Travail pratique 1 – Lecteur Multimedia utilisant ExoPlayer ( 10% de la session )

Objectifs :

* Utiliser les classes reliées à ExoPlayer pour diffuser de la musique
* Utiliser les librairies Volley et GSon afin de récupérer les ressources musicales sur un serveur distant
* Développer une application fonctionnelle avec un thème original
* Séparer autant que possible le modèle de données de la vue en utilisant le modèle de conception Observer
* Gérer le cycle de vie de l’app pour que l’application soit agréable à utiliser

## Partie 1 de 2 – Lecteur comme tel, opérations sur le player

Créez un nouveau projet API 26. Si ce n’est pas déjà le cas, changez le **compileSDK et le targetSDK à 34** ( à downloader à la maison si ce n’est pas déjà fait )

Développez votre idée de lecteur. Vous avez la liberté d’explorer la grande bibliothèque de classes ExoPlayer. Les fichiers à lire sont présents sur le serveur :

<https://api.jsonbin.io/v3/b/661ab8b1acd3cb34a837f284?meta=false>

Autre serveur disponible : <https://api.npoint.io/d4c29479e010376e6847>

Si le serveur ne répond pas : Arrêter l'émulateur. Le fermer. Le redémarrer seul et ensuite démarrer votre application

Attention ! Ce fichier JSON est appelé à être modifié à plusieurs reprises…la structure des objets ne changera pas cependant.

## Le modèle

Vous devrez lire les données en utilisant les librairies Volley et GSon de manière à vous bâtir une liste d’objets représentant votre librairie de fichiers multimédias. Cette partie doit être faite dans une classe n’étant pas une Activité.

Vous devrez trouver un moyen de détecter dans votre Vue que la réponse du serveur a été bien reçue dans le modèle 🡪 à venir à l’atelier 5

Cette / ces classes pourraient être l’occasion de filtrer les fichiers multimédias par catégorie ( artiste, album, genre… )

## La Vue

La Vue comprendra le ExoPlayer et les interfaces graphiques. Tel que vu dans l’annexe 4, on doit lier le ExoPlayer avec le composant graphique PlayerView. **Cependant, désactiver les contrôles automatiques sur le PlayerView en appliquant la méthode :**

***vue.setUseController(false);* où vue est l’objet PlayerView**

le but étant que vous développiez une interface originale où des boutons / autres composants peuvent être reliés à des actions ( clic sur un bouton 🡪 appel de la méthode play sur l’objet Player )

Autres directives sur la Vue

* Évitez de changer les couleurs au cas par cas, utilisez les thèmes, les styles et les StateListDrawables au besoin
* Faites en sorte que tourner le téléphone n’interrompt pas la musique diffusée ( fin de l’**annexe 4** )
* Faites en sorte de redéfinir certaines méthodes du cycle de vie de manière à ce que la musique s’arrête quand on quitte l’app ou qu’une autre app apparaisse ( appel téléphonique par exemple ) et qu’elle reprenne quand on revient sur l’app que vous développez.
* Vous pouvez avoir une ou plusieurs activités, une bien entendue doit être destinée aux contrôles et à diffusion de la musique.

## Vos premiers objectifs

* Lire le fichier JSON avec les chansons ( Atelier 2 )
* Utiliser GSON pour décomposer / recomposer les objets ( Atelier 2B )
* Contrôles de base ( PLAY, SUIVANT, PRÉCÉDENT, PAUSE, AVANCER DE 10 SECONDES … ) ( Atelier 4 )
* Détecter l’obtention de la réponse du modèle dans la vue ( Atelier 5 )

Cette partie ci-haut devrait être terminée au retour de la semaine de relance, soit dans la semaine du 21 octobre

Partie 2 : à venir : cours de jeudi 24 octobre

* Détecter les secondes passées pour faire avancer la seekBar ( Handler, Runnable, Thread ) car il n’y a pas d’écouteur dédié pour cela dans l’API d’ExoPlayer.
* Opération « Seek » pour avancer / reculer dans la chanson avec la SeekBar
* Playlistes
* Autres

Événements sur le ExoPlayer :

<https://developer.android.com/media/media3/exoplayer/listening-to-player-events?hl=fr>