

Kotlin & Android w Android Studio Giraffe (2022.3.1)

Multimedia

Tym razem zajmiemy się multimediami w naszym telefonie. Utworzymy aplikację, która demonstrować będzie sposób obsługi aparatu fotograficznego, zajmiemy się też tworzeniem galerii wyświetlającej obrazy zawarte w naszym telefonie. Ponadto zbudujemy odtwarzacz plików dźwiękowych i filmów wideo.

Tworzenie aplikacji wykorzystującej multimedia

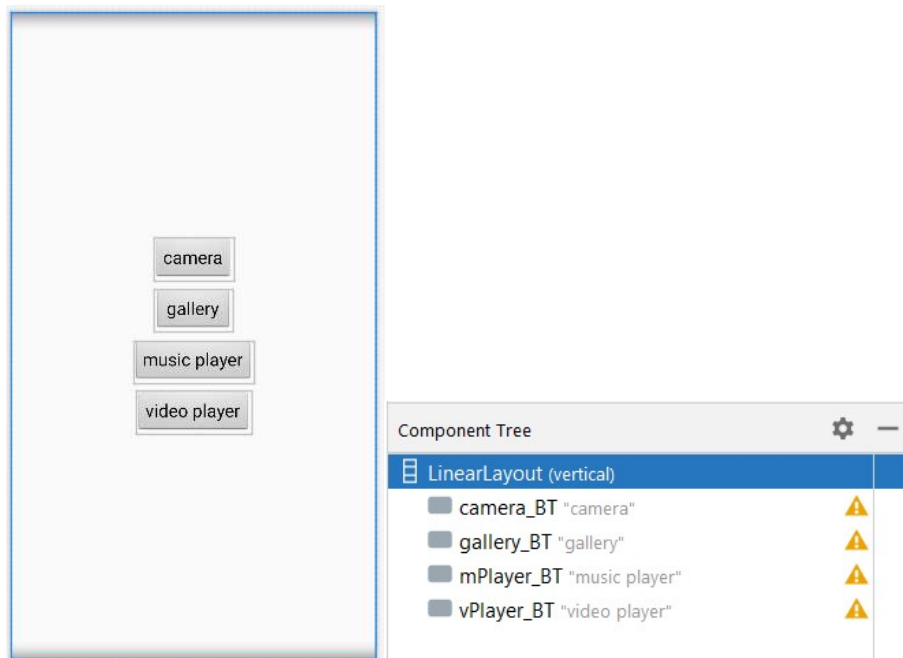
Nasza aplikacja bazować będzie na kilku aktywnościach i kilku layoutach. Zaczynamy od utworzenia głównego layoutu programu.

ZADANIE

Utwórz nowy projekt Android Studio o nazwie *MultimediaApp*, zawierający *EmptyActivity*. W pliku layoutu głównej aktywności użyj komponentu *LinearLayout(vertical)*. Ustaw dla layoutu parametr *gravity* na wartość *center*. Dodaj 4 przyciski *Button* zawierające następujące teksty:

- Camera (id = camera_BT);
- Gallery (id = gallery_BT);
- Music Player (id = mPlayer_BT);
- Video Player (id = vPlayer_BT).

Dla każdego przycisku ustaw: *layout_margin = 5dp*, *layout_width = wrap_content*, *textSize = 20sp*.



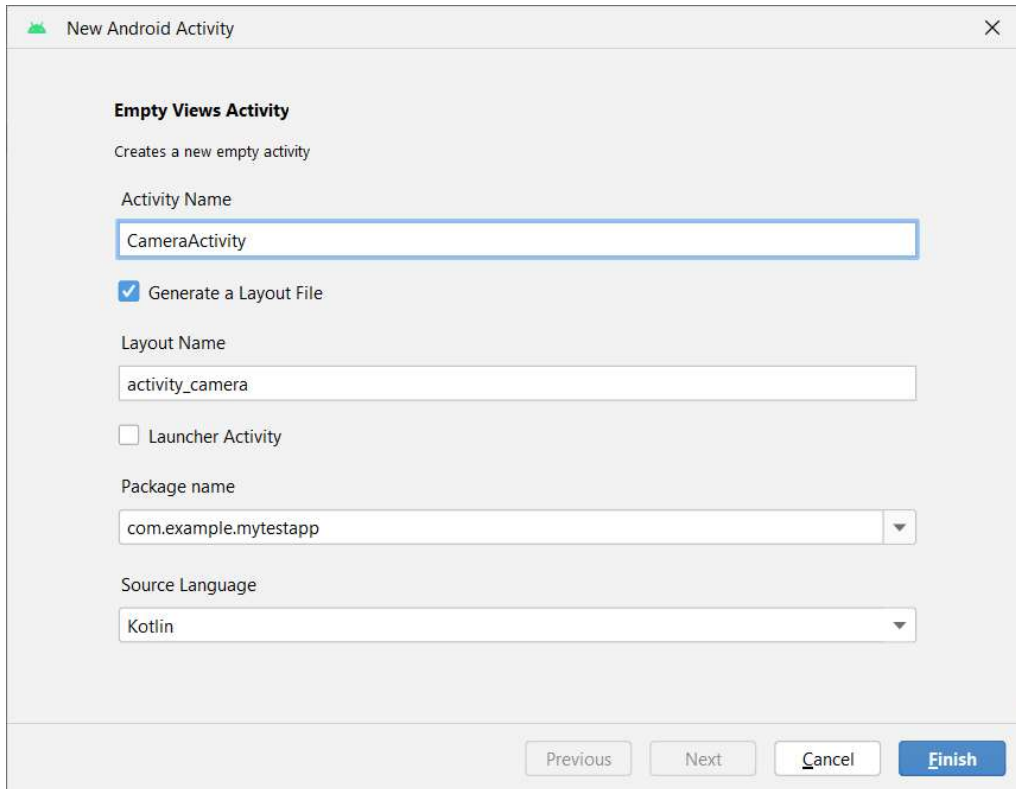
Przyciski posłużą nam do wywoływania aktywności odpowiedzialnych za realizację danych funkcjonalności naszej aplikacji.

Utwórzmy teraz pliki aktywności i layoutów dla każdej z funkcjonalności. Kod tych plików będziemy modyfikować nieco później.

ZADANIE

Utwórz w naszym projekcie 4 nowe pliki aktywności *EmptyActivity* o następujących nazwach

- *CameraActivity* (z plikiem layoutu *activity_camera*);
- *GalleryActivity* (z plikiem layoutu *activity_gallery*);
- *MusicActivity* (z plikiem layoutu *activity_music*);
- *VideoActivity* (z plikiem layoutu *activity_video*).



New Android Activity

Empty Views Activity
Creates a new empty activity

Activity Name
CameraActivity

☒ Generate a Layout File

Layout Name
activity_camera

☐ Launcher Activity

Package name
com.example.mytestapp

Source Language
Kotlin

Previous Next Cancel Finish

Przejdźmy do oprogramowania przycisków głównego layoutu aby uruchamiały wyżej utworzone aktywności.

ZADANIE

Zaimplementuj kod obsługi przycisków głównej aktywności.

```

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

    findViewById<Button>(R.id.camera_BT).setOnClickListener { it: View!
        val intentCamera = Intent(applicationContext, CameraActivity::class.java)
        startActivity(intentCamera)
    }

    findViewById<Button>(R.id.gallery_BT).setOnClickListener { it: View!
        val intentGallery = Intent(applicationContext, GalleryActivity::class.java)
        startActivity(intentGallery)
    }

    findViewById<Button>(R.id.mPlayer_BT).setOnClickListener { it: View!
        val intentMusic = Intent(applicationContext, MusicActivity::class.java)
        startActivity(intentMusic)
    }

    findViewById<Button>(R.id.vPlayer_BT).setOnClickListener { it: View!
        val intentVideo = Intent(applicationContext, VideoActivity::class.java)
        startActivity(intentVideo)
    }
}

```

Ponieważ w naszej aplikacji będziemy potrzebować dostępu do aparatu oraz do plików dodajmy w pliku manifestu odpowiednie uprawnienia.

ZADANIE

Dodaj w pliku *AndroidManifest* poniższy kod uprawnień naszej aplikacji:

```

<uses-permission android:name="android.permission.CAMERA" />
<uses-feature android:name="android.hardware.camera" android:required="true" />
<uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE" />
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />

```

Ponadto wewnątrz znacznika *<application* umieść kod:

```

<application
    android:preserveLegacyExternalStorage="true"
    android:requestLegacyExternalStorage="true"

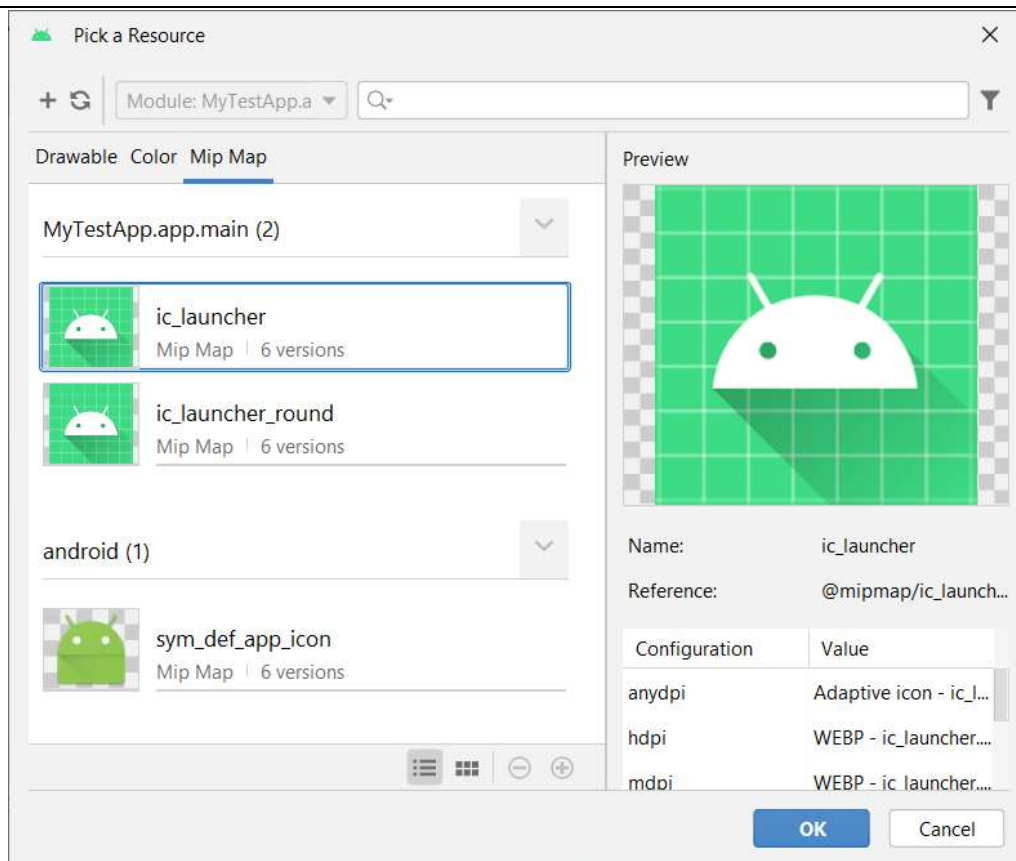
```

Obsługa aparatu

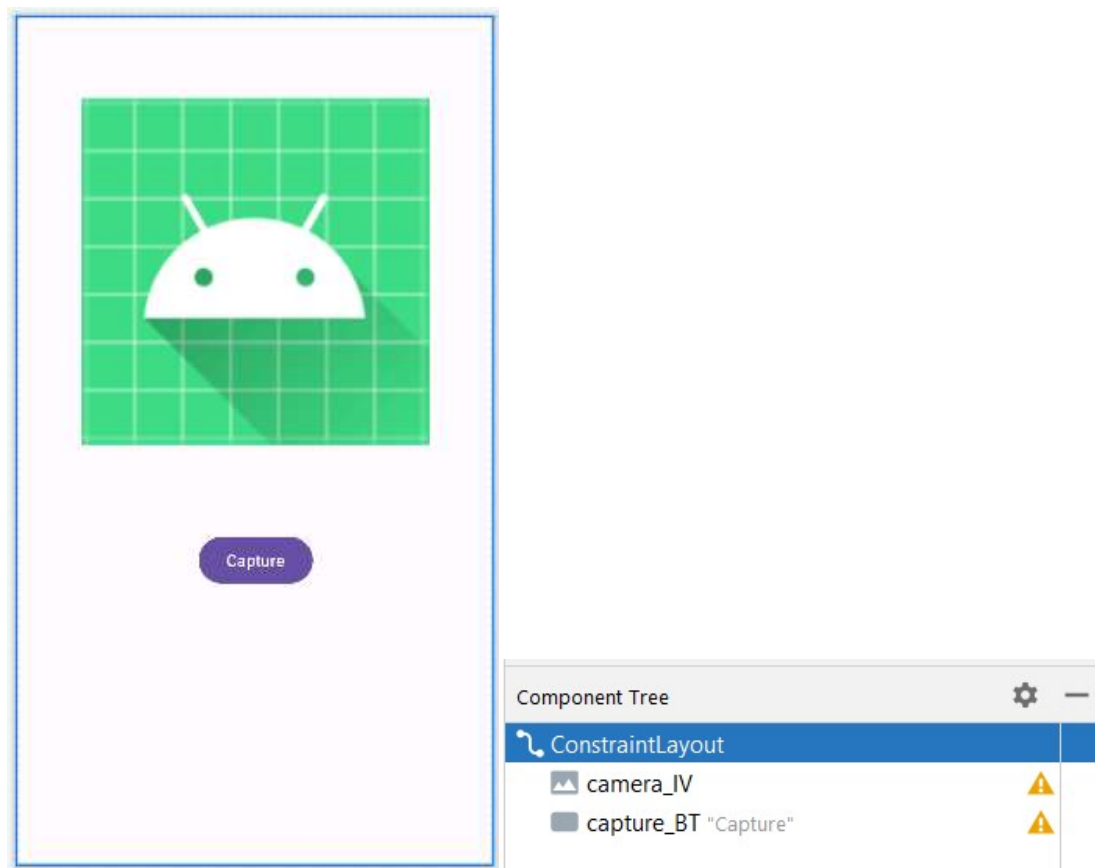
W tej sekcji zajmiemy się obsługą aparatu. Przygotujmy layout dla tej aktywności.

ZADANIE

Dodaj do layoutu *activity_camera* komponent *ImageView* i ustaw w nim obraz *ic_launcher*.



Dodaj również przycisk *Button* zawierający tekst *Capture*. Ustaw identyfikatory komponentów: dla *ImageView* id=*camera_IV*, dla *Button* id=*capture_BT*. Dopasuj rozmiar komponentów i ich położenie.



Layout aktywności obsługi aparatu jest gotowy. Teraz trzeba zaimplementować obsługę przycisku *Capture*. Skorzystamy tutaj z intencji wywołującej aplikację Aparat.

ZADANIE

Dodaj w *CameraActivity* kod odpowiedzialny za wykonywanie zdjęć.

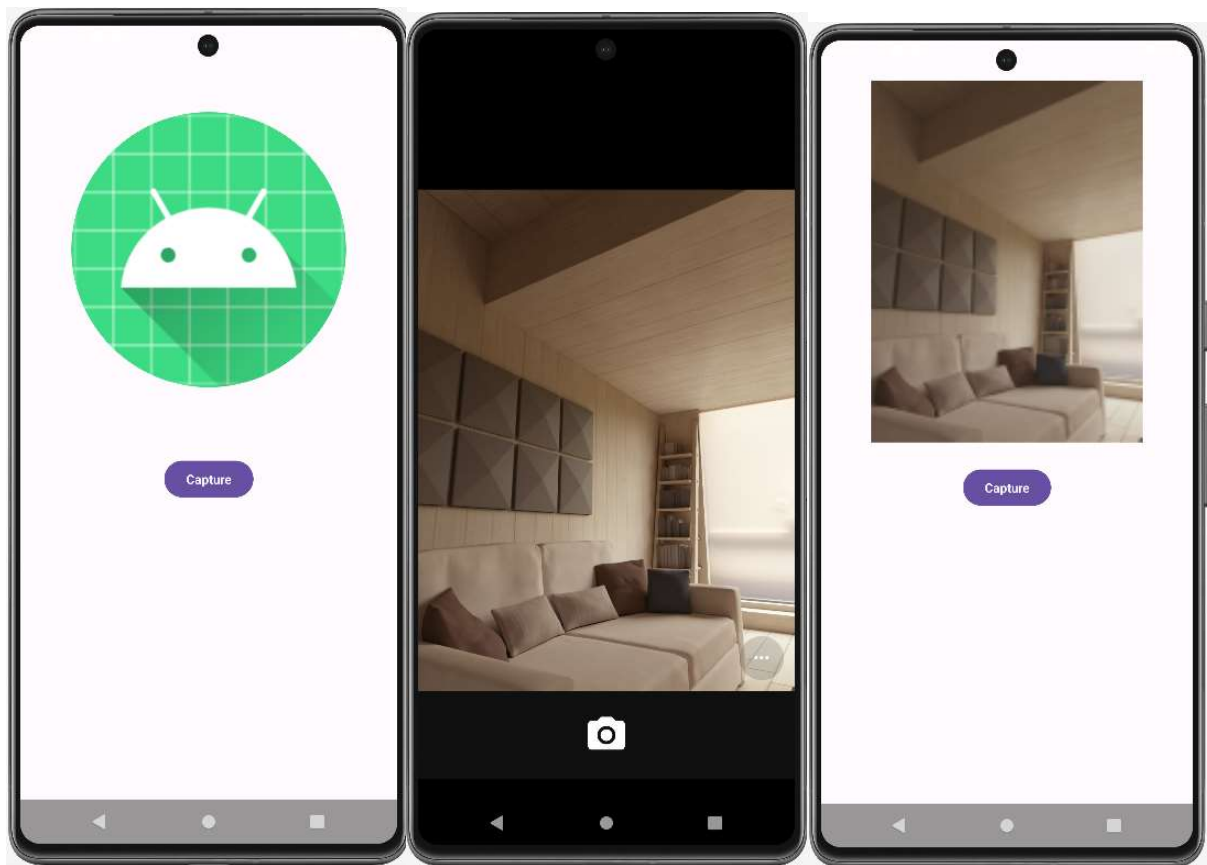
```
class CameraActivity : AppCompatActivity() {

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

        findViewById<Button>(R.id.capture_BT).setOnClickListener { it: View!
            // intencja wywołująca aplikację Aparat
            val intentCapture = Intent(MediaStore.ACTION_IMAGE_CAPTURE)
            // uruchomienie intencji z możliwością dostępu do rezultatu
            // jej działania - w tym przypadku do wykonanego zdjęcia
            startActivityForResult(intentCapture, requestCode: 123)
        }
    }

    // funkcja przetwarzająca wynik działania intencji
    override fun onActivityResult(requestCode: Int, resultCode: Int, data: Intent?) {
        super.onActivityResult(requestCode, resultCode, data)
        var bmp: Bitmap? = null
        if (data != null && requestCode == 123) {
            bmp = data.extras?.get("data") as Bitmap // pobranie obrazu
            // ustawienie obrazu w ImageView
            findViewById<ImageView>(R.id.camera_IV).setImageBitmap(bmp)
        }
    }
}
```

Korzystamy z metody *startActivityForResult()*, która nie tylko uruchamia intencję ale również pozwala na dostęp do wyniku jej działania. Do naszej intencji przypisujemy kod '123' aby później móc odnieść się do niej. Następnie w metodzie *onActivityResult()* sprawdzamy czy dane będące rezultatem działania intencji są niepuste (*data!=null*) oraz zgodność kodu nadanego wcześniej intencji (wynikowi jej działania). Ostatecznie przetwarzamy otrzymany rezultat – w naszym przypadku wyświetlamy zdjęcie w *ImageView*.

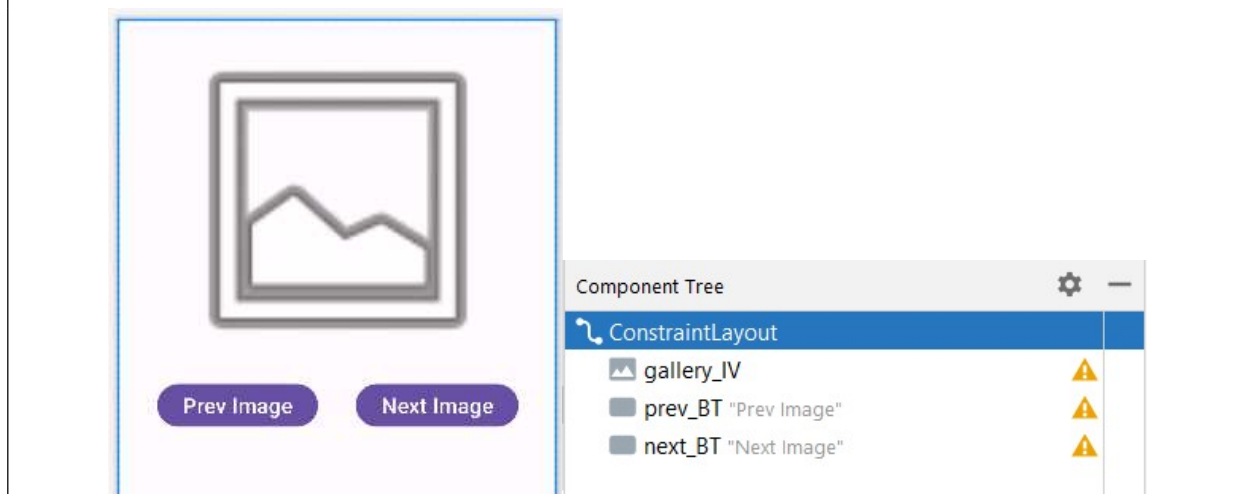


Tworzenie galerii zdjęć

Teraz zajmiemy się galerią zdjęć. Zaczynamy od zbudowania layoutu. Nasza galeria zbudowana będzie z komponentu *ImageView* do wyświetlania obrazów oraz 2 przycisków umożliwiających nawigowanie.

ZADANIE

Dodaj do layoutu *activity_gallery* komponent *ImageView* oraz 2 przyciski *Button* wg poniższego rysunku. Wprowadź identyfikatory komponentów i ustaw wyświetlany na nich tekst.



Mając gotowy layout uzupełniamy kod aktywności galerii. Na początku musimy zbudować listę wszystkich plików graficznych (ścieżek do plików) znajdujących się w pamięci telefonu aby następnie

móc swobodnie nawigować pomiędzy nimi. W tym celu utworzymy funkcję, która pozwoli nam utworzyć odpowiednią strukturę.

ZADANIE

Zaimplementuj w `Gallery_Activity` poniższą funkcję.

```
fun getAllImages():ArrayList<String>{
    val fileList: ArrayList<String> = ArrayList()
    // katalog główny urządzenia
    var path: String = Environment.getExternalStorageDirectory().absolutePath
    // pętla przechodząca po wszystkich katalogach, podkatalogach i plikach
    File(path).walk().forEach { it: File
        if(it.toString().endsWith( suffix: ".jpg")) // filtr plików JPG
            fileList.add(it.toString())
    }
    return fileList
}
```

Powyższa funkcja tworzy `ArrayList` ścieżek do plików graficznych w formacie JPG. Teraz wystarczy wywołać naszą funkcję a następnie oprogramować akcje przycisków nawigacyjnych naszej galerii.

ZADANIE

Zaimplementuj w metodzie `onCreate()` następujący kod:

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

    val gallery_IV = findViewById<ImageView>(R.id.gallery_IV)
    val prev_BT = findViewById<Button>(R.id.prev_BT)
    val next_BT = findViewById<Button>(R.id.next_BT)

    // ArrayLista wszystkich obrazów
    val allImagesPaths = getAllImages()
    // indeks aktywnie wyświetlanego obrazu
    var currImgIndex = 0

    // wyświetlenie pierwszego obrazu (indeks 0)
    gallery_IV.setImageURI(Uri.parse(allImagesPaths.get(currImgIndex)))

    // przejście do wyświetlania poprzedniego obrazu
    prev_BT.setOnClickListener { it: View!
        gallery_IV.setImageURI(Uri.parse(allImagesPaths.get(currImgIndex - 1)))
        currImgIndex--
    }

    // przejście do wyświetlania kolejnego obrazu
    next_BT.setOnClickListener { it: View!
        gallery_IV.setImageURI(Uri.parse(allImagesPaths.get(currImgIndex + 1)))
        currImgIndex++
    }
}
```

Nasza aplikacja wyszukuje obrazy i wyświetla je oraz umożliwia nawigowanie pomiędzy nimi. Jednak trzeba dodać pewne zabezpieczenia:

- Nie możemy wyświetlać obrazów jeżeli nie ma ich w telefonie. Wówczas przyciski powinny być nieaktywne.
- Jeśli istnieje tylko 1 obraz to wyświetlamy go, ale przyciski pozostają nieaktywne.
- Przy wyświetlaniu pierwszego obrazu z listy przycisk *Prev Image* powinien pozostać nieaktywny.
- Przy wyświetlaniu ostatniego pliku z listy przycisk *Next Image* powinien być nieaktywny.
- Odblokowanie przycisków nawigacyjnych powinno nastąpić po opuszczeniu skrajnych pozycji listy – odpowiednio *Prev Image* gdy jesteśmy za indeksem 0 oraz *Next Image* będąc przed ostatnim indeksem.

ZADANIE

Zmodyfikuj kod metody *onCreate()* wg poniższych obrazów.

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

    val gallery_IV = findViewById<ImageView>(R.id.gallery_IV)
    val prev_BT = findViewById<Button>(R.id.prev_BT)
    val next_BT = findViewById<Button>(R.id.next_BT)

    // ArrayLista wszystkich obrazow
    val allImagesPaths = getAllImages()
    // indeks aktualnie wyswietlanego obrazu
    var currImgIndex = 0

    if(allImagesPaths.size>0) {
        // wyswietlenie pierwszego obrazu (indeks 0)
        gallery_IV.setImageURI(Uri.parse(allImagesPaths.get(currImgIndex)))
    }
    prev_BT.isEnabled = false // na starcie przycisk nieaktywny
    if(allImagesPaths.size<=1)
        next_BT.isEnabled=false
}
```



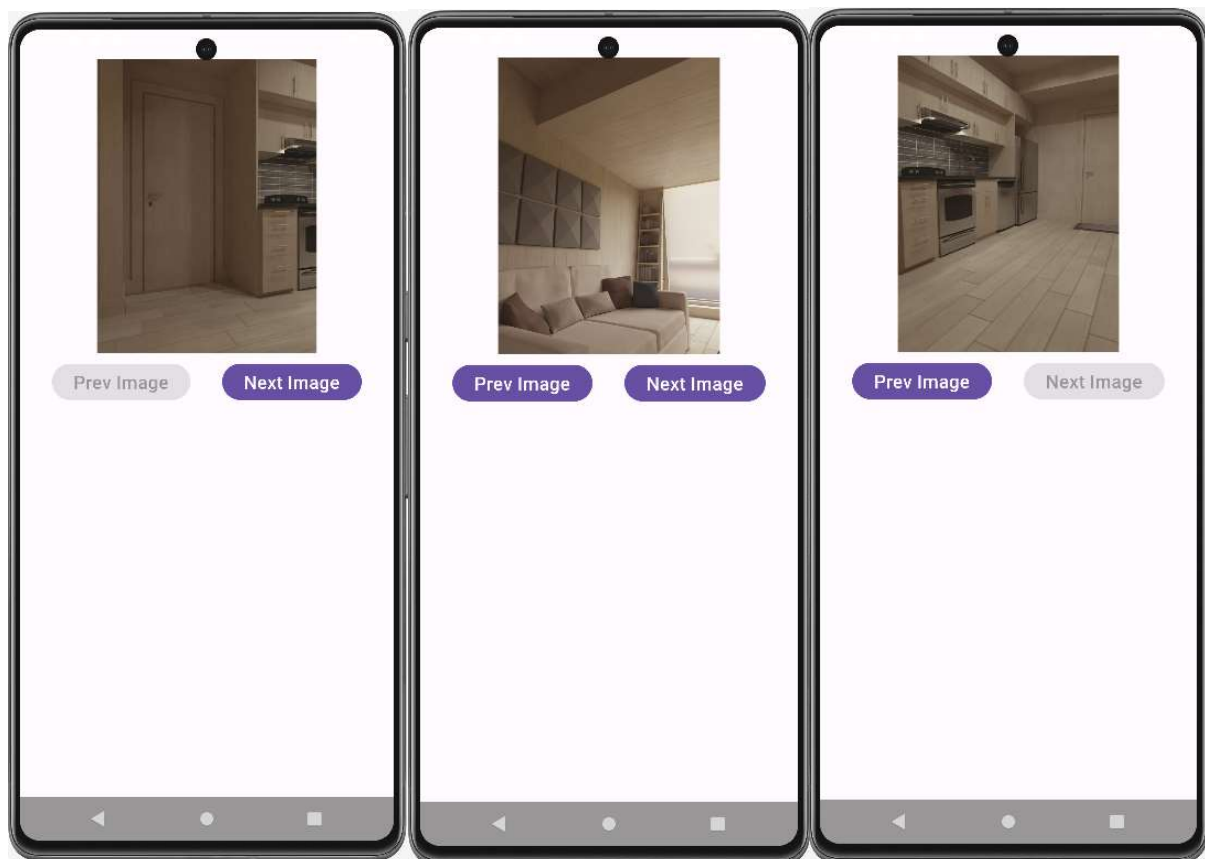
```

// przejście do wyświetlania poprzedniego obrazu
prev_BT.setOnClickListener { it: View!
    if (currImgIndex > 0) {
        gallery_IV.setImageURI(Uri.parse(allImagesPaths.get(currImgIndex - 1)))
        currImgIndex--
        if (!next_BT.isEnabled)
            next_BT.isEnabled = true
    }
    if (currImgIndex == 0)
        prev_BT.isEnabled = false
}

// przejście do wyświetlania kolejnego obrazu
next_BT.setOnClickListener { it: View!
    if (currImgIndex < allImagesPaths.size - 1) {
        gallery_IV.setImageURI(Uri.parse(allImagesPaths.get(currImgIndex + 1)))
        currImgIndex++
        if (!prev_BT.isEnabled)
            prev_BT.isEnabled = true
    }
    if (currImgIndex == allImagesPaths.size - 1)
        next_BT.isEnabled = false
}
}

```

Galeria gotowa, możemy bezpiecznie poruszać się między obrazami.



CDN.