dans le chapitre précédent ne pourrait pas fonctionner de façon optimale sans en arrière-plan une base de données relationnelle complète qui recense l'ensemble des informations utiles. Lors de ce travail, le système de gestion de bases de données relationnelles (SGBDR) choisi est MySQL. Sa licence est libre ou propriétaire selon le type d'usage. En concurrence avec Oracle, Microsoft SQL Server ou encore Microsoft Access 2010 pour ne citer qu'eux, MySQL reste néanmoins un des logiciels de gestion de bases de données les plus répandus au monde. Autant le grand public, pour l'application web principalement, que les professionnels l'utilise.[1] La modélisation, la structure ainsi que le contenu de la base de données crée seront décrits dans les sous-sections suivantes de ce travail.

3.1.1 Le schéma entité-association

Le schéma entité-association est le point de départ de la mise sur pied d'une base de données relationnelle. Celui établi lors de ce travail est représenté par la figure 1. Sa lecture est décrite ainsi : chaque client peut effectuer aucune, une ou plusieurs réservation(s). Chaque réservation se fait pour une seule séance mais peut comporter plusieurs places. Une salle dispose de plusieurs places. Chaque place correspond à une salle. Un multiplexe possède de nombreuses salles, où se déroule aucune, une ou plusieurs séance(s). Un film peut être diffusé lors de plusieurs séances. Un genre peut qualifier aucun, un ou plusieurs film(s). Chaque film peut être décrit par un ou plusieurs genre(s). Finalement, dans un film, il peut y avoir aucun, un ou plusieurs acteur(s) et chaque acteur peut jouer dans un, plusieurs voire aucun film. Ainsi, on dénombre 10 entités qui seront chacune transformées en tables. De plus, par la présence de relations *multiple-multiple*, 3 tables supplémentaires seront établies. La description de ces tables fera l'objet du point suivant de ce travail.

[1] Informations tirées du site internet Wikipédia, http://fr.wikipedia.org/wiki/MySQL (17.04.2015)

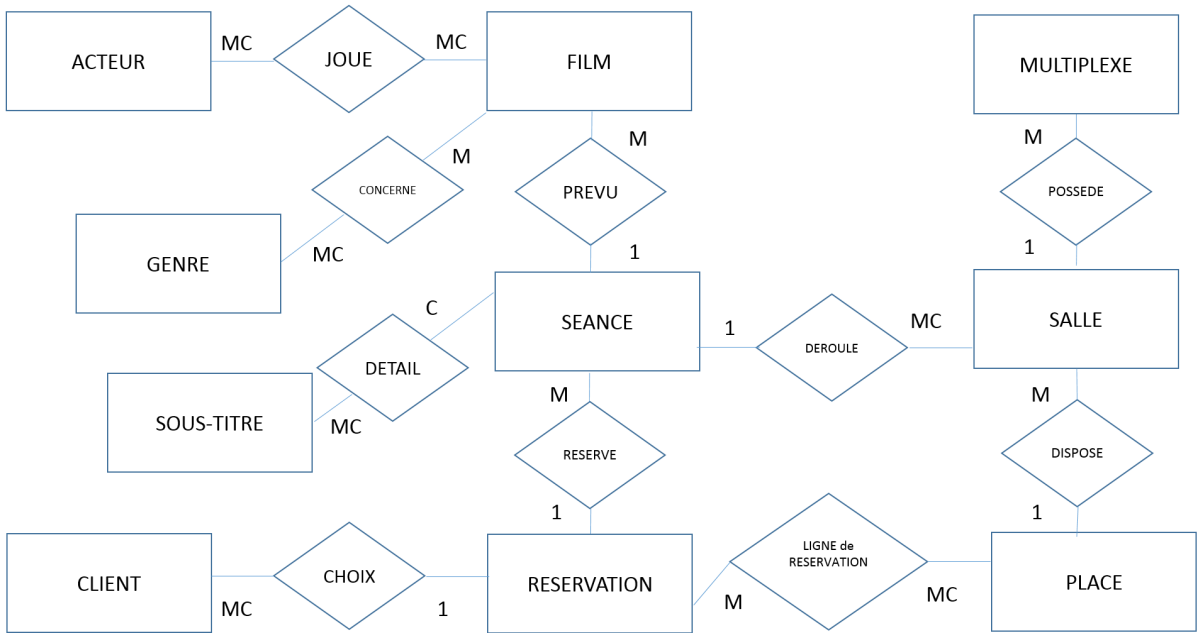

Figure 1 : *Schéma entité-association*

3.1.2 Les tables

Une fois le modèle entité-association établi, une base de données peut être créée. Le nom de la base est *cinema*. Comme précisé auparavant dans ce document, le but de ce travail est de fournir un aperçu d'un site de ventes de places de cinéma et non pas de créer une véritable interface. Dès lors, la base de données implémentée ici est adaptée à ce travail. En effet, chaque table n'est composée que d'un nombre restreint de données. Dans les lignes suivantes, chacune des tables sera décrite et illustrée d'une figure.

La table centrale est *seance*. Cette table regroupe toutes les informations concernant les séances comme la date, la langue de diffusion, la version qui peut être 2D ou 3D, s'il y a un entracte et son coût (14 pour une séance 2D et 16 pour une séance 3D). La clé primaire est *seanceID*. *FilmID*, *salleID* et *sous_titreID* sont des clés étrangères référençant les tables de mêmes noms. A noter que la dernière, *sous-titreID* peut être *NULL*, si la séance n'est pas sous-titrée. Le nombre de séances modélisées suffisant choisi pour ce travail est de onze.

seance

Colonne	Type	Null	Défaut	Relié à	Commentaires
seanceID	tinyint(4)	Non			
date_seance	datetime	Non			
langue	varchar(30)	Non			
version	varchar(2)	Non			
entracte	enum('oui', 'non')	Non			
cout	float	Non			
filmID	varchar(4)	Non			
salleID	varchar(4)	Non		salle -> salleID	
sous_titreID	tinyint(4)	Oui	NULL	sous_titre -> sous_titreID	

Nom de l'index	Type	Unique	Compressé	Colonne	Cardinalité	Interclassement	Null	Commentaire
PRIMARY	BTREE	Oui	Non	seanceID	10	A	Non	
filmID	BTREE	Non	Non	filmID	10	A	Non	
salleID	BTREE	Non	Non	salleID	10	A	Non	
sous_titreID	BTREE	Non	Non	sous_titreID	10	A	Oui	

Figure 2 : *Table seance*

Cité précédemment, *sous_titreID* est la clé primaire de la table *sous_titre* qui nous indique la langue dans laquelle sera sous-titrée la séance si *sous-titreID* n'était pas *NULL* dans la table *seance*. Le nombre de sous-titres modélisés est de 6.

sous_titre

Colonne	Type	Null	Défaut	Commentaires
sous_titreID	tinyint(4)	Non		
langue	varchar(30)	Non		

Nom de l'index	Type	Unique	Compressé	Colonne	Cardinalité	Interclassement	Null	Commentaire
PRIMARY	BTREE	Oui	Non	sous_titreID	5	A	Non	

Figure 3 : *Table sous_titre*

La table *acteur* comprend toutes les informations sur les 21 acteurs référencés. Les noms, les prénoms, les sexes et les nationalités de ces derniers y figurent. La clé primaire est *acteurID*. De plus, l'attribut *url* permet lors de la programmation du site de créer un lien vers une page plus détaillée de l'acteur concerné.

acteur

Colonne	Type	Null	Défaut	Commentaires
acteurID	tinyint(4)	Non		
nom	varchar(30)	Non		
prenom	varchar(30)	Non		
sexe	varchar(1)	Non		
nationalite	varchar(30)	Non		
url	varchar(100)	Non		

Nom de l'index	Type	Unique	Compressé	Colonne	Cardinalité	Interclassement	Null	Commentaire
PRIMARY	BTREE	Oui	Non	acteurID	21	A	Non	

Figure 4 : *Table acteur*

Il en va de même pour la table *film* qui possède aussi un attribut *url*, utilisé de la même façon que décrite précédemment pour la table acteur mais qui renvoie à un site proposant le visionnement d'un extrait du film en question. Les autres attributs décrivent le film en général hormis l'attribut *filmID* qui est la clé primaire. Le nombre de films sélectionnés est de 21.

film

Colonne	Type	Null	Défaut	Commentaires
filmID	tinyint(4)	Non		
intitule	varchar(30)	Non		
realisateur	varchar(30)	Non		
duree	varchar(6)	Non		
annee	year(4)	Non		
url	varchar(100)	Non		

Nom de l'index	Type	Unique	Compressé	Colonne	Cardinalité	Interclassement	Null	Commentaire
PRIMARY	BTREE	Oui	Non	filmID	21	A	Non	

Figure 5 : *Table film*

Les deux précédentes tables décrites ont été créées à partir d'entités. Cependant, sur la figure 1, on peut constater qu'une relation *multiple-multiple* lie l'entité *acteur* à l'entité *film*. Dès lors, une table *joue* a été établie. La clé primaire de cette dernière est formée de deux clés étrangères, issues de la table *acteur* (*acteurID*) et de la table *film* (*filmID*).

joue

Colonne	Type	Null	Défaut	Relié à	Commentaires
acteurID	tinyint(4)	Non		acteur -> acteurID	
filmID	tinyint(4)	Non		film -> filmID	

Nom de l'index	Type	Unique	Compressé	Colonne	Cardinalité	Interclassement	Null	Commentaire
PRIMARY	BTREE	Oui	Non	acteurID	22	A	Non	
				filmID	22	A	Non	
acteurID	BTREE	Non	Non	acteurID	22	A	Non	
filmID	BTREE	Non	Non	filmID	22	A	Non	

Figure 6 : *Table joue*

Comme on peut voir sur la figure 1, on a la même situation que précédemment entre l'entité film et genre. En effet, on trouve une relation *multiple-multiple*. Ainsi, en plus de créer une table *genre* où la clé primaire sera *genreID,* une autre table *concerne,* dont la clé primaire sera formée des deux clés étrangères (*genreID* et *filmID)* sera établie. A noter, que 16 genres de films ont été pris en compte.

genre

Colonne	Type	Null	Défaut	Commentaires
genreID	tinyint(4)	Non		
qualificatif	varchar(30)	Non		

Nom de l'index	Type	Unique	Compressé	Colonne	Cardinalité	Interclassement	Null	Commentaire
PRIMARY	BTREE	Oui	Non	genreID	16	A	Non	

Figure 7 : *Table genre*

concerne

Colonne	Type	Null	Défaut	Relié à	Commentaires
filmID	tinyint(4)	Non		film -> filmID	
genreID	tinyint(4)	Non		genre -> genreID	

Nom de l'index	Type	Unique	Compressé	Colonne	Cardinalité	Interclassement	Null	Commentaire
PRIMARY	BTREE	Oui	Non	filmID	26	A	Non	
				genreID	26	A	Non	
filmID	BTREE	Non	Non	filmID	26	A	Non	
fk_genre	BTREE	Non	Non	genreID	26	A	Non	

Figure 8 *: Table concerne*

Dans la table *salle*, la clé primaire est *salleID*, y figure aussi le nom de la salle ainsi que sa capacité qui pour ce travail sera fixée à 9 afin de faciliter l'implémentation. Dans ce même but, le nombre de salle par multiplexe qui selon sa définition[2] devrait en contenir 8 au minimum, ici ne sera composé que de 3 salles. Enfin, l'attribut *multiplexeID* est une clé étrangère renvoyant à la table *multiplexe* décrit dans le paragraphe suivant.

salle

Colonne	Type	Null	Défaut	Relié à	Commentaires
salleID	varchar(4)	Non			
nom	varchar(30)	Non			
multiplexeID	tinyint(4)	Non		multiplexe -> multiplexeID	
capacite	int(3)	Non			

Nom de l'index	Type	Unique	Compressé	Colonne	Cardinalité	Interclassement	Null	Commentaire
PRIMARY	BTREE	Oui	Non	salleID	12	A	Non	
multiplexeID	BTREE	Non	Non	multiplexeID	12	A	Non	

Figure 9 : *Table salle*

Citée précédemment, la table *multiplexe* nous informe du lieu des salles. En effet, comme chaque salle fait partie d'un multiplexe, le lieu de ce dernier et aussi celui de ses salles. La clé primaire est l'attribut *multiplexeID*. On y trouve aussi le nom du multiplexe et son exploitant. Comme énoncé précédemment, le nombre de places et de salles a été réduit par rapport à la réalité lors dans ce modèle. Dès lors, le nombre de multiplexe modélisé a lui aussi été diminué au nombre de 4.

multiplexe

Colonne	Type	Null	Défaut	Commentaires
multiplexeID	tinyint(4)	Non		
nom	varchar(30)	Non		
lieu	varchar(30)	Non		
exploitant	varchar(30)	Non		

Nom de l'index	Type	Unique	Compressé	Colonne	Cardinalité	Interclassement	Null	Commentaire
PRIMARY	BTREE	Oui	Non	multiplexeID	4	A	Non	

Figure 10 : *Table multiplexe*

La table *place* dont la clé primaire est *placeID*, précise la lettre de la rangée ainsi que le numéro de siège de la place située dans une salle qui est référencée par la clé étrangère *salleID*. Comme une place se trouve dans une salle, il a déjà été mentionné lors de la description de la table *salle* que 9 places sont modélisées par salle.

[2] Définition prise du site Wikipédia, http://fr.wikipedia.org/wiki/Multiplexe, (17.04.2015)

place

Colonne	Type	Null	Défaut	Relié à	Commentaires
placeID	tinyint(4)	Non			
salleID	varchar(4)	Non		salle -> salleID	
lettre_rangee	varchar(1)	Non			
numero_siege	tinyint(3)	Non			

Nom de l'index	Type	Unique	Compressé	Colonne	Cardinalité	Interclassement	Null	Commentaire
PRIMARY	BTREE	Oui	Non	placeID	99	A	Non	
salleID	BTREE	Non	Non	salleID	24	A	Non	

Figure 11 : *Table place*

A nouveau, entre l'entité *place* et l'entité *reservation* représentées sur la figure 1, on constate une relation *multiple-multiple*. Comme vu précédemment, un pareil cas nécessite la création d'une table supplémentaire, ici la table *ligne_de_reservation*. Cette dernière est formée d'une clé primaire composée de deux clés étrangères, *placeID*, référençant la table *place* et *reservationID*, renvoyant à la table *reservation.* De plus, par l'attribut *prix* y figure le montant de la réservation pour une place.

ligne_de_reservation

Colonne	Type	Null	Défaut	Relié à	Commentaires
reservationID	tinyint(4)	Non		reservation -> reservationID	
placeID	tinyint(4)	Non		place -> placeID	
prix	float	Non			

Nom de l'index	Type	Unique	Compressé	Colonne	Cardinalité	Interclassement	Null	Commentaire
PRIMARY	BTREE	Oui	Non	reservationID	29	A	Non	
				placeID	29	A	Non	
placeID	BTREE	Non	Non	placeID	29	A	Non	
reservationID	BTREE	Non	Non	reservationID	29	A	Non	

Figure 12 : *Table ligne_de_reservation*

Mentionné dans le paragraphe précédent, la table *reservation*, dont la clé primaire est *reservationID* , possède deux clés étrangères. La première *seanceID* renvoie à la table *seance* et la seconde *clientID* fait référence à la clé primaire de la table *client*, dernière table présentée juste après la figure 13. Le nombre de réservations insérées est de 15.

reservation

Colonne	Type	Null	Défaut	Relié à	Commentaires
reservationID	tinyint(4)	Non			
seanceID	tinyint(4)	Non		seance -> seanceID	
clientID	tinyint(4)	Non		client -> clientID	

Nom de l'index	Type	Unique	Compressé	Colonne	Cardinalité	Interclassement	Null	Commentaire
PRIMARY	BTREE	Oui	Non	reservationID	15	A	Non	
seanceID	BTREE	Non	Non	seanceID	15	A	Non	
clientID	BTREE	Non	Non	clientID	15	A	Non	

Figure 13 : *Table reservation*

La dernière table modélisée est dénommée *client*. On y trouve l'ensemble des informations sur les acheteurs, comme leur nom, leur prénom, leur adresse, leur email ainsi que leur téléphone qui peut prendre la valeur *NULL*. En effet, lors de son enregistrement, un client n'est pas obligé de le mentionner. De plus, figure aussi le pseudo et le mot de passe, choisis par le client, qui lui permettent de se connecter sur le site et de par exemple consulter ses précédentes réservations ou d'en effectuer de nouvelles. La clé primaire de cette dernière table est *clientID*. Seul 12 clients ont été enregistrés dans cette table.

client

Colonne	Type	Null	Défaut	Commentaires
clientID	tinyint(4)	Non		
nom	varchar(30)	Non		
prenom	varchar(30)	Non		
pseudo	varchar(30)	Non		
adresse	varchar(60)	Non		
email	varchar(60)	Non		
telephone	varchar(13)	Oui	*NULL*	
motdepasse	varchar(30)	Non		

Nom de l'index	Type	Unique	Compressé	Colonne	Cardinalité	Interclassement	Null	Commentaire
PRIMARY	BTREE	Oui	Non	clientID	11	A	Non	

Figure 14 : *Table client*

4
Page web

Nous avons vu dans le chapitre 2 de ce document que la création complète d'un site de ventes de places de cinéma est un travail conséquent. Néanmoins, dans le chapitre précédent, il a été possible, à l'aide de MySQL de donner un aperçu d'une base de données nécessaire à un tel site. Maintenant, il convient de présenter ce à quoi une interface partielle possible pourrait ressembler. C'est ce qui fera l'objet de ce chapitre.

4.1 Présentation

L'interface proposée, représentée par la figure 15 est un modèle simple. On trouve un titre *recherche*, le même que celui de la page, puis un formulaire qui permet à l'utilisateur d'avoir un aperçu rapide et global de ce qu'il recherche. Une première possibilité est d'effectuer une recherche par film en inscrivant un mot clé du titre dans la partie réservée à cela. Une deuxième possibilité est d'effectuer une recherche par genre en choisissant dans le menu déroulant un des genres proposés. Le troisième mode de recherche se fait en sélectionnant l'une des deux villes proposées. Après avoir rempli le ou les champ(s) désiré(s) et cliqué sur le bouton *Go*, une liste, contenant l'intitulé du film, la date et l'heure de la séance de ce dernier, l'année de la sortie du film ainsi que son genre, sa durée, la langue dans laquelle il sera projeté, sa version, le lieu de la salle où il sera diffusé et enfin un lien, *extrait* qui renvoie à un site proposant le visionnement d'une partie du film, apparaîtra en dessous du formulaire comme on peut le voir sur la figure 15. On peut aussi voir sur cette dernière que si aucun résultat n'a été trouvé, le texte *Aucun résultat trouvé !* s'affiche. Mais comment se fait-il que ces données apparaissent ? Le prochain sous-chapitre répondra à cette question.

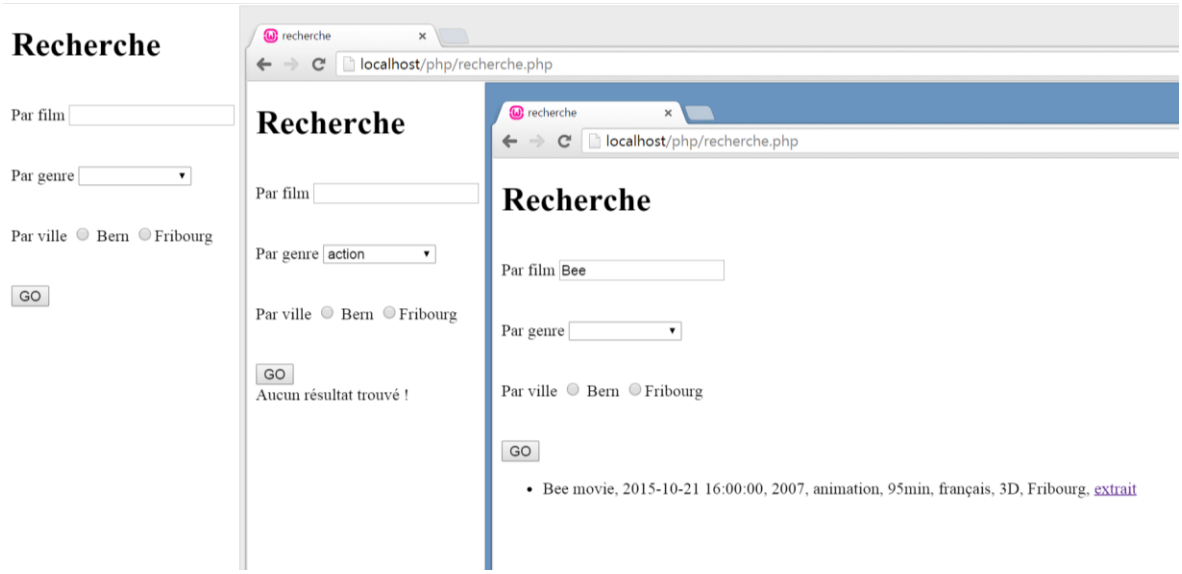

Figure 15: *Page recherche avant/après*

4.2 PHP

Les informations qui s'afficheront sur la page web, ne seront autres que celles contenues dans la base données présentée dans le chapitre 3. En effet, cela a déjà été mentionné, tout site web de ce type doit pouvoir être relié à une base de données. Afin d'établir ce lien, entre le site et la base de données, il existe la technologie PHP (Hypertext preprocessor). Il s'agit d'un langage de scripts libre. Ce langage est du côté serveur, autrement dit, le serveur (la machine hébergeant la page web) va interpréter le code PHP et générer dans la plupart des cas du code HTML afin que le navigateur puisse le lire et le traduire sous forme de pages web comme on les connait.

En suivant le schéma représenté par la figure 16, on peut décrire l'exécution d'un code PHP lors d'une utilisation web. Un visiteur demande à voir une page web(1). Son navigateur envoie une requête au serveur. Si ce dernier identifie la page comme étant un script PHP (fichier se terminant par l'extension .php), l'interprète PHP, appelé par le serveur, traitera et produira le code final de la page(2). Ce code final est renvoyé au serveur qui le transmettra au navigateur du client(3).

[3]

Figure 16 : *Fonctionnement du langage PHP*

Pour revenir à notre page *recherche*, le code PHP, inséré dans le code HTML de la page, sera de cette forme. Les premières lignes permettent de se connecter à la base de données.

```
1     <?php//ouverture code php
2
        $dbpass="MZ*41293";$dbhost="localhost";$dbuser="root";
      $dbname="cinema";
3             //$dbpass="";
4       $link=mysqli_connect($dbhost,$dbuser,$dbpass,$dbname);
5      //connexion à la base de données MySQL
```

Code 1 : *Connexion à MySQL*

Vient ensuite le code servant à formuler la requête SQL, qui nous sert à aller chercher les informations contenues dans la base de données et de les stocker dans la variable *$request*.

[3] Image prise sur le site Openclassrooms, http://openclassrooms.com/courses/concevez-votre-site-web-avec-php-et-mysql/introduction-a-php, (17.04.2015)

```
1     if(isset($_POST['search'])) {
      //On récupère les valeurs via une requête SQL et on stocke
      les données
2
3     $request = "SELECT intitule,duree,annee,date_seance,langue,
      version,entracte,cout,qualificatif,lieu,url
4     FROM film,seance,salle,multiplexe,genre,concerne
5     WHERE film.filmID=seance.filmID                          AND
6     salle.salleID=seance.salleID                              AND
      salle.multiplexeID=multiplexe.multiplexeID                AND
      concerne.filmID=film.filmID                               AND
      concerne.genreID=genre.genreID";
7
8       if(isset ($_POST['motcle']) && $_POST ['motcle'] != ''){
9                       $keyword = strtolower($_POST ['motcle']);
      // texte sera en minuscule
10      $keyword   =   mysqli_real_escape_string($link,   $keyword);
      // évite des injections
11    $request .= " AND LOWER(intitule) LIKE '%" . $keyword . "%'";
12                            }
13      if(isset ($_POST['genre']) && $_POST ['genre'] != ''){
14                      $keyword = strtolower($_POST ['genre']);
      // texte sera en minuscule
15      $keyword   =   mysqli_real_escape_string($link,   $keyword);
      // évite des injections
16    $request .= " AND concerne.genreID=" . $keyword . "";
17                            }
18      if(isset ($_POST['lieu']) && $_POST ['lieu'] != ''){
19                      $keyword = strtolower($_POST ['lieu']); //
      texte sera en minuscule
20      $keyword   =   mysqli_real_escape_string($link,   $keyword);
      // évite des injections
21      $request .= " AND LOWER(lieu) LIKE '%" . $keyword . "%'";
22                            }
```

Code 2 : *Requête et précisions*

Les données seront envoyées au serveur par la fonction *mysqli_query* ou s'il y a une erreur on l'affiche grâce à la fonction *mysqli_error*.

```
1     //echo $request;
2      $result_handle = mysqli_query($link,$request)or die("Aucun
      résultat trouvé!") or die(" la requête est fausse " .
      mysqli_error()) ;
3     //on envoie la requête au serveur et s'il y a une erreur
      affichage
```

Code 3 : *Envoi de la requête au serveur*

Puis, on récupère les données au moyen de la fonction *mysqli_fetch_array* et une boucle *while* qui fera « un tour » pour chaque enregistrement. Finalement, on affiche le résultat par ligne comme on peut le voir sur la figure 15.

```
1       $num_of_rows = mysqli_num_rows ($result_handle);
2
3       if ($num_of_rows == FALSE) {
4                               echo ("Aucun résultat trouvé ! ");
5     //si aucun résultat n'est trouvé
6                               }else{
7                                     echo "<ul>";
8     //sinon on récupère les données et on les affiche sous forme
      de liste
9       while ($row = mysqli_fetch_array($result_handle)){
10      $db_nom= $row ["intitule"];
11      $time= $row ["date_seance"];
12      $annee= $row ["annee"];
13      $genre= $row ["qualificatif"];
14      $duree= $row ["duree"];
15      $langue= $row ["langue"];
16      $version= $row ["version"];
17      $lieu= $row ["lieu"];
18      $url= $row ["url"];

19      echo utf8_encode
20    //permet que les caractères spéciaux s'affichent correctement
21    ("<li>  $db_nom,  $time,  $annee,  $genre,  $duree,  $langue,
      $version, $lieu, <a href='$url'> extrait</a> </li>");
22    //afficher ces attributs de la requête
23                                    }
24                                    echo "</ul>";
25                              }
26                        }//fermeture code php
27                ?>
```

Code 4: *Affichage des données*

5
Conclusion

Le but de ce travail de séminaire était de démontrer que malgré la complexité d'un site de vente de places de cinéma en ligne, il était néanmoins possible d'en donner un petit aperçu grâce aux technologies MySQL et PHP. En effet, ces deux technologies restent à la portée de tout débutant en informatique pour implémenter une petite base de données. Comme de nos jours, dans de nombreux domaines il est requis d'avoir des connaissances au sujet de ces deux technologies, il est pertinent de vouloir dès leur apprentissage mettre en œuvre les connaissances acquises afin de ne pas perdre la main. Ainsi, ce travail a constitué en une application, bien que sommaire, concrète des notions apprises durant les cours d'introduction à l'informatique de gestion des deux premiers semestres de bachelor en gestion d'entreprise à l'université de Fribourg.

A

CD des ressources

- Pour déployer ce site, il faut créer une base de données au moyen de phpMyAdmin, et y ajouter les tables SQL via le fichier *cinema.sql*. Ensuite, le fichier *recherche.php* s'ouvre à l'adresse *localhost/php/recherche.php* lorsqu'il aura été enregistré sous: C:\wamp\www\php\recherche.php.
- Ce site a été réalisé avec Apache 2.4.9 (Win64), PHP 5.5.12 et MySQL 5.6.17.

Bibliographie

[Collaud, Monnard, 2015]

G. Collaud, J. Monnard, *Informatique de gestion II*, Université de Fribourg, 2015.

[Pasquier, Albreshne, 2014]

J. Pasquier, A.Albreshne, *Informatique de gestion I*, Université de Fribourg, 2014.

[Openclasserooms, 2015]

Concevez votre site web avec PHP et MySQL, http://openclassrooms.com/courses/concevez-votre-site-web-avec-php-et-mysql, Dernière visite: 17.04.2015.

[Wikipédia, 2015]

Multiplexe, http://fr.wikipedia.org/wiki/Multiplexe, Dernière visite: 17.04.2015.

[Wikipédia, 2015]

MySQL, http://fr.wikipedia.org/wiki/MySQL, Dernière visite: 17.04.2015.