Compte rendu Jalon 3

Lien Github : https://github.com/zurchert/iutvalence-projetS2-2015-groupe18-

Ce projet a été l'occasion pour nous de découvrir le travail d'équipe et le suivi d'un planning. A travers l'avancée de notre projet, nous avons dû faire face à de nombreuses difficultés. Certaines ont rapide à résoudre et d'autres nous ont pris plusieurs heures.

Nous sommes partis sur l'idée de réaliser un lecteur audio. A première vue, ce projet nous semblait accessible. Mais ce fut qu'une simple vision. En effet, parmi les librairies permettant de lire un fichier audio, nous avons choisi d'utiliser JLayer, qui est une librairie de JavaZoom. Cette librairie nous permet de lire des fichiers audio au format MP3. Sauf qu'avant son utilisation pour le projet, nous ne l'avons jamais utilisée. Il nous a donc fallu comprendre certains de ces éléments et trouvé les classes et les types qui nous seraient utile pour notre projet JAVA. Par conséquent, les erreurs ce sont accumulés durant le codage à cause d'une mauvaise utilisation de la librairie.

Pour commencer, nous avons décidé de coder les éléments de bases, c'est-à-dire les classes Library, Media, Music, Radio, Playlist. La classe Library nous permet de créer une bibliothèque, que l'on enregistre sur un fichier XML, qui se trouvera dans le package, pour faciliter la création du fichier. Music et Radio héritent de Media, et leurs paramètres permettent de créer un Media. En effet, lorsque l'on créer un Media, on lui met un titre, le chemin permettant d'accéder à la musique ou l'URL permettant de lancer le flux radio, l'album, le nom de l'artiste, la longueur et la note. Ces 4 derniers paramètres sont pour Music, et son facultatif sauf pour la longueur. Dès qu'un media est créé, il sera classé dans le fichier XML et on lui attribuera un identifiant en fonction des autres médias présents dans le fichier. Nous créons donc une liste de médias que l'on range dans ce fichier. Nous avons eu des soucis pour cette classe avec la librairie JAudioTagger pour importer la longueur d'une musique. De plus dans cette classe, nous avons créé des méthodes qui nous permettent d'importer des médias, de les mettre à jour, et aussi de sauvegarder la librairie après chaque changement. Ensuite, nous avons créé la classe M3TPlayer qui nous permet de gérer les événements sur le lecteur audio. Nous avons donc créé plusieurs méthodes, dont une qui permet de lancer le média, une qui stoppe le média, une qui permet de passer au suivant et au précédent, une qui permet d'activer le mode aléatoire, et une qui change le volume. C'est la dernière qui nous a posé le plus de problèmes. En effet, nous avons tenté plusieurs codes et nous nous somme renseignés sur plusieurs forums pour trouver une solution. Le but de cette méthode était de pouvoir modifier le volume directement depuis l'application. Sauf que nous n'avons pas réussi à trouver un code qui fonctionne correctement, même en changeant de librairie. Nous avons mis en place une classe qui permet de faire remonter des exceptions si le média est inconnu. Ensuite, nous avons attaqué l'IHM de notre projet. Nous avons créé les classes suivantes : MainWindow, MusicListPanel, Launchner, StatusBar, ControlButtonsPanel. Cette dernière permet de créer tous les boutons de notre interface. La

classe MainWindow représente la plus grande partie pour l'IHM. Elle permet de faire afficher une fenêtre et ses caractéristiques mais aussi de créer des événements derrière les boutons de notre interface. Cette partie nous a posé beaucoup de problème également. En effet, notre premier problème fut que lorsque nous lançons une musique, nous ne pouvions plus interagir avec l'application car elle se figeait lors de l'exécution de la musique. Pour remédier à ce problème, nous avons décidé de créer un nouveau thread lors du lancement de la musique. Sauf que cela nous a à la fois résolu un problème et créer un nouveau problème. Nous avions mis du temps à trouver comment utiliser les attributs pour pouvoir les modifier dans le thread sans que cela génère des erreurs. De plus, nous n'avons pas réussi à trouver comment stopper le thread sans utiliser la méthode dépressive stop(). Nous avons voulu faire une interface facile à comprendre pour les utilisateurs, et pour cela, nous avons créer les boutons play, qui sert de bouton pause également, stop, random, previous, next et import. Nous avons gérer les évènements de ce bouton dans cette classe. Lorsque nous cliquons sur le bouton import, une fenêtre s'ouvre et permet d'aller récupérer directement nos musiques sur l'ordinateur. Ceci est plus agréable pour l'utilisateur. De plus, la musique s'affiche directement dans un tableau, sous forme de liste, qui se trouve en dessous des boutons, ce qui nous permet de voir qu'elles sont les musiques déjà importer. Nous pouvons directement lire ces musiques avec le bouton play, mais nous pouvons choisir aussi directement depuis le tableau quelle musique nous souhaitons lire en double-cliquant sur la ligne en question. Le codage fut plus long que prévu finalement et le temps et s'est très rapidement écoulé.

Pour notre organisation, nous avons mis nos idées sur brouillon dans les premières semaines du temps impartis. Une fois qu'on ait réussi à obtenir ce que l'on voulait nous avons décidé de commencer à la se pencher sur la partie de « codage ».

Pour notre expérience, nous avons donc découvert le travail d'équipe et le suivi d'un planning. Mais cela nous a appris aussi à travailler en autonomie car il a fallu apprendre à trouver les solutions tous seul à nos problèmes. Cette physionomie nous a permis de nous autoévaluer pendant tout le projet. De plus cela nous a donné notre première expérience en tant que « projet » et aider à améliorer notre niveau en JAVA.

Pour conclure, ce projet nous a donc permis de nous créer une expérience, mais aussi de découvrir de nouvelle librairie et des difficultés que nous pourrons de nouveau rencontrer lors des prochains projets.