L3 Miage
Université de Lorraine
13 rue Michel Ney
54037 Nancy Cedex

Département Miage

MusicaNubes

Application web pour ecouter de la musique

Rapport de projet pour le cours de programmation web
2014-2015

Thilbault Geoffroy
Jérôme Rollinger
Ersagun Yalcintepe

L3 Miage
Université de Lorraine
13 rue Michel Ney
54037 Nancy Cedex

Département Miage

MusicaNubes

Application web pour ecouter de la musique

Rapport de projet pour le cours de programmation web
2014-2015

Thilbault Geoffroy
Jérôme Rollinger
Ersagun Yalcintepe

Table des matières

Chapitre 1

Introduction

Dans ce projet nous avons réalisé une application web qui permet de créer un compte d'utilisateur, voir le profil des artistes, écouter des musiques et créer des playlist.

Chapitre 2

Fonctionnalités

2.1 Les fonctionnalités attendues

- Le site permet de rechercher des titres et de consulter les informations sur les artistes.
- Le site permet de construire des playlists et de les écouter.
- Le site permet de créer des comptes utilisateur et de sauvegarder les playlists

2.2 Les fonctionnalités réalisées

- Le site permet de rechercher des titres et de consulter les informations sur les artistes
- Le site permet de créer des comptes utilisateurr

Chapitre 3

Responsable des tâches

3.1 Responsable des tâches

- Thibault Geoffray - Gestion des utilisateurs
- Jérôme Rollinger - Les modèles et la classe base
- Ersagun Yalcintepe - Les modèles adjonctions
- Ersagun Yalcintepe - Gestion des utilisateurs adjonctions
- Ersagun Yalcintepe - Controlleur
- Ersagun Yalcintepe - Création HTML / CSS
- Ersagun Yalcintepe - Affichage des artistes et des chansons
- Ersagun Yalcintepe - Recherche d'une chanson
- Ersagun Yalcintepe - jsPrincipal + hash
- Ersagun Yalcintepe - Choix et la mise en place de l'architecture
- Ersagun Yalcintepe - Rédaction du rapport
- Ersagun Yalcintepe - Mettre en place le dépôt GIT
- Ersagun Yalcintepe - Commenter tous le code

Clémence Daguindeau a abandonnée.

Chapitre 4

Developpement

La vue est codée en HTML 5 et nous avons utilisés CSS 3. La page a passé le validateur W3C et aussi CSS validateur

4.1 Dépôt Git Hub

Le projet est développé sur git hub et le code est accessible en ligne. Nous avons décidé d'utiliser git hub pour ce projet car cela nous a permis de voir L'avancement de projet proprement.

URL :`https://github.com/ersagun/prog_web.git`

4.2 Outils et technologies

4.2.1 Pattern MVC

Pour avoir un code plus claire nous avons utilisées le pattern modèle vue contrôleur.

Nous avons donc séparé ;

- Les modelés qui sont seulement les objets de la base de données sous forme de classes.
- La vue qui est l'arbre HTML, le rendu de l'application web.

- Contrôleur qui s'occupe de récupérer l'information par la base de données et qui transmet à la vue.

4.2.2 Langages

– Javascript et JQUERY
– PHP 5
– HTML 5
– CSS 3

4.3 jsPrincipal.js

Contient principalement tous les évènements independant de l'ancre La recherche d'un artiste, écouter une musique ou l'envoi du formulaire via ajax est géré dans

ce script.
– Fonction searchBar()
– Fonction listenMusic(val)

4.4 hash.js

Tous les url l'ancré sont géré dans ce script. Dans un tableau de clé valeur nous avons stocké tous les noms des fonctions et la clé est l'ancré. Quand l'url continent hash nous vérifions si, il est dans le tableau si c'est le cas alors nous appelons la fonction sinon la page principale est affiché dans toutes autres cas. La page principale est allMusic.

Elle contient :

– Tableau clé valeur, similaire Contrôleur (appel des fonctions)
– la fonction ready pour ecouter l'url
– Pour chaque ancre possède une fonction, en fonction de l'ancre envoi des requetés Ajax au contrôleur récupère les données nécessaire et modifie l'arbre html principale.

Pour des cas comme :
– Formulaire de connexion
– Formulaire d'inscription
il modifie l'arbre html sans requête Ajax.

4.5 Contrôleur

Contrôleur fonction toujours de la même manière est développé pour récupérer des réponses post ou get. Il regarde tout d'abord si en post ou en get nous avons une réponse request[a] qui représente donc l'action demandé. S'il en trouve dans

post ou get, alors parcours le tableau qui contient l'ensemble des actions avec ensemble des fonction qu'on a dans le contrôleur. Quand l'action demandé post[a] ou get[a] dans le tableau des actions. Il regarde la valeur correspondant et il appelé la fonction qui correspond à ce demande.

Contrôleur est appelé à chaque fois que nous avons besoin de récupérer une information par le serveur. Donc chaque action de contrôleur a pour l'objectif de récupérer des données et les transférer à nos requêtes ajax sous forme de données JSON ou alors en format texte. Cela vient à dire que le contrôleur est communiqué seulement par nos requetes Ajax.

Elle contient :

Tableau de clé valeur :
- Tableau clé valeur
- les clés post[a] ou get[a] recuperé
- Les valeurs, le nom des fonctions appelé

Les fonctions :
- allArtist()
- allMusic()
- signIn()
- Search()
- InsertUser()
- Main()

Possède une fonction main et a chaque adressage d'une requête Ajax il fait : Controller : :main($_REQUEST) ; Toutes les fonctions font écho pour donner la réponse à l'Ajax

4.6 Affichage des artistes et des chansons

Pour affichage des chansons et les artistes, dans le hash.js nous avons ajouté une fonction allArtist et allMusic. La page principale est allMusic dans deux cas, si

une page non reconnu est appelé ou si une page sans hash est appelle. allMusic et allArtist font des requêtes Ajax et envoi au contrôleur en POST a=allMusic ou allArtist. Une fois la réponse est reçu, en cas de succès l'Ajax boucle sur la donnée json et modifie l'arbre html pour afficher les artistes ou les chansons

Dans le contrôleur une fonction allArtist et une fonction allMusic. Ces fonctions vont donner la réponse a l'Ajax, car Ajax communique avec controlleur.php. Ces fonctions appelle en fonction de la demande findAll de la classe Artist ou Track. Le tableau retourné est transformé en JSON et ensuite les deux fonctions font un écho pour que l'Ajax puisse récupérer la réponse.

Dans la vue nous avons créé un div vide avec id center. Une barre de navigation qui contient des buttons avec des liens comme # allArtist ou #allMusic pour rediriger l'utilisateur vers la page qu'il souhaite.

Dans le modèle la fonction findAll de la classe Artist est modifié. une condition order by name est ajouté a la requête. La fonction findAll de la classe Track a été changé de la meme façon.

Nous avons eu besoin d'implementer JsonSerializable pour chauqe classe pour eviter les problèmes d'encodage.

Les principales difficultés rencontrées étaient :

– Le nom des artistes et chansons n'affichait pas correctement nous avons tous essayé seul moyen etait de implmeneter JsonSerializable.

4.7 Recherche des artistes, des chansons

Pour la recherche des artistes et des chansons nous avons ajouté une fonction findArtistTrackLike qui prend en paramètre la valeur récupéré par la barre de recherche. Cette fonction findArtistTrackLike récupère toutes les artistes avec leurs

chansons et toutes les chansons ensuite vérifie le titre de la chanson avec la valeur donnée en paramètre. Une requête SQL like nous permet de récupérer les chansons mais aussi voir les chansons d'un chanteur si on donne le nom du chanteur en paramètre.

On écoute la barre de recherche avec le paramètre onkeyup et à chaque fois que l'utilisateur écrit nous appelons la fonction searchBar qui est dans le jsPrincipale et ensuite on envoie une requête http asynchrone avec Ajax. La requête est envoyée au contrôleur avec les paramètres renseigné.

Voici la requête SQL qui récupère la valeur de la barre de navigation et fais une recherche dans la base de données.

select from artists as a left join tracks as t on a.artist_idt.artist_id where t.title LIKE '".$val."%' OR a.name LIKE '".$val."%' UNION select * from artists as a right join tracks as t on a.artist_idt.artist_id where t.title LIKE '".$val."%' OR a.name LIKE '".$val."%'"

Grace à cette requête nous avons réussi récupérer les chansons d'un artiste si nous le cherchons par son nom.

Les principales difficultés rencontrées étaient :

– Les requêtes MySQL ne permets pas les jointures simple, nous avons eu des difficultés a construire la requêtes SQL.

4.8 Modèles

Pour pouvoir accéder aux données de la base de données ainsi que pour gérer les données de notre site, nous avons les classes « modèles » nécessaire. C'est donc dans ces classes que nous allons trouver les requêtes nécessaires au fonctionnement du site. Par exemple, les requêtes pour la recherche de tous les morceaux de

musique.

On a choisis d'utiliser une connexion PDO pour communiquer avec notre serveur de données MySQL. Le choix de PDO est dû au fait qu'il nous permet de sécuriser les requêtes et d'utiliser les requêtes préparées.

Les principales difficultés rencontrées étaient :

- la compréhension et l'utilisation de la classe singleton de connexion à la base.
- Lútilisation de la méthode Pdo fetch both permettant de boucler sur tous les résultats de la requêtes.
- La gestion des paramètres dans les requêtes.

4.9 Modèles adjonctions

Les fonctions des modèles n'étaient pas assez il n'était pas possible de comparer des utilisateurs, ou récupérer des chansons avec track id il n'était pas possible non plus de trouver des tracks by artist id.

De plus il fallait implémenter la classe JsonSerializable pour pouvoir encoder des objets de ces classes en json.

Les principales difficultés rencontrées étaient :

- Les données récupéré en JSON n'étaient pas en UTF-8 pour pouvoir les afficher il fallait implémenter la classe JsonSerializable

4.10 Connexion à la base de données

La classe Singleton « Base » est la classe qui va permettre d'établir la connexion avec la base de données MySql. La connexion sera définit en PDO, et nous utiliserons un fichier « config.php » afin de configurer les identifiants de connexion. Cette classe aura un attribut static « dblink » permettant de retourner une connexion et qui sera utilisé dans une fonction « getConnection() » afin de faciliter les connexions à la base dans les futurs classes modèles du patern MVC.

4.11 Gestion des utilisateurs

Pour la gestion utilisateur, il est nécessaire d'utiliser un modèle, fichier PHP représentant un utilisateur dans la base de données. Cette classe permet d'effectuer des requêtes en appelant des méthodes. « User.php »

Un visiteur quelconque a alors le choix de naviguer sur le site directement, ou de créer un compte, à parti de notre page « signUp ». Nous avons décidé d'utiliser AJAX pour ne pas que la page se recharge en cas d'erreur lors de la création de compte ou même de succès. De plus, le JavaScript nous permet d'afficher un message si l'adresse email entrée n'est pas correcte, et de bloquer l'envoie de données du formulaire tant qu'elle n'est pas bonne.

Dans notre fichier « hash.js » nous avons donc nos deux fonctions permettant d'afficher le code HTML de notre formulaire de création de compte et de connexion à la base. « function signUp () » et « function signIn () »

Puisque nous avons décidé d'utiliser un MVC, nous avons ensuite, dans notre « Controller.php », deux fonctions qui sont appelées lorsque les données sont envoyées au serveur. « function insertUser() » qui est la fonction appelée par le formulaire de création de compte. Elle récupère d'abords les données puis crée un objet « User ». Ensuite elle compare cet utilisateur aux autres de la base de données pour

voir s'il n'existe pas déjà un utilisateur ayant le même nom ou la même adresse mail. Si il y en a un, un message est envoyé à notre script « jsPrincipal.js ». Ce dernier permet l'interaction entre le serveur et le client sans que la page ne soit rechargée. Ce script affichera alors si l'utilisateur a réussi à créer un compte ou au contraire, s'il y a eu une erreur.

« function signIn() » qui est la fonction appelée par le formulaire de connexion. Elle récupère les données puis vérifie que l'utilisateur existe dans la base et que le mot de passe correspond. Si ce n'est pas le cas, elle envoie un message au script « jsPrincipal.js ». Ce dernier traite ce message et affiche le message d'erreur à l'utilisateur. Si la connexion est réussie, l'utilisateur est également notifié. Difficultés rencontrées :

Ne connaissant pas très bien le fonctionnement du MVC, j'ai eu quelques difficultés de compréhension au départ. De plus, j'ai rencontré des problèmes au niveau de l'affichage notamment à cause de la partie AJAX.

Chapitre 5

Bibliographie

Les liens ci-dessous sont valides en date du 5 janvier 2015.

– http ://stackoverflow.com
– http ://openclassrooms.com
– https ://developer.mozilla.org/en-US