



APLIKACJE MOBILNE

Wykład 6 - 7

AKTYWNOŚCI I INTENCJE

**Aplikacje zdalne - Wyświetlanie stron
www bezpośrednio w aplikacji**

dr Artur Bartoszewski

Intencje

- ✓ Intencje (obok Aktywności) są jednym z podstawowych komponentów z których zbudowane są aplikacje systemu Android.
- ✓ Są one odpowiedzialne przede wszystkim za obsługę rozkazów wydawanych przez użytkownika.
- ✓ Za pomocą intencji możemy wprowadzić komunikację pomiędzy aplikacjami (lub mniejszymi komponentami, jak usługi, aktywności itp.).
- ✓ Jednak najważniejszym zadaniem tego komponentu jest uruchamianie aplikacji lub aktywności.

Uruchamianie aktywności

Jawne (explicit) – w których wskazujemy obiekt, który chcemy stworzyć. W tym wypadku jednym z argumentów konstruktora Intencji jest obiekt typu Class wskazujący na klasę, której obiekt chcemy stworzyć.

Na przykład:

```
Intent intent = new Intent(context, MainActivity.class);
```

Tak zdefiniowana intencja uruchomi aktywność MainActivity.

Uruchamianie aktywności

Niejawne (implicit) – w których zawarta jest informacje o tym co chcemy zrobić, bez podawania konkretnych klas, które mają to zrealizować.

Najczęściej podawane są dwie informacje:

- co chcemy zrobić
- na jakich danych chcemy tą czynność wykonać.

System, za pomocą **Filtrów Intencji**, (o których wspominaliśmy na 1 wykładzie) decyduje jaka Aktywność ma być uruchomiona.

Na przykład: `Intent = new Intent (Intent.ACTION_VIEW,
Uri.parse("http://www.google.com"));`

Informujemy system o tym, że chcemy zobaczyć dane (Intent.ACTION_VIEW) zapisane pod adresem URI

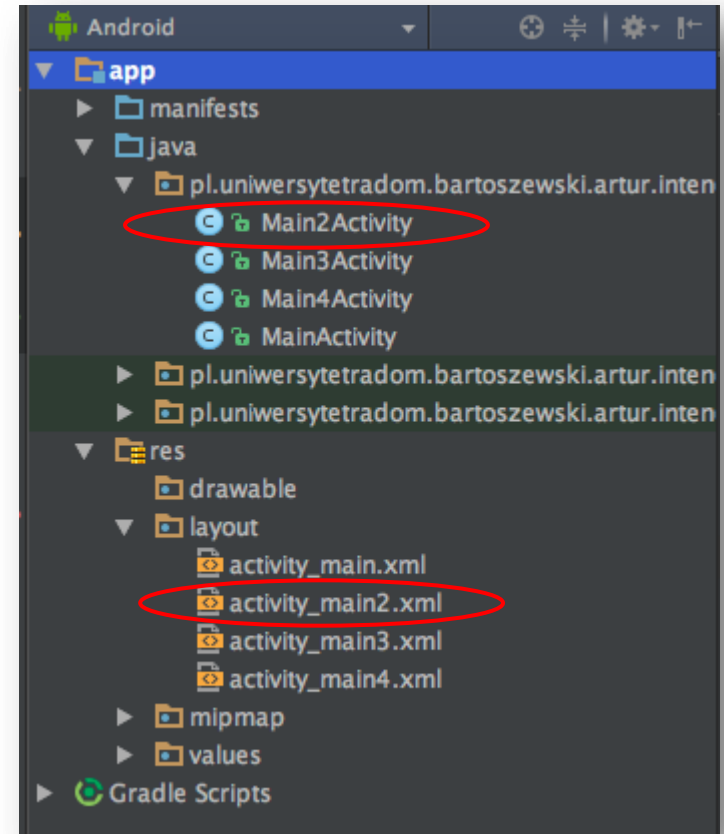
Przykład: Tworzymy aplikację, która zaprezentuje 5 różnych typów wykorzystania intencji do komunikacji pomiędzy aktywnościami

1. Otwieranie aktywności,
2. Przekazywanie danych do aktywności,
3. Odbieranie danych od aktywności;
4. Obsługa adresów WWW;
5. Obsługa adresów geoUri

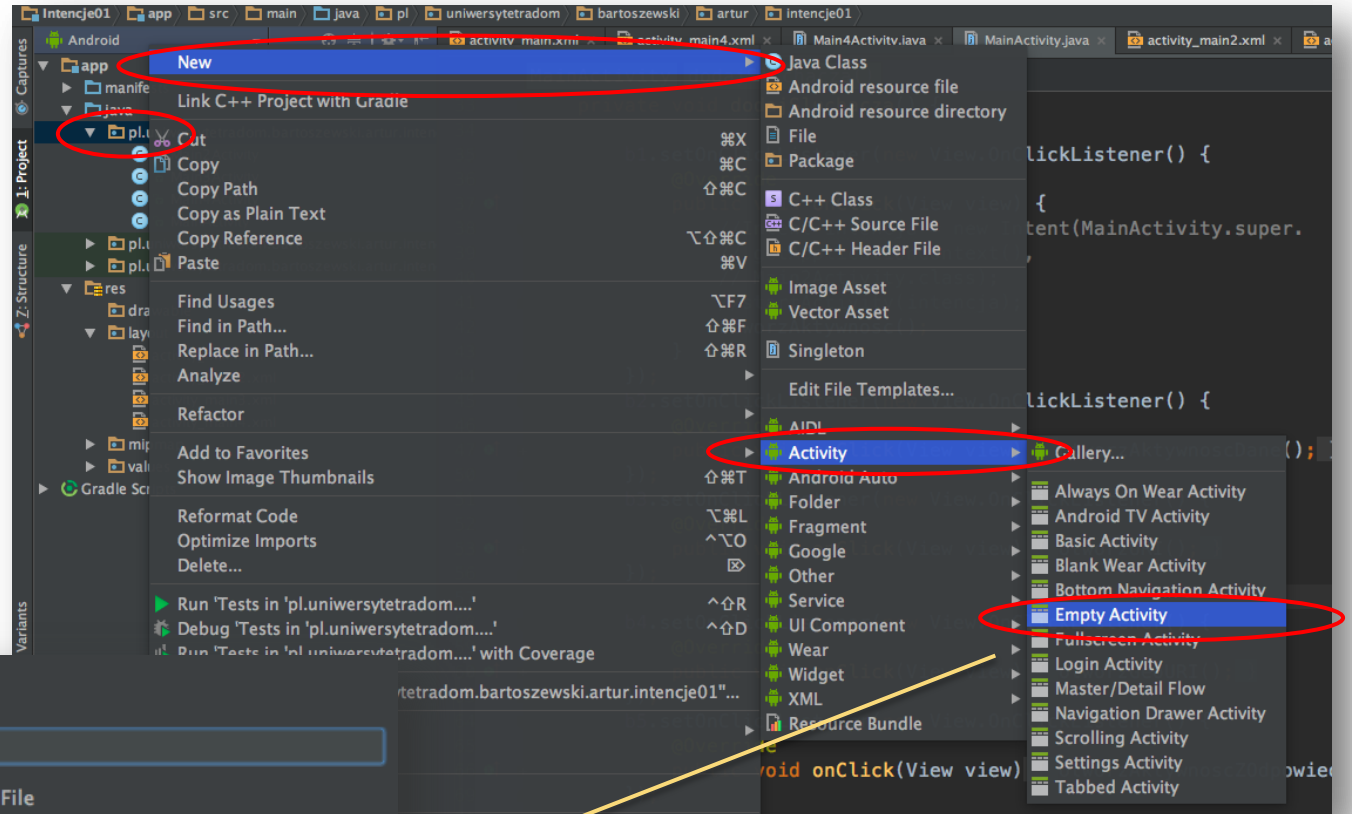


Przykład:

Rozpoczynamy od dodania nowej aktywności. Uzyskujemy plik .java oraz powiązany z nim plik .xml



Aplikacje mobilne



Creates a new empty activity

Activity Name:

☒ Generate Layout File

Layout Name:

☐ Launcher Activity

☒ Backwards Compatibility (AppCompat)

Package name:

Aplikacje mobilne

Przykład:

- W pliku `activity_main.xml` (startowa aktywność) dodajemy 5 komponentów `<Button>`
- Za pomocą `findViewById()` odnajdujemy uchwyty do nich. W naszym przykładzie nadałem im nazwy `b1` ; `b2` ; `b3`; `b4` i `b5`

```
17     private Button b1, b2, b3, b4, b5;
18     private TextView tekst01;
19
20     @Override
21     protected void onCreate(Bundle savedInstanceState) {
22         super.onCreate(savedInstanceState);
23         setContentView(R.layout.activity_main);
24         b1 = (Button) findViewById(R.id.button01);
25         b2 = (Button) findViewById(R.id.button02);
26         b3 = (Button) findViewById(R.id.button03);
27         b4 = (Button) findViewById(R.id.button04);
28         b5 = (Button) findViewById(R.id.button05);
29         tekst01 = (TextView) findViewById(R.id.textView01);
30         dodajSluchacza();
31     }
```


Przykład:

- Następnie stworzyłem funkcję `dodajSłuchacza()`, w której dodałem słuchacze zdarzeń do każdego przycisku (`b1`, `b2`, `b3`, `b4` i `b5`)

```
33 private void dodajSłuchacza() {  
34     b1.setOnClickListener((view) → {  
37         //...  
41         otworzAktywnosc();  
42     });  
44     b2.setOnClickListener((view) → { otworzAktywnoscDane(); });  
50     b3.setOnClickListener((view) → { otworzURI(); });  
56  
57     b4.setOnClickListener((view) → { otworzGeoURI(); });  
63     b5.setOnClickListener((view) → { otworzAktywnoscZ0dpowiedzia(); });  
69 }
```

Przykład cz.1: Otwieranie aktywności

- W `onClickListener()` przycisku `b1` wywołuje metodę `otworzAktywnosc()`,
- Można też kod otwierania umieścić bezpośrednio w słuchaczu (fragment kodu w komentarzu)

```
35 b1.setOnClickListener(new View.OnClickListener() {
36     @Override
37     public void onClick(View view) {
38         //Intent intencja = new Intent(MainActivity.super.
39         // getApplicationContext(),
40         // MainActivity.class);
41         //startActivity(intencja);
42         otworzAktywnosc();
43     }
44 });
```

Aplikacje mobilne

Przykład cz.1: Otwieranie aktywności

Kod aktywności [Main2Activity.java](#)

```
8 public class Main2Activity extends AppCompatActivity {
9
10     @Override
11     protected void onCreate(Bundle savedInstanceState) {
12         super.onCreate(savedInstanceState);
13         setContentView(R.layout.activity_main2);
14         ConstraintLayout cl = (ConstraintLayout) findViewById(R.id.layout02);
15         cl.setOnClickListener(new View.OnClickListener() {
16             @Override
17             public void onClick(View view) {
18                 finish();
19             }
20         });
21     }
22 }
```

Może oczywiście robić co tylko zaprogramujemy - w tym przykładzie tylko zamyka się po kliknięciu

Przykład cz.1: Otwieranie aktywności

Metoda `otworzAktywnosc()` zawiera dwa polecenia:

- Utworzenie nowej intencji – jej konstruktor wywołujemy z dwoma parametrami
 - `this` (kto utworzył)
 - `Main2Activity.class` (nazwa aktywności, która ma być utworzona)
- `startActivity()` z nazwą intencji w parametrze uruchamia nam aktywność wskazaną w intencji.

```
98  private void otworzAktywnosc() {  
99      Intent intencja = new Intent(this, Main2Activity.class);  
100     startActivity(intencja);  
101 }
```

Przykład cz.2: Otwieranie aktywności i przekazywanie danych

Metoda `otworzAktywnoscDane()` zawiera trzy polecenia:

- Utworzenie nowej intencji – jej konstruktor wywołujemy z dwoma parametrami: `--this` (kto utworzył) - `Main3Activity.class` (nazwa aktywności, która ma być utworzona)
- `.putExtra()` z dwoma parametrami (łańcuchami) – nazwa zmiennej do przekazania oraz jej zawartość (może być użyte kilka razy);
- `startActivity()` z nazwą intencji w parametrze uruchamia nam aktywność wskazaną w intencji.

```
103 private void otworzAktywnoscDane() {
104     Intent aktywnosc = new Intent(this, Main3Activity.class);
105     aktywnosc.putExtra("wartosc", "To jest przekazane z zewnatrz");
106     aktywnosc.putExtra("wartosc2", "\n i to tez");
107     startActivity(aktywnosc);
108 }
109
```

Przykład cz.2: Otwieranie aktywności i przekazywanie danych

Odebranie przekazanych danych następuje a docelowej aktywności:

```
7 public class Main3Activity extends AppCompatActivity {  
8  
9     TextView t3;  
10  
11     @Override  
12     protected void onCreate(Bundle savedInstanceState) {  
13         super.onCreate(savedInstanceState);  
14         setContentView(R.layout.activity_main3);  
15         t3 = (TextView) findViewById(R.id.textView03);  
16         Bundle paczka = getIntent().getExtras();  
17         String s = paczka.getString("wartosc");  
18         s += paczka.getString("wartosc2");  
19         t3.setText(s);  
20         //t3.setText(getIntent().getStringExtra("wartosc").toString());  
21     }  
22 }
```

Przykład cz.3: Otwieranie aktywności i odbieranie z niej danych

Metoda `otworzAktywnoscZOdpowiedzia()` zawiera 2 polecenia:

- Utworzenie nowej intencji – jej konstruktor wywołujemy z dwoma parametrami: - `-this` (kto utworzył) – `Main4Activity.class` (nazwa aktywności, która ma być utworzona)
- `startActivityForResult()` – używamy nieco inne metody otwarcia aktywności

```
71     private void otworzAktywnoscZOdpowiedzia() {  
72         Intent intencja = new Intent(this, Main4Activity.class);  
73         startActivityForResult(intencja, KOD_ODP_OKNA);  
74     }
```

Metoda `startActivityForResult()` przyjmuje jeszcze jeden argument. Jest nim identyfikator (liczba całkowita), na podstawie którego będzie można później stwierdzić skąd przyszła odpowiedź.

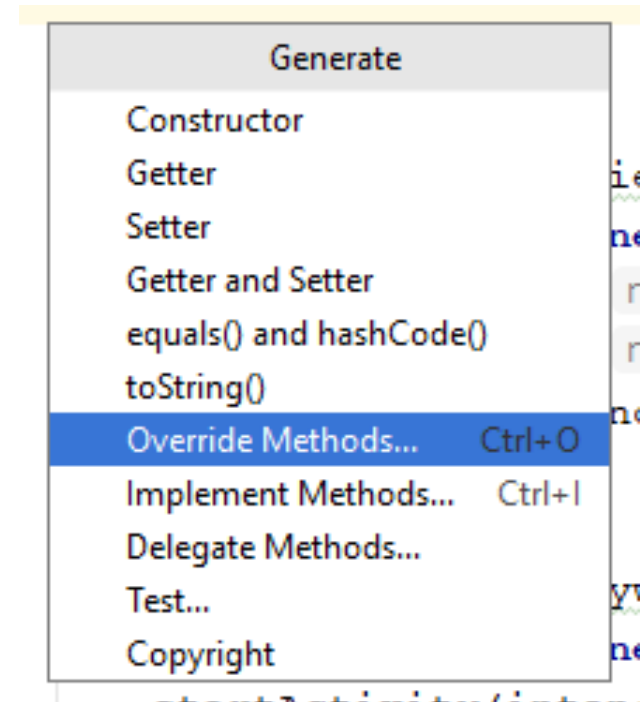
U nas identyfikatorem jest stała klasy MainActivity:

```
15     private static final int KOD_ODP_OKNA =1;
```

Przykład cz.3: Otwieranie aktywności i odbieranie z niej danych

Aby zinterpretować odpowiedź wywołanej aktywności należy (w aktywności wywołującej) przesłonić metodę systemową `onActivityResult()`.

Lewy przycisk myszy -> Generate ->



Przykład cz.3: Otwieranie aktywności i odbieranie z niej danych

```
76      @Override
77      protected void onActivityResult(int requestCode, int resultCode, Intent data) {
78          super.onActivityResult(requestCode, resultCode, data);
79          if (requestCode==KOD_ODP_OKNA) {
80              String odpowiedz = data.getStringExtra(Main4Activity.ODPOWIEDZ);
81              tekst01.setText("Odpowiedz aktywności to: \n"+odpowiedz);
82          }
83      }
```

Przyjmuje ona 3 argumenty:

- **int requestCode** – liczba całkowita, na podstawie której możemy zidentyfikować którą Aktywność zwróciła nam wynik (ta sama, którą ustawiliśmy w metodzie `startActivityForResult()`)
- **int resultCode** – liczba całkowita na podstawie której możemy określić stan wykonanej operacji (np. za pomocą stałych `Activity.RESULT_OK` lub `Activity.RESULT_CANCELED`).
- **Intent data** – dane zwrócone z Aktywności.

Przykład cz.3: Otwieranie aktywności i odbieranie z niej danych


W otwieranej aktywności (Main4Activity.java) przygotowujemy pole EditText o raz Button. Do Button-a dodajemy słuchacza, który uruchomi metodę przygotujOdpowiedz() i zamknie aktywność.

```
10 public class Main4Activity extends AppCompatActivity {
11
12     public static final String ODPOWIEDZ = "Odpowiedz";
13     EditText oknoOdp;
14     Button wykonaj;
15     @Override
16     protected void onCreate(Bundle savedInstanceState) {
17         super.onCreate(savedInstanceState);
18         setContentView(R.layout.activity_main4);
19         oknoOdp = (EditText) findViewById(R.id.editText01);
20         wykonaj = (Button) findViewById(R.id.buttonOdpowiedz);
21         wykonaj.setOnClickListener(new View.OnClickListener() {
22             @Override
23             public void onClick(View view) {
24                 przygotujOdpowiedz();
25                 finish();
26             }
27         });
28     }
```

Przykład cz.3: Otwieranie aktywności i odbieranie z niej danych

Metoda `przygotujOdpowiedz()` składa się z 3 elementów:

- Utworzenia obiektu `Intent`, który chcemy odesłać,
- Wpisania do niego danych za pomocą metod `putExtra()`,
- Ustawienia odpowiedzi za pomocą metody `setResult(int resultCode, Intent data)` –Argumenty ustawiane w tej metodzie pokrywają się z argumentami metody `onActivityResult()` aktywności nadrzędnej.

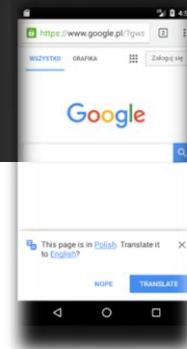
```
10  public class Main4Activity extends AppCompatActivity {  
  
29     private void przygotujOdpowiedz() {  
30         String s = oknoOdp.getText().toString();  
31         Intent wynik = new Intent();  
32         wynik.putExtra(ODPOWIEDZ, s);  
33         setResult(RESULT_OK, wynik);  
34     }  
35 }
```

Przykład cz.4: niejawne otwieranie aktywności - URL

W poprzednich przykładach wskazywaliśmy, którą aktywność chcemy uruchomić. Możemy jednak także zdefiniować zadanie do wykonania, a wybór Aktywności, która je obsłuży pozostawić systemowi (i nie musi być to aktywność naszej aplikacji)

Do Intencji prześlemy jedynie akcję oraz dane, na których będziemy chcieli ją przeprowadzić.

```
91 private void otworzURI() {  
92     Uri adres = Uri.parse("http://www.google.pl");  
93     Intent intencja = new Intent(Intent.ACTION_VIEW, adres);  
94     //intencja.setData(adres);  
95     startActivity(intencja);  
96 }
```



- Akcją którą chcemy wykonać jest `Intent.ACTION_VIEW` czyli obejrzyj.
- Danymi, na których wykonamy akcję jest adres url (ogólniej adres URI)

Takie wywołanie nie wymaga tworzenia nowej aktywności – przeglądarka jest w standardzie

Aplikacje mobilne

Przykład cz.5: niejawne otwieranie aktywności – współrzędne geograficzne

Innym typem danych URI, które mogą być obsługiwane przez system jest geoURI – czyli współrzędne geograficzne

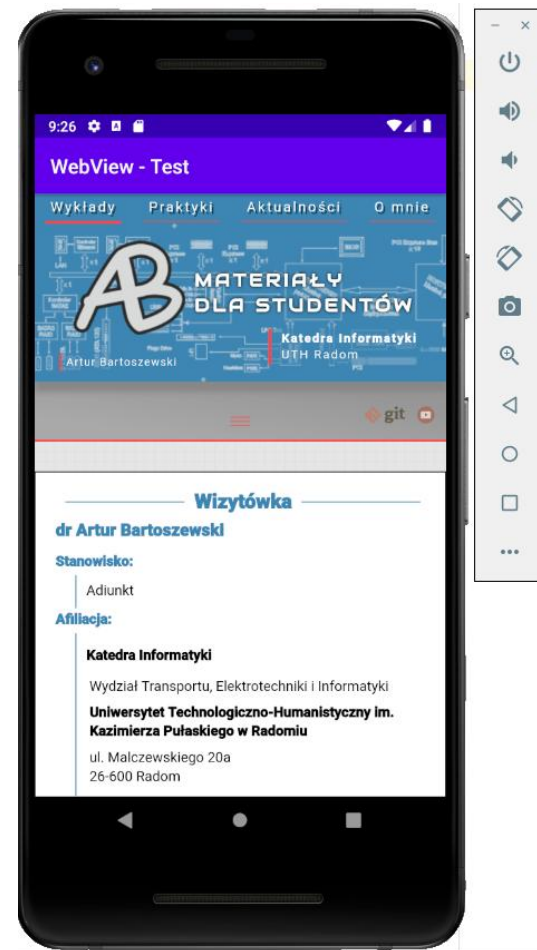
```
85 private void otworzGeoURI() {  
86     Uri geoAdres = Uri.parse("geo:51.405,21.1756");  
87     Intent intencja = new Intent(Intent.ACTION_VIEW, geoAdres);  
88     startActivity(intencja);  
89 }
```

Współrzędne należy przekonwertować na standardowe zmienną typu Uri za pomocą metody `Uri.parse()`

Takie wywołanie również nie wymaga tworzenia nowej aktywności – mapa Google też jest w standardzie



Aplikacje zdalne - Wyświetlanie stron www bezpośrednio w aplikacji



Aplikacje działające na systemie Android podzielić można na 3 grupy:

1. **Aplikacje natywne** - tworzone z wykorzystaniem SDK systemu oraz języków Java lub Kotlin. Przechowywane są na urządzeniu (plik .apk)
2. **Aplikacje internetowe** - realizowane jako aplikacje zdalne, działające na serwerze. Często są to po prostu odpowiednio przygotowane (responsywne) strony WWW.
3. **Aplikacje hybrydowe** - aplikacje internetowe, jednak oprócz wykorzystania zdalnych zasobów możliwa jest interakcja z lokalnymi zasobami smartfona (np. dostęp do czujników)

Komponent WebView - wstępnie zainstalowany na urządzeniu składnik systemu bazujący na Chrome. Umożliwia on aplikacjom na Androida wyświetlanie treści internetowych

Komponent WebView nie posiada interfejsu użytkownika przeglądarki – tylko renderuje stronę, bez standardowych menu, przycisków itp..

Klasa WebView zawiera metody:

- Ładowanie strony,
- nawigacja wprzód/wstecz przez historię przeglądarki,
- Przybliżanie i oddalanie,
- Wykorzystanie stylów CSS oraz skryptów JavaScript,
- Wymiana danych z urządzeniem przez interfejs JavaScript,
- Wyszukiwanie tekstu, pobieranie zdjęć,

Dodawanie obiektu WebView

```
<WebView
```

```
    android:id="@+id/webView01"  
    android:layout_width="0dp"  
    android:layout_height="0dp"  
    app:layout_constraintBottom_toBottomOf="parent"  
    app:layout_constraintLeft_toLeftOf="parent"  
    app:layout_constraintRight_toRightOf="parent"  
    app:layout_constraintTop_toTopOf="parent"  
/>
```

Obsługa obiektu WebView

```
public class MainActivity extends AppCompatActivity {  
  
    WebView webView01;  
    @Override  
    protected void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
        setContentView(R.layout.activity_main);  
        webView01 = findViewById(R.id.webView01);  
        webView01.getSettings().setJavaScriptEnabled(true);  
        webView01.loadUrl("http://www.google.pl/");  
    }  
}
```

Do załadowania strony służy metoda `.loadURL(http://...)`

Uprawnienia

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.webvirwtest">

    <uses-permission android:name="android.permission.INTERNET"/>

    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="WebVirwTest"
        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" />

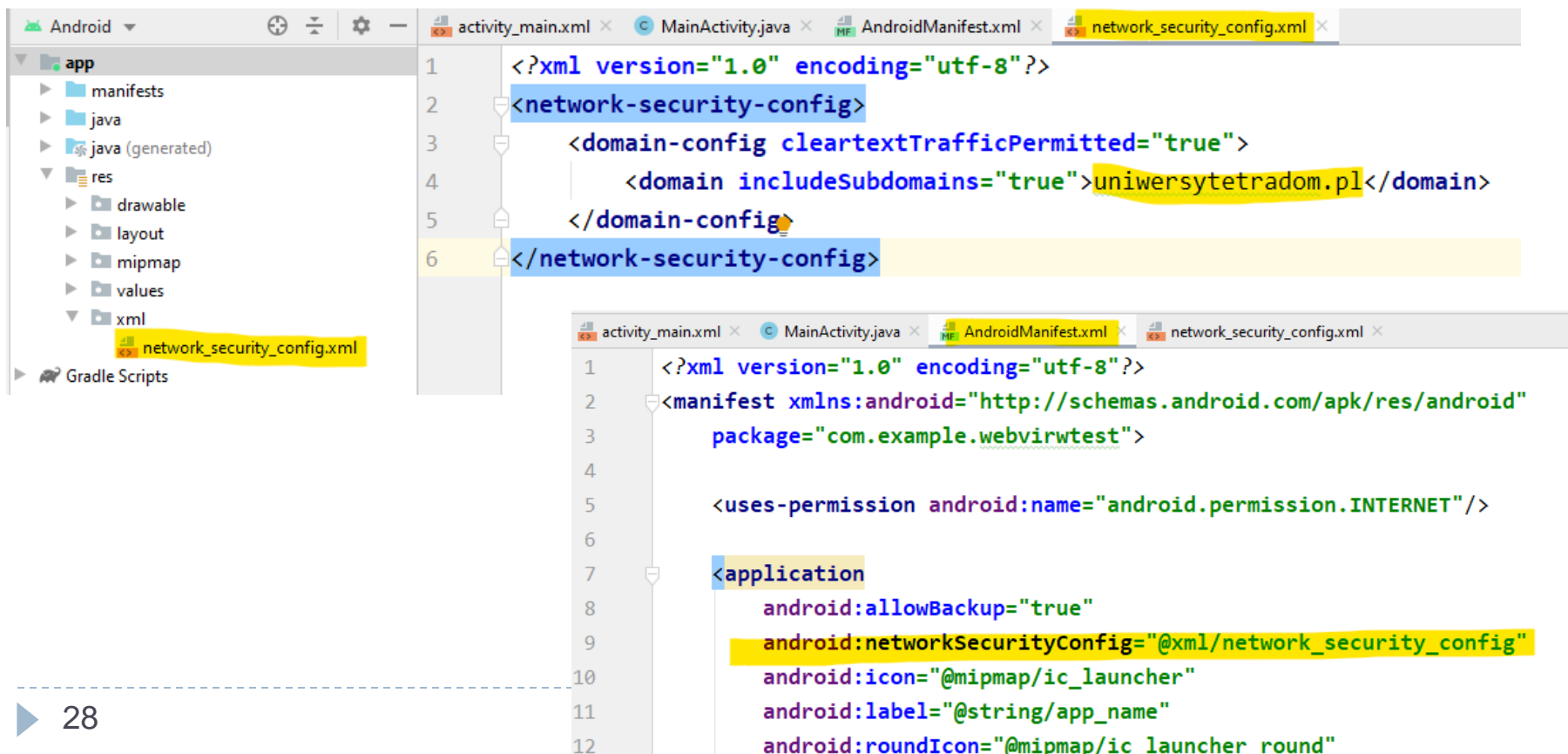
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>
</manifest>
```

W pliku AndroidManifest dodajemy uprawnienia do korzystania z Internetu

Problem z „brakiem s” w http://..

Próba wczytania do WebView strony o niezabezpieczonym protokole http:// zakończy się niepowodzeniem.

Jednym z rozwiązań jest dodanie domeny do listy wyjątków.



```
1 <?xml version="1.0" encoding="utf-8"?>
2 <network-security-config>
3   <domain-config cleartextTrafficPermitted="true">
4     <domain includeSubdomains="true">uniwersytetradom.pl</domain>
5   </domain-config>
6 </network-security-config>
```

```
1 <?xml version="1.0" encoding="utf-8"?>
2 <manifest xmlns:android="http://schemas.android.com/apk/res/android"
3   package="com.example.webvirwtest">
4
5   <uses-permission android:name="android.permission.INTERNET"/>
6
7   <application
8     android:allowBackup="true"
9     android:networkSecurityConfig="@xml/network_security_config"
10    android:icon="@mipmap/ic_launcher"
11    android:label="@string/app_name"
12    android:roundIcon="@mipmap/ic_launcher_round"
```

ZADANIE PRAKTYCZNE:

