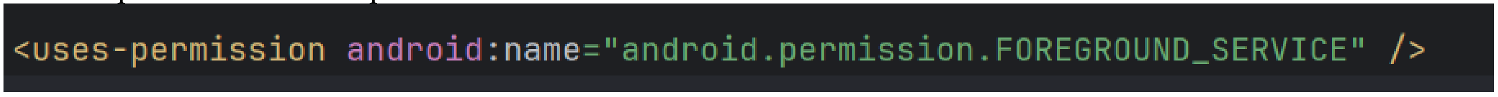
Ćwiczenia 23 – Android studio – Service

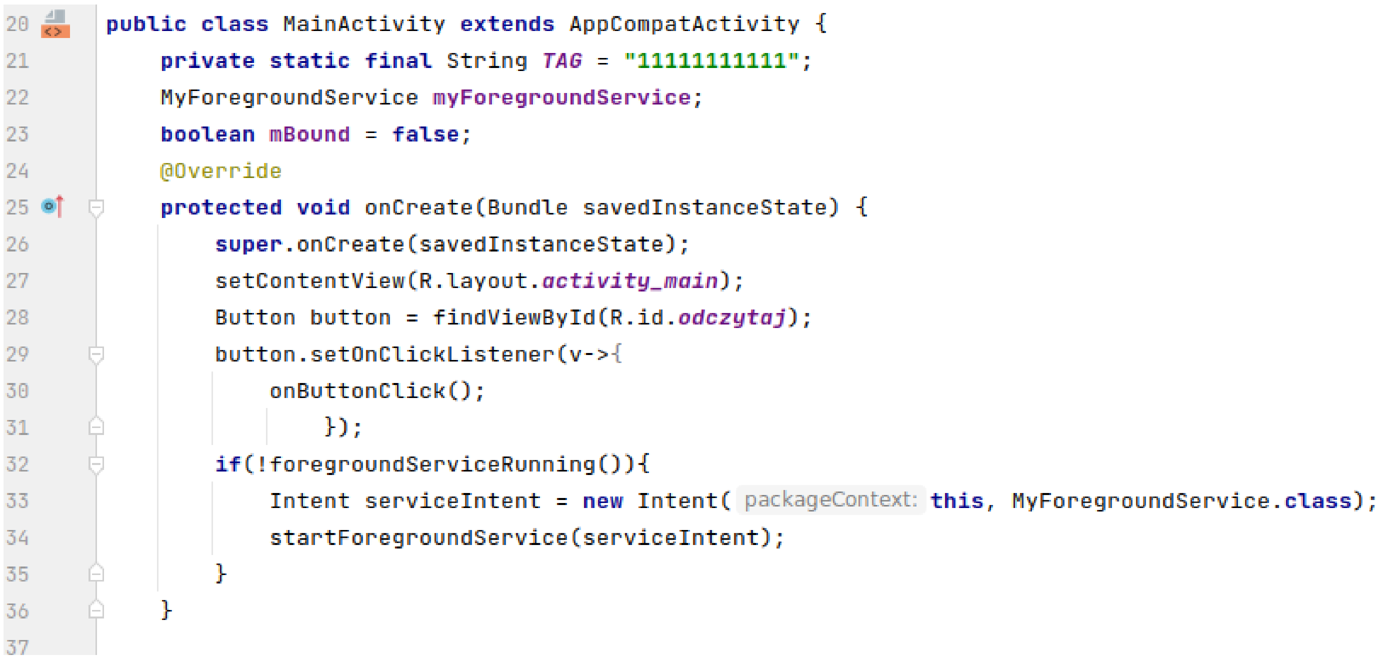
Na koniec zajęć prześlij pliki źródłowe (.xml, .java)+ obrazek do zasobu w teams.

1. Utwórz projekt o nazwie MySimpleServiceMusic na podstawie Empty Activity, dobierz odpowiednie API (28 – Android 9).
2. Otwórz dokumentację:
   https://developer.android.com/guide/components/foreground-services
   https://developer.android.com/reference/android/app/PendingIntent
3. Utwórz nowy service o nazwie MyForegroundService, New → Service → Service :
4. Sprawdź obecność wpisów w AndroidManifest.xml:



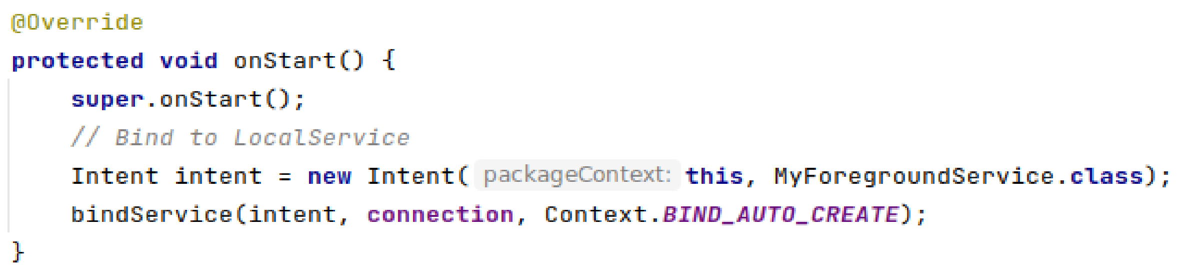
```xml
<uses-permission android:name="android.permission.FOREGROUND_SERVICE" />
```

5. W MainActivity:



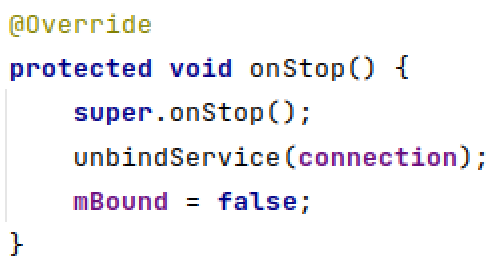
```java
public class MainActivity extends AppCompatActivity {
    private static final String TAG = "11111111111";
    MyForegroundService myForegroundService;
    boolean mBound = false;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        Button button = findViewById(R.id.odczytaj);
        button.setOnClickListener(v->{
            onButtonClick();
                });
        if(!foregroundServiceRunning()){
            Intent serviceIntent = new Intent( packageContext: this, MyForegroundService.class);
            startForegroundService(serviceIntent);
        }
    }
```

6. W metodzie onStart():



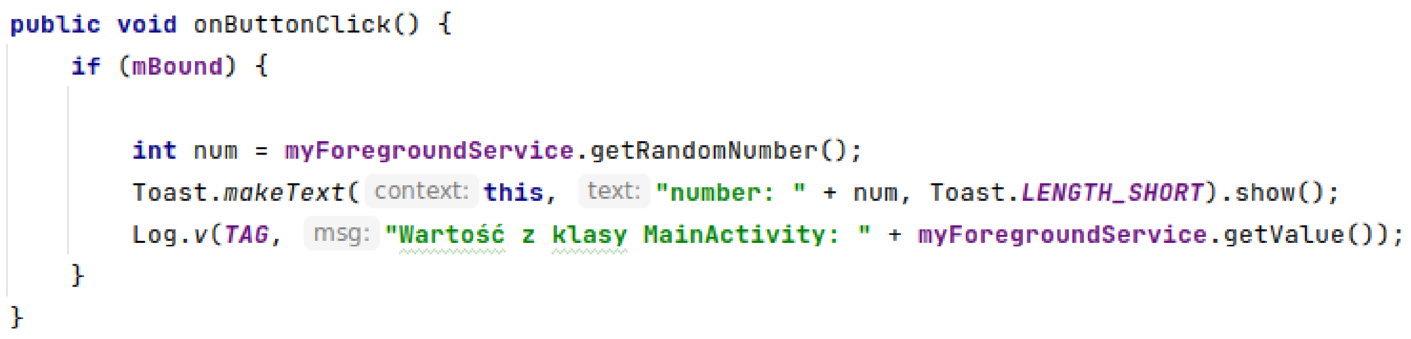
```java
@Override
protected void onStart() {
    super.onStart();
    // Bind to LocalService
    Intent intent = new Intent( packageContext: this, MyForegroundService.class);
    bindService(intent, connection, Context.BIND_AUTO_CREATE);
}
```

7. W metodzie onStop():



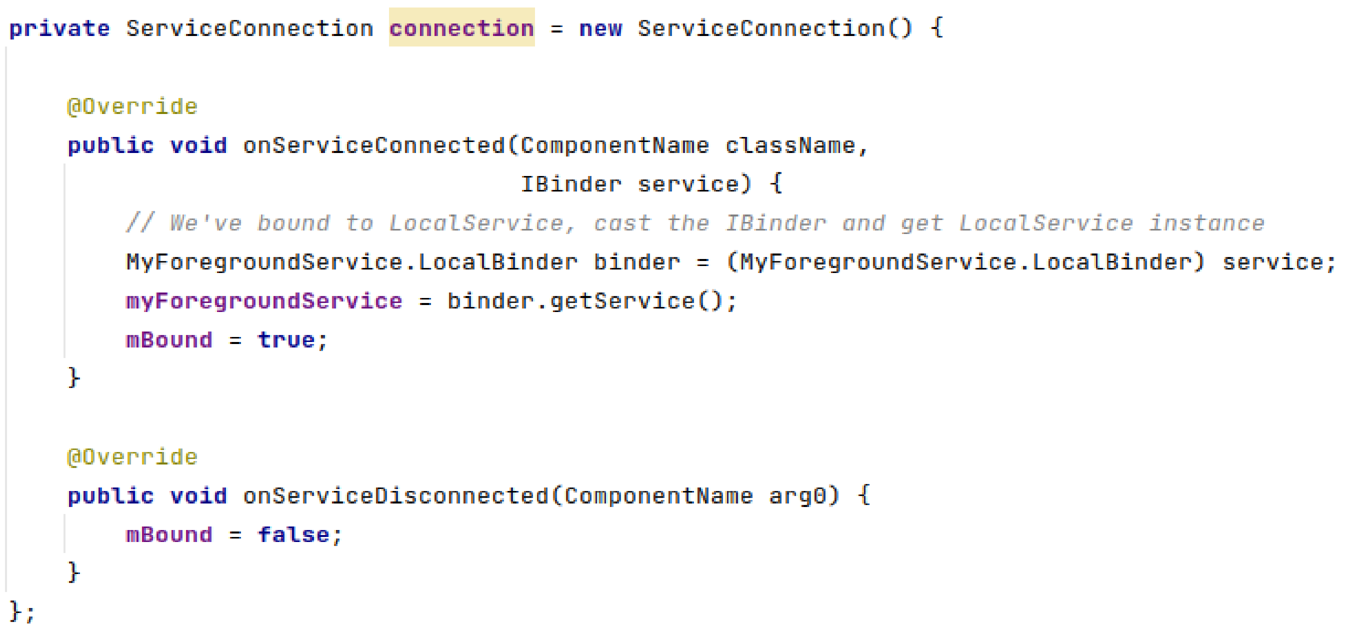
```java
@Override
protected void onStop() {
    super.onStop();
    unbindService(connection);
    mBound = false;
}
```

8. Szkielet metody onButtonClick():



```java
public void onButtonClick() {
    if (mBound) {

        int num = myForegroundService.getRandomNumber();
        Toast.makeText( context: this,  text: "number: " + num, Toast.LENGTH_SHORT).show();
        Log.v(TAG,  msg: "Wartość z klasy MainActivity: " + myForegroundService.getValue());
    }
}
```

9. Definiuje wywołania zwrotne dla wiązania usług, przekazywane do bindService():



```java
private ServiceConnection connection = new ServiceConnection() {

    @Override
    public void onServiceConnected(ComponentName className,
                                   IBinder service) {
        // We've bound to LocalService, cast the IBinder and get LocalService instance
        MyForegroundService.LocalBinder binder = (MyForegroundService.LocalBinder) service;
        myForegroundService = binder.getService();
        mBound = true;
    }

    @Override
    public void onServiceDisconnected(ComponentName arg0) {
        mBound = false;
    }
};
```

10. W klasie MyForegroundService zaimplementuj potrzebne metody do testów:

```java
public class MyForegroundService extends Service {
    private static final String TAG = "1111111111";
    // Binder given to clients
    private final IBinder binder = new LocalBinder();
    // Random number generator
    private final Random mGenerator = new Random();

    /**
     * Class used for the client Binder.  Because we know this service always
     * runs in the same process as its clients, we don't need to deal with IPC.
     */
    public class LocalBinder extends Binder {
        MyForegroundService getService() {
            // Return this instance of LocalService so clients can call public methods
            return MyForegroundService.this;
        }
    }
    public MyForegroundService() {
    }
```

11. Zapoznaj się z fragmentem:

Managing the lifecycle of a bound service

When a service is unbound from all clients, the Android system destroys it (unless it was also started with a `startService()` call). As such, you don't have to manage the lifecycle of your service if it's purely a bound service—the Android system manages it for you based on whether it is bound to any clients.

However, if you choose to implement the `onStartCommand()` callback method, then you must explicitly stop the service, because the service is now considered to be *started*. In this case, the service runs until the service stops itself with `stopSelf()` or another component calls `stopService()`, regardless of whether it is bound to any clients.

Additionally, if your service is started and accepts binding, then when the system calls your `onUnbind()` method, you can optionally return `true` if you would like to receive a call to `onRebind()` the next time a client binds to the service. `onRebind()` returns void, but the client still receives the `IBinder` in its `onServiceConnected()` callback. The following figure illustrates the logic for this kind of lifecycle.

12. Odśwież informacje o cyklu życia:

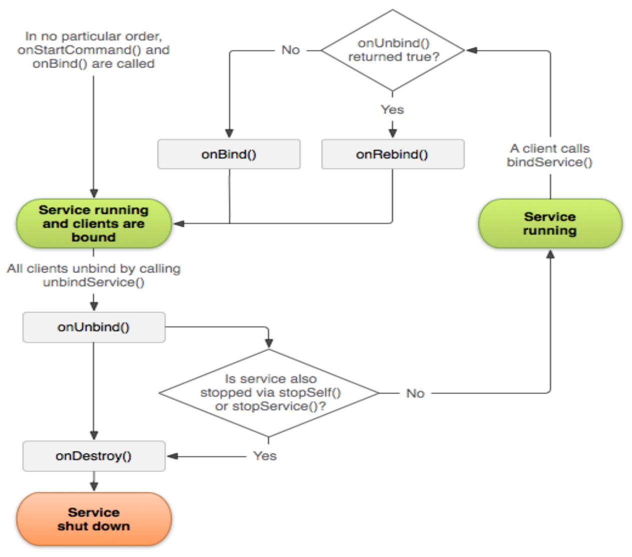

In no particular order, onStartCommand() and onBind() are called
onUnbind() returned true?
No
Yes
onBind()
onRebind()
A client calls bindService()
Service running and clients are bound
Service running
All clients unbind by calling unbindService()
onUnbind()
Is service also stopped via stopSelf() or stopService()?
No
Yes
onDestroy()
Service shut down

Figure 1. The lifecycle for a service that is started and also allows binding.

For more information about the lifecycle of a started service, see the Services document.

13. Dodaj metodę onStartCommand (jeśli musisz :)

```
@Override
public int onStartCommand(Intent intent, int flags, int startId) {
    new Thread(new Runnable() {
        @Override
        public void run() {
            while(true){

                Log.v(TAG,  msg: "service is running...");
                Log.v(TAG,  msg: "Wartość z klasy MyForegroundService: " + value);
                value--;
                try {
                    Thread.sleep( millis: 1000 * 5);
                } catch (InterruptedException e) {
                    e.printStackTrace();
                }
            }
        }
    }).start();
```

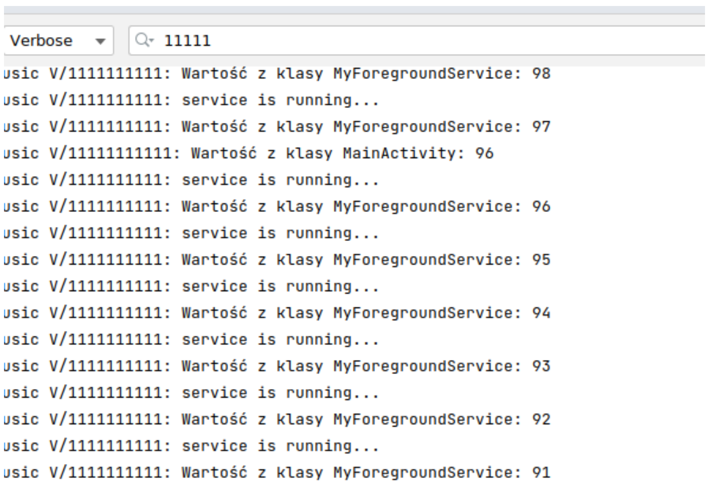
14. Przygotuj metody testowe:

```
/** method for clients */
public int getRandomNumber() { return mGenerator.nextInt( bound: 100); }
public int getValue() { return value; }
```

15. Sprawdź czy zwracasz binder:

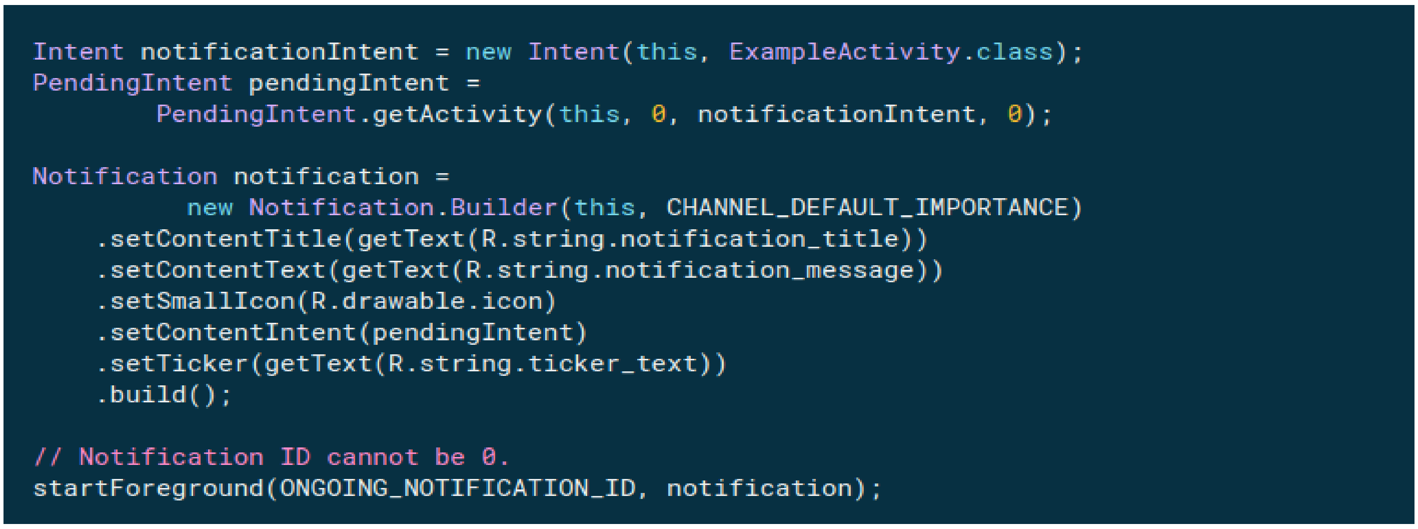
```
@Override
public IBinder onBind(Intent intent) { return binder; }
```

Przetestuj aplikację:



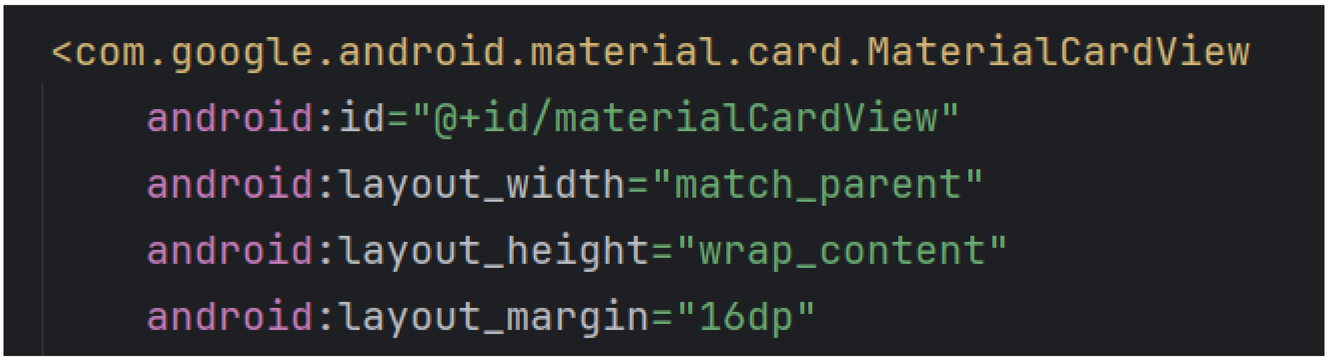
```
usic V/1111111111: Wartość z klasy MyForegroundService: 98
usic V/1111111111: service is running...
usic V/1111111111: Wartość z klasy MyForegroundService: 97
usic V/11111111111: Wartość z klasy MainActivity: 96
usic V/1111111111: service is running...
usic V/1111111111: Wartość z klasy MyForegroundService: 96
usic V/1111111111: service is running...
usic V/1111111111: Wartość z klasy MyForegroundService: 95
usic V/1111111111: service is running...
usic V/1111111111: Wartość z klasy MyForegroundService: 94
usic V/1111111111: service is running...
usic V/1111111111: Wartość z klasy MyForegroundService: 93
usic V/1111111111: service is running...
usic V/1111111111: Wartość z klasy MyForegroundService: 92
usic V/1111111111: service is running...
usic V/1111111111: Wartość z klasy MyForegroundService: 91
```

16. Upewnij się, że dodałeś/aś kanał:
    https://developer.android.com/training/notify-user/channels



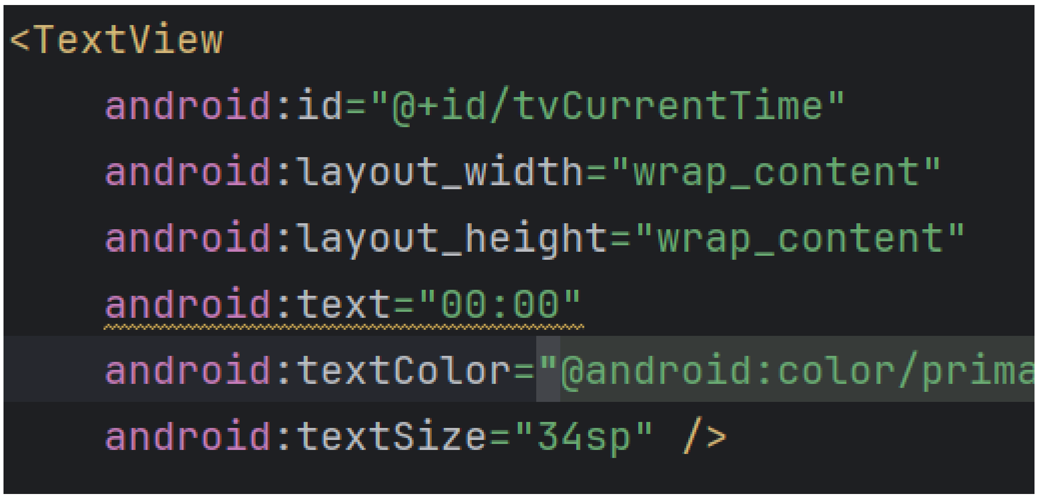
```java
Intent notificationIntent = new Intent(this, ExampleActivity.class);
PendingIntent pendingIntent =
        PendingIntent.getActivity(this, 0, notificationIntent, 0);

Notification notification =
          new Notification.Builder(this, CHANNEL_DEFAULT_IMPORTANCE)
    .setContentTitle(getText(R.string.notification_title))
    .setContentText(getText(R.string.notification_message))
    .setSmallIcon(R.drawable.icon)
    .setContentIntent(pendingIntent)
    .setTicker(getText(R.string.ticker_text))
    .build();

// Notification ID cannot be 0.
startForeground(ONGOING_NOTIFICATION_ID, notification);
```

17. Pozostałe zadania:
    a) przekształć service w usługę odtwarzającą muzykę dla aplikacji.
    b) Przetestuj działanie usługi podczas zamykania aplikacji, otwierania, wznawiania, minimalizacji
    c) dodaj nawigację do obsługi aplikacji, przyciski play, pause, resume i stop

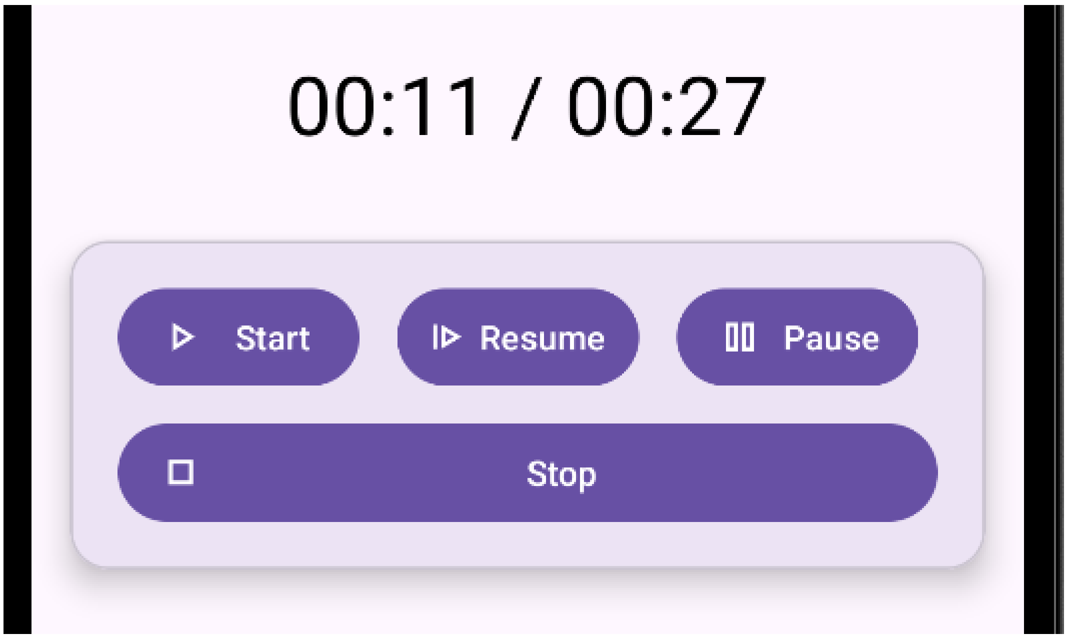


```xml
<com.google.android.material.card.MaterialCardView
    android:id="@+id/materialCardView"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_margin="16dp"
```

    d) dodaj całkowity czas utworu oraz aktualnie odtwarzany



```xml
<TextView
    android:id="@+id/tvCurrentTime"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="00:00"
    android:textColor="@android:color/prima
    android:textSize="34sp" />
```



e) dodaj TextView na nazwę utworu

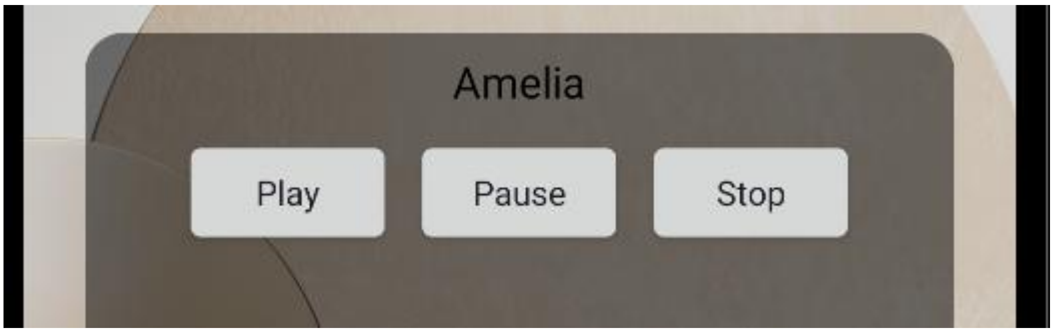
f) dodaj funkcjonalność odtworzenia muzyki z sieci z danego url, np.:

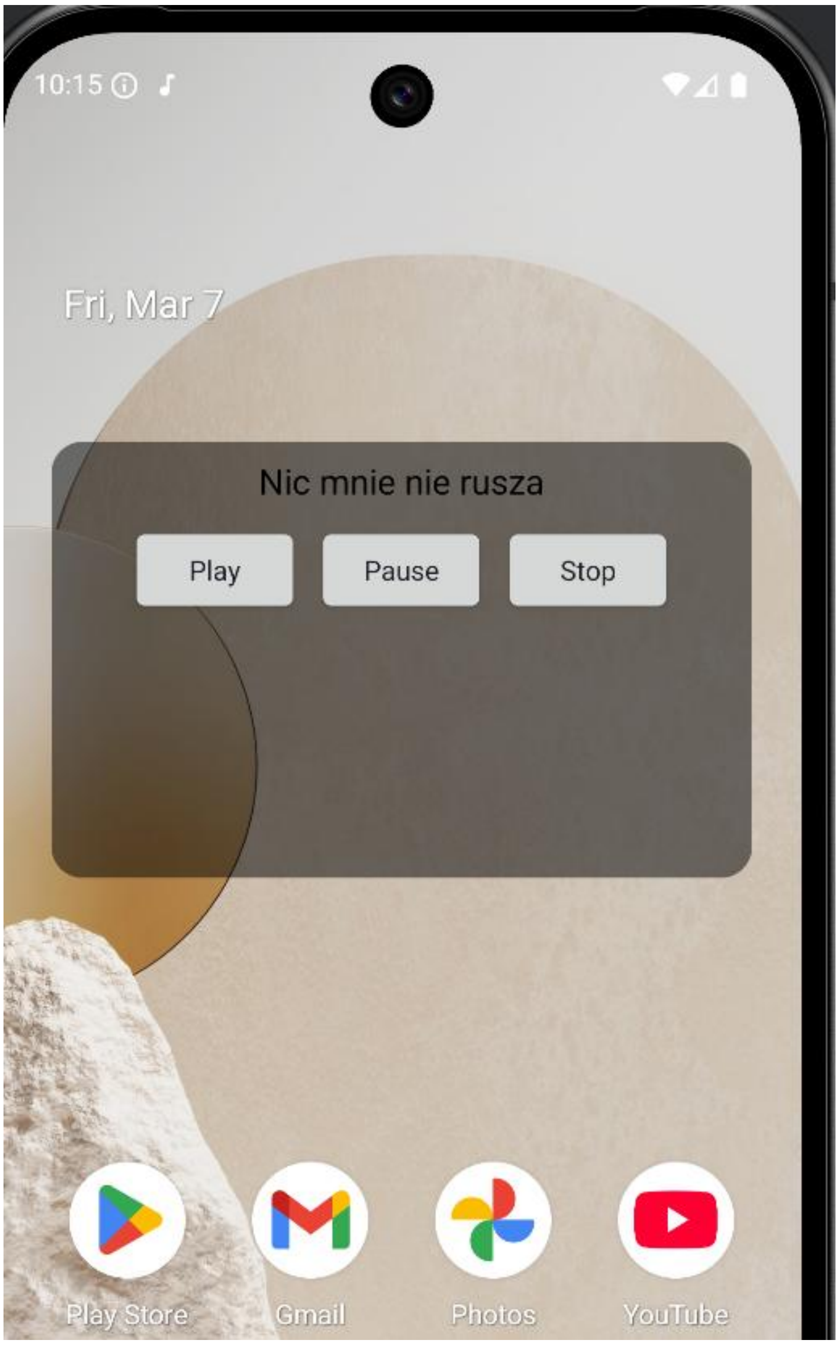
```
String url = "https://www.sample-videos.com/audio/mp3/crowd-cheering.mp3";
no usages
String url2 = "https://www.sample-videos.com/audio/mp3/wave.mp3";
```

g) dodaj odtwarzanie muzyki z plików *.mp3

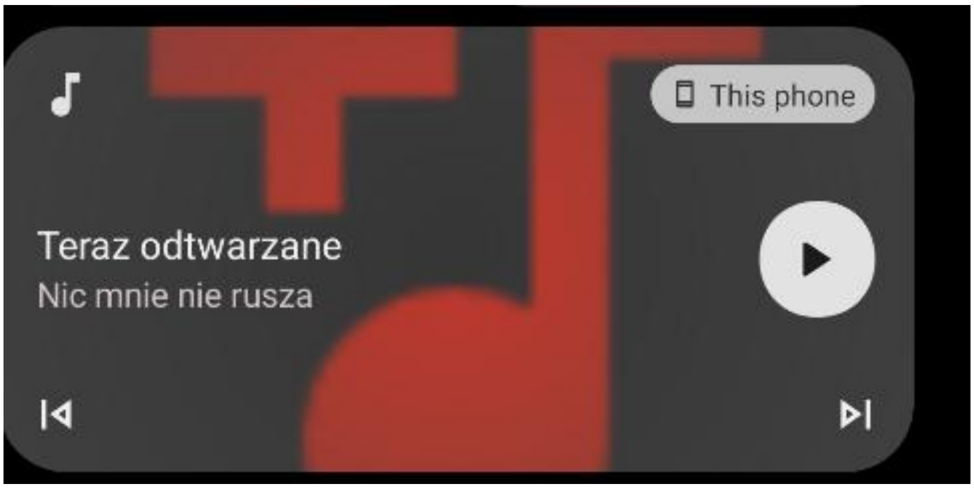
```
private int[] mp3Files = {
        R.raw.amelia,
```

h) dodaj widget z 3 przyciskami play, pause i stop

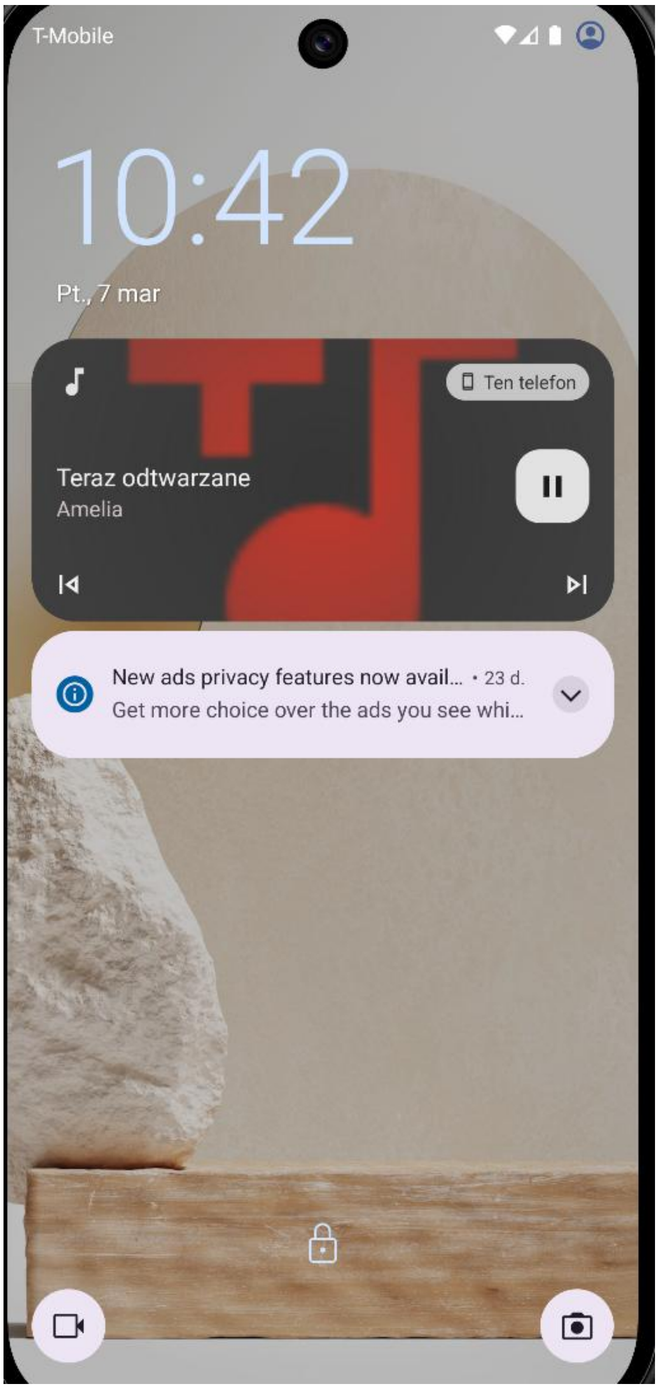




i) dodaj odtwarzanie na powiadomnieniu



j) dodaj odtwarzanie na ekranie blokady


T-Mobile
10:42
Pt., 7 mar
Ten telefon
Teraz odtwarzane
Amelia
New ads privacy features now avail... • 23 d.
Get more choice over the ads you see whi...

18. KONIEC.