# Projet Final – Lecteur MP3 Android

## Objectifs :

* Démontrer son savoir-faire dans le développement d’interfaces graphiques Android
* Démontrer sa capacité à créer des listes simples et complexes
* Démontrer sa capacité à utiliser un service afin de diffuser des mp3.
* Coder un modèle de données utilisant un Singleton

## But du projet :

Votre lecteur MP3 doit permettre :

* De récupérer tous les mp3 présents sur la carte SD d’un téléphone
* De les trier par artiste
* De les diffuser de manière continue ou aléatoire
* De pouvoir interrompre/reprendre la lecture ( pause ) , passer à la chanson suivante / précédente
* De pouvoir chercher un moment précis de la diffusion à l’aide d’une SeekBar
* BONUS : de pouvoir créer des listes de lecture et de pouvoir les utiliser.

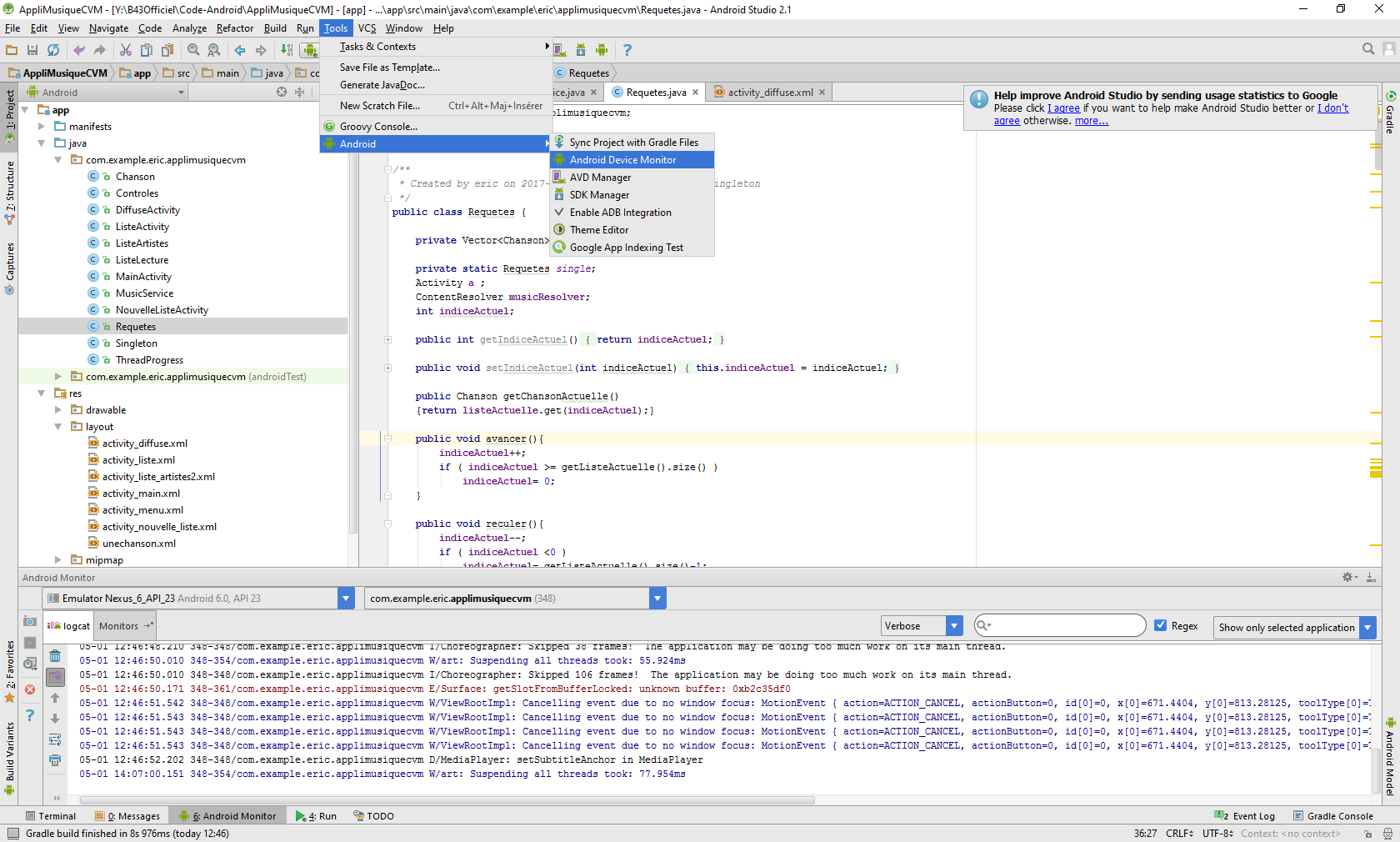
## Nouveautés

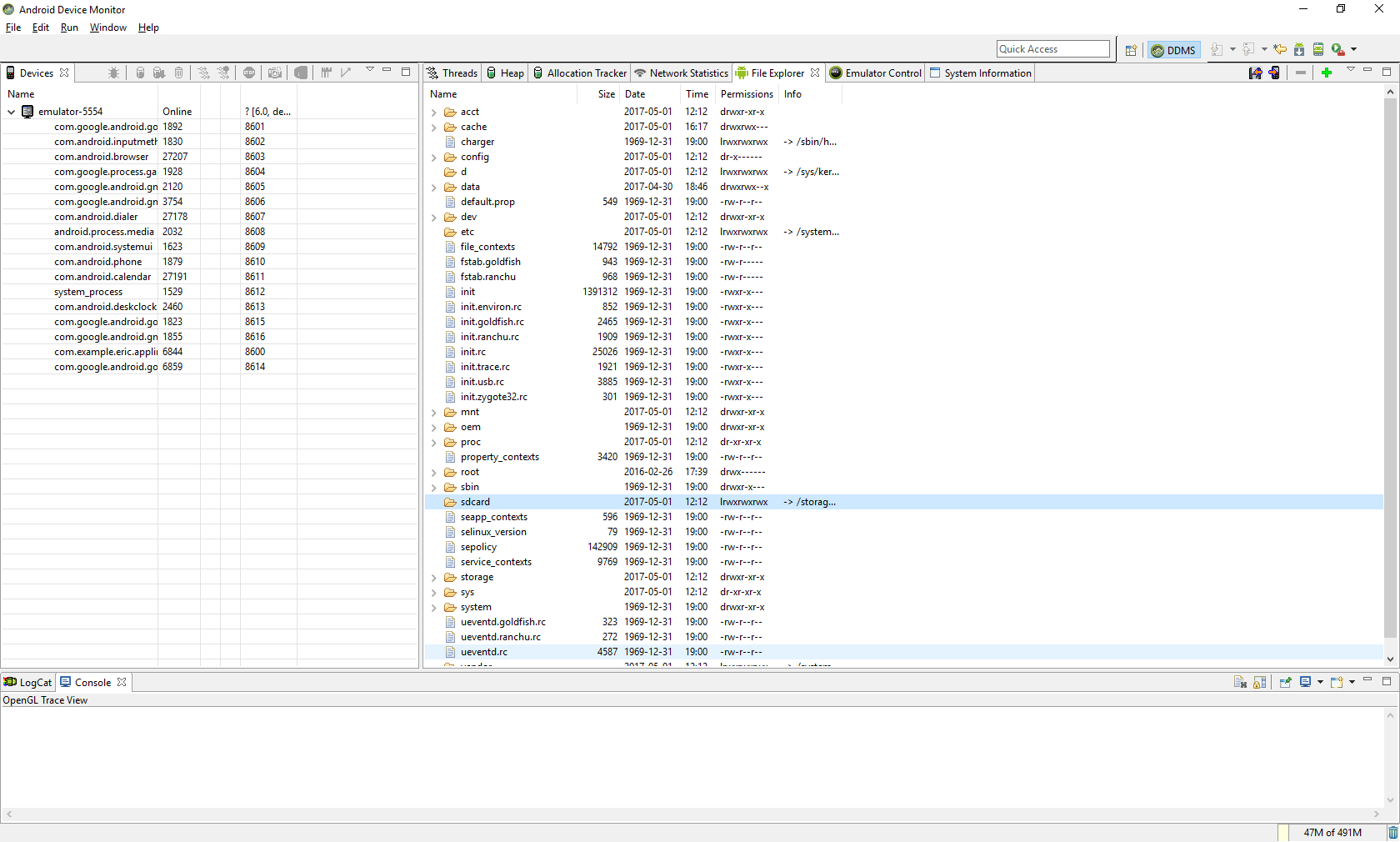
1. Qu’est-ce qu’un singleton ( patron de conception ) ?
2. Qu’est-ce qu’un Service Android ?
3. Qu’est-ce qu’un ContentResolver ?
4. Qu’est-ce qu’un Handler / Thread ?

Utiliser projet fourni:

* Permissions dans le fichier manifest
* Modification du target SDK dans fichier build.gradle

Pour ajouter des chansons sur la carte SD de votre émulateur





\*\*\*IMPORTANT, vous devez redémarrer votre émulateur pour qu’il tienne compte des nouveaux fichiers mp3.

\*\*\*IMPORTANT à la maison vous devez avoir l’API 22 installé ( pas seulement l’API 23 )

## 1.Modèle de données

1. Classe Chanson ( 15 min ) : répertorier toutes les informations importantes relatives aux mp3 que l’on diffusera. N’oubliez pas d’y inclure comme variable un int id car cette information est important afin de pouvoir diffuser les mp3.
2. Classe EnsembleChansons  ( 2h ) : C’est la pièce centrale de notre projet. Il s’agit d’un singleton qui contiendra notre Vecteur de Chansons qui changera dépendant des choix d’artistes ou de liste de lecture de notre utilisateur.

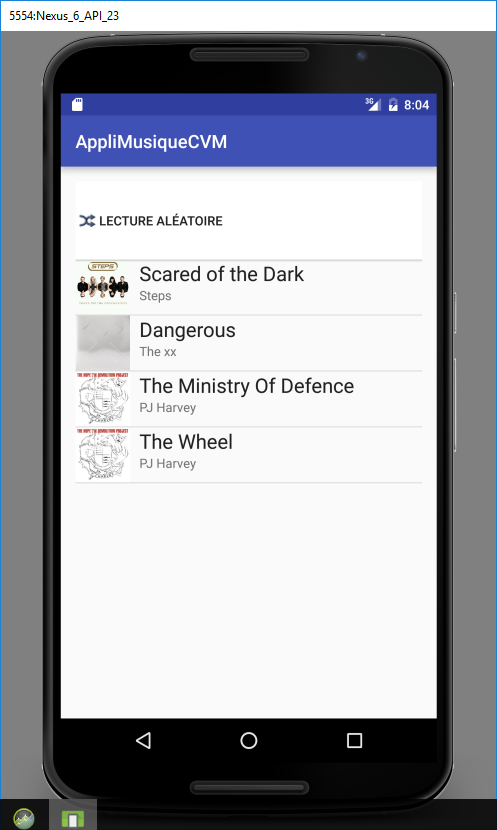
* Variables possibles / suggérées :
  + Référence static à EnsembleChansons ( voir notes sur le singleton )
  + Vector<Chanson> représentant en tout temps la liste des chansons qui doivent être diffusées ( appelé à changer )
  + Int : représentant l’indice de la chanson jouée présentement dans le Vector
  + ContentResolver : important pour faire nos requêtes sur la carte SD
* Méthodes possibles / suggérées :
  + Méthodes pour avancer ou reculer d’une chanson celle jouée présentement ( attention aux extrémités : doit revenir à la première si on faire suivant sur la dernière )
  + Méthodes d’accès / de mutations habituelles
  + Méthode permettant de remplir le Vector avec toutes les chansons présentes dans la carte SD
  + Méthode permettant de remplir le Vector avec toutes les chansons provenant d’un Artiste passé en paramètre
  + Méthode permettant de retourner une collection de tous les artistes dont les chansons sont présentes sur la carte SD.

1. Classe MusicService

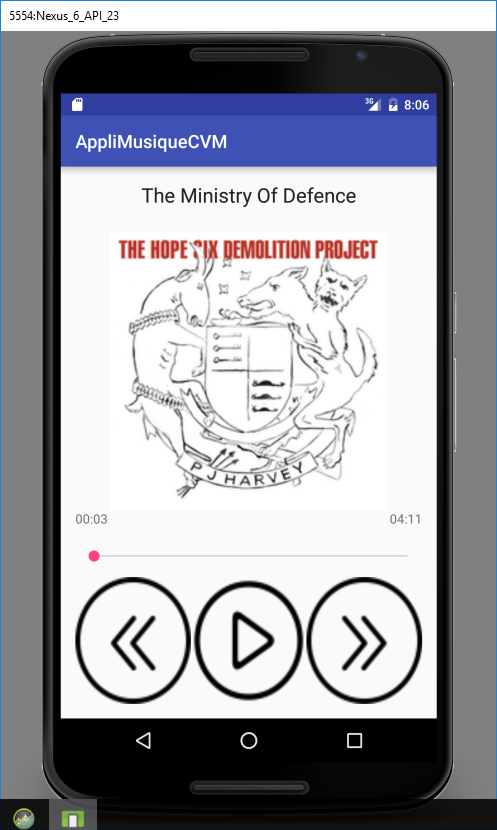
* Codée en grande partie pour vous; il vous reste à combler les commentaires par une instance de votre classe-singleton EnsembleChanson

## 2.Activités

* Je veux vous laisser de la liberté vis-à-vis le développement de votre app. Cependant, je dois y retrouver les fonctionnalités suivantes :
  + L’app doit s’ouvrir avec un logo de votre app ainsi que des boutons / menu permettant d’utiliser différentes options d’écoute ( liste de toutes les chansons, liste des artistes, création de listes de lecture, etc. )
  + L’app doit avoir une activité représentant l’ensemble des chansons, celles-ci représentées par la pochette de l’album, le nom de la chanson ainsi que l’artiste. Par exemple :



* + À partir de cette liste, on peut cliquer sur une chanson afin de nous amener vers une autre activité avec les contrôles de diffusion :



On peut ainsi appuyer sur PLAY pour diffuser la chanson, ce bouton se tranformera en PAUSE et vice-versa.

* + Les contrôles PRECEDENT et SUIVANT sont fonctionnels
  + Le seekBar permet de suivre le cours de la chanson, le nombre de min :sec ( utiliser un GregorianCalendar / SimpleDateFormat \_ est affiché.
  + Le décompte pourrait utilisera un Handler / un Thread ( Runnable )
  + On peut avancer la chanson en interagissant avec la seekBar.
  + 2 méthodes seront nécessaires pour lier votre activité de diffusion avec le service : à ajouter :

**private** ServiceConnection **musicConnection** = **new** ServiceConnection() {  
 @Override  
 **public void** onServiceConnected(ComponentName name, IBinder service) {  
 MusicService.MusicBinder binder = (MusicService.MusicBinder) service;  
 **musicSrv** = binder.getService();  
 *// musicSrv.setChansons(liste);* **musicBound** = **true**;  
 }  
  
 @Override  
 **public void** onServiceDisconnected(ComponentName name) {  
 **musicBound**=**false**;  
 }  
};

**protected void** onStart(){  
 **super**.onStart();  
 **if** ( **playIntent** == **null**)  
 {  
 **playIntent** = **new** Intent(**this**, MusicService.**class**);  
 bindService ( **playIntent**, **musicConnection**, Context.***BIND\_AUTO\_CREATE***);  
 startService(**playIntent**);  
  
  
 }  
}

* Il est possible d’avoir la liste de tous les artistes ayant au moins une chanson sur la carte SD et de faire un choix d’un artiste ; de cette façon le Vecteur de Chansons comprendra que les chansons de cet artiste.
* Il peut être possible que les chansons soient diffusées dans l’ordre ou dans le désordre ( aléatoire )
* Il peut être possible de sélectionner des chansons afin qu’elle fasse partie d’une liste de lecture : dans ce cas, les chansons de la liste de lecture seront diffusées. Vous devez trouver un moyen de sauvegarder ces listes de lecture afin de pouvoir les réutiliser.

## 3.Évaluation

14% de la session

Remise : 21 mai 17

Grille de correction sommaire :

|  |  |
| --- | --- |
| Modèle de données | 5% |
| Requêtes + activités listes | 25% |
| Musique diffusée | 15% |
| Contrôles de base + décompte | 30% |
| Contrôles avancés ( recherche )  ( diffusion aléatoire ) | 10% |
| Bonus : Listes de lectures | + 10% |
| Ergonomie, facile à utiliser | 5% |
| Commentaires / indentation / réutilisation / respect des principes orientés-objet | 10% |