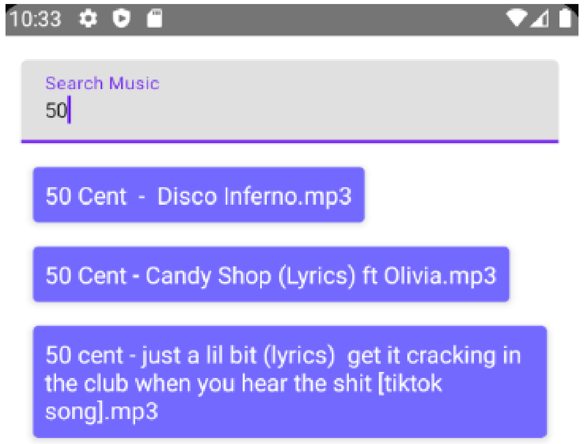
Android Jetpack Compose Music Player


10:33

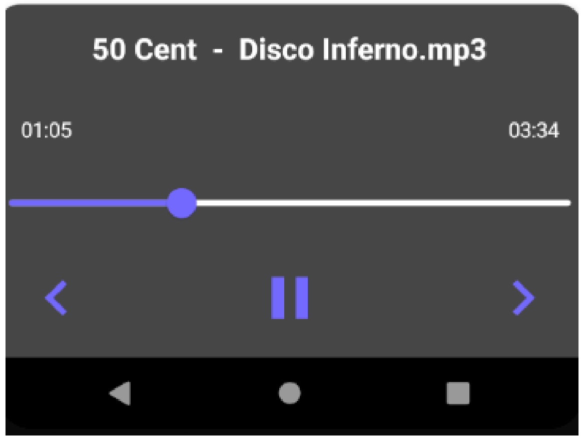
Search Music

50

50 Cent - Disco Inferno.mp3

50 Cent - Candy Shop (Lyrics) ft Olivia.mp3

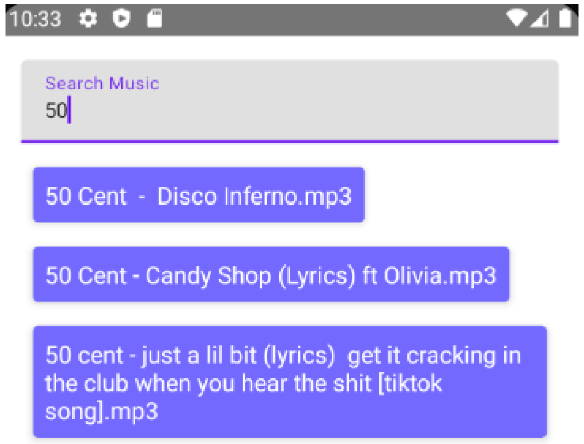
50 cent - just a lil bit (lyrics) get it cracking in the club when you hear the shit [tiktok song].mp3


50 Cent - Disco Inferno.mp3

01:05 03:34

Özellikler

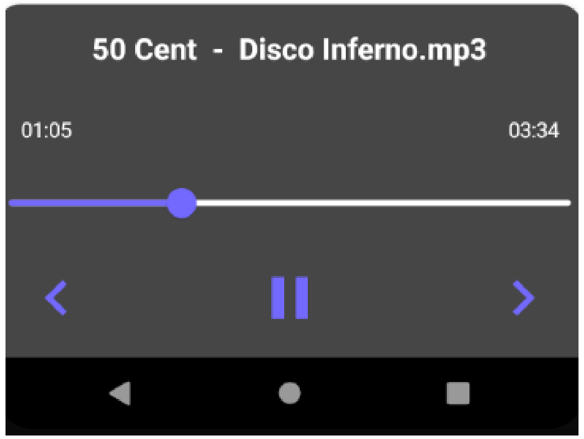
- Müzik Listeleme
- Sonraki ve Önceki müzikler arasında geçiş yapma
- Müzik saniyesini değiştirme çubuğu
- Müzik Durdurma ve Oynatma
- Müzik Arama



50 Cent - Disco Inferno.mp3

50 Cent - Candy Shop (Lyrics) ft Olivia.mp3

50 cent - just a lil bit (lyrics) get it cracking in the club when you hear the shit [tiktok song].mp3



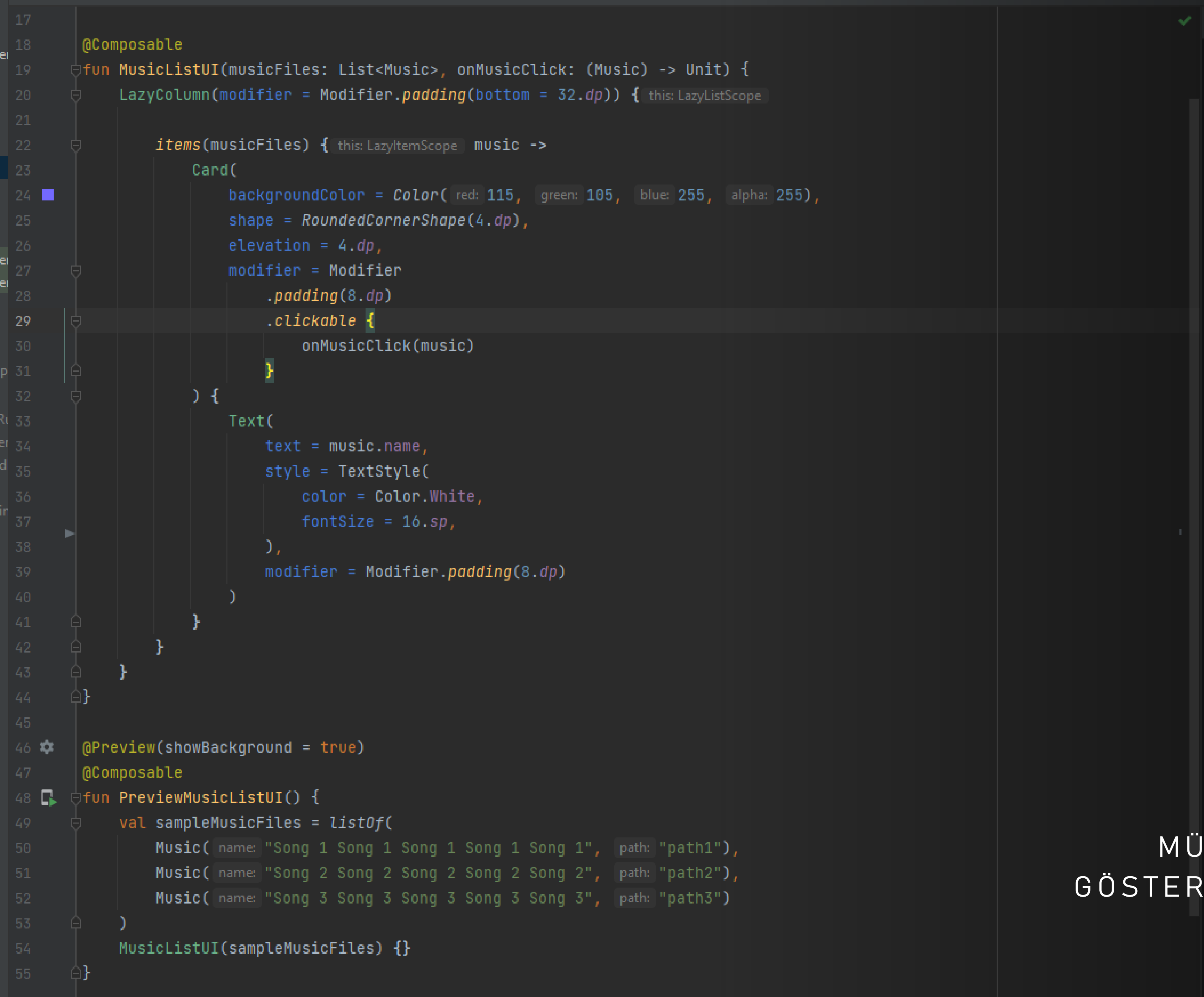
Kullanılan Kütüphaneler

- pub.devrel:easypermissions
- androidx.compose.material:material

Music Sınıfı

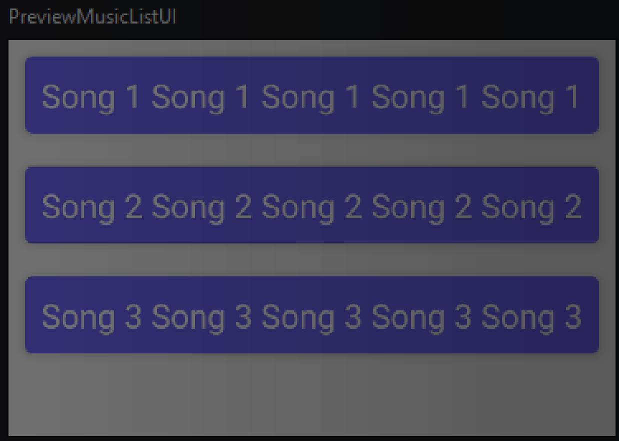
```
class Music(val name: String, val path: String)
```

- Müzik ile alakalı verileri saklayan sınıf.



```kotlin
@Composable
fun MusicListUI(musicFiles: List<Music>, onMusicClick: (Music) -> Unit) {
    LazyColumn(modifier = Modifier.padding(bottom = 32.dp)) { this: LazyListScope

        items(musicFiles) { this: LazyItemScope  music ->
            Card(
                backgroundColor = Color( red: 115,  green: 105,  blue: 255,  alpha: 255),
                shape = RoundedCornerShape(4.dp),
                elevation = 4.dp,
                modifier = Modifier
                    .padding(8.dp)
                    .clickable {
                        onMusicClick(music)
                    }
            ) {
                Text(
                    text = music.name,
                    style = TextStyle(
                        color = Color.White,
                        fontSize = 16.sp,
                    ),
                    modifier = Modifier.padding(8.dp)
                )
            }
        }
    }
}

@Preview(showBackground = true)
@Composable
fun PreviewMusicListUI() {
    val sampleMusicFiles = listOf(
        Music( name: "Song 1 Song 1 Song 1 Song 1 Song 1",  path: "path1"),
        Music( name: "Song 2 Song 2 Song 2 Song 2 Song 2",  path: "path2"),
        Music( name: "Song 3 Song 3 Song 3 Song 3 Song 3",  path: "path3")
    )
    MusicListUI(sampleMusicFiles) {}
}
```

MusicListUI



PreviewMusicListUI

Song 1 Song 1 Song 1 Song 1 Song 1

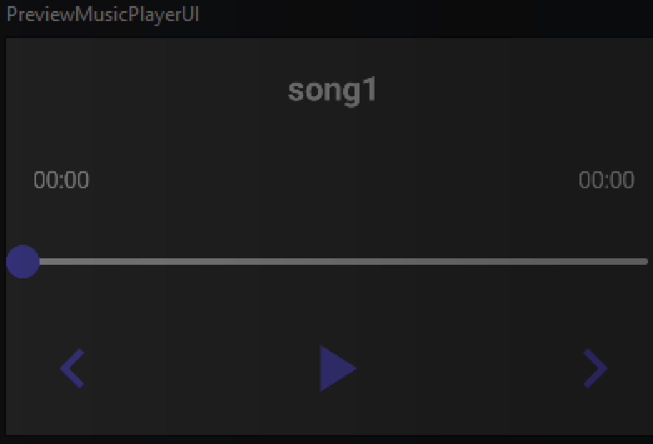
Song 2 Song 2 Song 2 Song 2 Song 2

Song 3 Song 3 Song 3 Song 3 Song 3

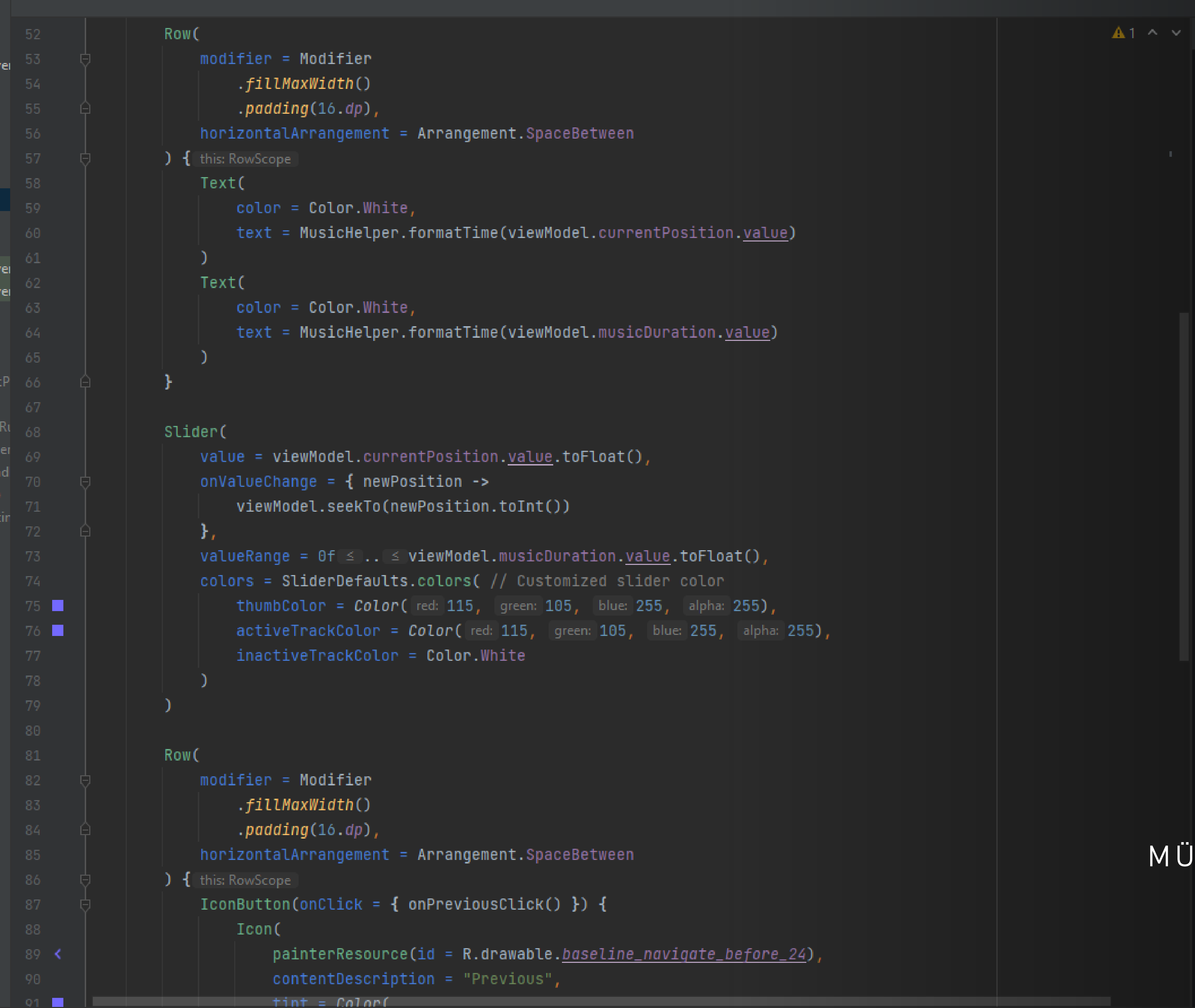
MÜZIK LISTESI VE ELEMANLARI GÖSTERMEYE YARAYAN KOMPONENT.

```kotlin
package com.serdarturan.musicplayer

import ...


@Composable
fun MusicPlayerUI(
    viewModel: MusicViewModel,
    onPreviousClick: () -> Unit,
    onPlayOrPauseClick: () -> Unit,
    onNextClick: () -> Unit
) {

    Column(
        modifier = Modifier
            .fillMaxWidth()
            .background(Color( red: 70, green: 70, blue: 70, alpha: 255)),
        verticalArrangement = Arrangement.Bottom
    ) { this: ColumnScope
        Text(
            color = Color.White,
            text = viewModel.currentMusicName.value,
            modifier = Modifier
                .align(Alignment.CenterHorizontally)
                .padding(16.dp),
            style = TextStyle(
                fontWeight = FontWeight.Bold,
                fontSize = 20.sp,
            )
        )
        Row(
            modifier = Modifier
                .fillMaxWidth()
                .padding(16.dp),
            horizontalArrangement = Arrangement.SpaceBetween
        ) { this: RowScope
            Text(
                color = Color.White,
                text = MusicHelper.formatTime(viewModel.currentPosition.value)
```

MusicPlayerUI

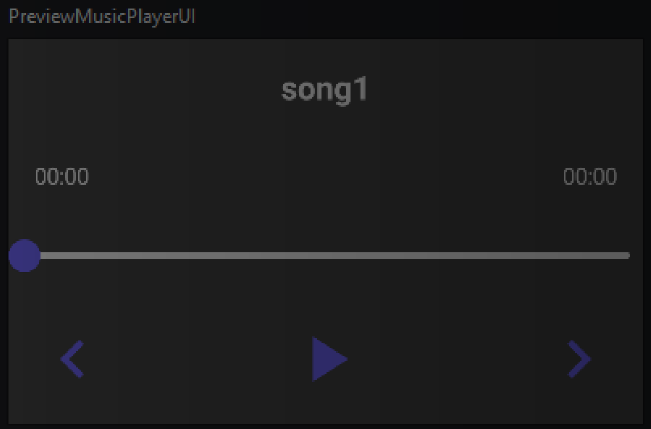


MÜZIK KONTROLLERİNİ YAPMAYA YARAYAN KOMPONENT.

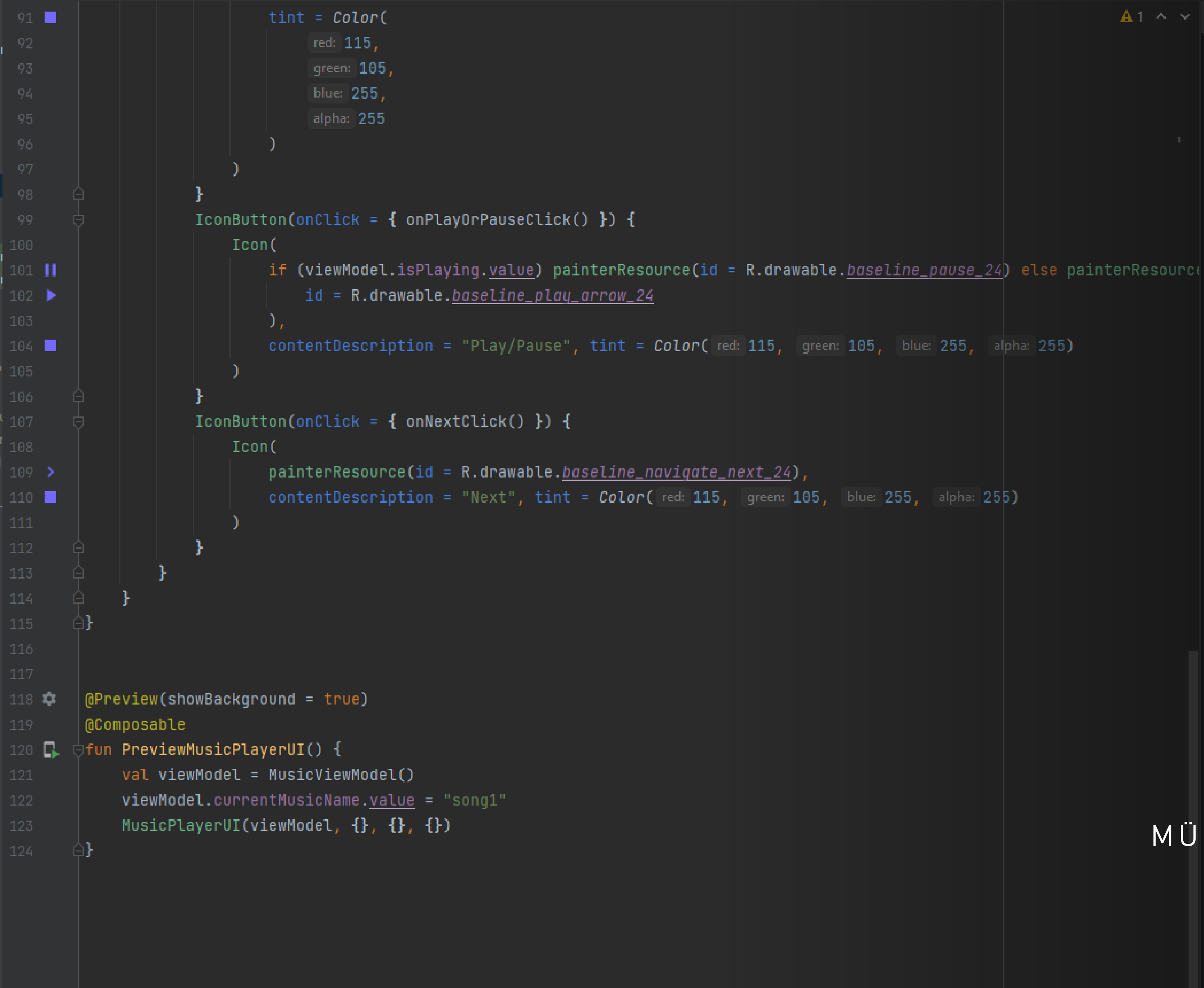


```
Row(
    modifier = Modifier
        .fillMaxWidth()
        .padding(16.dp),
    horizontalArrangement = Arrangement.SpaceBetween
) { this: RowScope
    Text(
        color = Color.White,
        text = MusicHelper.formatTime(viewModel.currentPosition.value)
    )
    Text(
        color = Color.White,
        text = MusicHelper.formatTime(viewModel.musicDuration.value)
    )
}

Slider(
    value = viewModel.currentPosition.value.toFloat(),
    onValueChange = { newPosition ->
        viewModel.seekTo(newPosition.toInt())
    },
    valueRange = 0f ≤ .. ≤ viewModel.musicDuration.value.toFloat(),
    colors = SliderDefaults.colors( // Customized slider color
        thumbColor = Color( red: 115,  green: 105,  blue: 255,  alpha: 255),
        activeTrackColor = Color( red: 115,  green: 105,  blue: 255,  alpha: 255),
        inactiveTrackColor = Color.White
    )
)

Row(
    modifier = Modifier
        .fillMaxWidth()
        .padding(16.dp),
    horizontalArrangement = Arrangement.SpaceBetween
) { this: RowScope
    IconButton(onClick = { onPreviousClick() }) {
        Icon(
            painterResource(id = R.drawable.baseline_navigate_before_24),
            contentDescription = "Previous",
            tint = Color(
```

MusicPlayerUI

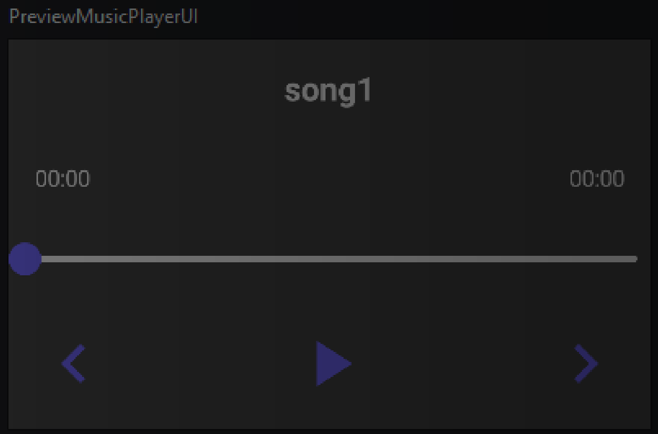


MÜZIK KONTROLLERİNİ YAPMAYA
YARAYAN KOMPONENT.

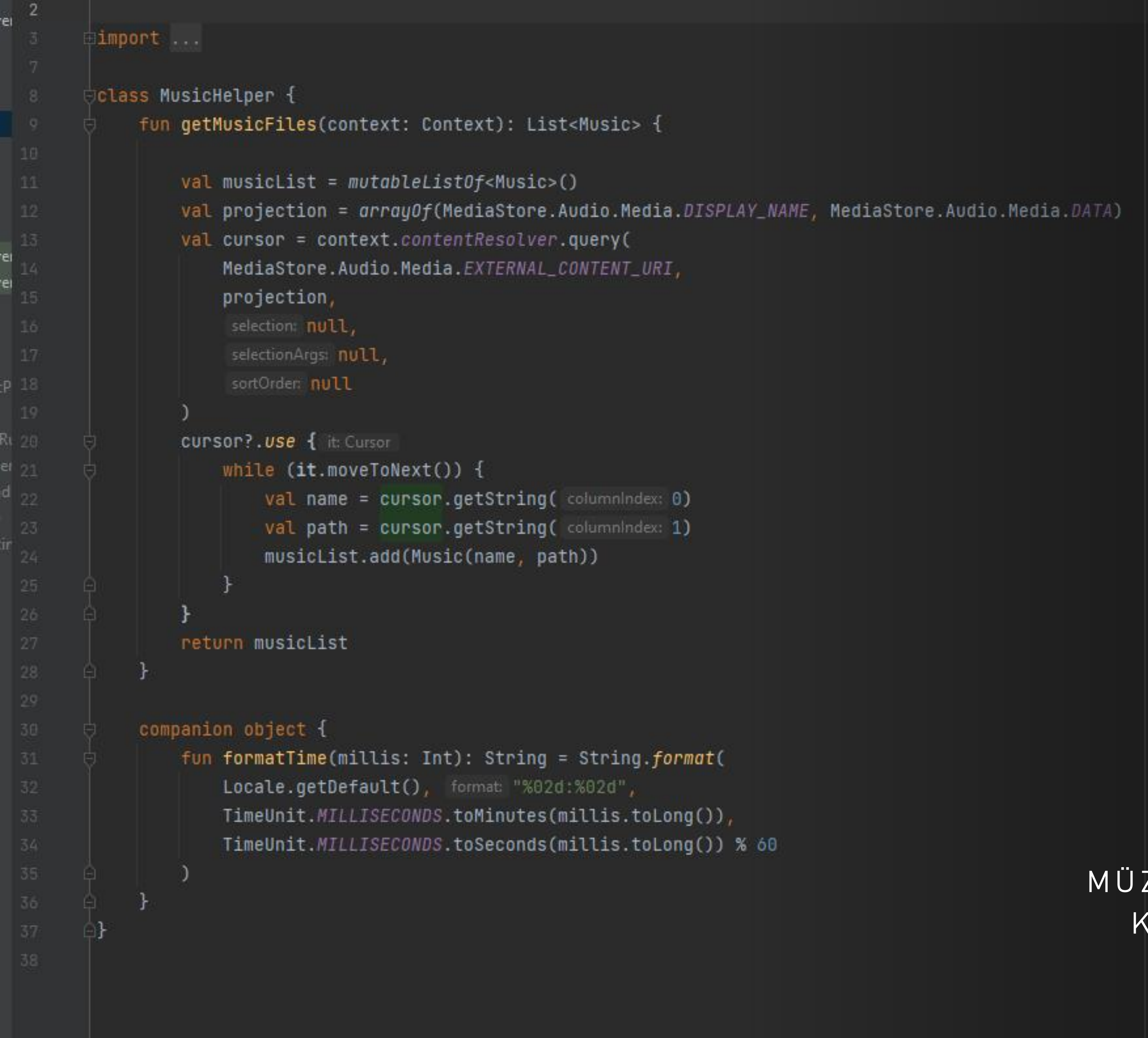


```
                    tint = Color(
                        red: 115,
                        green: 105,
                        blue: 255,
                        alpha: 255
                    )
                )
            }
            IconButton(onClick = { onPlayOrPauseClick() }) {
                Icon(
                    if (viewModel.isPlaying.value) painterResource(id = R.drawable.baseline_pause_24) else painterResourc
                        id = R.drawable.baseline_play_arrow_24
                    ),
                    contentDescription = "Play/Pause", tint = Color( red: 115, green: 105, blue: 255, alpha: 255)
                )
            }
            IconButton(onClick = { onNextClick() }) {
                Icon(
                    painterResource(id = R.drawable.baseline_navigate_next_24),
                    contentDescription = "Next", tint = Color( red: 115, green: 105, blue: 255, alpha: 255)
                )
            }
        }
    }
}


@Preview(showBackground = true)
@Composable
fun PreviewMusicPlayerUI() {
    val viewModel = MusicViewModel()
    viewModel.currentMusicName.value = "song1"
    MusicPlayerUI(viewModel, {}, {}, {})
}
```

MusicPlayerUI



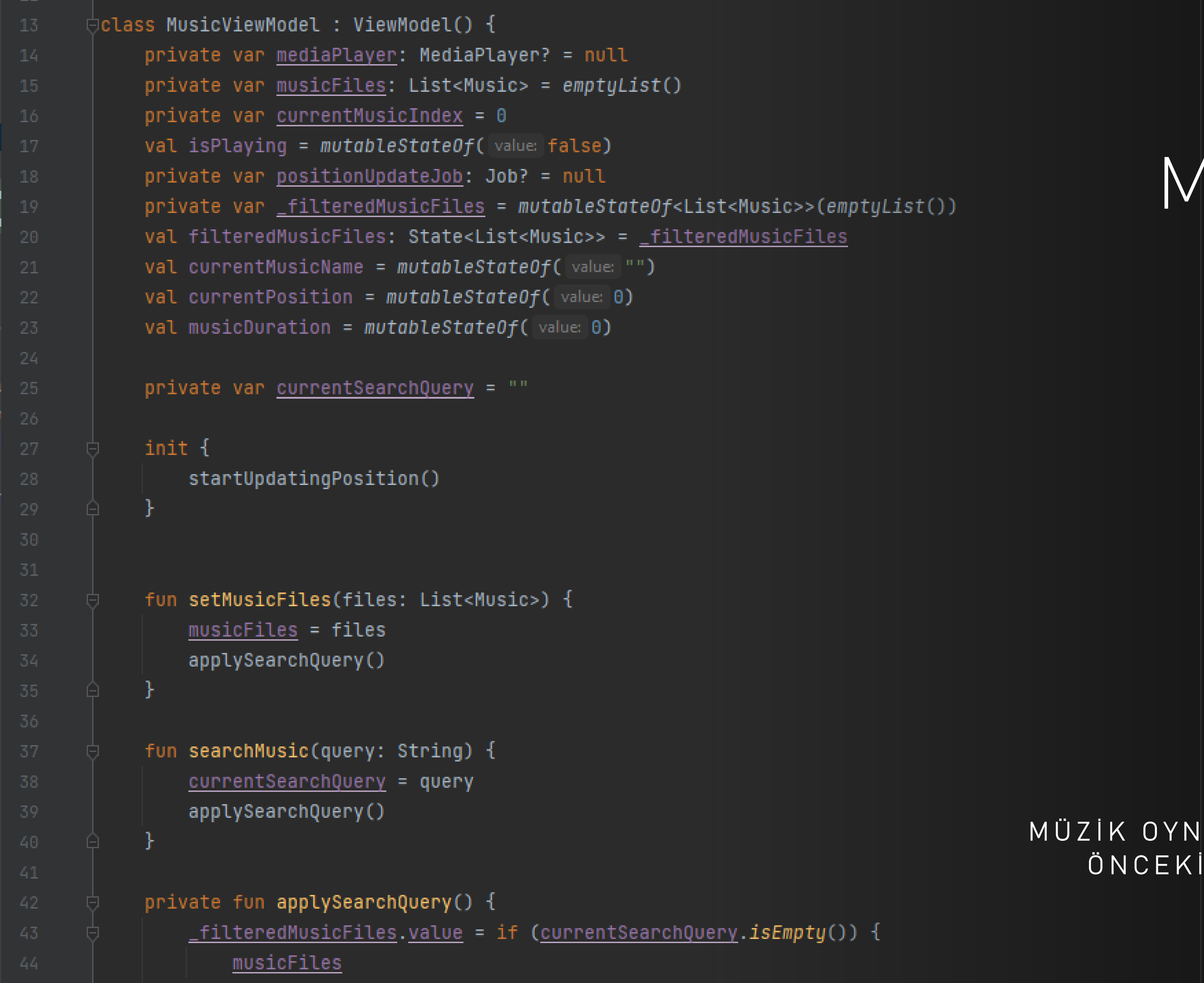
MÜZIK KONTROLLERİNİ YAPMAYA YARAYAN KOMPONENT.



```kotlin
import ...

class MusicHelper {
    fun getMusicFiles(context: Context): List<Music> {

        val musicList = mutableListOf<Music>()
        val projection = arrayOf(MediaStore.Audio.Media.DISPLAY_NAME, MediaStore.Audio.Media.DATA)
        val cursor = context.contentResolver.query(
            MediaStore.Audio.Media.EXTERNAL_CONTENT_URI,
            projection,
            selection: null,
            selectionArgs: null,
            sortOrder: null
        )
        cursor?.use { it: Cursor
            while (it.moveToNext()) {
                val name = cursor.getString( columnIndex: 0)
                val path = cursor.getString( columnIndex: 1)
                musicList.add(Music(name, path))
            }
        }
        return musicList
    }

    companion object {
        fun formatTime(millis: Int): String = String.format(
            Locale.getDefault(), format: "%02d:%02d",
            TimeUnit.MILLISECONDS.toMinutes(millis.toLong()),
            TimeUnit.MILLISECONDS.toSeconds(millis.toLong()) % 60
        )
    }
}
```

MusicHelper

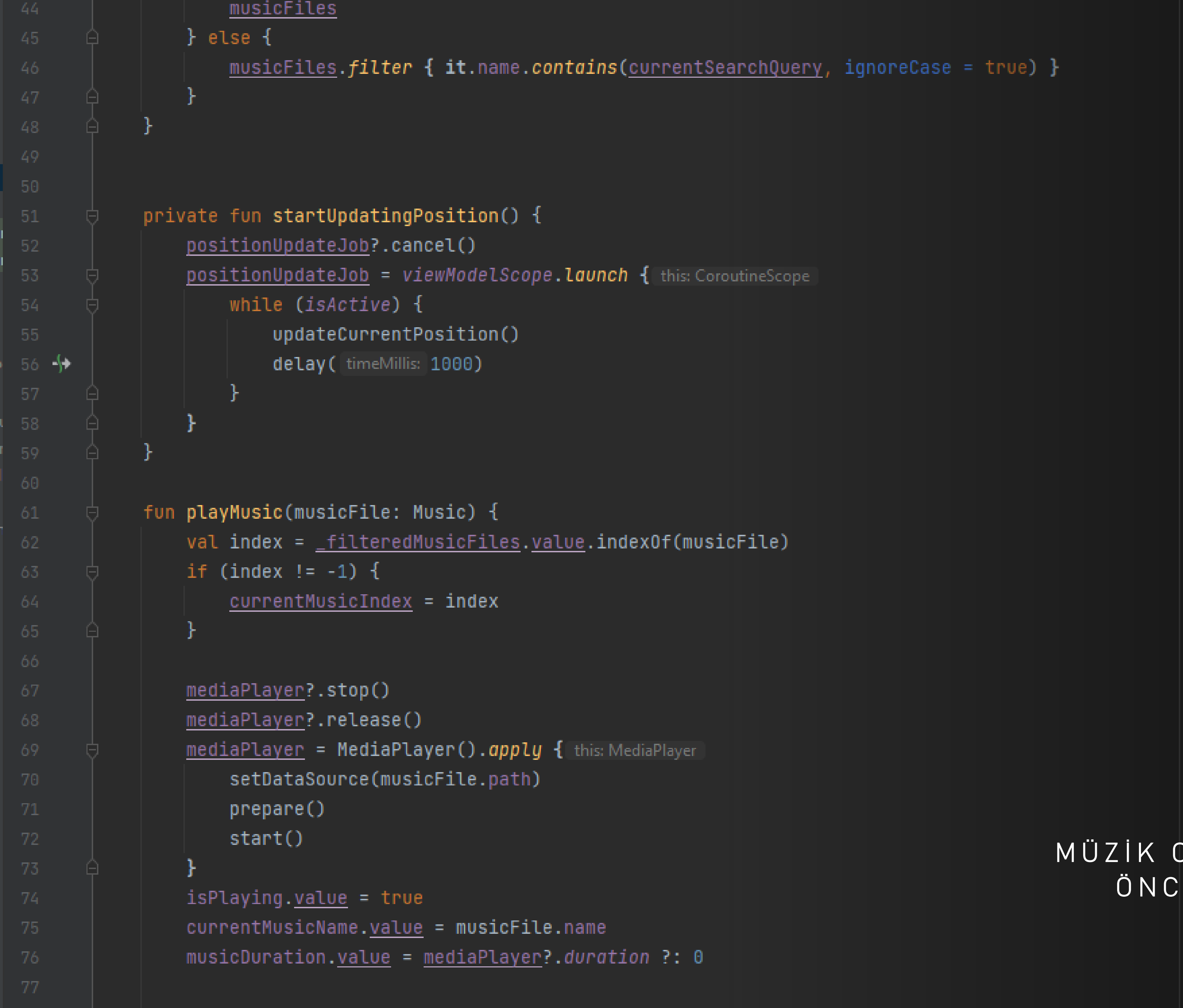
MÜZİKLERİ TELEFONDAN OKUMAYA VE KALAN SÜREYİ GÖSTERMEYE YARAR



```kotlin
class MusicViewModel : ViewModel() {
    private var mediaPlayer: MediaPlayer? = null
    private var musicFiles: List<Music> = emptyList()
    private var currentMusicIndex = 0
    val isPlaying = mutableStateOf( value: false)
    private var positionUpdateJob: Job? = null
    private var _filteredMusicFiles = mutableStateOf<List<Music>>(emptyList())
    val filteredMusicFiles: State<List<Music>> = _filteredMusicFiles
    val currentMusicName = mutableStateOf( value: "")
    val currentPosition = mutableStateOf( value: 0)
    val musicDuration = mutableStateOf( value: 0)

    private var currentSearchQuery = ""

    init {
        startUpdatingPosition()
    }


    fun setMusicFiles(files: List<Music>) {
        musicFiles = files
        applySearchQuery()
    }

    fun searchMusic(query: String) {
        currentSearchQuery = query
        applySearchQuery()
    }

    private fun applySearchQuery() {
        _filteredMusicFiles.value = if (currentSearchQuery.isEmpty()) {
            musicFiles
```

MusicViewModel

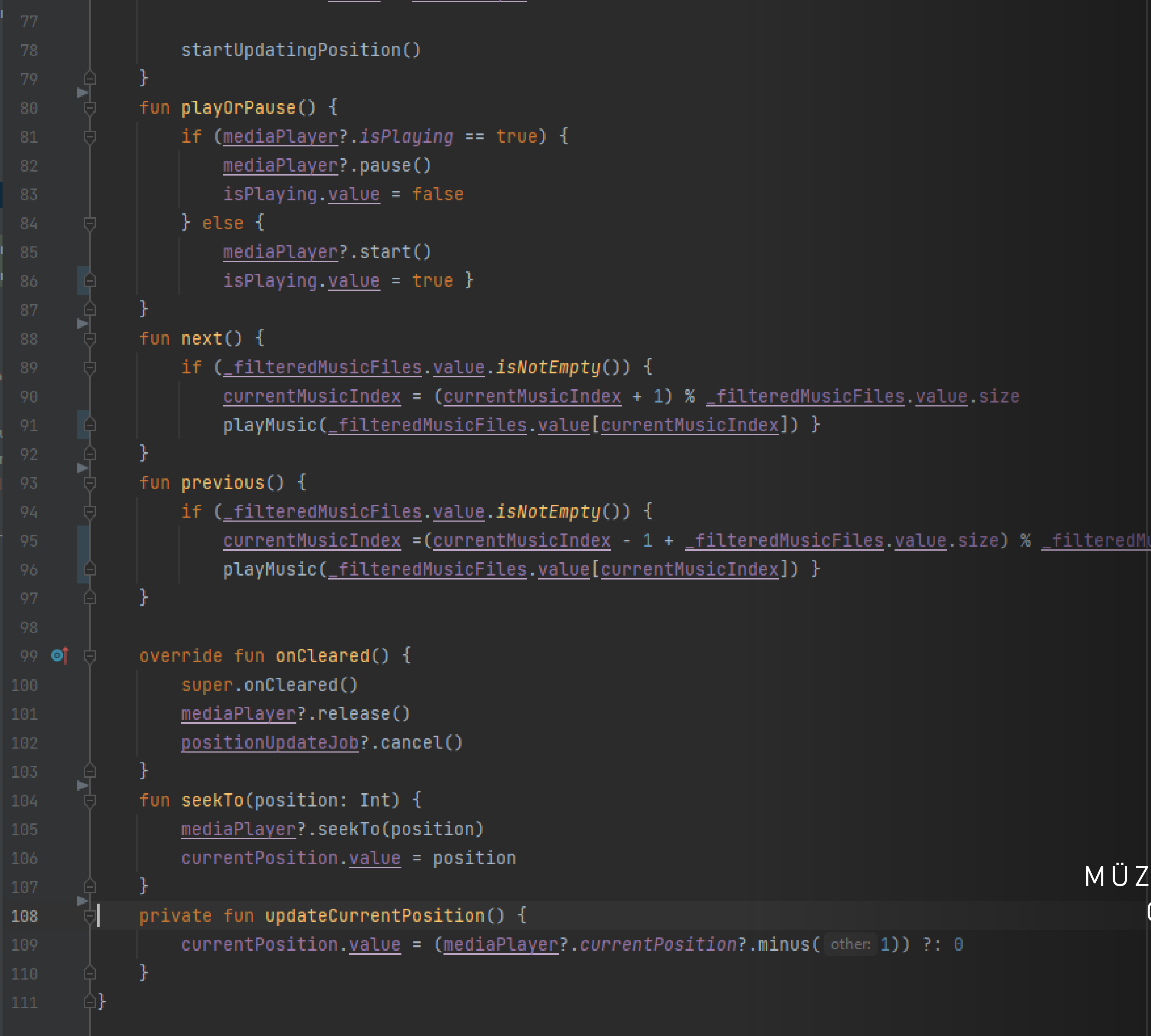
MÜZİK OYNATMA, DURDURMA, SONRAKİ-ÖNCEKİ, GEÇEN ZAMAN, ARAMA GİBİ İŞLEMLERİN YAPILDIĞI YER



```
            musicFiles
        } else {
            musicFiles.filter { it.name.contains(currentSearchQuery, ignoreCase = true) }
        }
    }


    private fun startUpdatingPosition() {
        positionUpdateJob?.cancel()
        positionUpdateJob = viewModelScope.launch { this: CoroutineScope
            while (isActive) {
                updateCurrentPosition()
                delay( timeMillis: 1000)
            }
        }
    }

    fun playMusic(musicFile: Music) {
        val index = _filteredMusicFiles.value.indexOf(musicFile)
        if (index != -1) {
            currentMusicIndex = index
        }

        mediaPlayer?.stop()
        mediaPlayer?.release()
        mediaPlayer = MediaPlayer().apply { this: MediaPlayer
            setDataSource(musicFile.path)
            prepare()
            start()
        }
        isPlaying.value = true
        currentMusicName.value = musicFile.name
        musicDuration.value = mediaPlayer?.duration ?: 0

```

MusicViewModel

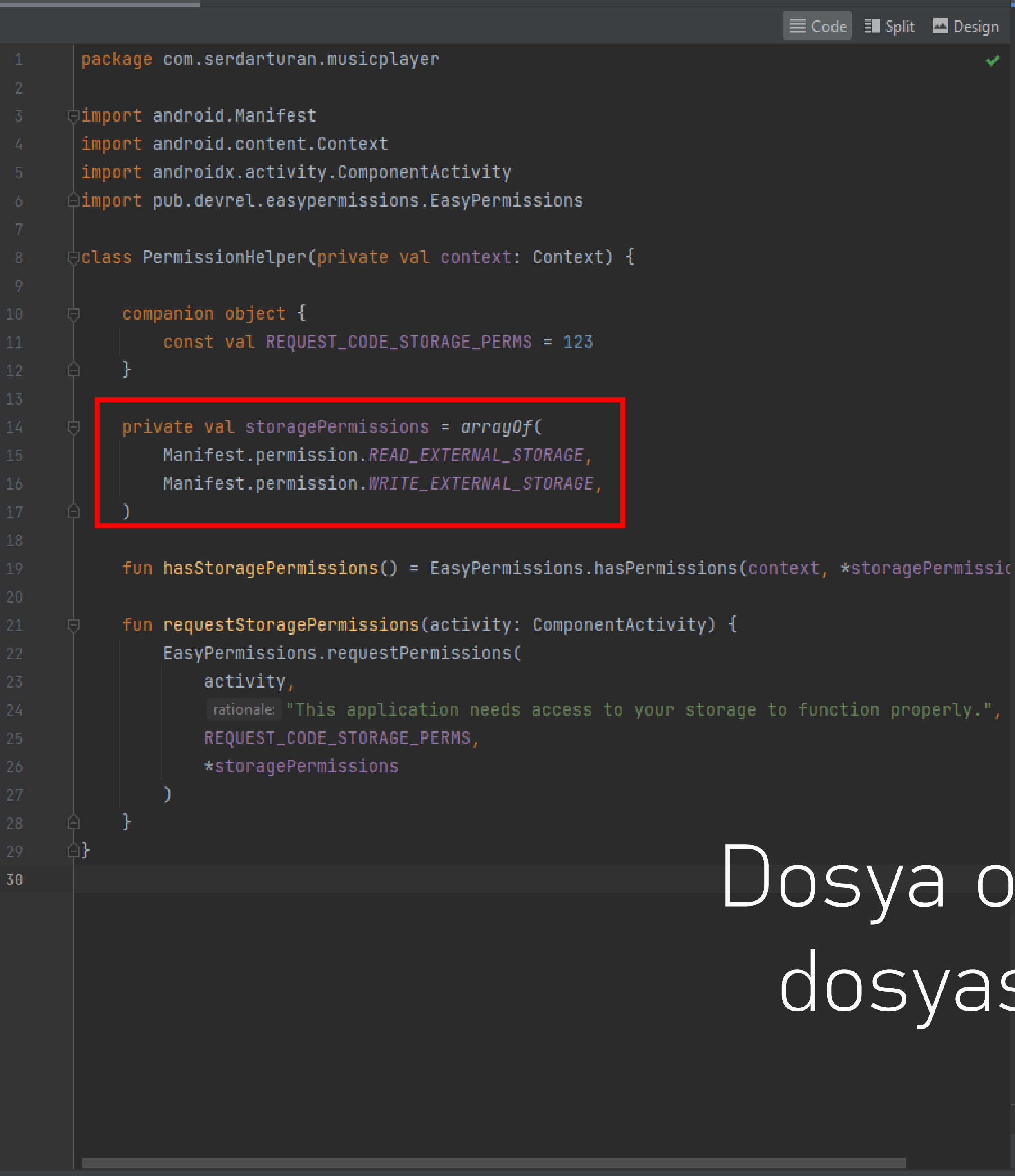
MÜZİK OYNATMA, DURDURMA, SONRAKİ-ÖNCEKİ, GEÇEN ZAMAN, ARAMA GİBİ İŞLEMLERİN YAPILDIĞI YER



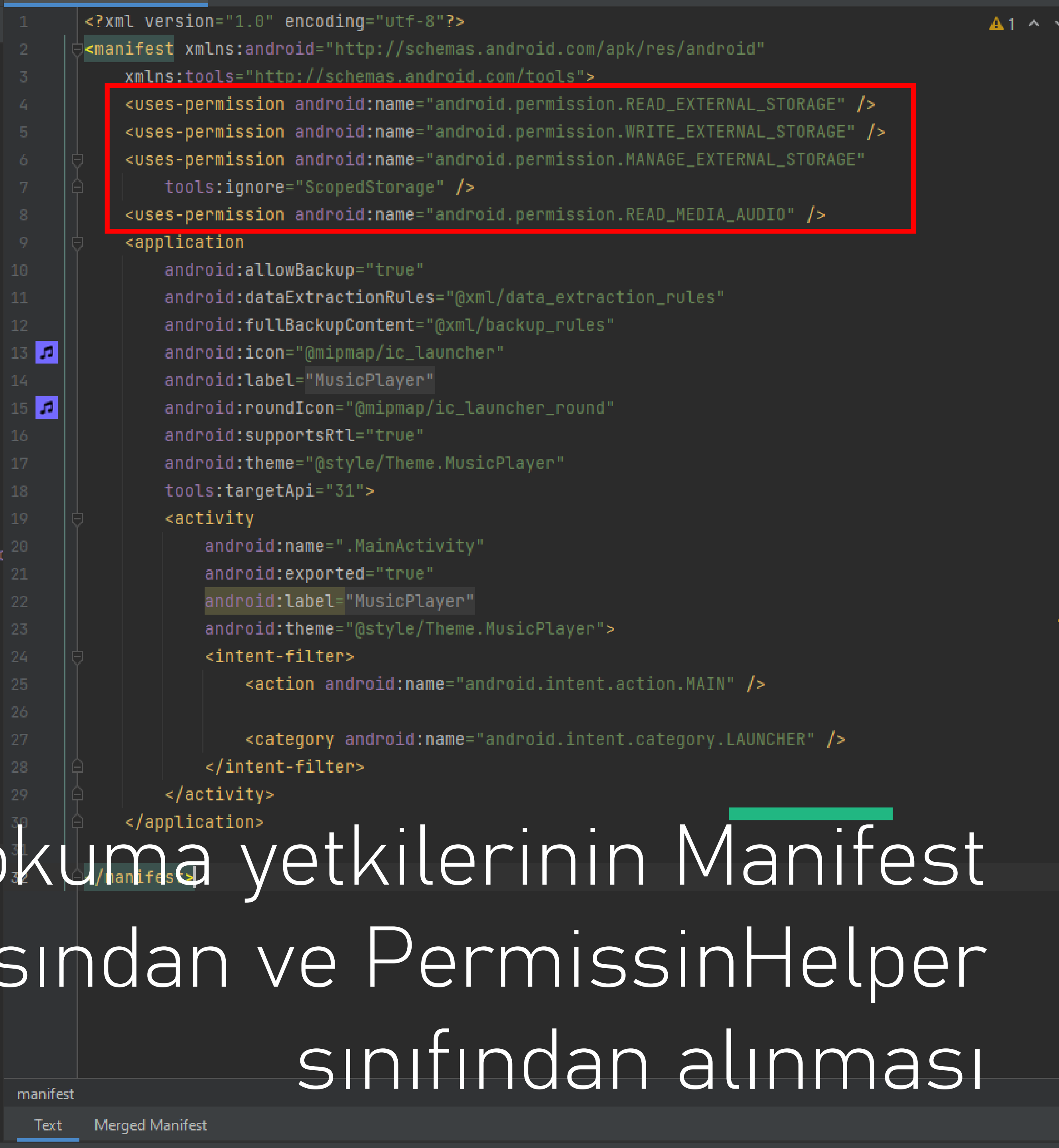
```kotlin

        startUpdatingPosition()
    }
    fun playOrPause() {
        if (mediaPlayer?.isPlaying == true) {
            mediaPlayer?.pause()
            isPlaying.value = false
        } else {
            mediaPlayer?.start()
            isPlaying.value = true }
    }
    fun next() {
        if (_filteredMusicFiles.value.isNotEmpty()) {
            currentMusicIndex = (currentMusicIndex + 1) % _filteredMusicFiles.value.size
            playMusic(_filteredMusicFiles.value[currentMusicIndex]) }
    }
    fun previous() {
        if (_filteredMusicFiles.value.isNotEmpty()) {
            currentMusicIndex =(currentMusicIndex - 1 + _filteredMusicFiles.value.size) % _filteredMusicFiles.value.size
            playMusic(_filteredMusicFiles.value[currentMusicIndex]) }
    }

    override fun onCleared() {
        super.onCleared()
        mediaPlayer?.release()
        positionUpdateJob?.cancel()
    }
    fun seekTo(position: Int) {
        mediaPlayer?.seekTo(position)
        currentPosition.value = position
    }
    private fun updateCurrentPosition() {
        currentPosition.value = (mediaPlayer?.currentPosition?.minus(1)) ?: 0
    }
}
```

MusicViewModel

MÜZİK OYNATMA, DURDURMA, SONRAKİ-
ÖNCEKİ, GEÇEN ZAMAN, ARAMA GİBİ
İŞLEMLERİN YAPILDIĞI YER



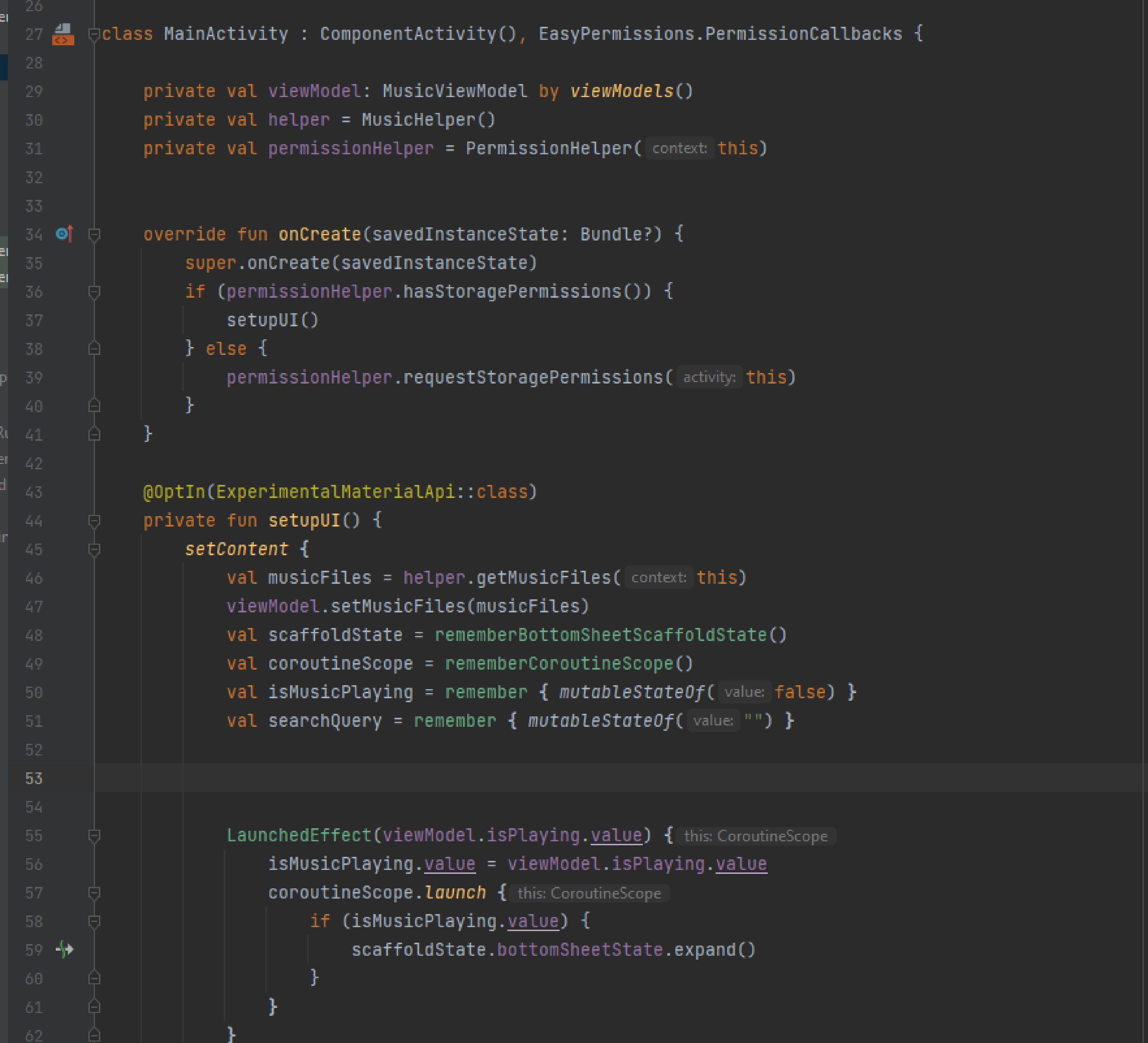
```kotlin
package com.serdarturan.musicplayer

import android.Manifest
import android.content.Context
import androidx.activity.ComponentActivity
import pub.devrel.easypermissions.EasyPermissions

class PermissionHelper(private val context: Context) {

    companion object {
        const val REQUEST_CODE_STORAGE_PERMS = 123
    }

    private val storagePermissions = arrayOf(
        Manifest.permission.READ_EXTERNAL_STORAGE,
        Manifest.permission.WRITE_EXTERNAL_STORAGE,
    )

    fun hasStoragePermissions() = EasyPermissions.hasPermissions(context, *storagePermissic

    fun requestStoragePermissions(activity: ComponentActivity) {
        EasyPermissions.requestPermissions(
            activity,
            rationale: "This application needs access to your storage to function properly.",
            REQUEST_CODE_STORAGE_PERMS,
            *storagePermissions
        )
    }
}
```



```xml
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools">
    <uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE" />
    <uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
    <uses-permission android:name="android.permission.MANAGE_EXTERNAL_STORAGE"
        tools:ignore="ScopedStorage" />
    <uses-permission android:name="android.permission.READ_MEDIA_AUDIO" />
    <application
        android:allowBackup="true"
        android:dataExtractionRules="@xml/data_extraction_rules"
        android:fullBackupContent="@xml/backup_rules"
        android:icon="@mipmap/ic_launcher"
        android:label="MusicPlayer"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:theme="@style/Theme.MusicPlayer"
        tools:targetApi="31">
        <activity
            android:name=".MainActivity"
            android:exported="true"
            android:label="MusicPlayer"
            android:theme="@style/Theme.MusicPlayer">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>

</manifest>
```

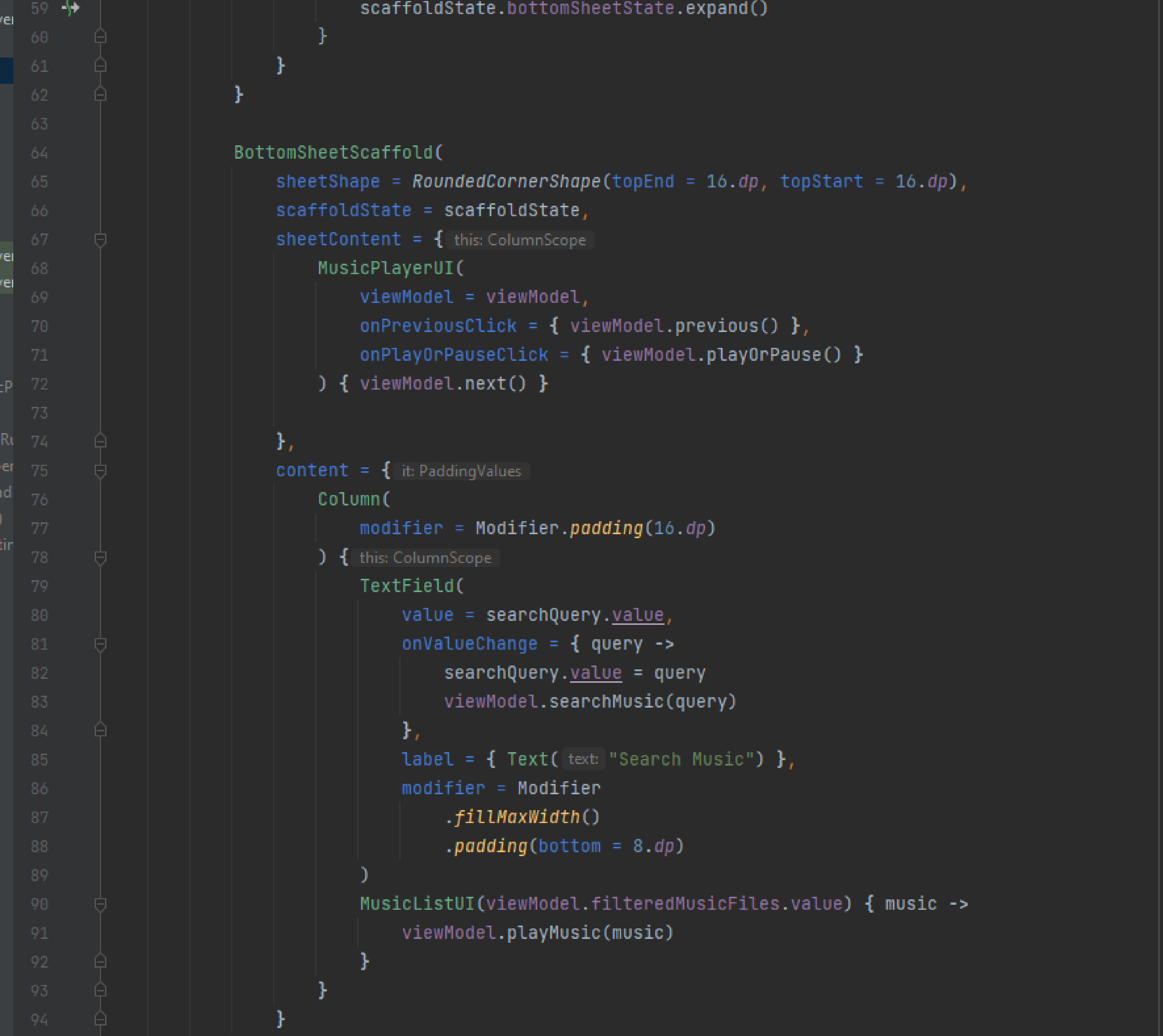
Dosya okuma yetkilerinin Manifest dosyasından ve PermissinHelper sınıfından alınması

MainActivity



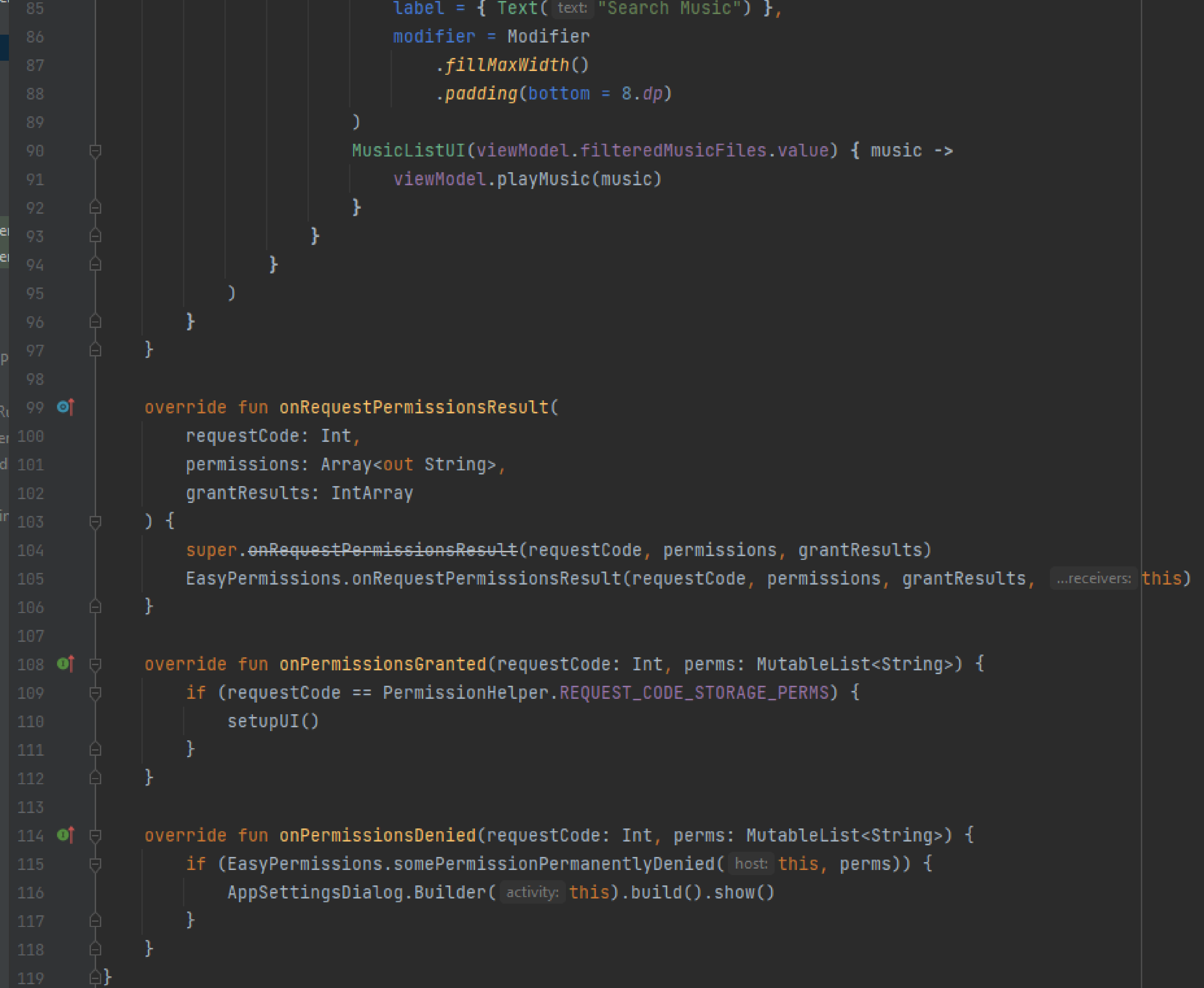
```kotlin
class MainActivity : ComponentActivity(), EasyPermissions.PermissionCallbacks {

    private val viewModel: MusicViewModel by viewModels()
    private val helper = MusicHelper()
    private val permissionHelper = PermissionHelper( context: this)


    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        if (permissionHelper.hasStoragePermissions()) {
            setupUI()
        } else {
            permissionHelper.requestStoragePermissions( activity: this)
        }
    }

    @OptIn(ExperimentalMaterialApi::class)
    private fun setupUI() {
        setContent {
            val musicFiles = helper.getMusicFiles( context: this)
            viewModel.setMusicFiles(musicFiles)
            val scaffoldState = rememberBottomSheetScaffoldState()
            val coroutineScope = rememberCoroutineScope()
            val isMusicPlaying = remember { mutableStateOf( value: false) }
            val searchQuery = remember { mutableStateOf( value: "") }



            LaunchedEffect(viewModel.isPlaying.value) { this: CoroutineScope
                isMusicPlaying.value = viewModel.isPlaying.value
                coroutineScope.launch { this: CoroutineScope
                    if (isMusicPlaying.value) {
                        scaffoldState.bottomSheetState.expand()
                    }
                }
            }
```

MainActivity



```
            scaffoldState.bottomSheetState.expand()
        }
    }
}

BottomSheetScaffold(
    sheetShape = RoundedCornerShape(topEnd = 16.dp, topStart = 16.dp),
    scaffoldState = scaffoldState,
    sheetContent = { this: ColumnScope
        MusicPlayerUI(
            viewModel = viewModel,
            onPreviousClick = { viewModel.previous() },
            onPlayOrPauseClick = { viewModel.playOrPause() }
        ) { viewModel.next() }

    },
    content = { it: PaddingValues
        Column(
            modifier = Modifier.padding(16.dp)
        ) { this: ColumnScope
            TextField(
                value = searchQuery.value,
                onValueChange = { query ->
                    searchQuery.value = query
                    viewModel.searchMusic(query)
                },
                label = { Text( text: "Search Music") },
                modifier = Modifier
                    .fillMaxWidth()
                    .padding(bottom = 8.dp)
            )
            MusicListUI(viewModel.filteredMusicFiles.value) { music ->
                viewModel.playMusic(music)
            }
        }
    }
```

MainActivity



```kotlin
                            label = { Text( text: "Search Music") },
                            modifier = Modifier
                                .fillMaxWidth()
                                .padding(bottom = 8.dp)
                        )
                        MusicListUI(viewModel.filteredMusicFiles.value) { music ->
                            viewModel.playMusic(music)
                        }
                    }
                }
            )
        }
    }

    override fun onRequestPermissionsResult(
        requestCode: Int,
        permissions: Array<out String>,
        grantResults: IntArray
    ) {
        super.~~onRequestPermissionsResult~~(requestCode, permissions, grantResults)
        EasyPermissions.onRequestPermissionsResult(requestCode, permissions, grantResults, ...receivers: this)
    }

    override fun onPermissionsGranted(requestCode: Int, perms: MutableList<String>) {
        if (requestCode == PermissionHelper.REQUEST_CODE_STORAGE_PERMS) {
            setupUI()
        }
    }

    override fun onPermissionsDenied(requestCode: Int, perms: MutableList<String>) {
        if (EasyPermissions.somePermissionPermanentlyDenied( host: this, perms)) {
            AppSettingsDialog.Builder( activity: this).build().show()
        }
    }
}
```

kaynak

- https://www.youtube.com/watch?v=dDNpblR7T8w
- https://www.youtube.com/watch?v=32HjHtoyGvQ
- https://github.com/DawinderGill/MusicPlayer-JetpackCompose
- https://github.com/jslowinski/MusicPlayer