Sprawozdanie z realizacji projektu aplikacji mobilnej „PRMusics”

W ramach przedmiotu Programowanie Urządzeń Mobilnych

Reszczyński Patryk

Prowadzący: dr Aleksander Klosow

Legnica

4 stycznia 2021

Spis treści

Rozdział 1

Wstęp

PRMusics jest odtwarzaczem muzyki pozwalającym na streaming treści z internetowej bazy danych. Aplikacja jest wzorowana na popularnym programie Spotify. Program obsługuje przewijanie do dowolnego fragmentu utworu oraz do kolejnej lub poprzedniej piosenki. Utwory zostają automatycznie pobrane na urządzenie mobilne podczas uruchomienia aplikacji z dostępnych w bazie aplikacji osadzonej usłudze Firebase. Użytkownik tak samo jak na Spotify nie ma możliwości dodawania utworów a jest zmuszony do odsłuchu tylko dostępnych materiałów.

Rozdział 2

Przegląd narzędzi i technologii

2.1 Wykorzystane technologie

- Kotlin – język użyty do zaprogramowania aplikacji

2.2 Wykorzystane narzędzia

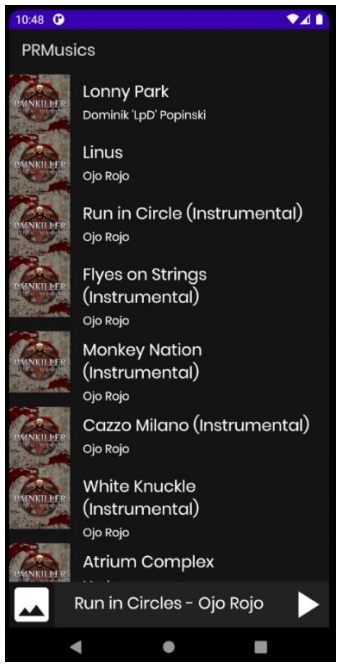
- Android Studio – zintegrowane środowisko programistyczne dla systemu operacyjnego Android
- Github – system kontroli wersji Git
- Firebase – baza danych przechowująca pliki muzyczne

Rozdział 3

Interfejs

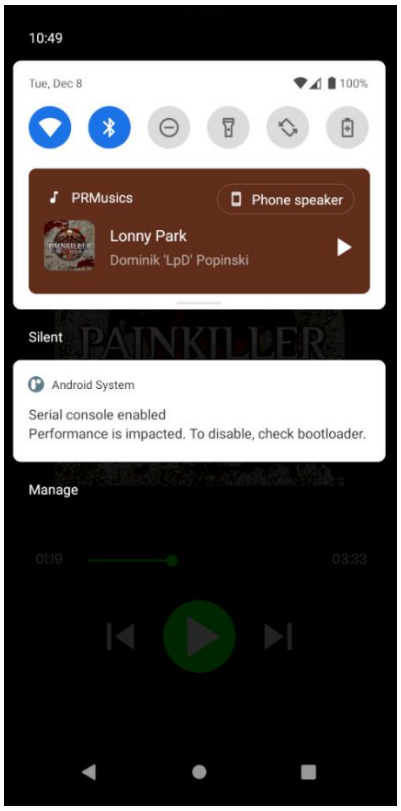
Wideo prezentacja działania aplikacji dostępna pod adresem:

https://youtu.be/U_flsNtQOds

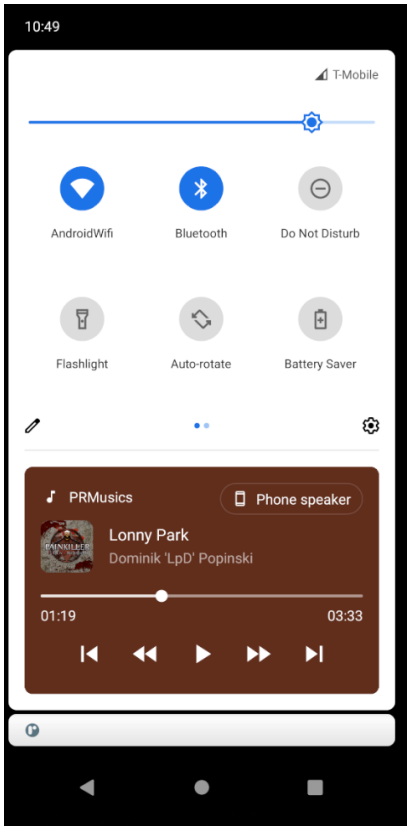


Rysunek 1 Lista piosenek

Rysunek 2 Odtwarzany utwór



Rysunek 3 Nawigacja w tle



Rysunek 4 Rozszerzona nawigacja w tle

Rozdział 4

Dokumentacja kodu

W aplikacji zawarto jedną encję Song. Zawiera ona wszystkie niezbędne informacje o odtwarzanej piosence takie jak numer identyfikacyjny, tytuł, autora, adres URL piosenki oraz adres URL albumu piosenki.

```
data class Song (
    val mediaId: String = "",
    val title: String = "",
    val subtitle: String = "",
    val songUrl: String = "",
    val imageUrl: String = ""
    )
```

Piosenki pobierane są z bazy Firebase. Połączenie z bazą Firebase następuje w klasie MusicDatabase.

```
class MusicDatabase {
    private val firestore : FirebaseFirestore  = FirebaseFirestore.getInstance()
    private val songCollection : CollectionReference  = firestore.collection(SONG_COLLECTION)

    suspend fun getAllSongs(): List<Song> {
        return try {
            songCollection.get().await().toObjects(Song::class.java)
        } catch (e: Exception) {
            emptyList()
        }
    }
}
```

Pobieranie piosenek z bazy następuje według poniższej metody fetchMediaData()

```
suspend fun fetchMediaData() = withContext(Dispatchers.IO) { this: CoroutineScope
    state = STATE_INITIALIZING
    val allSongs = musicDatabase.getAllSongs()
    songs = allSongs.map { song ->
        MediaMetadataCompat.Builder()
            .putString(METADATA_KEY_ARTIST, song.subtitle)
            .putString(METADATA_KEY_MEDIA_ID, song.mediaId)
            .putString(METADATA_KEY_TITLE, song.title)
            .putString(METADATA_KEY_DISPLAY_TITLE, song.title)
            .putString(METADATA_KEY_DISPLAY_ICON_URI, song.imageUrl)
            .putString(METADATA_KEY_MEDIA_URI, song.songUrl)
            .putString(METADATA_KEY_ALBUM_ART_URI, song.imageUrl)
            .putString(METADATA_KEY_DISPLAY_SUBTITLE, song.subtitle)
            .putString(METADATA_KEY_DISPLAY_DESCRIPTION, song.subtitle)
            .build()
    }
    state = STATE_INITIALIZED
}
```

Aby ochronić użytkownika przed nieprzewidywanym zerwaniem połączenia z bazą, zastosowano mechanizm obsługi błędów połączenia.

```
private inner class MediaBrowserConnectionCallback(
    private val context: Context
) : MediaBrowserCompat.ConnectionCallback() {

    override fun onConnected() {
        mediaController = MediaControllerCompat(context, mediaBrowser.sessionToken).apply {
            registerCallback(MediaControllerCallback())
        }
        _isConnected.postValue(Event(Resource.success( data: true)))
    }

    override fun onConnectionSuspended() {
        _isConnected.postValue(Event(Resource.error(
            message: "The connection was suspended", data: false
        )))
    }

    override fun onConnectionFailed() {
        _isConnected.postValue(Event(Resource.error(
            message: "Couldn't connect to media browser", data: false
        )))
    }
}
```

Android Studio posiada wbudowaną obsługę plików multimedialnych dzięki MediaSessionCompat. Aby jednak odtwarzacz był w stanie rozróżnić czy aktualnie wybrany utwór jest grany czy zatrzymany zdefiniowano poniższą logikę.

```
inline val PlaybackStateCompat.isPrepared : Boolean
    get() = state == PlaybackStateCompat.STATE_BUFFERING ||
            state == PlaybackStateCompat.STATE_PLAYING ||
            state == PlaybackStateCompat.STATE_PAUSED

inline val PlaybackStateCompat.isPlaying : Boolean
    get() = state == PlaybackStateCompat.STATE_BUFFERING ||
            state == PlaybackStateCompat.STATE_PLAYING

inline val PlaybackStateCompat.isPlayEnabled : Boolean
    get() = actions and PlaybackStateCompat.ACTION_PLAY != 0L ||
            (actions and PlaybackStateCompat.ACTION_PLAY_PAUSE != 0L &&
                    state == PlaybackStateCompat.STATE_PAUSED)

inline val PlaybackStateCompat.currentPlaybackPosition: Long
    get() = if(state == STATE_PLAYING) {
        val timeDelta = SystemClock.elapsedRealtime() - lastPositionUpdateTime
        (position + (timeDelta * playbackSpeed)).toLong()
    } else position
```

Obsługę przełączania piosenek obsługują poniższe metody:

skipToNextSong() – przełącza na następną piosenkę na liście,

skipToPreviousSong() – przełącza na poprzednią piosenkę na liście

seekTo() – przewija piosenkę do wybranego fragmentu

playOrToggleSong() – odtwarza na wybrany utwór lub wstrzymuje aktualny i uruchamia wybrany

```
fun skipToNextSong() {
    musicServiceConnection.transportControls.skipToNext()
}

fun skipToPreviousSong() {
    musicServiceConnection.transportControls.skipToPrevious()
}

fun seekTo(pos: Long) {
    musicServiceConnection.transportControls.seekTo(pos)
}

fun playOrToggleSong(mediaItem: Song, toggle: Boolean = false) {
    val isPrepared = playbackState.value?.isPrepared ?: false
    if(isPrepared && mediaItem.mediaId ==
        curPlayingSong.value?.getString(METADATA_KEY_MEDIA_ID)) {
        playbackState.value?.let { playbackState ->
            when {
                playbackState.isPlaying -> if(toggle) musicServiceConnection.transportControls.pause()
                playbackState.isPlayEnabled -> musicServiceConnection.transportControls.play()
                else -> Unit
            }
        }
    } else {
        musicServiceConnection.transportControls.playFromMediaId(mediaItem.mediaId,  extras: null)
    }
}
```

Rozdział 5

Instrukcja uruchamiania aplikacji

5.1 Lokalne uruchomienie aplikacji

Aplikacja dostępna jest na repozytorium git pod adresem:

https://github.com/reszczyk/PRMusics.git

Zaleca się użycie oprogramowania Android Studio do uruchomienia aplikacji.

5.2 Uruchomienie aplikacji na Androidzie

Pod poniższym adresem dostępny jest plik „app-release.apk”:

https://github.com/reszczyk/PRMusics/tree/main/app/release

Należy go pobrać i zainstalować na urządzenie z androidem.