Instrukcja

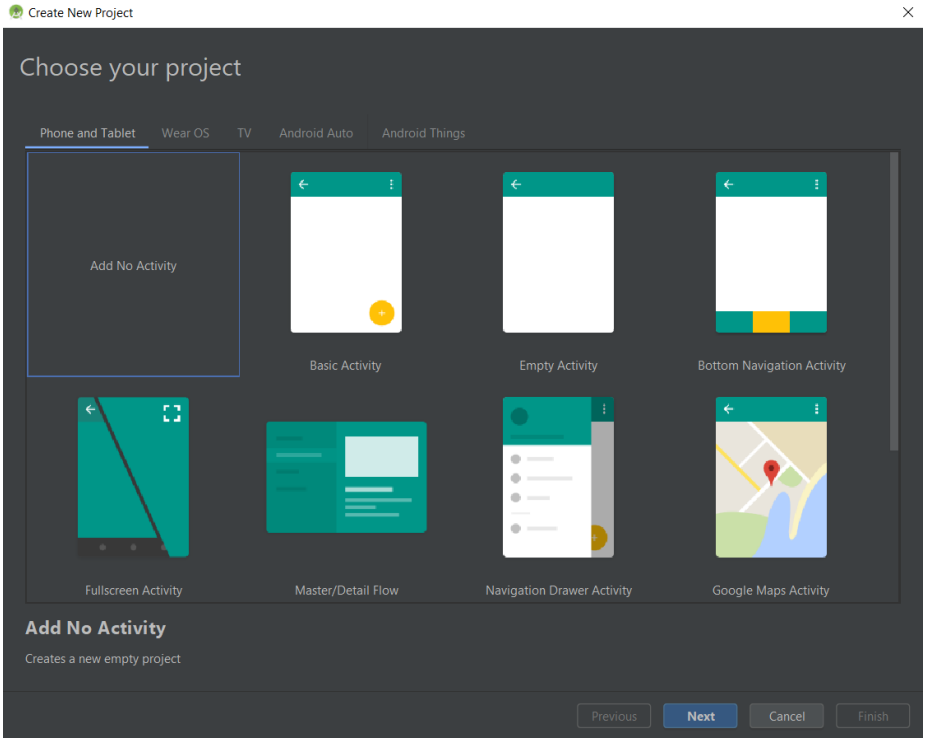
Spis treści

1. Tworzenie projektu 2
2. Tworzenie klasy do przechowywania danych 3
3. Tworzenie danych 4
4. Tworzenie własnego adaptera listy 5
5.Tworzenie aktywności listy 8
6. Tworzenie własnego adaptera tabbed 9
7. Tworzenie aktywności tabbed 12
8. Tworzenie intencji wyboru obrazu 13

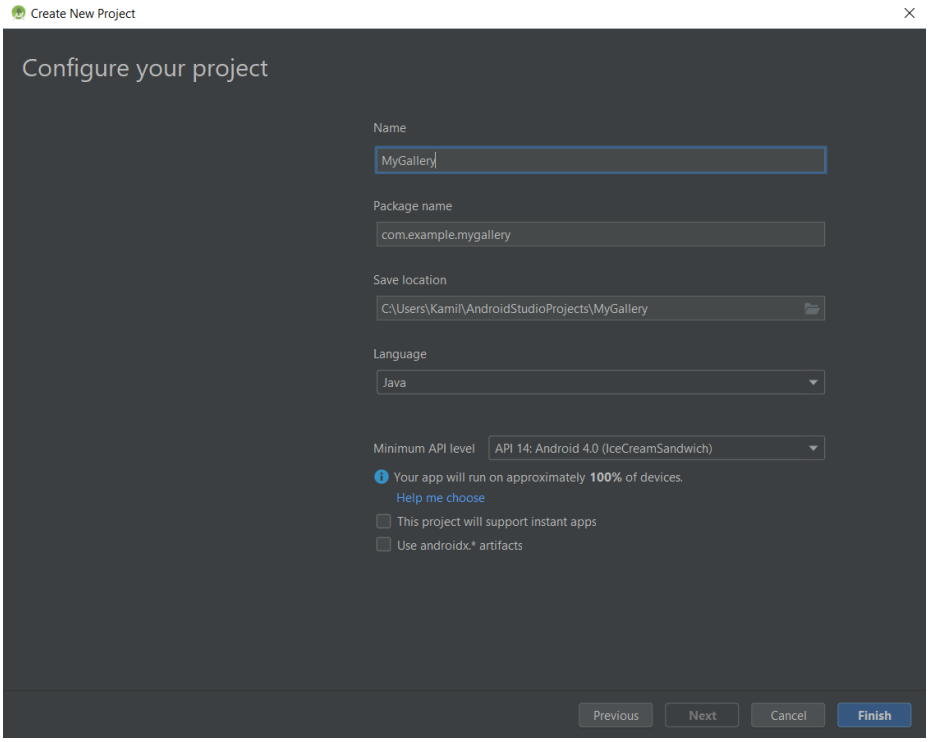
1. Tworzenie projektu

Jeżeli chcesz dodać galerię do już istniejącego projektu pomiń tworzenie nowego projektu i przejdź do tworzenia klasy do przechowywania danych!

1. Utwórz nowy projekt w Android Studio
2. Wybierz „Empty Activity” i kliknij „next”



3. Wprowadź nazwę projektu (Name), wybierz język (Language) Java i ustaw minimalną wersję systemu android (Minimum API level) i kliknij „Finish”



2. Tworzenie klasy do przechowywania danych

1. Stwórz nową klasę „ImageEntity” w pakiecie „com.example.[nazwa projektu]”. Zadaniem klasy będzie przechowywać informację o Obrazie.

2. Utwórz wewnątrz klasy prywatne pola przechowujące wybrane informacje o obrazie. Możesz swobodnie dodać nowe informacje np. rok powstania, wymiary, itp.

```
private final String title;
private final String author;
private final String place;
private final String description;
private final int image;
```

3. Utworzone pola zostaną podkreślone na czerwono, ponieważ nigdy nie zostaną przypisane im wartości. Utwórz konstruktor w którym nadasz polom wartości.

```
ImageEntity(String title, String author, String place, String description, int image)
{
    this.title = title;
    this.author = author;
    this.place = place;
    this.description = description;
    this.image = image;
}
```

4. Utwórz gettery dla każdego pola, aby można było się odwołać do nich z poza klasy.

```
String getTitle() {
    return title;
}

String getAuthor() {
    return author;
}

String getPlace() {
    return place;
}

String getDescription() {
    return description;
}

int getImage() {
    return image;
}
```

3. Tworzenie danych

1. Stwórz nową klasę „ImageGenerator” w pakiecie „com.example.[nazwa projektu]”. Zadaniem klasy będzie stworzyć dane o obrazach.

2. Wrzuć do pakietu „res .drawable” kilka obrazków. W moim przypadku są to obrazy image1, image2, image3 i image4. Możesz umieścić dowolne obrazy i nazwać je inaczej.
   Pamiętaj aby nazwy nie zawierały spacji.

3. Stwórz i zaimplementuj metodę „generateImages” która zwróci listę obrazów. Pamiętaj żeby w „R.drawable.” po kropce podać nazwę pliku z obrazem, który umieściliśmy w folderze „drawable”. Dla lepszego efektu możesz zamienić teks "Tutaj wstaw długi opis…" na kilkuset znakowy teks. Możesz skorzystać z generatora tekstu lub skopiować go z Wikipedii.
   W prawdziwej aplikacje dane pobiera się z bazy danych lub pamięci urządzania.
   Nie należy ich hard kodować jak w poniższym przykładzie!

```
ImageEntity[] generateImages() {
    return new ImageEntity[] {

            new ImageEntity(
                    "Trzy słoneczniki w wazonie",
                    "Vincent van Gogh",
                    "Muzeum Vincenta van Gogha",
                    "Tutaj wstaw długi opis…",
                    R.drawable.image1),

            new ImageEntity(
                    "Wazon z piętnastoma słonecznikami",
                    "Vincent van Gogh",
                    "National Gallery w Londynie",
                    "Tutaj wstaw długi opis…",
                    R.drawable.image2),

            new ImageEntity(
                    "Wazon z dwunastoma słonecznikami",
                    "Vincent van Gogh",
                    "Nowa Pinakoteka",
                    "Tutaj wstaw długi opis…",
                    R.drawable.image3),

            new ImageEntity(
                    "Czaszka z palącym się papierosem",
                    "Vincent van Gogh",
                    "Muzeum Vincenta van Gogha",
                    "Tutaj wstaw długi opis…",
                    R.drawable.image4)
    };
}
```

4. Tworzenie własnego adaptera listy

1. Stwórz nową klasę „MyImageListAdapter” w pakiecie „com.example.[nazwa projektu]”. Zadaniem klasy jest tworzenie pojedynczego elementu listy obrazów.

2. Utwórz wewnątrz klasy prywatne pola. Typ „ImageEntity[]” określa kolekcję obiektów które chcemy wyświetlić. W przypadku tworzenia własnej listy możesz stworzyć własną kolekcję obiektów np. „Animal[] animals”, „List<Car> cars”, „Buildings buildings”

```
private Context context;
private int layoutResourceId;
private ImageEntity[] images;
```

3. Jeżeli Android studio zapyta się o import klas to zawsze wyrażaj zgodę. Jeżeli jakaś klasa będzie zaznaczona na czerwono, kliknij na nią lewym przyciskiem myszy. Android Studio zasugeruje import klasy. Kliknij ALT + ENTER, w razie konieczności wybierz z listy „Import class”. W razie potrzeby importuj brakujące klasy.

4. Rozszerz klasę o „ArrayAdapter<ImageEntity>”, w razie potrzeby zaimportuj brakującą klasę. W przypadku tworzenia własnego adaptera należy zamienić „<ImageEntity>” na klasę której obiekty chcemy wyświetlić.

```
public class MyImageListAdapter extends ArrayAdapter<ImageEntity> {
    ...
}
```

5. Deklaracje klasy zostanie podkreślona na czerwono, ponieważ brakuje konstruktora. Utwórz konstruktor w którym nadasz polom wartości.

```
MyImageListAdapter(Context context, int layoutResourceId, ImageEntity[] images) {
    super(context, layoutResourceId, images);
    this.layoutResourceId = layoutResourceId;
    this.context = context;
    this.images = images;
}
```

6. Stwórz nowy layout „fragment_image_list” w pakiecie „res.layout”, określający wygląd pojedynczego elementu listy.

4. Uzupełnij layout poniższym kodem lub stwórz własny layout. Spróbuj poeksperymentować z różnymi układami aż osiągniesz satysfakcjonujący efekt.
   Pamiętaj aby pola które chcesz uzupełnić wartościami z obiektu posiadały unikatowy ID. Pozostałe elementu layoutu nie muszą ich posiadać.

```
<?xml version="1.0" encoding="utf-8"?>
<android.support.constraint.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:id="@+id/row"
    android:layout_width="match_parent"
    android:layout_height="match_parent">

    <android.support.constraint.Guideline
        android:id="@+id/guideline2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
```

```
    android:orientation="vertical"
    app:layout_constraintGuide_begin="156dp" />

<android.support.constraint.Guideline
    android:id="@+id/guideline1"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:orientation="horizontal"
    app:layout_constraintGuide_begin="226dp" />

<ImageView
    android:id="@+id/imageView"
    android:layout_width="0dp"
    android:layout_height="0dp"
    android:layout_marginStart="8dp"
    android:layout_marginLeft="8dp"
    android:layout_marginTop="8dp"
    android:layout_marginEnd="8dp"
    android:layout_marginRight="8dp"
    android:layout_marginBottom="8dp"
    android:contentDescription="@string/app_name"
    app:layout_constraintBottom_toTopOf="@+id/guideline1"
    app:layout_constraintEnd_toStartOf="@+id/guideline2"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toTopOf="parent"
    app:srcCompat="@android:drawable/ic_menu_crop" />

<LinearLayout
    android:layout_width="0dp"
    android:layout_height="0dp"
    android:layout_marginStart="8dp"
    android:layout_marginLeft="8dp"
    android:layout_marginTop="8dp"
    android:layout_marginEnd="8dp"
    android:layout_marginRight="8dp"
    android:layout_marginBottom="8dp"
    android:orientation="vertical"
    app:layout_constraintBottom_toTopOf="@+id/guideline1"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toStartOf="@+id/guideline2"
    app:layout_constraintTop_toTopOf="parent">

    <TextView
        android:id="@+id/textViewTitle"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="@android:string/unknownName"
        android:textSize="24sp" />

    <TextView
        android:id="@+id/textViewAuthor"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="@android:string/unknownName" />

    <TextView
        android:id="@+id/textViewPlace"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="@android:string/unknownName" />
```

```
        <TextView
            android:id="@+id/textViewDescription"
            android:layout_width="match_parent"
            android:layout_height="match_parent"
            android:text="@android:string/unknownName" />
    </LinearLayout>

</android.support.constraint.ConstraintLayout>
```

7. Teraz przypisz wartości pól obrazów do pól w layoucie. Wróć do klasy „MyImageListAdapter” i nadpisz metodę „getViev”. Zwróć uwagę na:
   „((TextView)currentView.findViewById(R.id.idPolaTekstowego)).setText(image.getPole());”
   Pierwszy nawias ma za zadanie znaleźć pole tekstowe o id „idPolaTekstowego” znajdującego się w layoucie. Następnie wywołuje się jego metodę „setText()” jak nazwa wskazuje służy ona ustawieniu tekstu w polu. Jako parametr wprowadzamy „image.getPole()” jest to odwołanie się do getera zwracającego wartość wskazanego pola.
   W razie potrzeby zaimportuj brakujące klasy.

```
@Override
public View getView(int position, View convertView, ViewGroup parent) {
    LayoutInflater inflater = ((Activity)context).getLayoutInflater();
    View currentView = inflater.inflate(layoutResourceId, parent, false);
    ImageEntity image = images[position];

    ((TextView)currentView.findViewById(R.id.textViewTitle)).setText(image.getTitle());

((TextView)currentView.findViewById(R.id.textViewAuthor)).setText(image.getAuthor());
    ((TextView)currentView.findViewById(R.id.textViewPlace)).setText(image.getPlace());

((TextView)currentView.findViewById(R.id.textViewDescription)).setText(image.getDescri
ption());

((ImageView)currentView.findViewById(R.id.imageView)).setImageResource(image.getImage(
));

    return currentView;
}
```

5.Tworzenie aktywności listy

1. Jeżeli rozszerzasz istniejącą aplikację stwórz nową aktywność „ImageListActivity”
   Pamiętaj aby ustawić intencję umożliwiającą przejście z twojej aplikacji do nowej aktywności!
   Jeżeli tworzysz aplikację od początku wykorzystaj już istniejącą aktywność „MainActivity”. W pakiecie „com.example.[nazwa projektu]” kliknij prawym przyciskiem myszy na „MainActivity”, wybierz „Refactor” i „Rename”. Zmień nazwę aktywności na „ImageListActivity”. Analogicznie zmień nazwę layoutu „activity_main” w pakiecie „res.layout” na „activity_image_list”

2. Przejdź do edycji layoutu „activity_image_list”. Usuń pole tekstowe „Hello World!”, zamień „ConstraintLayout” na „LinearLayout”, zmieść w środku „ListView”. Rozciągnij go na cały ekran i ustaw id na „imagesList”. **Możesz też zastąpić kod poniższym kodem**

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".ImageListActivity">

<ListView
    android:id="@+id/imagesList"
    android:layout_width="match_parent"
    android:layout_height="match_parent"/>
</LinearLayout>
```

3. Przejdź do edycji klasy „ImageListActivity”. Do metody „onCreate()” dopisz poniższy kod. W razie potrzeby zaimportuj brakujące klasy.
   Pierwsza linia tworzy kolekcję obiektów „ImagesEntity” (obrazy)
   Druga linia tworzy adapter który deklarowaliśmy w poprzednim punkcie.
   Trzecia linia pobiera „ListView” znajdujący się w layoucie „activity_image_list”.
   Czwarta linia dodaje do „ListView” adapter, co powoduje zapełnienie „ListView” elementami według reguł określonych w adapterze.

```
ImageEntity[] images = new ImageGenerator().generateImages();

MyImageListAdapter myImageListAdapter = new MyImageListAdapter(this,
R.layout.fragment_image_list, images);
ListView listView = findViewById(R.id.imagesList);

listView.setAdapter(myImageListAdapter);
```

4. Uruchom aplikację i przetestuj działanie listy. Możesz spróbować edytować layout „fragment_image_list”, żeby uzyskać inny wygląd.

6.Tworzenie własnego fragmentu tabbed

1. Stwórz nowy layout „fragment_image_tabbed” określający wygląd pojedynczego taba

2. Uzupełnij layout poniższym kodem lub stwórz własny layout. Spróbuj poeksperymentować z różnymi układami aż osiągniesz satysfakcjonujący efekt.
   Pamiętaj aby pola które chcesz uzupełnić wartościami z obiektu posiadały unikatowy ID.
   Z tego powodu każde id w poniższym kodzie ma na końcu „2”.
   Pozostałe elementu layoutu nie muszą ich posiadać.

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:layout_width="match_parent"
    android:layout_height="match_parent">

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

        <LinearLayout
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:orientation="vertical">

            <ImageView
                android:id="@+id/imageView2"
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:contentDescription="@string/app_name"
                android:adjustViewBounds="true"
                app:srcCompat="@android:drawable/ic_menu_crop" />

            <TextView
                android:id="@+id/textViewTitle2"
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:text="@android:string/unknownName"
                android:layout_marginTop="8dp"
                android:textSize="24sp" />

            <TextView
                android:id="@+id/textViewAuthor2"
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:layout_marginTop="8dp"
                android:text="@android:string/unknownName" />

            <TextView
                android:id="@+id/textViewPlace2"
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:layout_marginTop="8dp"
                android:text="@android:string/unknownName" />

            <TextView
                android:id="@+id/textViewDescription2"
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
```

```
            android:layout_marginTop="8dp"
            android:text="@android:string/unknownName" />
      </LinearLayout>
   </ScrollView>

</RelativeLayout>
```

3. Stwórz nową klasę „MyImageListTabbedFragment” w pakiecie „com.example.[nazwa projektu]”. Zadaniem klasy jest tworzenie widoku poszczególnych tabów.

4. Stwórz wewnątrz klasy prywatne pole przechowujące obiekt na podstawie którego ma powstać fragment. W przypadku tworzenia własnego fragmentu należy użyć własnego obiektu który będzie reprezentacją danych.

```
private ImageEntity image;
```

5. Dodane pole zostanie podkreślone na czerwono, dodaj konstruktor

```
public void setImage(ImageEntity image) {
   this.image = image;
}
```

6. Rozszerz klasę „MyImageTabbedFragment” o klasę „Fragment”

```
public class MyImageTabbedFragment extends Fragment {
      ...
}
```

7. Nadpisz metodę „onCreate”, zasada taka sama jak w klasie „MyImageListAdapter”.

```
@Override
public View onCreateView(LayoutInflater inflater, ViewGroup container, Bundle
savedInstanceState) {
   View currentView = inflater.inflate(R.layout.fragment_image_tabbed, container,
false);

   ((TextView)
currentView.findViewById(R.id.textViewTitle2)).setText(image.getTitle());
   ((TextView)
currentView.findViewById(R.id.textViewAuthor2)).setText(image.getAuthor());
   ((TextView)
currentView.findViewById(R.id.textViewPlace2)).setText(image.getPlace());
   ((TextView)
currentView.findViewById(R.id.textViewDescription2)).setText(image.getDescription());
   ((ImageView)
currentView.findViewById(R.id.imageView2)).setImageResource(image.getImage());

   return currentView;
}
```

7. Tworzenie własnego adaptera tabbed

1. Stwórz nową klasę „MyImageTabbedAdapter” w pakiecie „com.example.[nazwa projektu]”.

2. Utwórz wewnątrz klasy prywatne pole przechowujące listę fragmentów.

```
private final List<Fragment> mFragmentList = new ArrayList<>();
```

3. Rozszerz klasę o „FragmentStatePagerAdapter” w razie potrzeby zaimportuj brakującą klasę.

```
public class MyImageTabbedAdapter extends FragmentStatePagerAdapter {
    ...
}
```

4. Deklaracje klasy zostanie podkreślona na czerwono, ponieważ brakuje konstruktora. Utwórz konstruktor w którym będą tworzone fragmenty oraz funkcję prywatną tworzącą fragmenty.

```
MyImageTabbedAdapter(FragmentManager fragmentManager, ImageEntity[] images) {
    super(fragmentManager);
    for (ImageEntity image : images) {
        mFragmentList.add(createFragment(image));
    }
}

private MyImageTabbedFragment createFragment (ImageEntity image)
{
    MyImageTabbedFragment fragment = new MyImageTabbedFragment();
    fragment.setImage(image);
    return fragment;
}
```

5. Nadpisz metodę „getItem” zwracającą fragment dla danej pozycji i „getCount” zwracającą ilość tabów to stworzenia.

```
@Override
public Fragment getItem(int position) {
    return mFragmentList.get(position);
}

@Override
public int getCount() {
    return mFragmentList.size();
}
```

8. Tworzenie aktywności tabbed

1. Utwórz nową aktywność „ImageTabbedActivity”

2. Przejdź do edycji layoutu „activity_image_tabbed”. Usuń pole tekstowe „Hello World!” (jeżeli zostało wygenerowane), zamień „ConstraintLayout” na „linearLayout”, umieść w środku „TabLayout”, ustaw id na „TabLayout”, dodaj „ViewPager” rozciągnij go na cały ekran i ustaw id na „viewPager”. **Możesz też zastąpić kod poniższym kodem**

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
   android:layout_width="match_parent"
   android:layout_height="match_parent">

   <android.support.design.widget.TabLayout
      android:id="@+id/tabLayout"
      android:layout_width="0dp"
      android:layout_height="0dp"/>

   <android.support.v4.view.ViewPager
      android:id="@+id/viewPager"
      android:layout_width="match_parent"
      android:layout_height="match_parent">

   </android.support.v4.view.ViewPager>
</LinearLayout>
```

3. Przejdź do edycji klasy „ImageTabbedActivity”. Do metody „onCreate()” dopisz poniższy kod. W razie potrzeby zaimportuj brakujące klasy.

   Pierwsza linia tworzy kolekcję obiektów „ImagesEntity” (obrazy)
   Druga linia tworzy adapter który deklarowaliśmy w poprzednim punkcie.
   Trzecia i czwarta linia pobiera „ViewPager” i „TabLayout” z layoutu „activity_image_tabbed”.
   Piąta linia pobiera z intecji numer wybranego obrazu.
   Szusta linia ustawienie adaptera
   Siódma linia ustawianie wybranego obrazu jako aktywny
   Ósma linia dodanie do layoutu viewPagera

```
ImageEntity[] images = new ImageGenerator().generateImages();

MyImageTabbedAdapter tabAdapter = new yImageTabbedAdapter(getSupportFragmentManager(),
images);
ViewPager viewPager = findViewById(R.id.viewPager);
TabLayout tabLayout = findViewById(R.id.tabLayout);
int selectedImage = getIntent().getIntExtra("imageSelected", 0);

viewPager.setAdapter(tabAdapter);
viewPager.setCurrentItem(selectedImage);
tabLayout.setupWithViewPager(viewPager);
```

9. Tworzenie intencji wyboru obrazu

1. Przejdź do klasy ImageListActivity i w metodzie onCreate dopisz deklarację intencji przejścia do nowej aktywności z przekazaniem parametru określającego numer wybranego obrazu

```
listView.setOnItemClickListener(new AdapterView.OnItemClickListener() {
   public void onItemClick(AdapterView<?> arg0, View arg1, int position, long arg3) {
      Intent intent = new Intent(getApplicationContext(), ImageTabbedActivity.class);
      intent.putExtra("imageSelected", position);
      startActivity(intent);
   }
});
```

2. Finalnie klasa ImageListActivity powinna wyglądać tak

```
public class ImageListActivity extends AppCompatActivity {

   @Override
   protected void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.activity_image_list);

      ImageEntity[] images = new ImageGenerator().generateImages();

      MyImageListAdapter myImageListAdapter = new MyImageListAdapter(this,
R.layout.fragment_image_list, images);
      ListView listView = findViewById(R.id.imagesList);

      listView.setAdapter(myImageListAdapter);

      listView.setOnItemClickListener(new AdapterView.OnItemClickListener() {
         public void onItemClick(AdapterView<?> arg0, View arg1, int position, long
arg3) {
            Intent intent = new Intent(getApplicationContext(),
ImageTabbedActivity.class);
            intent.putExtra("imageSelected", position);
            startActivity(intent);
         }
      });
   }
}
```

3. Uruchom aplikację i przetestuj działanie. Możesz edytować layouty fragment_image_list i fragment_image_tabbed aby uzyskać inny efekt wizualny.