TP service: Android
TI et IQL
S5 – 2021

Objectif :

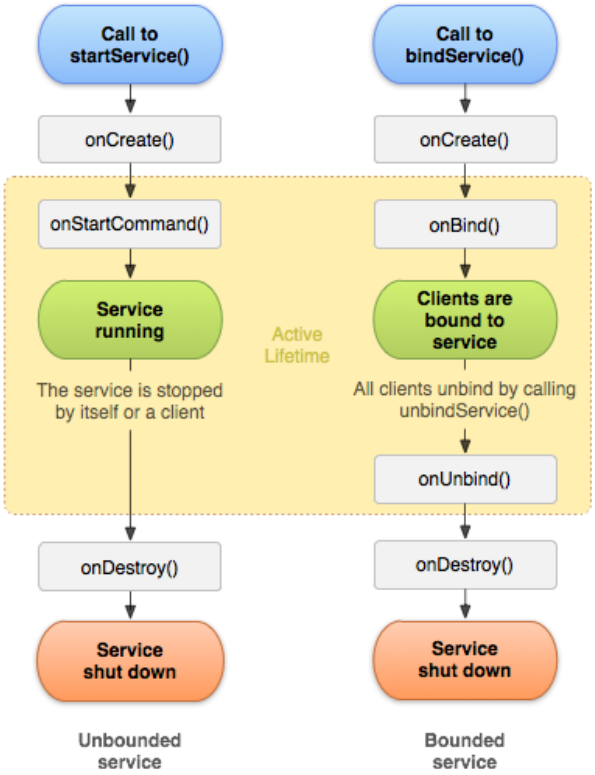
- Implémentation et interaction avec un service
- Utilisation de la classe Media Player
- Utilisation de la liste comme élément de l'interface graphique
- Le développement doit se faire en Natif (java ou Kotlin)
- Tout ajout personnel sera considéré comme bonus. Soyez inventifs. L'énoncé du TP est à titre indicatif.
- Votre projet doit contenir les fichiers de chaque étape du TP

Introduction (à lire) :

- **Service :** Un service est un élément Android qui hérite de la classe Service et qui n'a pas d'interface graphique associée. Le service s'exécute en arrière-plan et peut interagir avec une activité ou un autre élément.

  Une Activité peut créer et démarrer son propre service ou bien elle peut se connecter (Bind) à un service existant.

  Le cycle de vie d'un service dépend de comment on le sollicite : en le démarrant ou on s'y connectant. Voir le schéma suivant :



Lorsque le service est démarré `startService(Intent)`, il s'exécutera **indéfiniment** (ou jusqu'à ce que le système manque de ressources et il le tuera). Il faut donc penser à l'arrêter. Un service peut être arrêté soit par lui-même : `stopSelf()` ou par un autre élément en appelant `stopService(Intent)`

- **Media Player** : MediaPlayer est une classe qui va permettre à l'utilisateur de faire jouer/contrôler un fichier audio/vidéo ou stream. Pour ce TP nous l'utiliseront pour jouer de la musique (audio). Les méthodes importantes de la classe MediaPlayer sont :
  - La méthode statique **create(this, IDfichierAudio)** avec **IDfichierAudio=R.raw.*VotreFichierAudio*** permet de créer un objet *MediaPlayer* pour le fichier *VotreFichierAudio* stocké dans le répertoire res/raw de votre projet.
  - La méthode **setLooping(Boolean)** permet de spécifier si la répétition (c-à-d, faire rejouer) est permise ou pas sur l'objet MediaPlayer.
  - La méthode **release()** permet de libérer toutes les ressources liées l'objet MediaPlayer.
  - La méthode **start()** permet de lancer et démarrer l'objet MediaPlayer.
  - La méthode **stop()** permet d'arrêter le Media player
  - La méthode **pause()** permet de suspendre momentanément l'objet MediaPlayer
  - La méthode **isPlaying()** vérifie si le media player est déjà en marche ou pas
  - La méthode **reset()** permet de réinitialiser l'objet MediaPlayer.

Travail à faire :

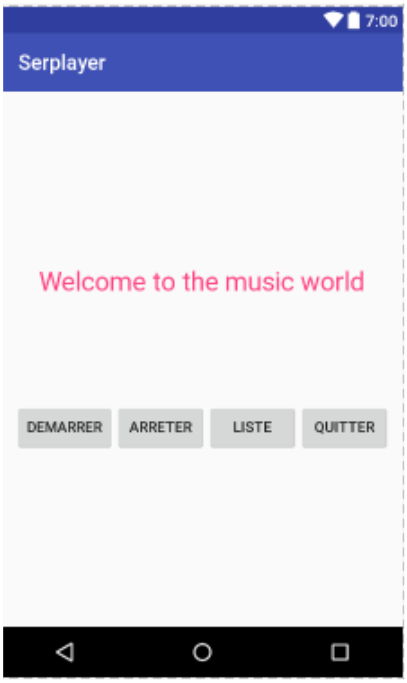
Implémenter une application avec une liste de chansons. En choisissant une chanson de la liste un service démarre et joue la musique correspondante.

Créez un projet TP4_Player avec une activité « HomeActivite » contenant 3 boutons : Jouer, arrêter, Quitter et Liste.

Etape 1 :

Dans cette étape nous implémenterons d'abord les boutons : Quitter, Jouer et arrêter.

- Le bouton Quitter : terminer l'application.
- Le bouton Démarrer : démarre un service qui joue un fichier mp3 en particulier
- Le bouton Arrêter : Arrête la musique
- Le bouton Liste : permet de démarrer l'activité MyPlayList de l'étape 3 (voir plus bas). Ce bouton ne sera pas traité pour le moment.



Ensuite, créez un service « ServiceMusique » qui permet de jouer un fichier mp3 en arrière-plan. Pour créer un service : Bouton droit sur le projet → new → Service → Service

Ps. Un IntentService c'est un service qui démarre dans un Thread.

Le fichier mp3 sera placé dans le répertoire **res/raw**.

Le service sera contrôlé à partir de l'activité « HomeActivite » : Le bouton *Démarrer* fera jouer le son, le bouton *Arrêter* arrêtera le son et *Quitter* pour terminer l'application.

Squelette de votre classe service :

```
public class ServiceMusique extends Service {
            MediaPlayer player = new MediaPlayer();
      @Override
      public IBinder onBind(Intent intent) {
        throw new UnsupportedOperationException("Not yet implemented");
      }

      @Override
      public void onCreate() {
            Toast.makeText(this, "Service Créé",Toast.LENGTH_LONG).show();
      }

      @Override
      public void onDestroy() {
            Toast.makeText(this, "Service détruit",Toast.LENGTH_LONG).show();
      }

      @Override
        public int onStartCommand(Intent intent, int flags, int startId) {
            Toast.makeText(this, "Service démmaré",Toast.LENGTH_LONG).show();
            }
}}
```

cycle de vie du service. C'est dans le service qu'il faut manipuler la classe MediaPlayer. **Le fichier à jouer sera codé directement et explicitement dans le service.**

Pour lancer le service à partir d'une activité, on utilise la méthode **startService()** avec un objet Intent comme paramètre. De même, pour qu'une activité arrête un service elle doit appeler la méthode **stopService()** avec un objet Intent comme paramètre.

Etape 2 :

Modifier votre application pour que le nom du fichier musique soit transmis par l'activité au service. Le service doit recevoir un nom de fichier, le convertir en ID et démarrer le *Player* avec ce fichier.

En cliquant sur le bouton *Démarrer* l'activité va envoyer un nom donné de fichier mp3 au service qui jouera la musique correspondante.

Le bouton *Arrêter* arrêtera la musique.

Utile : Pour obtenir un ID à partir d'un nom de fichier et son répertoire :

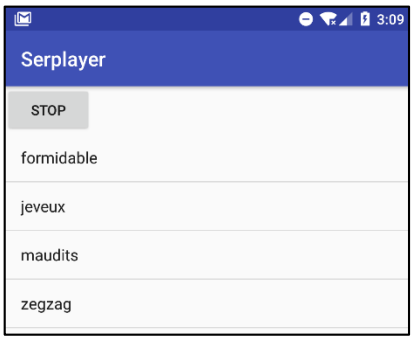
`getResources().getIdentifier(nom_fichier, répertoire, getPackageName());`

Retourne l'ID du fichier *nom_fichier*, en principe raw est le repertoire.

Etape 3 :

Objectif : Utiliser une ListView pour lister un ensemble de titres de musique et réagir au clic en jouant le fichier correspondant.

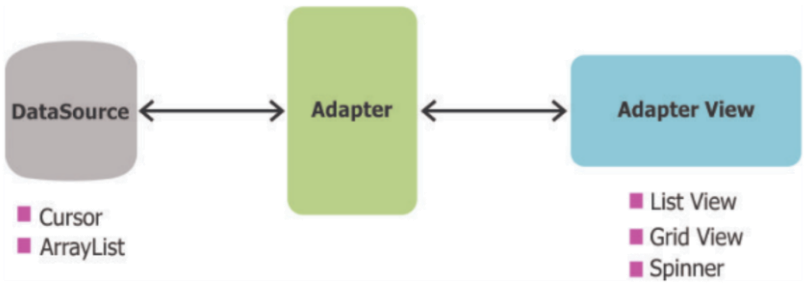
Créer une activité **MyPlayList.java** qui contient une ListView et un Bouton Stop pour arrêter la musique. Elle affichera une liste de titres donnés en entrée sous forme de tableau : `String songs[] = {}`



Le bouton **Liste** de **HomeActivite** permet de démarrer **MyPlayList**

Ajouter le nécessaire dans votre projet pour jouer le fichier musique correspondant au choix cliqué dans la liste.

Pour afficher dans une liste il faut utiliser un adapter



ArrayAdapter<String> ***name*** = new ArrayAdapter<String>(***activity***, ***layout***, ***array***);

Layout = le layout de la liste (chaque ligne) – layout prédéfini :

android.R.layout.simple_list_item_1

Array = tableau source de données

Ne pas oublier de lier l'adapter à la liste:

ListView mylist = (ListView) findViewById(R.id.***maList***);

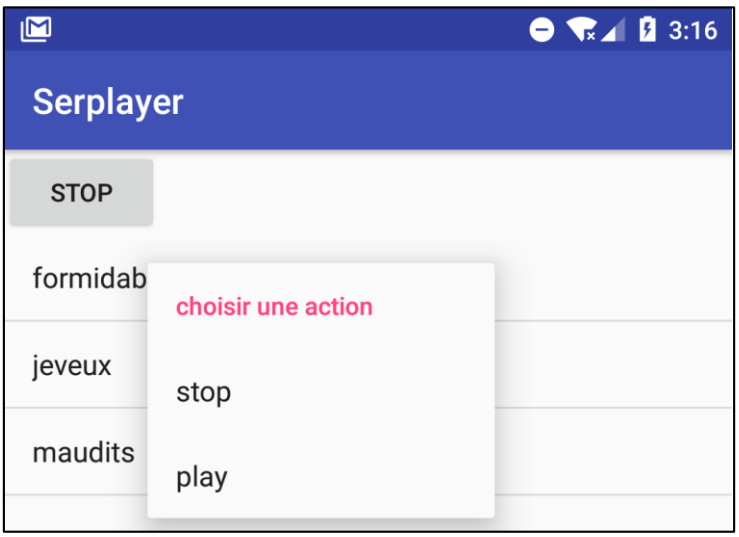
mylist.setAdapter(myAdapter);

Pour répondre au clic :

```
mylist.setOnItemClickListener(new AdapterView.OnItemClickListener() {
   @Override
   public void onItemClick(AdapterView<?>      parent, View view, int pos, long id) {
     String p = parent.getItemAtPosition(pos).toString();  //element cliqué
    }
});
```

Etape 4 : Ajouter un Menu contextuel à la ListView

Nous aimerions ajouter un menu contextuel lors d'un long clic sur un élément de la liste qui permet de faire deux actions : arrêter le fichier sur lequel on a cliqué ou bien jouer le fichier sur lequel on a cliqué



Pour cela il faut :

a. Créer un menu : bouton droit sur le répertoire menu → new → menu resource file
Et y ajouter ce contenu :

```
<?xml version="1.0" encoding="utf-8"?>
<menu xmlns:android="http://schemas.android.com/apk/res/android">
    <item android:id="@+id/stop"
          android:title="stop"/>
    <item android:id="@+id/play"
          android:title="play"/>
</menu>
```

b. Enregistrer votre liste pour qu'elle réponde au menu contextuel :

```
registerForContextMenu(nom_de_la_liste);
```

c. Redéfinir la méthode suivante qui permet de créer le menu contextuel suite à un long clic :

```
@Override
public void onCreateContextMenu(ContextMenu menu, View v,
ContextMenu.ContextMenuInfo menuInfo) {

    super.onCreateContextMenu(menu, v, menuInfo);

    menu.setHeaderTitle("Mettez Votre Titre ici");
    MenuInflater inflater = getMenuInflater();
    inflater.inflate(R.menu.nom_fichier_menu, menu);
}
```

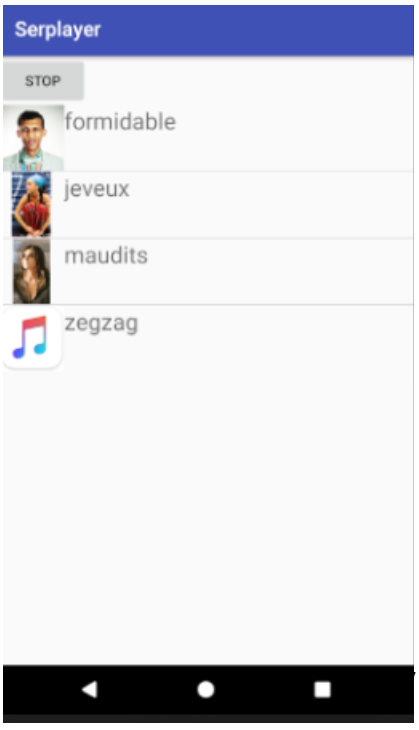
d. Redéfinir la méthode suivante qui permet de spécifier le comportement de l'application lors du clic sur un des éléments du menu (**à Compléter**)

```
@Override
public boolean onContextItemSelected(MenuItem item) {

//Obtenir le titre de la chanson objet du long clic

  AdapterView.AdapterContextMenuInfo info =
(AdapterView.AdapterContextMenuInfo) item.getMenuInfo();

String title = votre_liste.getItemAtPosition(info.position).toString();

    switch (item.getItemId()) {
        case R.id.stop:
            // do something useful


            return true;

              case R.id.play:
               // do something interesting

                  return true;

              default:
                  return super.onContextItemSelected(item);
        }
    }
```

Etape 5 : Modifier votre liste simple en liste complexe

Nous aimerions donner plus d'esthétique à la liste. Chaque ligne de la liste devra contenir le titre de la chanson et une image de l'interprète et le titre de la chanson/album.

Créer le Layout et l'Adapter correspondants pour modifier l'affichage de la liste.



1- Il faut créer un layout correspondant à une ligne

de la liste : une imageView et un TextView

2- Créer une classe monAdaptateur qui hérite de
la classe ArrayAdapter et redéfinir sa méthode
getView() qui permer de positionner chaque élément de données dans sa place (dans la liste)