## Equalizer

Bouton de lancement : **android:id="@+id/tonalite".**

Méthode associée au clic du bouton : **public void openEqualizer(View v)** ligne 612 du fichier **AAP.java**. Cette méthode se contente de lancer l’activity **EqualizerActivity** du package fr.unice.aap.

EqualizerActivity initialise ensuite tout ce qui est nécessaire au fonctionnement de l’equalizer et du visualizer. Comme décrit dans le rapport, l’Equalizer va rechercher le nombre de bandes de fréquence dont est composé le morceau en cours de lecture (ligne 74). Ce morceau est récupéré (ligne 69) via un champ static de la classe AAP :

mEqualizer = **new** Equalizer(0, AAP.*mPlayer*.getAudioSessionId());

Les Seekbar qui permettent la variation de la valeur des bandes de fréquence sont alors générées « en dur » et non pas via un fichier xml.

L’equalizer est réinitialisé à chaque changement de chanson dans le fichier **AAP.java**, pour cela :

1. L’EqualizerActivity est stockée dans AAP.java grâce à la ligne suivante de EqualizerActivity.java (ligne 51)

AAP.*equalizerActivity* = **this**;

1. Lors de chaque changement de piste (appel des méthodes musiqueSuivante et musiquePrécédente) la méthode réinitialisation est appelée : ***equalizerActivity*.resetEqualizer();**

## Lyrics / Paroles de chansons

Bouton de lancement : **android:id="@+id/paroles"**.

Méthode associée au clic du bouton : **public void findLyrics(View v)** ligne 630 du fichier AAP.java. Comme toutes les méthodes issues d’un clic de notre interface, le passage en paramètre d’une vue est obligatoire, cependant la vraie méthode findLyrics sera déclanchée autrement que via le bouton c’est pourquoi la surcharge ayant pour paramètre une vue est créée.

**public void findLyrics()** ligne 634 réalise donc le vrai travail :

* Elle déclenche l’animation pour le retrait du menu de fonctionnalités,
* Elle rend visible la ScrollView qui contiendra toutes les informations,
* Elle appelle ensuite la méthode **setLyricsTextResult(String artist, String song, boolean estRetour)** avec comme paramètres la chanson en cours de lecture, ainsi une première recherche est effectuée automatiquement sans que l’utilisateur ne rentre d’informations.

La méthode **setLyricsTextResult(…)** effectue alors la requête sur la base de donnée de **LyricsWiki** de la sorte :

* Si tous les champs artist et song sont renseignés, on récupère le résultat sous format xml, on le parse pour afficher l’échantillon de paroles ainsi qu’un bouton permettant d’ouvrir un navigateur web pour afficher l’intégralité des paroles.
* Si seul l’artiste est renseigné et que l’appel de la méthode n’est pas un retour (c’est un retour lorsque l’utilisateur quitte le navigateur web ouvert à l’aide des boutons dédiés, ce booléen est nécessaire car la fermeture du navigateur appel la méthode setLyricsTextResult) un navigateur internet est alors ouvert sur la page de cet artiste affichant ainsi la liste complète de ses chansons et albums.
* Si la recherche n’a donné aucun résultat, un simple « Not Found » est affiché.

Lorsque le navigateur web a besoin d’être ouvert, la méthode **openBrowser()** est appelée. Cette méthode rend le conteneur de la page de recherche invisible et celui du navigateur visible.

Deux lignes importantes de cette méthode :

* **lyricsView.getSettings().setJavaScriptEnabled(true)**

Active le JavaScript (les paroles sont affichées en javaScript sur le site LyricsWiki).

* **lyricsView.setWebViewClient(new WebViewClient())**

Force l’ouverture de nouvelles pages dans une WebView. Par défaut, lorsqu’une nouvelle page doit être ouverte (click d’un lien) le navigateur internet par défaut du téléphone est lancé, sortant ainsi du cadre de notre application. En forcant l’ouverture d’une WebView on permet alors à l’utilisateur de quitter à tout moment.

## Enregistreur

Bouton de lancement : **android:id="@+id/rec"**.

Méthode associée au clic du bouton : **public void openRecordLayout(View v)**. Cette méthode déclenche l’animation pour le retrait du menu de fonctionnalités et rend visible la fenêtre d’enregistrement : **android:id="@+id/RecordLayout"**.

Ce conteneur est composé d’un champ modifiable correspondant au nom de l’enregistrement, d’un bouton qui déclenchera et arrêtera l’enregistrement et enfin d’un bouton pour minimiser la fonctionnalité.

La méthode principale liée à cette fonctionnalité est donc **recordSound(View v)** ligne 751. Cette méthode démarre un nouveau **Recorder** (fr.unice.aap.recorder.Recorder.java) dont le paramètre est le chemin du fichier qui sera enregistré. Ce chemin sera formaté pour assurer le bon fonctionnement de l’enregistrement.

Quelques lignes importantes :

* recorder.setAudioSource(MediaRecorder.AudioSource.*MIC*);

Défini le microphone du téléphone comme source d’enregistrement.

* recorder.setOutputFormat(MediaRecorder.OutputFormat.*THREE\_GPP*);

Défini le format d’enregistrement en .3gp.