

**RAPPORT MIDDLEWARE**

Développement d’un serveur d’application de diffusion de musique en streaming

**Adonis SETTOUF**

**Adnane LOUARTI**

**Sara KHOUYIBABA**

**Jun WANG**

Sommaire

[1. Présentation générale 3](#_Toc379835593)

[2. Architecture de l’application 3](#_Toc379835594)

[2.1. Description générale/ Outils utilisés 3](#_Toc379835595)

[Maven 3](#_Toc379835596)

[Git 4](#_Toc379835597)

[Glassfish 4](#_Toc379835598)

[EJB 4](#_Toc379835599)

[MySQL 5](#_Toc379835600)

[OpenJPA 5](#_Toc379835601)

[2.2. Description des EJB 6](#_Toc379835602)

[Les entités 6](#_Toc379835603)

[Les EJB 8](#_Toc379835604)

[3. Conception de la base de données 8](#_Toc379835605)

# Présentation générale

Le projet consiste à mettre des technologies telles que les EJB et autres au service des utilisateurs d’applications de musique en streaming. En effet, une bonne gestion des pistes de musique permet une aisance à utiliser l’application. Pour ce fait, dans le cadre du projet d’IHM et de développement mobile, le but est de ré implémenter l’API déjà existante à l’aide de technologies middleware assurant la persistance des données ainsi que la communication avec les contrôleurs qui doivent assurer la prise en charge des requêtes REST effectuées par un terminal mobile.

# Architecture de l’application

## Description générale/ Outils utilisés

Le but de notre projet est d’utiliser l’API mise à notre disposition, y ajouter les technologies JAVA, mettre en place les EJB nécessaires et de faire une sauvegarde dans notre base de données. Pour ce fait, nous avons utilisé différents outils tels que : Maven, Git, Glassfish, EJB, OpenJPA.

### Maven

Maven sert à gérer le cycle de vie de l’application, il impose une structure de fichier et gère l’arborescence à l’aide du fichier pom.xml. Toutes les libraires nécessaires doivent être indiquées à Maven via les dépendances du Pom. Elles peuvent soit exister localement, sinon Maven les récupère depuis un repository public. De plus, le concept POM permet d'avoir une compréhension plus simple et plus complète du projet. Et finalement, Un élément clé et relativement spécifique de Maven est son aptitude à fonctionner en réseau.

Les dépendances ayant été ajoutées dans notre Pom :

openjpa-kernel, openjpa-persistence, mysql-connector-java, hsqldb, javaee-api, openjpa-persistence-jdbc, openjpa-jdbc, openjpa-lib, junit.

### Git

L’avantage le plus de Git est de pouvoir travailler en groupe sur un même projet. C’est est un logiciel de gestion de versions décentralisé caractérisé par sa rapidité de mettre à jour les informations modifiées.Et contrairement à SVN, Git ne surveille pas les fichiers mais leur contenu. Cela permet de faire des choses qui auraient été impossibles autrement, comme savoir qu’une fonction a été déplacée d’un fichier à un autre.

Un avantage certain de Git est de pouvoir rollbacker à une version précédente en cas de code trop inutilisé, de créer des branches lors d’ajout de fonctionnalités différentes à un « tronc » de base (ici ce n’est pas le cas).

### Glassfish

GlassFish est un serveur d'applications compatible Java EE. Son implémentation recouvre plusieurs fonctionnalités que nous utiliserons pour le déploiement de notre projet. En effet, c’est lui qui va gérer les requêtes SQL grâce à un pool de connexions à la base SQL, et gérer étroitement l’EntityManager fournie par OpenJPA permettant d’assurer la persistance des données.

### 

### EJB

Un EJB est un ensemble de JavaBeans constituant une architecture de composants logiciels côté serveur pour la plateforme de développement Java EE.

Les principaux avantages des EJB:  
- Ils supportent un nombre illimité de connexions simultanées grâce au serveur d'application qui se duplique en cas de surcharge.  
- Ils permettent persistance la gestion d'une Base De Données (suppression, modification et ajout).

-Ils permettent de « peupler » une DAO sans avoir à l’instancier

-Ils permettent de faire l’interface entre le contrôleur et la DAO gérant l’accès à la base de données.  
- Sont accessibles par plusieurs types de clients.

### MySQL

Nous avons préféré utiliser une base de données de type MySQL car celle-ci se caractérise par:

* Sa rapidité : Le serveur MySQL est très rapide grâce aux algorithmes de recherche qu’il implémente
* Ses API diverses : On peut effectuer diverses opérations sur une base MySQL
* Sa gestion de toutes les jointures grâce à l’opération Join (contrairement à MongoDB qui n’offre pas cette fonction)
* Son langage de requêtage facile à utiliser

### OpenJPA

OpenJPA implémente la Java Persistence API. Nous l’avons utilisé suite au choix de MySQL en tant que Base de Données pour permettre une rapidité d’exécution grâce à son mapping objet-relationnel (ORM) pour le langage Java ainsi que de par notre connaissance du framework Hibernate qui se base sur OpenJPA

Le fichier persistence.xml est créé dans le répertoire META-INF, c’est le fichier  de configuration de l'API JPA.



## Description des EJB

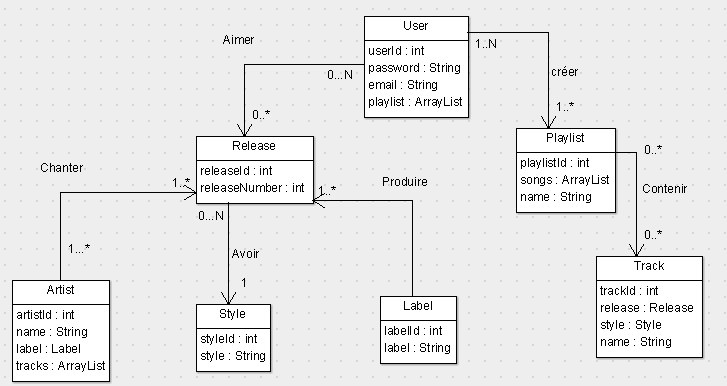
### Les entités

|  |  |  |
| --- | --- | --- |
| **ENTITE** | **FONCTION** | **ATTRIBUTS** |
| **Artist** | **gère les artistes** | artistId  name  label  tracks : propriété gérée par la relation one to many |
| **Label** | **gère les labels attribués aux artistes** | labelId  label |
| **Release** | **Cette entité gère les releases** | release\_Id  releaseNumber |
| **Playlist** | **Gère les listes de musique** | playlistId  name  songs : attribut géré par une relation one to many entre les classes track et playlist |
| **Style** | **Gère le style de chaque album** | styleId  style. |
| **Track** | **Gère les chansons** | trackId  release  style  name |
| **User** | **gère les utilisateurs de l'API** | userId  email  password  playlist : gérée par une relation one to many entre les classes user et playlist |

### Les EJB

|  |  |
| --- | --- |
| **ArtistDao** | A faire |
| **LabelDao** | A faire |
| **StyleDao** | A faire |
| **TrackDao** | A faire |
| **PlaylistDao** | A faire |
| **ReleaseDao** | A faire |
| **UserDao** | A compléter |

# Conception de la base de données



**Conclusion**

Ce projet a été l’occasion de mettre en œuvre un middleware, en partant uniquement des spécifications techniques, à l’aide de technologies classiques du monde J2E. De la persistance des données avec l’utilisation d’OpenJPA, aux contrôleurs assurant la communication en RESTful avec les clients, en passant par l’utilisation d’EJB permettant d’assurer facilement la liaison avec la base de données.