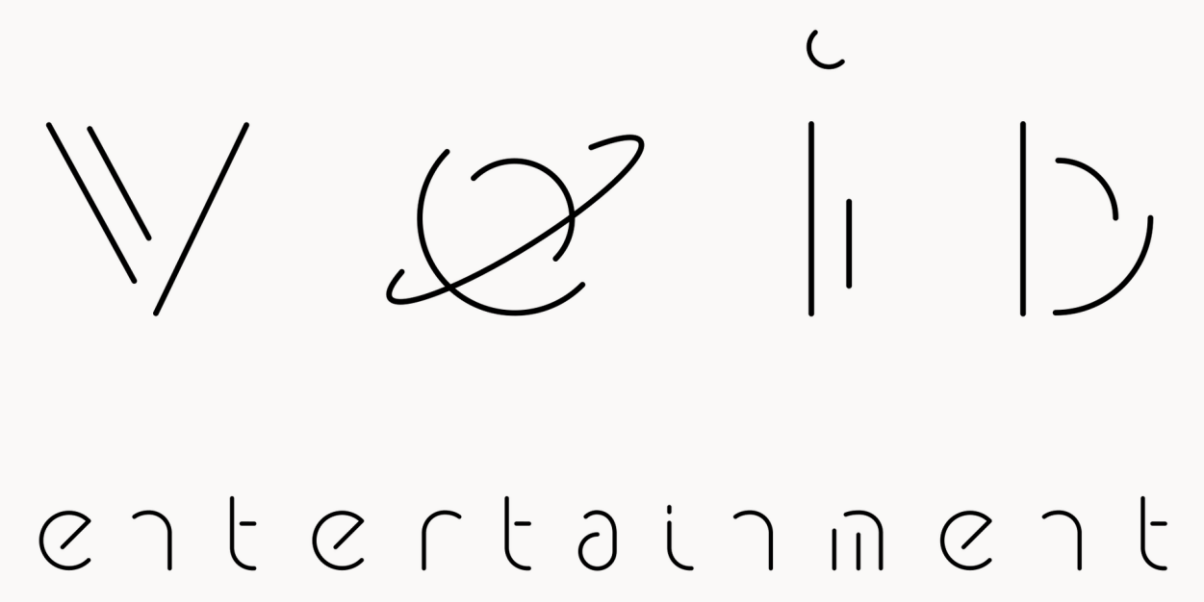

Projet Bibliothèque musicale en ligne

Glenan Bolou

Rémi Haverland

Kilian Guennec

Ruben Galindo

Table des matières :

Introduction

Nous comptons révolutionner le monde de la musique en créant un site internet qui doit permettre à tout un chacun d'écouter les meilleures musiques.

Deux grandes parties au projet :

– La partie Analyse UML ;

– La partie développement web.

Après une première analyse du projet qui nous a permis d'établir des bases solides pour la mise en place de la bibliothèque musicale, nous avons pu réaliser l'implémentation JEE de notre site de streaming musical. Ce document reprend les différents éléments nécessaires à la bonne compréhension de nos choix d'implémentations.

I-Structure générale du projet

Dans le but d'optimiser au maximum le développement de notre site, notre équipe a choisi d'utiliser une architecture de type MVC (Model – View – Controller). Cette architecture consiste à séparer notre architecture en trois couches jouant des rôles bien précis :

- La couche Model qui regroupe l'ensemble des données de l'application avec en particulier les classes permettant d'interagir avec notre base de données ;
- La couche View qui regroupe les fichiers responsables de l'interface graphique du site et de l'affichage des données ;
- La couche Controller qui a pour but d'assurer la liaison entre la couche Model et la couche View.

Cette architecture présente de nombreux intérêts puisqu'elle permet d'une part d'obtenir un projet plus clair en termes d'organisation, et d'autre part assure une assez grande indépendance entre les différentes couches pour pouvoir travailler sur celles-ci en parallèle.

Afin de pouvoir conserver les informations essentielles au bon fonctionnement de la plateforme, la mise en place d'une base de données a été nécessaire. Cette base de données permet de stocker d'une part les différentes informations des utilisateurs mais aussi la liste des albums et des contenus musicaux disponibles sur la plateforme.

Les différentes fonctionnalités que notre équipe a choisies de développer pour notre plateforme sont les suivantes :

- L'inscription et la connexion d'un utilisateur ;
- La modification des informations personnelles d'un utilisateur ;
- La gestion du catalogue musicale ;
- La lecture d'un élément du catalogue.

L'utilisateur peut se déplacer entre différentes pages telles que :

- La page d'accueil permettant de naviguer sur le site ;
- Les pages d'inscription et de connexion ;
- La page de lecture d'un contenu musical ;
- La page des nouveautés ;
- La page de gestion du catalogue ;
- La page de gestion des informations personnelles.

Un diagramme de classe se trouve en annexe et facilitent la bonne compréhension des parties suivantes.

II- La base de données

Comme expliqué précédemment, une base de données est nécessaire au bon fonctionnement de notre plateforme musicale.

Cette base de données s'oriente autour de deux parties majeures : l'aspect utilisateur (table Utilisateur) et l'aspect musique (tables musique, album et linkerAlbMus). La jonction de ces aspects est réalisée par la table likes.

La table Utilisateur regroupe les informations d'un utilisateur représenté par son nom, son prénom, sa civilité et son adresse, ainsi que ses identifiants de connexion (email et mot de passe). On associe de plus à chaque utilisateur un identifiant unique qui lui est propre ainsi que la date à laquelle il a souscrit son dernier abonnement afin de vérifier sa validité.

La table musique regroupe les informations liées à un morceaux de musique, c'est-à-dire son nom, l'artiste associé au morceau, ainsi qu'un identifiant unique qui le désigne. La table album représente un rassemblement de morceaux sous la forme d'un album identifié par son titre, son genre (pop, rock, électro, etc..), son artiste, le nombre de morceaux qu'il contient et un identifiant unique. La table linkerAlbMus regroupe les identifiants d'un morceau et de l'album auquel il appartient en tant que clé étrangère.

Enfin la table likes utilise les identifiants d'un utilisateur et d'un morceau pour indiquer à quel morceau cet utilisateur a ajouté un j'aime.

III- L'inscription

Comme toute plateforme musicale, il était indispensable pour notre bibliothèque de posséder un système d'authentification qui permet à un utilisateur d'accéder aux services proposés par notre plateforme. Or l'authentification implique au préalable qu'un utilisateur puisse se créer un compte utilisateur à l'aide d'un service d'inscription. Cette partie aborde en détails le processus d'inscription et les différentes classes qui interviennent dans ce processus.

Comme nous l'avons vu précédemment, notre équipe a choisi une architecture MVC pour notre projet et l'inscription est un exemple parfait pour illustrer cette architecture.

Tout d'abord nous avons la couche View : dans le cadre de l'inscription, la couche View est composée du fichier FormulaireInscription.jsp. Ce fichier permet de générer une page HTML qui s'affichera pour que l'utilisateur puisse s'inscrire. En effet, les fichiers JSP ont pour but d'être généré par le serveur afin d'afficher une page HTML qui servira d'interface entre l'utilisateur et l'application en elle-même. Ici, FormulaireInscription.jsp affiche une page contenant un formulaire d'inscription contenant les principaux champs nécessaires à l'inscription d'un utilisateur.

Supposons que l'utilisateur ait fini de remplir le formulaire d'inscription, il veut alors envoyer le formulaire complété pour être enregistré en tant qu'utilisateur de la plateforme : il va alors cliquer sur le bouton Inscription et une requête est alors envoyée à l'application. Nous passons à la couche Controller représentée par la servlet Inscription.java.

La servlet Inscription.java a pour but de traiter les requêtes d'inscription émises par des utilisateurs afin de rendre l'inscription effective en utilisant les différentes classes de la couche Model. Pour cela, nous allons utiliser la méthode doPost de la servlet : cette méthode va dans un premier temps créer une nouvelle instance de la classe InscriptionFormulaire, qui permet de créer un utilisateur. La servlet va ensuite créer une nouvelle instance de la classe utilisateur et remplir ses différents paramètres à l'aide de la méthode inscrireUtilisateur de la classe InscriptionFormulaire.

La méthode inscrireUtilisateur peut être divisée en deux parties : dans un premier temps, la méthode va récupérer l'ensemble des valeurs des différents champs du formulaire d'inscription qui sont stockées dans l'objet HttpServletRequest request. La méthode va ensuite comparer les différentes valeurs récupérées afin de vérifier leur validité (mail de la forme …@…, mot de passe et confirmation possèdent la même valeur, etc.), puis les attribuer à l'instance d'utilisateur créée précédemment à l'aide des constructeurs de la classe utilisateur.

Une fois l'ensemble des champs validés, le mot de passe est hashé (méthode SHA-1) et la méthode AddUser de la classe utilisateur est appelée afin d'ajouter le nouvel utilisateur dans la base de données : la méthode utilisateur utilise la méthode AjoutUtilisateur de la classe AccessBDD qui regroupe l'ensemble des méthodes qui permettent d'interagir avec la BDD.

La méthode AjoutUtilisateur va envoyer une requête de création d'utilisateur à la base de données. Pour cela, elle va charger dans un premier temps le Driver qui va permettre à l'application web de communiquer avec la BDD, puis envoyer la requête d'ajout à la BDD.

Une fois l'ensemble de ce processus effectué, la servlet va indiquer à l'utilisateur la bonne réalisation de l'inscription.

Cependant, ce n'est pas la première fois qu'intervient la servlet dans le processus d'inscription : en effet, la servlet est à l'origine de l'affichage de la page d'inscription puisque c'est lui qui, à l'aide de sa méthode doGet, va demander au fichier JSP de charger la page d'inscription.

IV- Connexion

L'utilisateur a accès à plusieurs fonctionnalités en fonction de son statut.

Un utilisateur non connecté va évidemment pouvoir s'inscrire ou se connecter mais aussi écouter les tendances du moment et les recommandations qui lui sont proposées après l'écoute d'une tendance.

Une fois connecté, il va pouvoir accéder à un plus large choix de fonctionnalités puisqu'il va pouvoir modifier ses informations personnelles et accéder à l'intégralité de la bibliothèque musicale en plus des autres fonctionnalités.

Enfin, un utilisateur peut aussi être connecté sous le statut d'administrateur afin de pouvoir modifier les éléments du catalogue en ajoutant ou supprimant des morceaux en plus des précédentes fonctionnalités.

Le principe de fonctionnement de la connexion est très proche de l'inscription : l'utilisateur va renseigner un formulaire de connexion puis émette une requête à l'application. Celle-ci va demander à la servlet Login de traiter cette requête. La servlet va utiliser la méthode checkUser de la classe AccessBdd pour vérifier si un utilisateur possédant les identifiants renseignés existe bien dans la base de données. Si c'est le cas, l'application conservera le statut de l'utilisateur en tant qu'utilisateur connecté jusqu'à ce qu'il décide de se déconnecter à l'aide d'une instance de la classe HttpSession. La servlet va ensuite renvoyer l'utilisateur sur la page d'accueil en lui indiquant que la connexion est effectuée.

L'utilisateur peut à tout moment choisir de se déconnecter à l'aide du bouton Déconnexion présent en haut de son écran. Le comportement de la déconnexion est analogue au comportement de la connexion si ce n'est qu'aucune interaction avec la base de données est requise puisque la déconnexion consiste simplement pour la servlet à supprimer l'instance de la classe HttpSession qui indiquait que l'utilisateur était connecté.

V-Musique

Lorsque l'utilisateur clique sur une musique, la page réservée à son écoute s'affiche. Le titre, l'auteur, l'album et la pochette apparaissent aussi, et l'utilisateur est libre d'appuyer sur le bouton play s'il désire jouer la musique.

Nous avons fait le choix, dans la modélisation de la musique, qu'il s'agirait d'une classe Java à part entière. En particulier, chaque musique a un identifiant unique, pratique par conséquent pour intégrer cette musique à la base de données.

Nous avons alors commencé à intégrer dans la base de données les musiques en tant que telles, afin qu'une simple requête SQL nous permette de récupérer un fichier mp3, ou wav, ou toute autre extension associée à un fichier sonore. C'est la technologie des blobs qui nous auraient permis de ne conserver aucun fichier physique dans notre application.

Nous avons finalement opté pour cette dernière solution : si les blobs semblent idéaux, leur utilisation n'est pas aisée, de même que le téléchargement dudit blob sur l'ordinateur lors de son appel, et pas plus lorsqu'il s'agissait de rafraîchir une musique.

Les fichiers musicaux sont donc conservés au sein même de l'application, et chaque appel d'écoute d'une musique s'accompagne d'une recherche dans la base de données afin de récupérer l'identifiant de la musique spécifiée. Une fois l'identifiant récupéré, il ne reste plus qu'à appeler la page d'écoute de musique.

Car c'est en passant cet identifiant en paramètre de la page JSP correspondante, que nous parvenons à afficher une page personnalisée pour la musique correspondante, et tout de même réussir à écouter cette musique.

Mais revenons plus concrètement sur le fonctionnement interne du chargement d'une page d'écoute de musique.

Lorsque l'utilisateur clique sur une musique, une requête est envoyée à la servlet, ici le MainServletMusique. Immédiatement, cette servlet récupère l'identifiant associé afin d'afficher les bonnes informations à la page JSP dédiée à l'écoute de musique, listeningMusic.jsp. La servlet va ainsi chercher dans la base de données le titre, l'album et l'auteur de cette musique.

La musique est alors modélisée par l'instanciation de la classe Musique, dont les attributs sont renseignés au mieux dans la servlet, et cette musique est transmise au fichier jsp.

Là, la page affiche toutes les informations intéressantes pour l'utilisateur. Si l'opérateur est connecté, il peut même liker la musique afin que cela soit pris en compte dans la base de données.

Conclusion

Ainsi, nous vous avons présenté les fonctionnalités majeurs développées par notre équipe pour notre plateforme musicale telles que l'inscription, la connexion, la lecture de musique ou encore la gestion du catalogue musicale. Cependant, de nombreuses autres fonctionnalités sont à l'étude pour améliorer encore la qualité de notre application. Nous pensons en particulier à la mise au point du système de blob pour la musique de manière parfaitement fonctionnelle, l'ajout d'une distinction entre deux types d'administrateur (les administrateurs musique et clients) qui auraient accès à des fonctionnalités différentes ou encore l'ajout d'un système de statistiques qui permettrait aux administrateurs de consulter diverses informations comme les morceaux les plus écouter pour un artiste donné, ou encore le nombre de visites par jour.