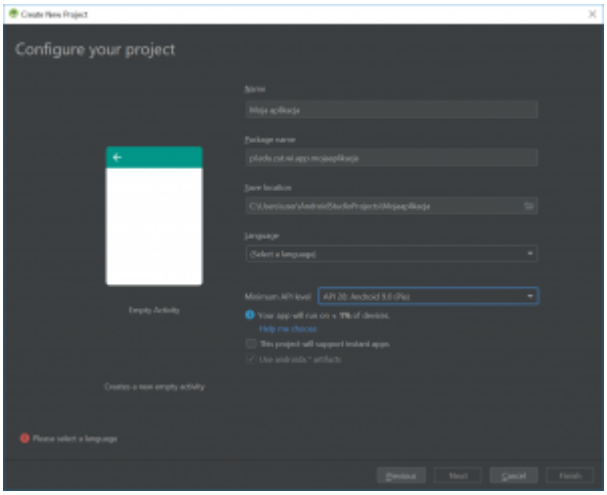
# Android 1

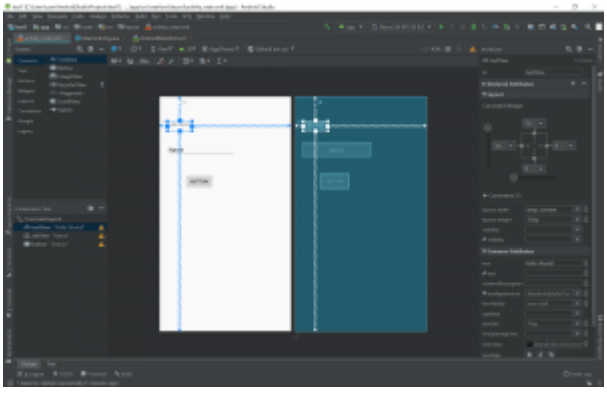
written by archi | 16 października 2019

Po konfiguracji pakietu uruchamiamy środowisko Android Studio wybieramy nowy projekt typu „Empty Activity”

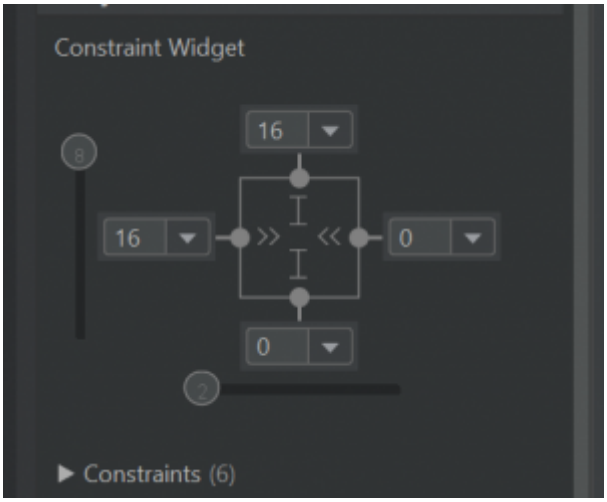
Ustawiamy nazwę pakietu !!!



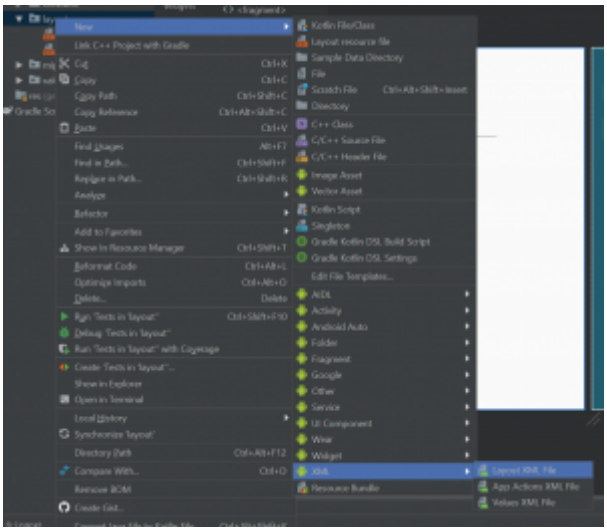

Zobaczysz ekran środowiska (w tym przypadku już zmodyfikowany o dodatkowe elementy)



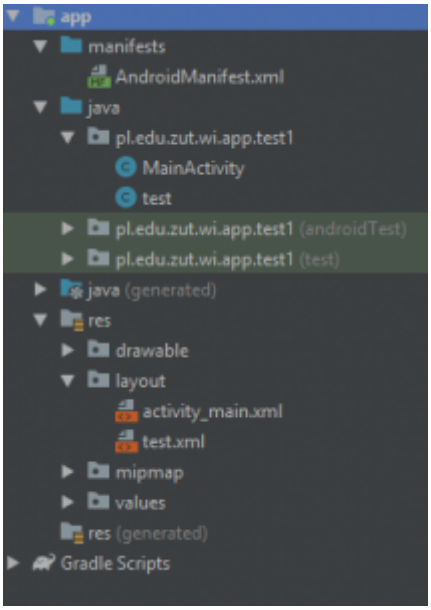
Na ekranie powyżej zostały dodane pola: editText, Button. Wykorzystamy je do dalszej pracy. Zauważ, że standardowo wszystkie elementy na formie będą centrowane i aby ustawić elementy na formie należy zmienić sposób metody powiązania z krawędziami lu innymi elementami na formie.




Kolejnym krokiem jest dodanie nowej formy i wywołanie jej z podstawowej wraz z obsługą stanu aplikacji i możliwości systemowego powrócenia do poprzedniej formy. Tworzymy nową formę „Layout” :



Wywołamy Layout z wykorzystaniem głównej formy poprzez przycisk tam zdefiniowany. W tym celu musimy zdefiniować klasę w gałęzi Java i dalej kontekst aplikacji poprzez menu kontekstowe New->JavaClass. Nadajemy nazwę (np. test) i zatwierdzamy.




Wewnątrz class wykonujemy extends AppCompatActivity. Zostanie automatycznie dodany odpowiedni dodatek. Plik będzie wyglądał jak poniżej:

```
import android.support.v7.app.AppCompatActivity;
public class test extends AppCompatActivity {
}
```

Dodajemy obsługę Layout który wykonaliśmy „test” jak poniżej:

```
import android.support.v7.app.AppCompatActivity;
public class test extends AppCompatActivity{
 @Override
 public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.test);
 }
}
```

W pliku AndroidManifest.xml dodajemy definicję Activity:

```
<?xml version="1.0" encoding="utf-8"?>
<manifest
xmlns:android="http://schemas.android.com/apk/res/android"
    package="pl.edu.zut.wi.app.test1">

    <application
        android:allowBackup="true"
```

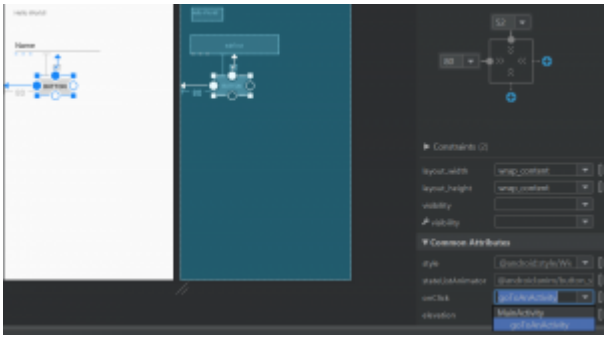
```
        android:icon="@mipmap/ic_launcher"
        android:label="Aplikacja"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:theme="@style/AppTheme">
        <activity android:name=".MainActivity">
            <intent-filter>
                <action android:name="android.intent.action.MAIN"
/>

                <category
android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
        <activity android:name=".test"
            android:label="Druga forma"
            android:parentActivityName=".MainActivity">
        </activity>
    </application>

</manifest>
```

Głównym naszym celem jest wykonanie obsługi przycisku do uruchomienia nowej activity. Wykonany to poprzez zdefiniowanie procedury obsługi OnClick i przypisaniu procedury do przycisku. W tym celu definiujemy:

```
package pl.edu.zut.wi.app.test1;

import android.content.Intent;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.view.View;
import android.widget.EditText;

public class MainActivity extends AppCompatActivity {

    public static final String EXTRA_MESSAGE = "" ;

    @Override
```

```
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
    }

    public void goToAnActivity(View view) {
        Intent intent = new Intent(this, test.class);
        startActivity(intent);
    }

}
```

W atrybutach przycisku wybieramy stworzoną procedurę w akcji onClick.



Po wykonaniu powyższych czynności uzyskamy aplikację z dwoma activity, które potrafią wspólnie działać.

Budujemy aplikację i uruchamiamy .

Po wykonaniu możemy zobaczyć uruchomioną aplikację na urządzeniu AVD

Kolejnym krokiem będzie przebudowanie aplikacji do przekazania danych z pola edycyjnego do drugiej instancji i tam wyświetlenie tej informacji.

W tym celu:

1. Rozbudowujemy procedurę uruchomienia nowej instancji w MainActivity (goToAnActivity)

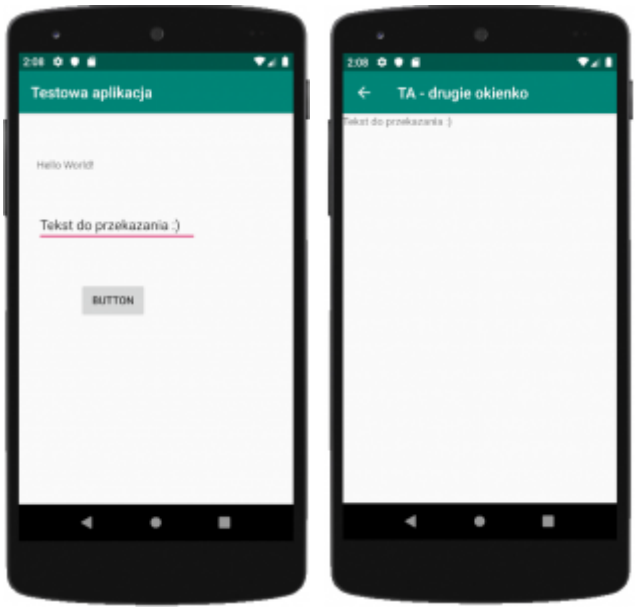
```
public void goToAnActivity(View view) {
Intent intent = new Intent(this, test.class);
EditText editText = (EditText) findViewById(R.id.editText);
String message = editText.getText().toString();
intent.putExtra(EXTRA_MESSAGE, message);
startActivity(intent);
}
```

oraz dodajemy predefinicję zmiennej w MainActivity na samym początku definicji klasy:

```
public static final String EXTRA_MESSAGE = "" ;
```

2. dodajemy pole tekstowe TextView w definicji layout test i obsługę komunikatów w clasie test

```
@Override
```

```
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.test);

    Intent intent = getIntent();
    String message =
intent.getStringExtra(MainActivity.EXTRA_MESSAGE);

    TextView textView = findViewById(R.id.textView2);
    textView.setText(message);

}
```

Budujemy aplikację i uruchamiamy .