
~~ Rapport de projet ~~

Music Player ViRap

Semestre 5

Développement Mobile

L3 Informatique

Viet NGUYEN – 20006303
Raphaël MOSCATELLI – 21006543

Table des matières

Introduction

Dans le cadre du cours universitaire "Développement Mobile" de troisième année, nous avons conçu et développé "Music Player ViRap", une application Android open source dédiée à la gestion et à la lecture de musique. Cette application, développée en Kotlin, se distingue par sa facilité d'utilisation et ses fonctionnalités innovantes. “ViRap” est composé des premières lettres dans notre prénom: Vi-et et Rap-haël.

"Music Player ViRap" offre aux utilisateurs une expérience immersive de navigation, de lecture et de gestion de leurs fichiers audio locaux sur des appareils Android. Parmi les principales fonctionnalités, nous avons intégré un lecteur de musique intuitif, des commandes de lecture avancées, la création de playlists favorites et personnalisées, des notifications interactives, ainsi qu'un égaliseur sophistiqué pour une expérience audio optimisée.

L'architecture du code de l'application est un modèle de bonnes pratiques de programmation Android, garantissant ainsi une performance et une stabilité exceptionnelles. Notre démarche a été guidée par l'objectif de créer une application à la fois fonctionnelle et esthétiquement agréable.

Ce projet est le fruit d'une collaboration étroite entre deux étudiants, Viet et Raphaël. Outre l'utilisation de GitHub pour le versionnage et la collaboration, Discord a joué un rôle crucial dans notre communication et gestion de projet.

Dans ce rapport, nous aborderons en détail les fonctionnalités de l'application, illustrées par des extraits de code. Nous discuterons également des axes d'amélioration potentiels et expliquerons les raisons qui nous ont poussés à choisir ce projet. Enfin, nous fournirons un lien vers une démonstration pratique de l'utilisation de l'application, témoignant de son efficacité et de son intérêt pour les utilisateurs finaux.

Contexte et Justification

- **Contexte Technologique** : À l'ère actuelle, où la technologie mobile évolue à un rythme effréné, les applications de musique jouent un rôle central dans l'expérience utilisateur quotidienne. Avec une multitude d'applications disponibles, la nécessité de développer une solution plus intuitive et performante pour la lecture de musique sur Android est devenue évidente. Notre choix de développer "MusicPlayerVirap" s'inscrit dans ce contexte, en réponse à la demande croissante d'applications musicales à la fois fiables et conviviales.

- **Justification du Projet** : L'idée de "Music Player ViRap" a germé après une réflexion approfondie sur plusieurs concepts innovants. Initialement, nous avons envisagé des applications centrées sur la photographie et l'animation, mais ces idées, bien que séduisantes, ne répondaient pas pleinement à un besoin utilisateur spécifique. C'est alors que nous avons identifié plusieurs lacunes dans les applications de musique existantes, notamment en termes d'ergonomie et de compatibilité avec différents appareils Android.

Nombre de ces applications présentaient des problèmes tels que des interfaces utilisateur peu intuitives, des problèmes de performance sonore variables selon les appareils, et des baisses soudaines de volume. Ces observations nous ont conduit à concevoir une application de musique qui non seulement résout ces problèmes, mais offre également une expérience utilisateur optimale.

Notre ambition était de créer une application de lecteur de musique qui soit ergonomique, simple d'utilisation, et adaptée à une large gamme d'appareils Android, tout en nécessitant peu de ressources système. En somme, "Music Player ViRap" vise à offrir une solution élégante et efficace

pour la lecture de musique, répondant ainsi aux attentes des utilisateurs modernes.

Objectifs du Projet

L'objectif principal de ce projet est de développer une application de lecteur de musique pour appareils Android, offrant une expérience d'écoute hors ligne, gratuite et sans publicités. Cette application intégrera toutes les fonctionnalités essentielles des applications de musique modernes, tout en fournissant une interface utilisateur intuitive et conviviale. Les objectifs spécifiques du projet "Music Player ViRap" sont les suivants :

- **Développement d'une Interface Utilisateur Efficace et Conviviale** :
  - La page d'accueil présentera une liste complète de tous les fichiers musicaux disponibles sur l'appareil de l'utilisateur, accompagnée de boutons permettant d'accéder aux pages "Favoris" et "Playlist d'Albums".
  - Une barre de défilement sera disponible pour naviguer facilement dans une longue liste de chansons.
  - Une fonction de recherche permettra aux utilisateurs de trouver rapidement des chansons par nom.

- **Création de Fonctionnalités Spécifiques pour l'Amélioration de l'Expérience Utilisateur** :
  - La page "Favoris" stockera les chansons marquées comme préférées par l'utilisateur.

- La page "Playlist" permettra aux utilisateurs de créer des albums personnalisés pour organiser leurs chansons préférées en fonction de divers thèmes ou humeurs (comme des playlists pour la relaxation, les fêtes, ou pour des moments de stress).

- Le "Lecteur de Musique" s'activera lorsqu'une chanson est sélectionnée, affichant l'image de l'album si disponible, et offrira des contrôles de lecture standards (Play/Pause, Next & Previous, favoris, répétition automatique, minuterie, égaliseur, et partage).

- **Intégration de Fonctionnalités Supplémentaires** :
  - Une page de paramètres pour personnaliser l'interface, comme le changement de couleur de fond et l'organisation des chansons.
  - Une page de retour d'information pour permettre aux utilisateurs d'envoyer directement leurs commentaires et suggestions par courriel.

En atteignant ces objectifs, ce projet vise à offrir une solution complète et personnalisable pour l'écoute de musique sur les appareils Android, répondant aux besoins et préférences des utilisateurs modernes.

Méthodologie

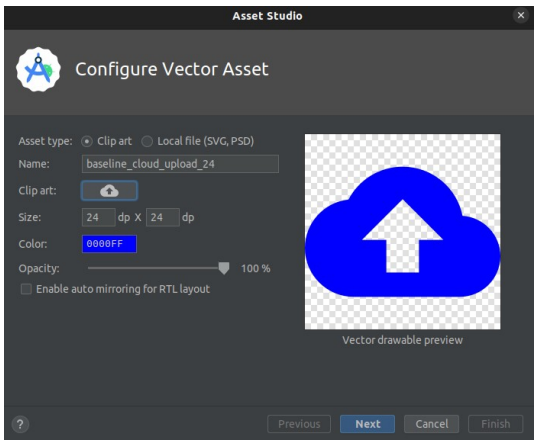
Dans le développement de notre application “Music Player ViRap” pour Android, nous avons adopté une approche méthodique et structurée, détaillée comme suit :

Création d'un Répertoire sur GitHub :

- Pour faciliter la collaboration, un repo GitHub a été établi dès le début du projet. (Lien: https://github.com/Viet281101/MusicPlayerVirap)
- Les premières étapes ont impliqué la création de maquettes pour l'agencement des pages, en utilisant LibreSprite pour esquisser les interfaces et discuter de la disposition des boutons.

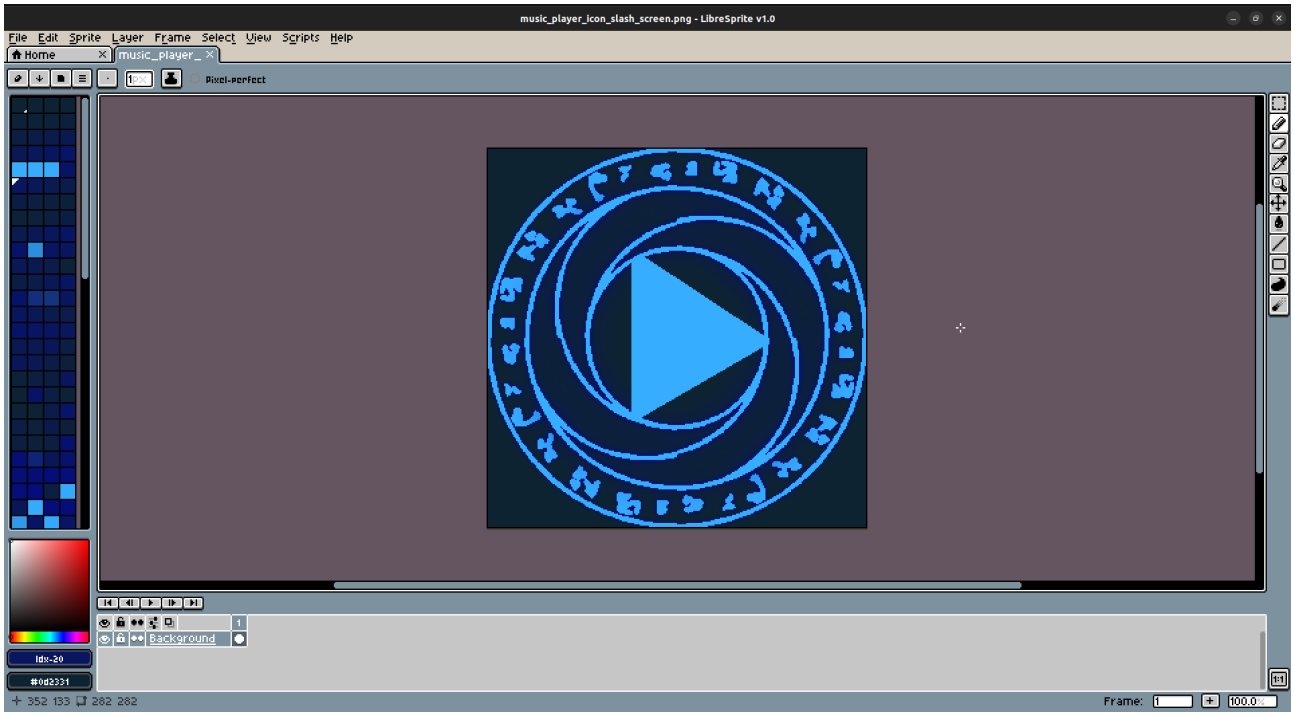
Conception des Icônes et de l'Interface :

- Au lieu de rechercher des icônes gratuites, nous avons opté pour l'utilisation des icônes Vector Assets intégrées dans Android Studio.



- La couleur principale choisie pour les icônes et l'interface est le bleu océan. Des options supplémentaires de personnalisation, y compris des gradients et des animations, seront envisagées pour les versions futures.

- La création de l'icône principale de l'application a été confiée à Viet, qui l'a conçue avec LibreSprite. Cette icône sert également pour les fichiers musicaux sans image.



Mise en Place de l'Architecture du Code :

- Pour chaque fonctionnalité ou page de l'application, une activité "Empty Views Activity" a été créée dans Android Studio, générant des fichiers Activity.kt et activity.xml correspondants. Il ajoute aussi des <activity> dans le fichier AndroidManifest.xml .
- Cette méthode simplifie la gestion des différentes sections de l'application et facilite la répartition des tâches entre les membres de l'équipe.

Intégration des Fichiers Musicaux :

- L'ajout de fichiers musicaux dans l'application se fait via l'utilisation de Uri dans Kotlin pour accéder à l'explorateur de fichiers des téléphones et afficher des fichiers tels que mp3, wav, ogg, etc.

- La permission de l'utilisateur est demandée pour accéder aux fichiers, et une fois accordée, donc requestRuntimePermission() = true et l'application affiche les chansons disponibles.
- Raphael a été chargé de l'implémentation de cette fonctionnalité, tandis que Viet a corrigé des erreurs et conçu l'interface utilisateur pour afficher ces fichiers en liste.

Développement des Fonctionnalités des Boutons :

- Chaque bouton dans l'interface a été assigné un identifiant unique dans les fichiers activity.xml. Par exemple: android:id="@+id/favoriteBtn"
- Les actions des boutons ont été programmées en utilisant la syntaxe standard binding.id.setOnClickListener{ code }.

Gestion des Playlists et Favoris :

- Pour les fonctionnalités des playlists et des favoris, des ArrayList<> ont été crées pour stocker les sélections musicales des utilisateurs.
- Les chansons ajoutées à ces listes sont gérées à l'aide de la bibliothèque Glide pour le chargement et l'affichage.

Fonction de Recherche de Musique :

- Une liste spécifique musicListSearch: ArrayList<Music> a été créée pour gérer la fonction de recherche, ajoutant des chansons dont les titres correspondent dans cette liste à la saisie de l'utilisateur.

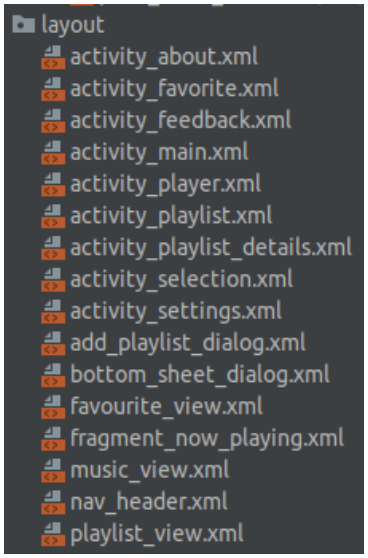
Cette méthodologie couvre plusieurs parties de notre approche pour le développement de l'application “Music Player ViRap”, un projet impliquant une série de choix techniques et de méthodes de travail collaboratif pour atteindre nos objectifs.

Développement

Le développement de notre application “Music Player ViRap” pour Android a été exécuté avec une attention particulière à l'architecture du système et aux fonctionnalités clés, comme suit :

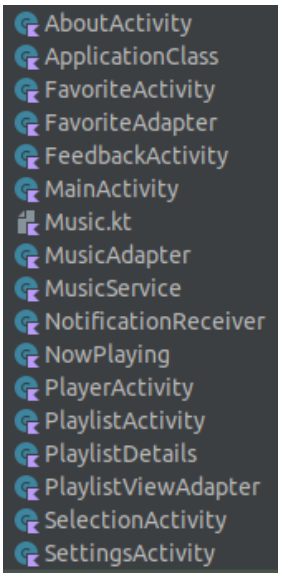
Architecture du système :

- Chaque page de l'application correspond à une activité "Empty Views Activity" distincte dans Android Studio, générant deux fichiers essentiels : un fichier Kotlin pour la logique fonctionnelle et un fichier XML pour l'interface utilisateur.
- Le processus a commencé avec MainActivity, la page principale affichant la liste globale de toutes les chansons, servant de base pour l'intégration d'autres pages.



Organisation des Fichiers:

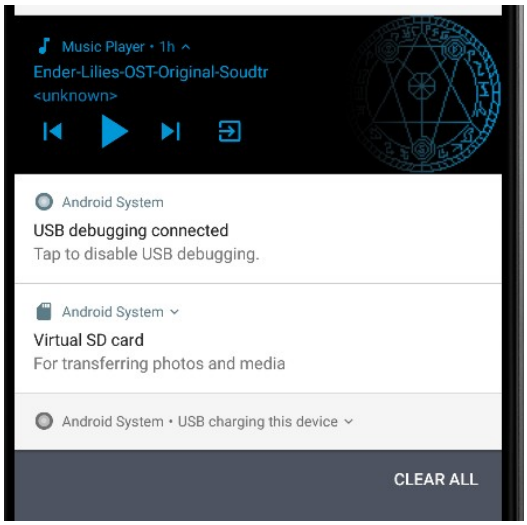
- Fichiers XML (app/src/main/res/layout/*) : Chaque fichier XML, comme **activity_main.xml**, **activity_player.xml**, **activity_playlist.xml**, **activity_favorite.xml**, etc., a été soigneusement conçu pour structurer les interfaces des différentes pages, y compris les listes de chansons, les boutons, et les vues de playlist.



- Fichiers Kotlin :
  - Des fichiers tels que **MainActivity.kt**, **PlayerActivity.kt**, **PlaylistActivity.kt**, et **FavoriteActivity.kt** contiennent la logique fonctionnelle des pages respectives, y compris la gestion des listes de chansons, les fonctionnalités de lecture, et les interactions utilisateur.

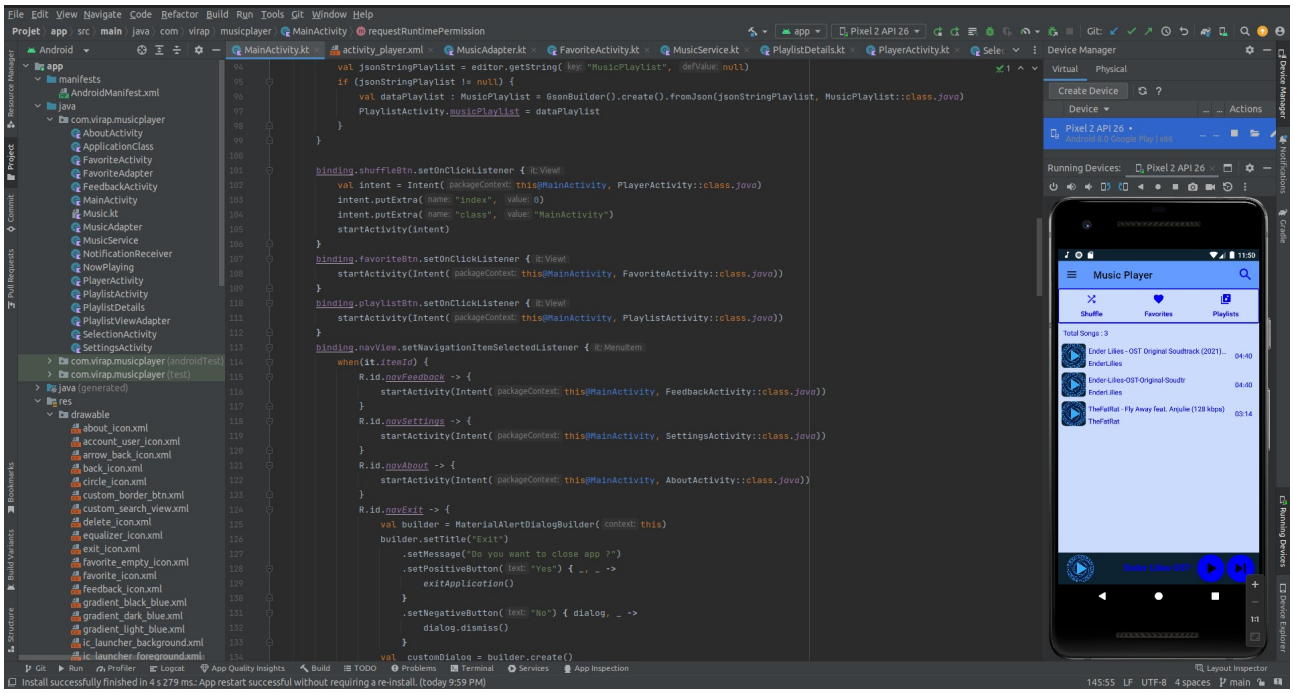
  - Des fichiers supplémentaires comme **Music.kt**, **MusicAdapter.kt**, **MusicService.kt**,

PlaylistViewAdapter.kt, et **FavoriteAdapter.kt** définissent et gèrent les opérations liées aux chansons, playlists, et favoris.



- Le fichier **NotificationReceiver.kt** est dédié à la gestion des notifications de lecture sur les appareils.

- Gestion des Ressources et Thèmes :
  - Des fichiers tels que **colors.xml**, **strings.xml**, et **themes.xml** ont été utilisés pour centraliser la gestion des couleurs, des chaînes de caractères, et des thèmes de l'application, assurant une cohérence visuelle et facilitant les mises à jour futures.



Fonctionnalités clés :

Brève explication du code dans certaines fichiers du projet :

MainActivity.kt :

Elle offre une gestion avancée des permissions pour accéder et gérer les fichiers audio. La navigation intuitive est facilitée par un menu latéral et des boutons dédiés pour les fonctionnalités principales comme les playlists, les favoris, et le mode aléatoire. L'application prend en charge divers formats audio et assure un classement efficace des pistes selon différents critères (date d'ajout, titre, taille). L'interface utilisateur est hautement personnalisable, avec des options pour changer les thèmes et les couleurs de fond. De plus, une barre de recherche intégrée permet aux utilisateurs de trouver rapidement des chansons spécifiques. Enfin, l'application gère la persistance des données des playlists et des favoris, assurant une expérience utilisateur cohérente et personnalisée.

PlayerActivity.kt :

L'activité 'PlayerActivity' est au cœur de l'expérience utilisateur de “Music Player ViRap”, offrant une interface interactive pour la lecture de musique. Cette classe gère la lecture des pistes audio, les interactions avec l'utilisateur et l'intégration avec le service de musique en arrière-plan.

- La fonction onCreate() initialise l'interface utilisateur et configure les interactions. Si l'intention de démarrage contient des données audio, elles sont récupérées et utilisées pour préparer la lecture. Les boutons pour la lecture, la pause, le morceau précédent et suivant, ainsi que la barre de progression de la lecture, sont configurés pour répondre aux actions de l'utilisateur.

- La gestion de la liste de lecture est effectuée via musicListPA, une liste dynamique de l'objet Music. Les fonctions playMusic et pauseMusic contrôlent le flux de musique en utilisant le service de musique, et prevNextSong gère le passage à la chanson suivante ou précédente.

- L'interaction avec l'égaliseur audio est également intégrée, permettant à l'utilisateur de personnaliser les effets sonores via l'intention

AudioEffect.ACTION_DISPLAY_AUDIO_EFFECT_CONTROL_PANEL.

- Un aspect unique de cette activité est la gestion de minuteries pour la fin automatique de la lecture, via showBottomSheetDialog, offrant des options pour arrêter la musique après un temps spécifié.

- L'activité interagit étroitement avec MusicService pour maintenir la lecture en arrière-plan, gérer la barre de progression, et assurer une expérience utilisateur fluide et continue, même lorsque l'application n'est pas au premier plan.

- Enfin, PlayerActivity intègre des options de partage et de gestion des favoris, permettant aux utilisateurs de partager des pistes audio et de marquer des chansons comme favorites, renforçant ainsi l'aspect social et personnalisable de l'expérience utilisateur.

Music.kt :

Encapsulant les modèles de données et les fonctions utilitaires pour la gestion de la musique.

- La classe Music est une structure de données représentant une piste musicale, avec des champs pour l'ID, le titre, l'album, l'artiste, la durée, le chemin d'accès et l'URI de l'illustration. Cette définition de données facilite la manipulation et l'affichage des informations de musique dans l'application.

- Playlist et MusicPlaylist sont des classes pour organiser les pistes musicales. Playlist représente une liste de lecture individuelle, tandis que MusicPlaylist sert de conteneur pour plusieurs playlists, permettant une organisation et une gestion efficaces des collections de musique.

- La fonction formatDuration() convertit la durée de la musique en millisecondes en un format lisible (minutes:secondes), essentiel pour l'affichage de la durée des pistes dans l'interface utilisateur.

- getImgArt extrait les données d'image intégrées à partir du chemin de la musique, utilisant MediaMetadataRetriever. Cette fonction est utilisée pour récupérer les illustrations des albums pour l'affichage.

- setSongPosition ajuste la position de la piste en cours de lecture dans la liste, en tenant compte du mode répétition. Cette fonction est cruciale pour la navigation entre les pistes.

- exitApplication gère la fermeture de l'application et la libération des ressources audio. Elle est appelée lors de la fermeture de l'application pour assurer une sortie propre.

- favouriteChecker vérifie si une piste musicale donnée fait partie des favoris, en fournissant une intégration avec la fonctionnalité de favoris de l'application.

- Enfin, checkPlaylist valide l'existence des fichiers musicaux dans une liste de lecture, garantissant que toutes les pistes disponibles sont jouables.

MusicAdapter.kt :

Le MusicAdapter est un composant essentiel dans MusicPlayerVirap, facilitant l'affichage et la gestion des pistes musicales dans une vue de liste recyclable. Cette classe étend RecyclerView.Adapter et est utilisée pour relier les données de musique à une vue recyclable, permettant ainsi un défilement efficace et une utilisation optimisée de la mémoire.

- Chaque instance de MusicAdapter est initialisée avec une liste de pistes musicales (musicList) et des paramètres pour gérer des cas spécifiques comme les détails d'une playlist (playlistDetails) ou une activité de sélection (selectionActivity).

- La fonction onCreateViewHolder crée une nouvelle vue pour chaque élément de la liste de musique. onBindViewHolder lie

chaque piste musicale de musicList à une vue, en définissant le titre, l'album, la durée et l'image de la piste.

- Les interactions utilisateur avec chaque élément de la liste sont gérées dans onBindViewHolder. Selon le contexte (détails de playlist, activité de sélection, ou recherche), des intentions différentes sont envoyées pour initier PlayerActivity avec la piste sélectionnée.

- updateMusicList et refreshPlaylist sont des fonctions pour mettre à jour la liste de musique affichée dans l'adapter, assurant que l'interface utilisateur reflète les dernières modifications des données.

MusicService.kt :

Le MusicService est un service crucial dans MusicPlayerVirap, gérant la lecture de musique en arrière-plan. Ce service étend Service et implémente AudioManager.OnAudioFocusChangeListener pour une gestion efficace des ressources audio.

- Le service utilise MediaPlayer pour la lecture de musique et MediaSessionCompat pour la gestion de session média. La fonction showNotification crée et affiche une notification personnalisée, permettant le contrôle de la musique depuis l'écran de verrouillage ou la barre de notifications.
- La liaison du service (onBind) retourne un Binder personnalisé, fournissant un accès direct au service depuis d'autres composants de l'application.
- createMediaPlayer initialise et prépare MediaPlayer avec la piste musicale sélectionnée, gérant également l'affichage des informations de la piste et la mise à jour de la barre de progression.

- seekBarSetup met en place un Runnable pour mettre à jour régulièrement la position de la barre de progression en fonction de la position actuelle de la musique.
- Enfin, onAudioFocusChange gère les changements de focus audio, mettant en pause ou reprenant la lecture en fonction des interruptions du système (comme les appels entrants).

NotificationReceiver.kt :

Le NotificationReceiver est un élément clé dans la gestion des actions de contrôle de la musique à partir des notifications. Ce BroadcastReceiver réagit aux actions déclenchées par l'utilisateur depuis la notification de lecture musicale.

- Lorsque l'utilisateur sélectionne une action (précédent, lecture/pause, suivant, quitter), onReceive intercepte cette action et appelle la fonction correspondante dans l'application. Les actions de lecture et de pause utilisent playMusic et pauseMusic pour contrôler le flux de musique.
- Les fonctions prevNextSong gèrent le passage à la chanson précédente ou suivante, en mettant à jour la position de la chanson et en relançant la lecture.
- Ce mécanisme permet une interaction fluide avec la barre de notifications, offrant à l'utilisateur un contrôle pratique sur la lecture de musique, même lorsque l'application n'est pas au premier plan.

Conclusion

Les fichiers analysés ci-dessus représentent des composants clés du projet MusicPlayerVirap, démontrant l'architecture soignée et la logique cohérente de l'application. Bien que ces fichiers soient cruciaux, il est important de noter que le projet comprend également d'autres fichiers aux fonctions complémentaires, suivant une logique similaire, ou remplissant des rôles périphériques sans impacter l'essence même du projet.

Tests et Validation

Dans le cadre du développement de notre application Music Player ViRap, une série de tests a été méticuleusement planifiée et exécutée pour s'assurer de la qualité et de la fonctionnalité de l'application. Les détails sont les suivants :

Stratégie de Test :

- En raison de l'absence de dispositifs Android physiques parmi les membres de l'équipe, nous avons opté pour l'utilisation de l'émulateur de dispositif virtuel disponible dans Android Studio.
- Le dispositif choisi pour ces tests était le Google Pixel 2 API 26. Ce choix n'a pas été motivé par des raisons spécifiques, mais plutôt par sa disponibilité immédiate suite à l'installation d'Android Studio.

Résultats des Tests :

- L'application s'est avérée fonctionner de manière satisfaisante sur l'émulateur Google Pixel 2 API 26. Les fonctionnalités clés, y compris la navigation entre les différentes pages, la gestion des fichiers musicaux, et l'utilisation des boutons, ont été testées avec succès.
- Cependant, nous reconnaissons que des tests supplémentaires sur d'autres appareils Android sont nécessaires pour assurer une compatibilité et une performance optimales sur une variété de dispositifs et de versions d'Android.

Ces tests initiaux ont fourni des indications précieuses sur la stabilité et l'efficacité de l'application. Toutefois, une phase de test plus approfondie est envisagée pour garantir une expérience utilisateur fluide et sans faille sur une plus grande diversité d'appareils Android.

Conclusion et Perspectives d’Avenir

Le projet “Music Player ViRap” s'est avéré être une initiative enrichissante et instructive, débouchant sur la création d'une application de lecteur de musique efficace et bien conçue pour les appareils Android. Parmi les réalisations notables, nous avons :

Résumé des réalisations :

- La capacité de lire et de gérer des fichiers musicaux stockés localement.
- Une interface utilisateur ergonomique, facilitant la navigation entre les pistes et les albums.
- La possibilité de créer et d'éditer des playlists personnalisées.
- Une fonctionnalité pour marquer rapidement les pistes préférées.
- L'intégration de notifications et d'un mode veille.
- Un égaliseur intégré avec divers préréglages.
- Une architecture de code modulaire et extensible.
- La mise en place d'une approche de développement ouvert et collaboratif sur GitHub.

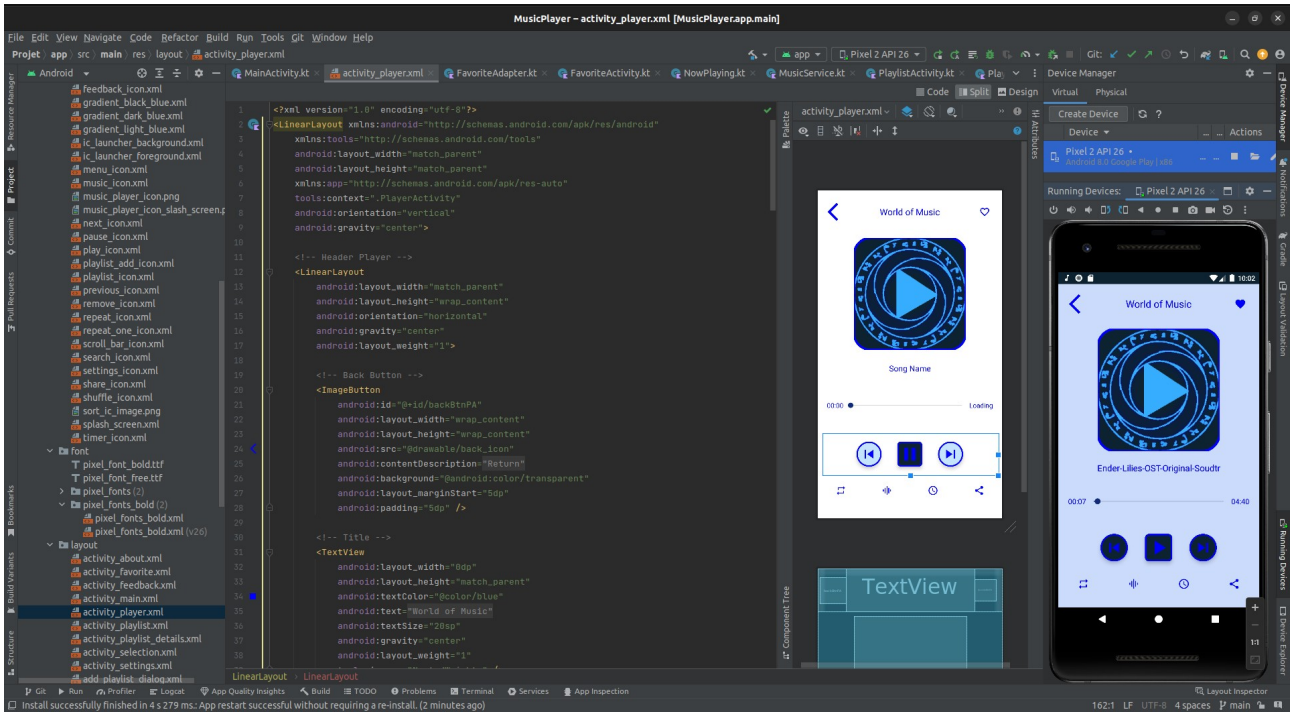
Limitations et améliorations futures :

Toutefois, ce projet a également révélé des domaines nécessitant des améliorations et des développements futurs. Parmi ceux-ci, l'ajout de fonctionnalités uniques pour distinguer notre application sur le marché concurrentiel des lecteurs de musique est primordial. De plus, l'expérience acquise dans le cadre de ce projet a mis en lumière la complexité et les défis inhérents au développement sur Android Studio, en particulier pour une équipe avec une expérience limitée dans ce domaine.

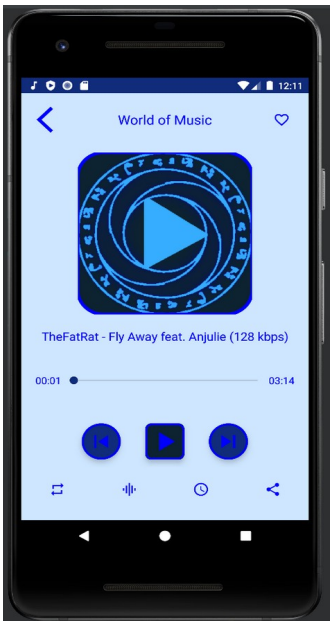
En perspective, nous envisageons d'approfondir notre expertise en développement Android et d'explorer de nouvelles fonctionnalités pour améliorer et enrichir l'expérience utilisateur. Ce projet a ouvert la voie à de

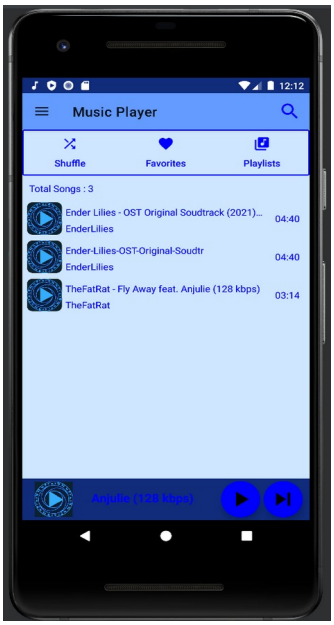
futures explorations dans le domaine des applications mobiles, et nous sommes impatients de poursuivre cette aventure passionnante.

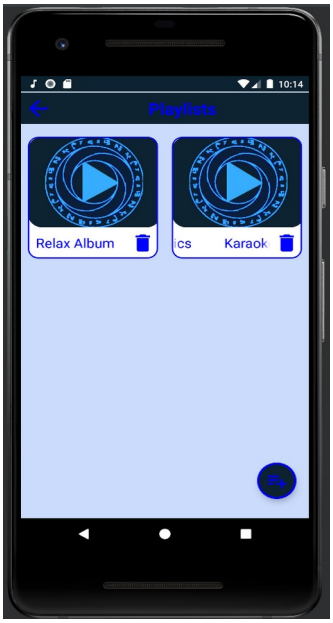
Annexes



- Debug :







Références

- Sur Kotlin Docs:
  - https://kotlinlang.org/docs/home.html
- Sur YouTube:
  - Create Music Player App | Android Studio | Kotlin | 2023
  - How to create a Splash Screen in Android Studio (Kotlin 2020)