Міністерство освіти і науки України
КПІ ім. Ігоря Сікорського
Факультет інформатики та обчислюваної техніки
Кафедра інформаційних систем та технологій

Лабораторна робота №4

з дисципліни: «Розробка мобільних застосунвань під Android»

Перевірив:
Викладач, PhD
Орленко С.П.

Виконав:
студент гр. ІК-23
Волнушкін В.І.

2025 р.

Мета роботи: дослідити яким чином платформа Андроїд надає можливість оброблювати аудіо-файли та відео-файли та отримати практичні навички щодо використання інструментів відтворення медіа-даних.

Завдання:

– Виконано

– Почато, але буде дороблено, якщо буде час

БАЗОВЕ (12/20 балів). Написати програму під платформу Андроїд, яка має інтерфейс для запуску аудіо-файлів та відео-файлів. Мінімально інтерфейс має надавати можливість Програвати/Зупиняти/Призупиняти відтворення відео-файлу або аудіо-файлу, який зберігається у внутрішньому сховищі.

ПОВНЕ (20/20). Функціональність базового додатку додатково розширюється наступними можливостями:

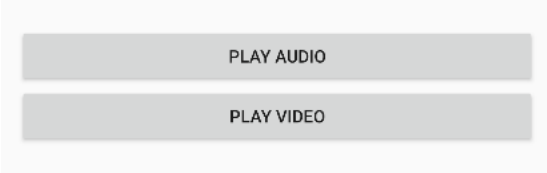
- надати вибір типу файлу для відтворення (аудіо або відео) з будь-якого сховища на мобільному пристрої;

- надати вибір завантаження файлу з Інтернету;

- використовувати для реалізації обробки медіа-даних спеціалізовані інструменти (особливу увагу приділити програванню відео).

Хід виконання

Створення MainActivity з двома кнопками (4-ма – якщо буде час доробити на більше балів, а поки основна ціль – закрити хоч на 60) для відтворення аудіо і відео відповідно.



Кнопки для програвання з інтернету приховані в activity_main.xml:

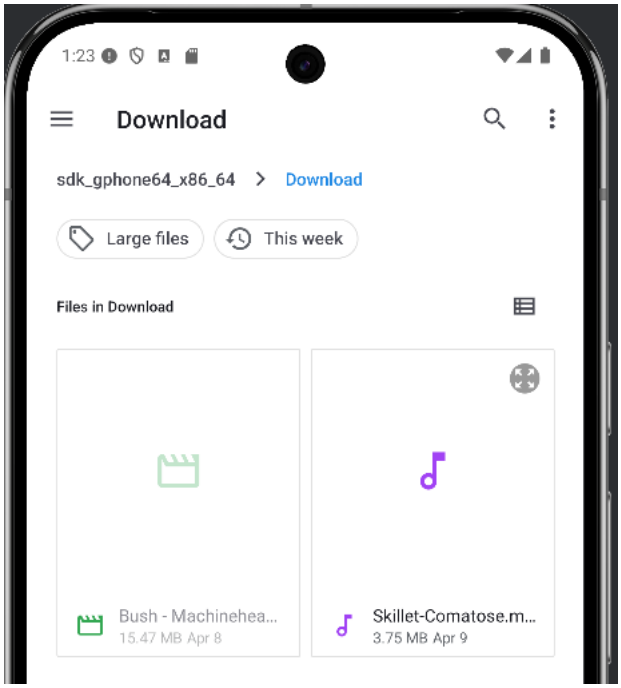
```
<Button
   android:id="@+id/playAudioOnlineButton"
   android:layout_width="match_parent"
   android:layout_height="wrap_content"
   android:text="Play Audio Online"
   android:visibility="gone" />

<Button
```

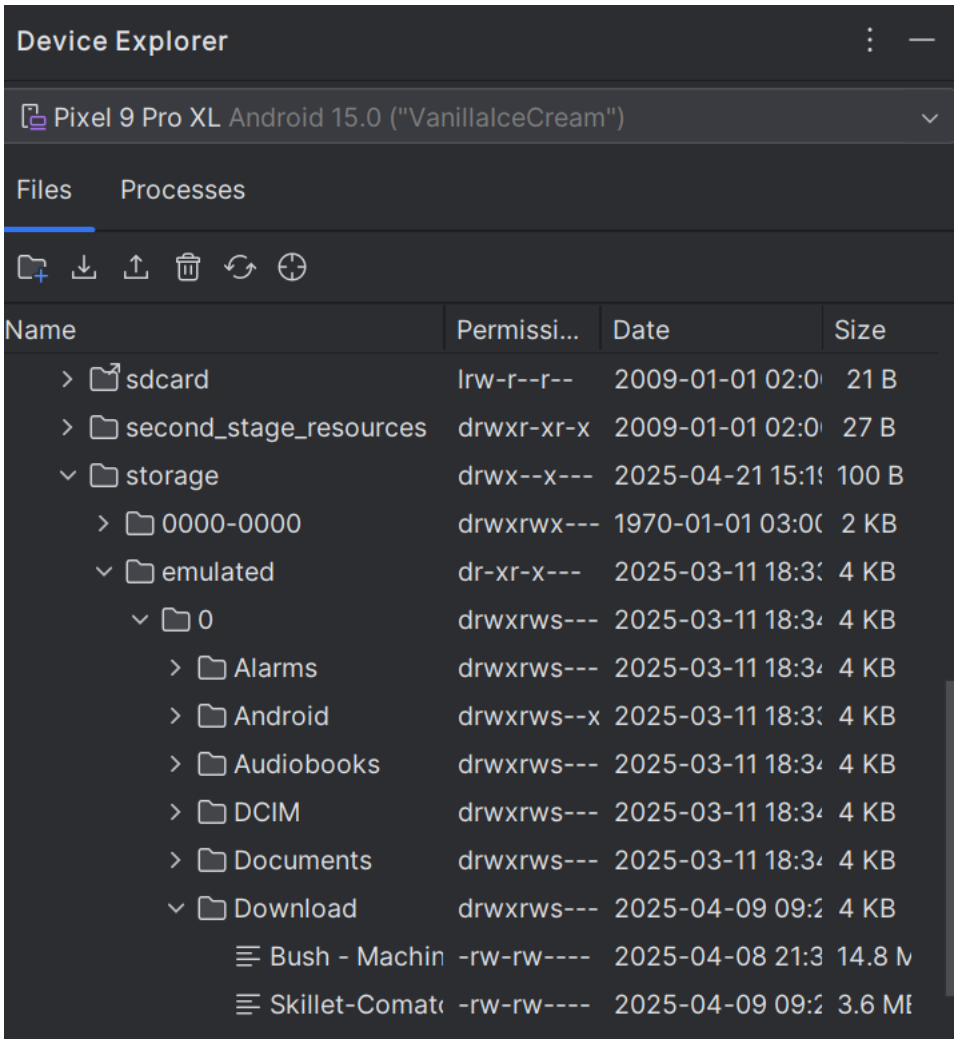
```
android:id="@+id/playVideoOnlineButton"
android:layout_width="match_parent"
android:layout_height="wrap_content"
android:text="Play Video Online"
android:visibility="gone" />
```

Ось, як виглядає початковий екран:

При натисненні на “PLAY AUDIO” або “PLAY VIDEO” – відкривається меню сховища пристрою, де можна обрати необхідні файли:

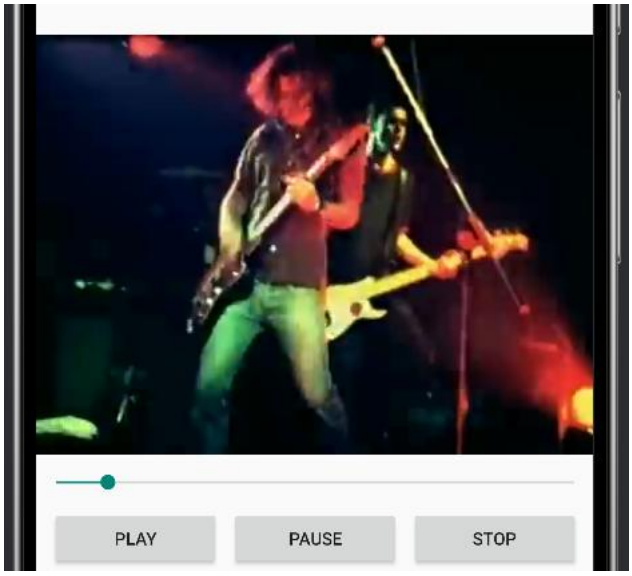


Файли розташовані за наступник шляхів в Device Explorer:



Ось так виглядає меню самого аудіо-плеєра:

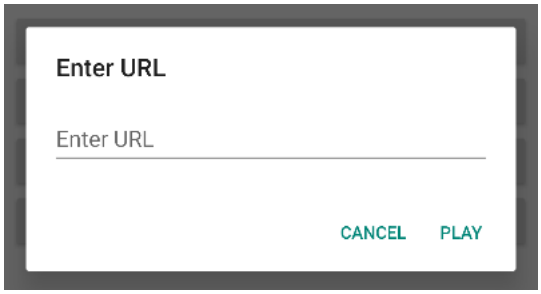
Ось так виглядає меню відео-плеєра (videoView):



Та при повороті пристрою відкриття відео на повний екран (наскільки можливо з MATCH_PARENT):

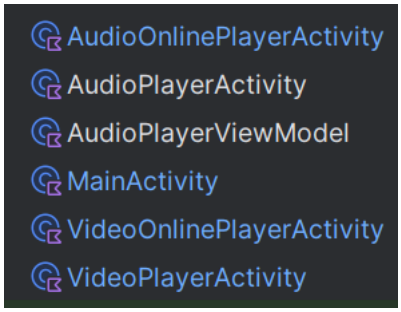
Якщо «показати» дві інші кнопки, то повинно спливати додаткове підменю, куди потрібно вставляти посилання на відео та обробляти стан, коли url немає, але воно не коректне:

```
android:visibility="visible"
```

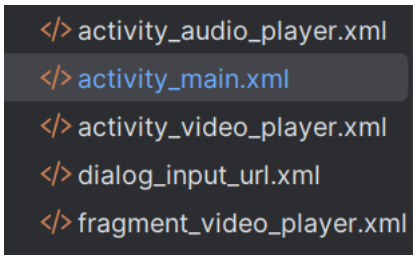


Код програми

Класи, серед яких нас цікавлять “AudioPlayerActivity.kt”, “AudioPlayerViewModel.kt”, “VideoPlayerActivity.kt” та “MainActivity.kt”, бо класи з приставкою «Online» в назві – не готові:



А тут цікавлять перші три layouts:



Та AndroidManifest.xml з дозволами на доступ до інтернету та читання з сховища:

```
<?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.INTERNET" />
    <uses-permission
android:name="android.permission.READ_EXTERNAL_STORAGE"/>
    <application
        android:allowBackup="true"
        android:dataExtractionRules="@xml/data_extraction_rules"
        android:fullBackupContent="@xml/backup_rules"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:theme="@style/Theme.MyApplication"
```

```
        tools:targetApi="31">
        <activity
            android:name=".MainActivity"
            android:exported="true"
            android:label="@string/app_name"
            android:theme="@style/Theme.MyApplication">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER"
/>
            </intent-filter>
        </activity>
        <activity android:name=".AudioPlayerActivity" />
        <activity android:name=".VideoPlayerActivity"
            android:configChanges="orientation|keyboardHidden|screenSize"/>
    </application>

</manifest>
```

Реалізація аудіо-плеєра розбита на два класи “AudioPlayerActivity.kt”:

```
package com.example.myapplication

import android.net.Uri
import android.os.Bundle
import android.widget.SeekBar
import androidx.activity.viewModels
import androidx.appcompat.app.AppCompatActivity
import androidx.core.net.toUri
import com.example.myapplication.databinding.ActivityAudioPlayerBinding

class AudioPlayerActivity : AppCompatActivity() {

    private lateinit var binding: ActivityAudioPlayerBinding
    private val viewModel: AudioPlayerViewModel by viewModels()
    private lateinit var mediaUri: Uri
    private var isUserSeeking = false

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        binding = ActivityAudioPlayerBinding.inflate(layoutInflater)
        setContentView(binding.root)

        mediaUri = intent.getStringExtra("mediaUri")?.toUri() ?: return

        if (!viewModel.isPrepared) {
            viewModel.initMediaPlayer(mediaUri)
        }

        binding.seekBar.max = 100
        viewModel.setSeekBarUpdateListener { progress ->
            if (!isUserSeeking) {
                binding.seekBar.progress = progress
            }
        }

        binding.seekBar.setOnSeekBarChangeListener(object :
SeekBar.OnSeekBarChangeListener {
            override fun onStartTrackingTouch(seekBar: SeekBar?) {
```

```
                isUserSeeking = true
            }

            override fun onStopTrackingTouch(seekBar: SeekBar?) {
                isUserSeeking = false
                viewModel.getDuration().takeIf { it > 0 }?.let { duration -
>
                    val newPos = binding.seekBar.progress * duration / 100
                    viewModel.seekTo(newPos)
                }
            }

            override fun onProgressChanged(seekBar: SeekBar?, progress:
Int, fromUser: Boolean) {}
        })

        binding.playButton.setOnClickListener {
            viewModel.play()
        }

        binding.pauseButton.setOnClickListener {
            viewModel.pause()
        }

        binding.stopButton.setOnClickListener {
            viewModel.stop(mediaUri)
        }
    }
}
```

та “AudioPlayerViewModel.kt”:

```
package com.example.myapplication

import android.app.Application
import android.media.MediaPlayer
import android.net.Uri
import android.os.Handler
import android.os.Looper
import androidx.lifecycle.AndroidViewModel

class AudioPlayerViewModel(application: Application) :
AndroidViewModel(application) {

    private var mediaPlayer: MediaPlayer? = null
    private var handler: Handler = Handler(Looper.getMainLooper())
    private var updateCallback: ((Int) -> Unit)? = null

    var isPrepared = false
        private set

    fun initMediaPlayer(uri: Uri) {
        if (mediaPlayer == null) {
            mediaPlayer = MediaPlayer().apply {
                setDataSource(getApplication(), uri)
                prepare()
                isPrepared = true
                setOnCompletionListener {
                    stopSeekBarUpdates()
                    updateCallback?.invoke(0)
```

```
                }
            }
        }
    }

    fun play() {
        mediaPlayer?.start()
        startSeekBarUpdates()
    }

    fun pause() {
        mediaPlayer?.pause()
    }

    fun stop(uri: Uri) {
        mediaPlayer?.apply {
            stop()
            reset()
            setDataSource(getApplication(), uri)
            prepare()
        }
        updateCallback?.invoke(0)
        stopSeekBarUpdates()
    }

    fun isPlaying(): Boolean {
        return mediaPlayer?.isPlaying == true
    }

    fun getDuration(): Int {
        return mediaPlayer?.duration ?: 0
    }

    fun getCurrentPosition(): Int {
        return mediaPlayer?.currentPosition ?: 0
    }

    fun seekTo(position: Int) {
        mediaPlayer?.seekTo(position)
    }

    fun setSeekBarUpdateListener(callback: (Int) -> Unit) {
        updateCallback = callback
    }

    private fun startSeekBarUpdates() {
        handler.post(object : Runnable {
            override fun run() {
                if (mediaPlayer?.isPlaying == true) {
                    val progress = (mediaPlayer!!.currentPosition * 100) /
mediaPlayer!!.duration
                    updateCallback?.invoke(progress)
                    handler.postDelayed(this, 1000)
                }
            }
        })
    }

    private fun stopSeekBarUpdates() {
        handler.removeCallbacksAndMessages(null)
    }
```

```
    override fun onCleared() {
        super.onCleared()
        mediaPlayer?.release()
        mediaPlayer = null
        stopSeekBarUpdates()
    }
}
```

Відео плеєр реалізовано одним класом “VideoPlayerActivity.kt”, в якому є всі необхідні адаптації для зміни орієнтації пристрою та програвання (функції onConfigurationChanged, adjustLayoutForOrientation, onSaveInstanceState, onRestoreInstanceState, etc.)

та

<activity android:name=".VideoPlayerActivity" android:configChanges="orientation|keyboardHidden|screenSize"/> з маніфесту:

```
package com.example.myapplication

import android.content.res.Configuration

import android.os.Bundle
import android.view.View
import android.widget.LinearLayout
import androidx.appcompat.app.AppCompatActivity
import androidx.core.net.toUri
import com.example.myapplication.databinding.ActivityVideoPlayerBinding

class VideoPlayerActivity : AppCompatActivity() {

    private lateinit var binding: ActivityVideoPlayerBinding

    private var currentPosition: Int = 0
    private var isPlaying: Boolean = false

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        binding = ActivityVideoPlayerBinding.inflate(layoutInflater)
        setContentView(binding.root)

        val mediaUri = intent.getStringExtra("mediaUri")?.toUri()
        binding.videoView.setVideoURI(mediaUri)

        binding.seekBar.max = 100

        binding.seekBar.setOnSeekBarChangeListener(object :
android.widget.SeekBar.OnSeekBarChangeListener {
            override fun onProgressChanged(seekBar:
android.widget.SeekBar?, progress: Int, fromUser: Boolean) {
                if (fromUser && binding.videoView.duration > 0) {
                    binding.videoView.seekTo(progress *
binding.videoView.duration / 100)
                }
            }

            override fun onStartTrackingTouch(seekBar:
```

```
android.widget.SeekBar?) {}
            override fun onStopTrackingTouch(seekBar:
android.widget.SeekBar?) {}
        })

        binding.playButton.setOnClickListener {
            if (!binding.videoView.isPlaying) {
                binding.videoView.start()
                isPlaying = true
                updateSeekBar()
            }
        }

        binding.pauseButton.setOnClickListener {
            if (binding.videoView.isPlaying) {
                binding.videoView.pause()
                isPlaying = false
            }
        }

        binding.stopButton.setOnClickListener {
            binding.videoView.pause()
            binding.videoView.stopPlayback()
            binding.videoView.setVideoURI(mediaUri)
            isPlaying = false
            currentPosition = 0
            binding.seekBar.progress = 0
        }

        // Повернення стану
        if (savedInstanceState != null) {
            currentPosition = savedInstanceState.getInt("currentPosition",
0)
            isPlaying = savedInstanceState.getBoolean("isPlaying", false)
            binding.videoView.seekTo(currentPosition)
            if (isPlaying) {
                binding.videoView.start()
                updateSeekBar()
            }
        }

        // Перевірка орієнтації
        adjustLayoutForOrientation(resources.configuration.orientation)
    }

    private fun updateSeekBar() {
        Thread {
            while (isPlaying) {
                runOnUiThread {
                    try {
                        // Перевіряємо, чи відео все ще відтворюється
                        if (binding.videoView.isPlaying) {
                            binding.seekBar.progress =
binding.videoView.currentPosition * 100 / binding.videoView.duration
                        }
                    } catch (e: Exception) {
                        // Логуємо помилку, якщо вона виникає
                        e.printStackTrace()
                    }
                }
                Thread.sleep(100)
```

```
            }
        }.start()
    }

    override fun onConfigurationChanged(newConfig: Configuration) {
        super.onConfigurationChanged(newConfig)
        adjustLayoutForOrientation(newConfig.orientation)
    }

    private fun adjustLayoutForOrientation(orientation: Int) {
        if (orientation == Configuration.ORIENTATION_LANDSCAPE) {
            // Повноекранний режим - кнопки сховані
            binding.seekBar.visibility = View.GONE
            binding.controlsLayout.visibility = View.GONE
            supportActionBar?.hide()

            // Розтягнути відео на весь екран
            val layoutParams = binding.videoView.layoutParams
            layoutParams.width = LinearLayout.LayoutParams.MATCH_PARENT
            layoutParams.height = LinearLayout.LayoutParams.MATCH_PARENT
            binding.videoView.layoutParams = layoutParams

        } else {
            // Портретний режим — кнопки видно
            binding.seekBar.visibility = View.VISIBLE
            binding.controlsLayout.visibility = View.VISIBLE
            supportActionBar?.show()

            // Розміри відео для портретного режиму
            val layoutParams = binding.videoView.layoutParams
            layoutParams.width = LinearLayout.LayoutParams.MATCH_PARENT
            layoutParams.height = LinearLayout.LayoutParams.MATCH_PARENT
            binding.videoView.layoutParams = layoutParams
        }
    }

    override fun onSaveInstanceState(outState: Bundle) {
        super.onSaveInstanceState(outState)
        outState.putInt("currentPosition",
binding.videoView.currentPosition)
        outState.putBoolean("isPlaying", binding.videoView.isPlaying)
    }

    override fun onRestoreInstanceState(savedInstanceState: Bundle) {
        super.onRestoreInstanceState(savedInstanceState)
        currentPosition = savedInstanceState.getInt("currentPosition", 0)
        isPlaying = savedInstanceState.getBoolean("isPlaying", false)
        binding.videoView.seekTo(currentPosition)
        if (isPlaying) {
            binding.videoView.start()
            updateSeekBar()
        }
    }

    override fun onDestroy() {
        super.onDestroy()
        binding.videoView.stopPlayback()
    }
}
```

MainActivity.kt

```
package com.example.myapplication

import android.content.Intent
import android.os.Bundle
import android.widget.Button
import android.widget.EditText
import androidx.activity.result.contract.ActivityResultContracts
import androidx.appcompat.app.AlertDialog
import androidx.appcompat.app.AppCompatActivity

class MainActivity : AppCompatActivity() {

    private lateinit var playAudioButton: Button
    private lateinit var playVideoButton: Button
    private lateinit var playAudioOnlineButton: Button
    private lateinit var playVideoOnlineButton: Button

    // Реєстрація вибору аудіофайлів
    private val audioPicker =
registerForActivityResult(ActivityResultContracts.GetContent()) { uri ->
        uri?.let {
            val intent = Intent(this, AudioPlayerActivity::class.java)
            intent.putExtra("mediaUri", it.toString())
            startActivity(intent)
        }
    }

    // Реєстрація вибору відеофайлів
    private val videoPicker =
registerForActivityResult(ActivityResultContracts.GetContent()) { uri ->
        uri?.let {
            val intent = Intent(this, VideoPlayerActivity::class.java)
            intent.putExtra("mediaUri", it.toString())
            startActivity(intent)
        }
    }

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)

        playAudioButton = findViewById(R.id.playAudioButton)
        playVideoButton = findViewById(R.id.playVideoButton)
        playAudioOnlineButton = findViewById(R.id.playAudioOnlineButton)
        playVideoOnlineButton = findViewById(R.id.playVideoOnlineButton)

        playAudioButton.setOnClickListener {
            audioPicker.launch("audio/*")
        }

        playVideoButton.setOnClickListener {
            videoPicker.launch("video/*")
        }

        playAudioOnlineButton.setOnClickListener {
            showMediaDialog("audio")
        }

        playVideoOnlineButton.setOnClickListener {
```

```
            showMediaDialog("video")
        }

        // Приховані кнопки для онлайн-програвання (на всякий випадок, хоча
вони вже в XML з visibility="gone")
        //playAudioOnlineButton.visibility = android.view.View.GONE
        //playVideoOnlineButton.visibility = android.view.View.GONE
    }

    private fun showMediaDialog(mediaType: String) {
        val dialogView = layoutInflater.inflate(R.layout.dialog_input_url,
null)
        val input = dialogView.findViewById<EditText>(R.id.urlEditText)

        val builder = AlertDialog.Builder(this)
            .setTitle("Enter URL")
            .setView(dialogView)
            .setPositiveButton("Play") { _, _ ->
                val url = input.text.toString()
                if (mediaType == "audio") {
                    playAudioOnline(url)
                } else {
                    playVideoOnline(url)
                }
            }
            .setNegativeButton("Cancel", null)

        builder.show()
    }

    private fun playAudioOnline(url: String) {
        val intent = Intent(this, AudioOnlinePlayerActivity::class.java)
        intent.putExtra("mediaUri", url)
        startActivity(intent)
    }

    private fun playVideoOnline(url: String) {
        val intent = Intent(this, VideoOnlinePlayerActivity::class.java)
        intent.putExtra("mediaUri", url)
        startActivity(intent)
    }
}
```

Інші .xml файли додавати не став, бо і так надто багато сторінок вийшло.

Контрольні питання

1. Наведіть способи підключення Інтернет ресурсів до мобільного застосунку.

   • **HTTP-запити вручну:** через `HttpURLConnection` або `OkHttp` (бібліотека для HTTP-запитів).
   • **REST API:** підключення до серверів через RESTful API, часто із використанням бібліотек.

- **Retrofit:** популярна бібліотека для спрощеної роботи з API та автоматичного парсингу JSON.
- **Volley:** бібліотека від Google для роботи з мережевими запитами (HTTP, JSON, зображення).
- **Firebase:** надає доступ до онлайн-бази даних, хмарного сховища, аутентифікації тощо.
- **WebSockets:** для підключення до серверів у реальному часі (чат, ігри, тощо).
- **GraphQL:** альтернативний підхід до REST для запитів до серверу.

2. Поясніть різницю між внутрішнім та зовнішнім сховищем.

Властивість	Внутрішнє сховище	Зовнішнє сховище
Доступність	Доступне лише цьому застосунку	Може бути доступне іншим застосункам та користувачу
Захищеність	Надійно захищене	Менш захищене, особливо без дозволів
Приклад API	`getFilesDir()`, `getCacheDir()`	`getExternalFilesDir()`, `Environment.getExternalStorageDirectory()`
Необхідні дозволи	Не потрібні	Потрібні дозволи (`READ/WRITE_EXTERNAL_STORAGE`) для Android < 10

3. Наведіть категорії файлів при збереженні в зовнішньому сховищі.

Environment.DIRECTORY_MUSIC – для музики
Environment.DIRECTORY_PODCASTS – для подкастів
Environment.DIRECTORY_RINGTONES – для рингтонів
Environment.DIRECTORY_ALARMS – для будильників
Environment.DIRECTORY_NOTIFICATIONS – для звуків сповіщень
Environment.DIRECTORY_PICTURES – для зображень
Environment.DIRECTORY_MOVIES – для відео
Environment.DIRECTORY_DOWNLOADS – для завантажень
Environment.DIRECTORY_DOCUMENTS – для документів

4. Опишіть властивості спеціалізованих інструментів для відтворення аудіо-файлів.

- **MediaPlayer** (вбудований Android-клас):
    - Підтримує формати: MP3, AAC, WAV тощо
    - Можна запускати з ресурсу, Uri чи потоку
    - Підтримка станів: `prepare()`, `start()`, `pause()`, `stop()`
- **ExoPlayer**:
    - Більш гнучкий, ніж MediaPlayer
    - Підтримка потокового аудіо, DASH, HLS

- Кастомні контролери, буферизація, адаптивний стрімінг

- **SoundPool**:

  - Для коротких звуків (ефекти, ноти)
  - Низька затримка

5. Опишіть властивості спеціалізованих інструментів для відтворення відео-файлів.

   - **VideoView**:

     - Простий у використанні
     - Працює з MediaPlayer
     - Обмежений функціонал (немає кастомного контролю)

   - **ExoPlayer + PlayerView**:

     - Гнучкість у налаштуванні
     - Підтримка багатьох відеоформатів (MP4, MKV, HLS, DASH)
     - Повний контроль над UI (Play/Pause, Seek, Buffering)
     - Підтримка субтитрів, реклами, DRM