# Kotlin & Android w Android Studio Narwhal 3

**(2025.11.30)**
**Oleksii Nawrocki**

## Wykonywanie zdjęć, nagrywanie filmów i dźwięku

### Wprowadzenie

Obsługa multimediów to kluczowa funkcja wielu aplikacji mobilnych. Android udostępnia mechanizmy pozwalające na integrację aparatu i mikrofonu. Możemy to zrobić na dwa główne sposoby:

1. **Intencje (Intents):** Wykorzystanie systemowej aplikacji aparatu do zrobienia zdjęcia lub nagrania filmu. Jest to najprostsza metoda, nie wymagająca skomplikowanej obsługi sprzętowej.
2. **API Camera2 / CameraX:** Bezpośrednia kontrola nad sprzętem kamery (bardziej zaawansowane).
3. **MediaRecorder:** Klasa służąca do bezpośredniego nagrywania dźwięku i wideo.

W tym laboratorium skupimy się na **Intencjach** do obsługi zdjęć i wideo (korzystając z systemowej aplikacji aparatu) oraz klasie **MediaRecorder** do nagrywania dźwięku.

### Kluczowe pojęcia

- **FileProvider:** Specjalny mechanizm, który ułatwia bezpieczne udostępnianie plików powiązanych z naszą aplikacją innym aplikacjom (np. aplikacji aparatu). Od Androida 7.0 (API 24) bezpośrednie przesyłanie ścieżek plików (file://) jest zabronione. Zamiast tego używamy bezpiecznego URI typu content://.
- **Scoped Storage:** Nowoczesny model dostępu do pamięci, który promuje zapisywanie plików w prywatnych katalogach aplikacji (Android/data/...), co zwiększa bezpieczeństwo i nie wymaga drastycznych uprawnień w nowszych systemach.
- **ViewBinding:** Mechanizm pozwalający na bezpieczniejszy i szybszy dostęp do widoków w kodzie (zamiast findViewById).

## Krok 1: Konfiguracja Uprawnień (AndroidManifest.xml)

Aby aplikacja mogła korzystać z aparatu i mikrofonu, musimy zadeklarować odpowiednie uprawnienia w pliku AndroidManifest.xml. Wymagane są uprawnienia do kamery (CAMERA), nagrywania dźwięku (RECORD_AUDIO) oraz odpowiednie uprawnienia do odczytu mediów w zależności od wersji systemu Android (API 33+ wymaga READ_MEDIA_...).

```
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools">

    <!-- Hardware Features -->
    <uses-feature android:name="android.hardware.camera" android:required="true" />

    <!-- Standard Permissions -->
    <uses-permission android:name="android.permission.CAMERA" />
    <uses-permission android:name="android.permission.RECORD_AUDIO" />

    <!-- Storage Permissions (Android 13+ / API 33+) -->
    <uses-permission android:name="android.permission.READ_MEDIA_IMAGES" />
    <uses-permission android:name="android.permission.READ_MEDIA_VIDEO" />
    <uses-permission android:name="android.permission.READ_MEDIA_AUDIO" />

    <application
```

Dodatkowo musimy zarejestrować FileProvider w sekcji <application>, aby móc bezpiecznie udostępniać pliki (zdjęcia/wideo) zewnętrznej aplikacji aparatu. Ważne jest również ustawienie odpowiedniego motywu aplikacji (Theme.AppCompat), aby uniknąć błędów przy korzystaniu z AppCompatActivity.

```
    <provider
        android:name="androidx.core.content.FileProvider"
        android:authorities="${applicationId}.provider"
        android:exported="false"
        android:grantUriPermissions="true">
        <meta-data
            android:name="android.support.FILE_PROVIDER_PATHS"
            android:resource="@xml/file_paths" />
    </provider>
```

**ZADANIE**
Otwórz plik app/manifests/AndroidManifest.xml. Zmodyfikuj go, dodając uprawnienia, konfigurację FileProvider oraz zmieniając motyw aplikacji na “@style/Theme.AppCompat.DayNight.DarkActionBar”.

## Krok 2: Konfiguracja Ścieżek Plików (FileProvider)

Musimy stworzyć plik XML, który powie systemowi, które foldery naszej aplikacji są "publiczne" dla innych aplikacji (np. dla kamery).

1. W drzewie projektu kliknij prawym przyciskiem myszy na folder **res**.
2. Wybierz **New -> Android Resource Directory**.
3. Wybierz typ **xml** i kliknij OK.
4. W folderze res/xml utwórz nowy plik: **New -> XML Resource File**.
5. Nazwij go **file_paths**.

```xml
<?xml version="1.0" encoding="utf-8"?>
<paths>
    <!-- Udostepniamy katalog zewnetrzny plików naszej aplikacji -->
    <external-files-path name="my_images" path="/" />
</paths>
```

**ZADANIE**
Stwórz plik XML, który powie systemowi, które foldery naszej aplikacji są "publiczne" dla innych aplikacji.

## Krok 3: Włączenie ViewBinding

Użyjemy ViewBinding, aby kod był czystszy i bezpieczniejszy. Dla tego przechodzimy do “build.gradle” i dodajemy poniższy kod.

```
    kotlinOptions {
        jvmTarget = "11"
    }
    buildFeatures {
        compose = true
        viewBinding = true
    }
}
```

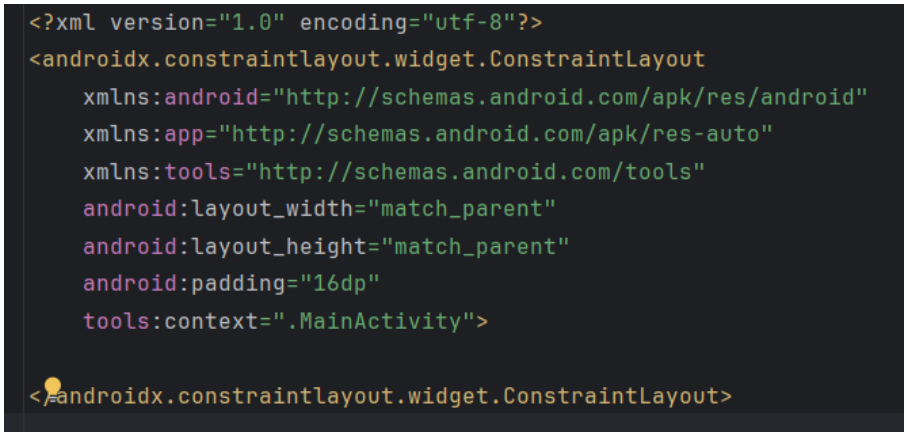
**ZADANIE**
Włącz ViewBinding.

## Krok 4: Tworzenie Layoutu (Interfejs)

Zaprojektujemy interfejs użytkownika krok po kroku, używając ConstraintLayout.

### Krok 4.1: Główny Kontener

Otwórz plik res/layout/activity_main.xml. Usuń całą zawartość i wstaw podstawowy kontener ConstraintLayout



```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:padding="16dp"
    tools:context=".MainActivity">

</androidx.constraintlayout.widget.ConstraintLayout>
```

**ZADANIE**
Otwórz plik res/layout/activity_main.xml i stwórz glówny kontener

widoku.

**Krok 4.2: Przyciski Foto i Wideo**

Dodaj dwa przyciski wewnątrz `ConstraintLayout`. Pierwszy przypinamy do góry ekranu, a drugi pod nim.

```
<Button
    android:id="@+id/btnTakePhoto"
    android:layout_width="0dp"
    android:layout_height="wrap_content"
    android:text="Zrób Zdjecie"
    app:layout_constraintTop_toTopOf="parent"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintEnd_toEndOf="parent"
    android:layout_marginTop="16dp"/>

<Button
    android:id="@+id/btnRecordVideo"
    android:layout_width="0dp"
    android:layout_height="wrap_content"
    android:text="Nagraj Wideo"
    app:layout_constraintTop_toBottomOf="@id/btnTakePhoto"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintEnd_toEndOf="parent"
    android:layout_marginTop="8dp"/>
```

**ZADANIE**
Dodaj przeciski Foto i Wideo

**Krok 4.3: Sekcja Audio**

Dodaj poziomy LinearLayout zawierający dwa przyciski (Start/Stop) do obsługi nagrywania dźwięku. Umieść go pod przyciskiem wideo.

```
<LinearLayout
    android:id="@+id/audioButtonsLayout"
    android:layout_width="0dp"
    android:layout_height="wrap_content"
    android:orientation="horizontal"
    android:weightSum="2"
    app:layout_constraintTop_toBottomOf="@id/btnRecordVideo"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintEnd_toEndOf="parent"
    android:layout_marginTop="16dp">

    <Button
        android:id="@+id/btnStartAudio"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:layout_weight="1"
        android:text="Start Audio"
        android:layout_marginEnd="4dp"/>

    <Button
        android:id="@+id/btnStopAudio"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:layout_weight="1"
        android:text="Stop Audio"
        android:enabled="false"
        android:layout_marginStart="4dp"/>
</LinearLayout>
```

**ZADANIE**
Zrób sekcję audio.

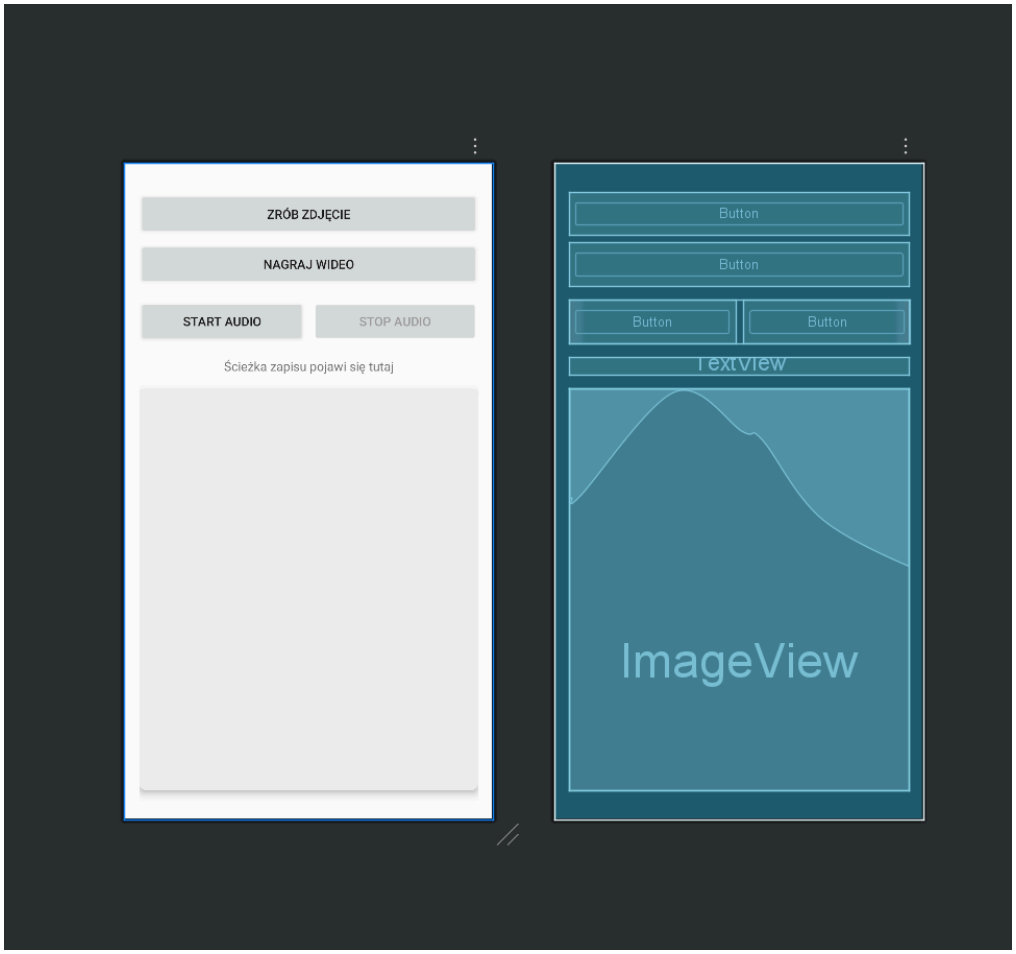
### Krok 4.4: Informacje i Podgląd

Na koniec dodaj pole tekstowe wyświetlające status oraz CardView (kontener z zaokrąglonymi rogami), w którym umieścimy ImageView oraz VideoView do podglądu mediów.

```
<!-- Container for Previews (Image & Video) -->
<androidx.cardview.widget.CardView
    android:id="@+id/previewContainer"
    android:layout_width="0dp"
    android:layout_height="0dp"
    app:cardCornerRadius="8dp"
    app:cardElevation="4dp"
    app:layout_constraintTop_toBottomOf="@id/tvInfo"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintBottom_toBottomOf="parent"
    android:layout_marginTop="16dp"
    android:layout_marginBottom="16dp">

    <FrameLayout
        android:layout_width="match_parent"
        android:layout_height="match_parent">

        <ImageView
            android:id="@+id/imageView"
            android:layout_width="match_parent"
            android:layout_height="match_parent"
            android:background="#EEEEEE"
            android:scaleType="fitCenter"
            android:contentDescription="Podgląd zdjęcia" />

        <VideoView
            android:id="@+id/videoView"
            android:layout_width="match_parent"
            android:layout_height="match_parent"
            android:visibility="gone"
            android:layout_gravity="center"/>

    </FrameLayout>
</androidx.cardview.widget.CardView>
```

**ZADANIE**
Zrób sekcję z informacją i podglądem.

Na końcu powinniśmy otrzymać następny widok:



## Krok 5: Logika Aplikacji (MainActivity.kt)

Przystępujemy do programowania logiki w języku Kotlin. Na początek musimy przygotować klasę MainActivity. Zdefiniujemy niezbędne zmienne do przechowywania URI plików (zdjęcia/wideo), obsługi nagrywania dźwięku oraz inicjalizacji bindingu. W metodzie onCreate ukryjemy również pasek akcji (ActionBar), aby interfejs wyglądał czyściej i nie zasłaniał widoku kamery.

```
class MainActivity : AppCompatActivity() {
    19 Usages
    private lateinit var binding: ActivityMainBinding // ViewBinding
    5 Usages
    private var currentPhotoUri: Uri? = null
    5 Usages
    private var currentVideoUri: Uri? = null
    3 Usages
    private var mediaRecorder: MediaRecorder? = null
    3 Usages
    private var audioFilePath: String = ""
    3 Usages
    private var isRecordingAudio = false

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)

        // Fix: Hide the top bar (ActionBar) so it doesn't overlap the layout
        supportActionBar?.hide()

        // Initialize ViewBinding
        binding = ActivityMainBinding.inflate(layoutInflater)
        setContentView(binding.root)

        // Set up click listeners using binding
        binding.btnTakePhoto.setOnClickListener {
            if (checkPermissions()) takePhoto()
        }

        binding.btnRecordVideo.setOnClickListener {
            if (checkPermissions()) recordVideo()
        }

        binding.btnStartAudio.setOnClickListener {
            if (checkPermissions()) startAudioRecording()
        }

        binding.btnStopAudio.setOnClickListener {
            stopAudioRecording()
        }
    }
```

**ZADANIE 5.1: Przygotowanie Klasy i Zmiennych**
Otwórz plik MainActivity.kt. Dodaj niezbędne importy, zmienne klasowe oraz inicjalizację ViewBinding. Ukryjemy też domyślny pasek aplikacji (ActionBar).

Android wymaga, aby aplikacje prosiły użytkownika o wrażliwe uprawnienia (np. dostęp do kamery, mikrofonu) w czasie działania

aplikacji (Runtime Permissions). Stworzymy metodę checkPermissions, która sprawdzi listę wymaganych uprawnień. Jeśli jakieś uprawnienie nie zostało jeszcze przyznane, metoda poprosi o nie użytkownika.

```
3 Usages
private fun checkPermissions(): Boolean {
    val permissions = mutableListOf<String>()

    permissions.add(Manifest.permission.CAMERA)
    permissions.add(Manifest.permission.RECORD_AUDIO)

    // Android 13+ (API 33) uses different permissions for media
    if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.TIRAMISU) {
        permissions.add(Manifest.permission.READ_MEDIA_IMAGES)
        permissions.add(Manifest.permission.READ_MEDIA_VIDEO)
        permissions.add(Manifest.permission.READ_MEDIA_AUDIO)
    } else {
        permissions.add(Manifest.permission.WRITE_EXTERNAL_STORAGE)
        permissions.add(Manifest.permission.READ_EXTERNAL_STORAGE)
    }

    val listPermissionsNeeded = ArrayList<String>()
    for (p in permissions) {
        if (ContextCompat.checkSelfPermission( context = this, permission = p) != PackageManager.PERMISSION_GRANTED) {
            listPermissionsNeeded.add(p)
        }
    }

    if (listPermissionsNeeded.isNotEmpty()) {
        ActivityCompat.requestPermissions( activity = this, permissions = listPermissionsNeeded.toTypedArray(), requestCode = 100)
        return false
    }
    return true
}
}
```

**ZADANIE 5.2: Obsługa Uprawnień**
Dodaj metodę checkPermissions na końcu klasy MainActivity. Będzie ona sprawdzać, czy mamy wymagane uprawnienia, a jeśli nie – poprosi o nie użytkownika.

Aby zrobić zdjęcie, wykorzystamy systemową aplikację aparatu. Proces składa się z trzech kroków:

1. **Stworzenie pliku:** Metoda createImageFile utworzy pusty plik w pamięci urządzenia, do którego aparat zapisze zdjęcie.
2. **Wywołanie intencji:** Metoda takePhoto wygeneruje bezpieczny adres URI do tego pliku za pomocą FileProvider i uruchomi aparat.
3. **Odebranie wyniku:** takePhotoLauncher to callback, który zostanie wywołany, gdy aparat zakończy pracę. Jeśli zdjęcie zostało zrobione pomyślnie, wyświetlimy je w ImageView.

```
59          private val takePhotoLauncher = registerForActivityResult( contract = ActivityResultContracts.TakePicture()) { success ->
60              if (success && currentPhotoUri != null) {
61                  binding.imageView.visibility = android.view.View.VISIBLE
62                  binding.videoView.visibility = android.view.View.GONE
63                  binding.imageView.setImageURI(currentPhotoUri)
64
65                  val msg = "Zapisano zdjęcie:\n$currentPhotoUri"
66                  binding.tvInfo.text = msg
67                  Toast.makeText( context = this,  text = "Zapisano w: Android/data/com.example.multimedia/files/Pictures",  duration = Toast.LENGTH_LONG).show()
68              }
69          }
70
            1 Usage
71          private fun takePhoto() {
72              try {
73                  val photoFile = createImageFile()
74                  currentPhotoUri = FileProvider.getUriForFile(
75                      context = this,
76                      authority = "${applicationContext.packageName}.provider",
77                      photoFile
78                  )
79                  takePhotoLauncher.launch( input = currentPhotoUri)
80              } catch (e: Exception) {
81                  e.printStackTrace()
82                  Toast.makeText( context = this,  text = "Błąd kamery: ${e.message}",  duration = Toast.LENGTH_SHORT).show()
83              }
84          }
85
            1 Usage
86          private fun createImageFile(): File {
87              val timeStamp: String = SimpleDateFormat( pattern = "yyyyMMdd_HHmmss", Locale.getDefault()).format(Date())
88              val storageDir: File? = getExternalFilesDir( type = Environment.DIRECTORY_PICTURES)
89              return File.createTempFile( prefix = "JPEG_${timeStamp}_",  suffix = ".jpg",  directory = storageDir)
90          }
```

**ZADANIE 5.3: Implementacja Obsługi Zdjęć**
Dodaj logikę tworzenia pliku zdjęcia (createImageFile), wywoływania aparatu (takePhoto) oraz odbierania wyniku (takePhotoLauncher).

Mechanizm nagrywania wideo jest analogiczny do robienia zdjęć. Użyjemy innej intencji (CaptureVideo) oraz innego katalogu zapisu (DIRECTORY_MOVIES). Po nagraniu filmu wyświetlimy go w komponencie VideoView, który pozwala na odtwarzanie materiałów wideo.

```
1 Usage
private val takeVideoLauncher = registerForActivityResult( contract = ActivityResultContracts.CaptureVideo()) { success ->
    if (success && currentVideoUri != null) {
        binding.imageView.visibility = android.view.View.GONE
        binding.videoView.visibility = android.view.View.VISIBLE
        binding.videoView.setVideoURI(currentVideoUri)
        binding.videoView.start()

        val msg = "Zapisano wideo:\n$currentVideoUri"
        binding.tvInfo.text = msg
        Toast.makeText( context = this, text = "Zapisano w: Android/data/com.example.multimedia/files/Movies", duration = Toast.LENGTH_LONG).show()
    }
}

1 Usage
private fun recordVideo() {
    try {
        val videoFile = createVideoFile()
        currentVideoUri = FileProvider.getUriForFile(
            context = this,
            authority = "${applicationContext.packageName}.provider",
            videoFile
        )
        takeVideoLauncher.launch( input = currentVideoUri)
    } catch (e: Exception) {
        Toast.makeText( context = this, text = "Błąd wideo: ${e.message}", duration = Toast.LENGTH_SHORT).show()
    }
}

1 Usage
private fun createVideoFile(): File {
    val timeStamp: String = SimpleDateFormat( pattern = "yyyyMMdd_HHmmss", Locale.getDefault()).format(Date())
    val storageDir: File? = getExternalFilesDir( type = Environment.DIRECTORY_MOVIES)
    return File.createTempFile( prefix = "VIDEO_${timeStamp}_", suffix = ".mp4", directory = storageDir)
}
```

**ZADANIE 5.4: Implementacja Obsługi Wideo**
Podobnie jak przy zdjęciach, dodaj metody do obsługi nagrywania wideo (createVideoFile, recordVideo, takeVideoLauncher).

Do nagrywania dźwięku nie użyjemy zewnętrznej aplikacji, lecz klasy MediaRecorder wbudowanej w Android SDK. Pozwala ona na bezpośrednie przechwytywanie strumienia audio z mikrofonu i zapisywanie go do pliku. Zaimplementujemy metody startujące i zatrzymujące nagrywanie oraz metodę updateAudioUI, która będzie blokować/odblokowywać przyciski w zależności od stanu nagrywania.

```
1 Usage
private fun startAudioRecording() {
    val timeStamp: String = SimpleDateFormat( pattern = "yyyyMMdd_HHmmss", Locale.getDefault()).format(Date())
    val storageDir: File? = getExternalFilesDir( type = Environment.DIRECTORY_MUSIC)
    val audioFile = File.createTempFile( prefix = "AUDIO_${timeStamp}_", suffix = ".3gp", directory = storageDir)
    audioFilePath = audioFile.absolutePath

    mediaRecorder = MediaRecorder().apply {
        setAudioSource(MediaRecorder.AudioSource.MIC)
        setOutputFormat(MediaRecorder.OutputFormat.THREE_GPP)
        setAudioEncoder(MediaRecorder.AudioEncoder.AMR_NB)
        setOutputFile(audioFilePath)
        try {
            prepare()
            start()
            isRecordingAudio = true
            updateAudioUI( isRecording = true)
            binding.tvInfo.text = "Nagrywanie audio..."
        } catch (e: IOException) {
            e.printStackTrace()
        }
    }
}
```

```
1 Usage
148     private fun stopAudioRecording() {
149         if (isRecordingAudio) {
150             mediaRecorder?.apply {
151                 stop()
152                 release()
153             }
154             mediaRecorder = null
155             isRecordingAudio = false
156             updateAudioUI( isRecording = false)
157
158             val msg = "Zapisano audio:\n$audioFilePath"
159             binding.tvInfo.text = msg
160             Toast.makeText( context = this, text = "Nagranie zapisane!", duration = Toast.LENGTH_SHORT).show()
161         }
162     }
163
        2 Usages
164     private fun updateAudioUI(isRecording: Boolean) {
165         binding.btnStartAudio.isEnabled = !isRecording
166         binding.btnStopAudio.isEnabled = isRecording
167     }
```

**ZADANIE 5.5: Implementacja Nagrywania Dźwięku**
Dodaj metody obsługujące MediaRecorder: startAudioRecording, stopAudioRecording oraz pomocniczą updateAudioUI.

## Zakończenie

Po wykonaniu wszystkich kroków:

1. **Odinstaluj starą wersję aplikacji** z telefonu/emulatora (jeśli zmieniałeś nazwę pakietu).
2. Wykonaj **Sync Project with Gradle Files**.
3. Uruchom aplikację.

Powinna ona teraz poprawnie obsługiwać aparat, nagrywać filmy i dźwięk, a pliki znajdziesz w katalogu Android/data/<twoja.paczka>/files/.

**Powstala aplikacja:**