Par Dufour Vivien et André-Masse Maxime

MusiLib

MusiLib est une application répertoriant des partitions de musique pour aider les gens à apprendre à les jouer. Vous pouvez écouter la partition et utiliser le métronome, vous pouvez régler leur vitesse et donc vous entrainez à jouer cette partition.

Nous avons choisi d'utiliser une structure comportant 3 classes, la classe Partition, la classe Métronome et la classe Trier.

La classe Partition comporte les attributs suivants : un nom, une liste d'images, un auteur, une complexité, une description, un instrument, un son.

La liste d'images permet de stocker toutes les images de partitions afin de pouvoir les jouer / apprendre. Le son comporte la chaîne de caractères du son en .mid, .wav … afin de pouvoir le lancer.

Cette classe comporte seulement 2 méthodes, une pour ajouter des images et l'autre pour en enlever.

La classe Metronome comporte les attributs suivants : un AudioPlayer, un booléen pour vérifier si la musique se joue et un second pour vérifier si la musique déjà a été lancé (même si elle est en pause).

Le AudioPlayer utilise le plugin Maui.Audio afin de pouvoir jouer des sons sur Android, Windows et iOS.

Cette classe comporte une méthode pour créer le lecteur de musique et lancer une musique (ou le métronome) à partir du chemin du fichier donné en paramètre (l'attribut son de la partition). Il y a aussi une méthode pour jouer / relancer la musique si elle est en pause, une méthode pour la mettre en pause, une pour l'arrêter, une pour régler le tempo de la musique ou du métronome.

La classe Trier comporte les attributs suivants : une liste de Partition pour stocker les partitions initiales et une autre pour stocker les partitions filtrées.

Cette classe comporte 3 méthodes de tri, une pour trier par instrument, une pour trier par complexité et une pour trier par ordre alphabétique. Ces classes prennent en paramètre un attribut correspondant à ce qu'elles font, un attribut instrument pour trier par instrument et ainsi de suite.

Nous avons aussi une classe Manager comportant comme attributs : une interface de IPersistanceManager, une liste de partitions favorites et une liste de partitions.

Nous utilisons cette classe pour gérer toute l'application en stockant les partitions et les favoris, elle sert également à charger et sauvegarder nos données sur le téléphone afin de les récupérer à la réouverture de l'appli pour ne pas perdre ses favoris.

L'interface IPersistenceManager contient les définitions des fonctions de la classe Manager.

Nous utilisons également une interface IAllowClick qui nous permet de savoir si l'on peut cliquer sur un bouton ou si le délai n'est pas encore passé, cela évite les bugs sur téléphone permettant d'ouvrir plusieurs fois la même page en appuyant plein de fois dessus.

Nous utilisons aussi un Stub nous permettant de charger les données la première fois que l'on lance l'application. Dedans, on initialise toutes les partitions avec tous leurs attributs et on les ajoute à la liste de partitions.

Nous avons également recours à une classe DataContractPers qui nous permet de récupérer les données sur le téléphone quand on relance l'application, elle permet aussi de sauvegarder des données sur le téléphone notamment les partitions qu'on a en favori qui se sauvegarde dès qu'on les ajoute en favori.

Notre application contient 3 pages qui sont : la page d'Accueil, la page Favoris, et la page d'une partition, et respectivement géré par les classes Accueil, Favoris et PartitionView.

La classe Accueil contient un attribut MyManager de type Manager, 2 attributs partitionsInitiales et partitionsFiltrees de type List de Partition et un attribut trieur de type Trier.

Cette classe contient les méthodes :

GoToFavorisButton, permettant l'accès à la page Favoris

GoToPartitionButton, permettant l'accès à la page d'une partition

ChargerPartitions, permettant d'afficher toutes les images de partitions

GetImageButtonMargin, qui récupère des valeurs pour les margin de chargerFavoris et d'avoir un affichage aligné

SearchBar_TextChanged, permettant d'afficher différemment les partitions selon ce que l'on tape dans la barre de recherche

TriButton_Clicked, lançant un popup pour choisir le type de tri souhaité

TrierParTypeButton, TrieParDifficulteButton et TrieParOrdreAlphabetiqueButton permettant de trier différents les partitions selon notre choix

Et ReinitialiserButton, afin de réinstaller l'affichage par défaut après un tri.

La classe Favoris est très similaire à la classe Accueil étant donné qu'elle contient les mêmes fonctions et attributs, mais contient une méthode de plus, OnAppearing, permettant de recharger les partitions en favoris sur la page quand on revient d'une page de PartitionView si jamais une partition a été supprimé des favoris.

La classe PartitionView, quant à elle, contient comme attribut, MyManager de type Manager et 2 variables de type Metronome, music et metronome.

Elle contient comme méthodes :

Les fonctions Play_Music et Play_Metronome, qui lancent respectivement la musique d'une partition ou le son du métronome.

Les fonctions Stop_Music et Stop_Metronome, qui stoppent respectivement la musique d'une partition si elle est lancée ou le son du métronome.

Les fonctions OnBackButtonPressed, qui coupe la musique et le métronome quand on revient en arrière avec le bouton retour du téléphone

Les fonctions OnDisappearing et Shell_Navigating, sont très similaires à OnBackButtonPressed mais fonctionnent avec le bouton retour de l'app.

La fonction InitializeButton, qui permet de mettre l'image de l'étoile si la partition est en favori ou non.

La fonction ChargerPartitionsSimilaires, qui initialise et créer des imageButton des partitions du même auteur et instrument pour les afficher automatiquement.

Les fonctions GoToPartitionButton et AddFavoriButton, qui sont similaires aux méthodes des classes Accueil et Favoris

Et les fonctions TempoSlider et BPMSlider, permettant respectivement d'envoyer la valeur du slider pour modifier la vitesse de lecture d'une musique et d'un métronome.