

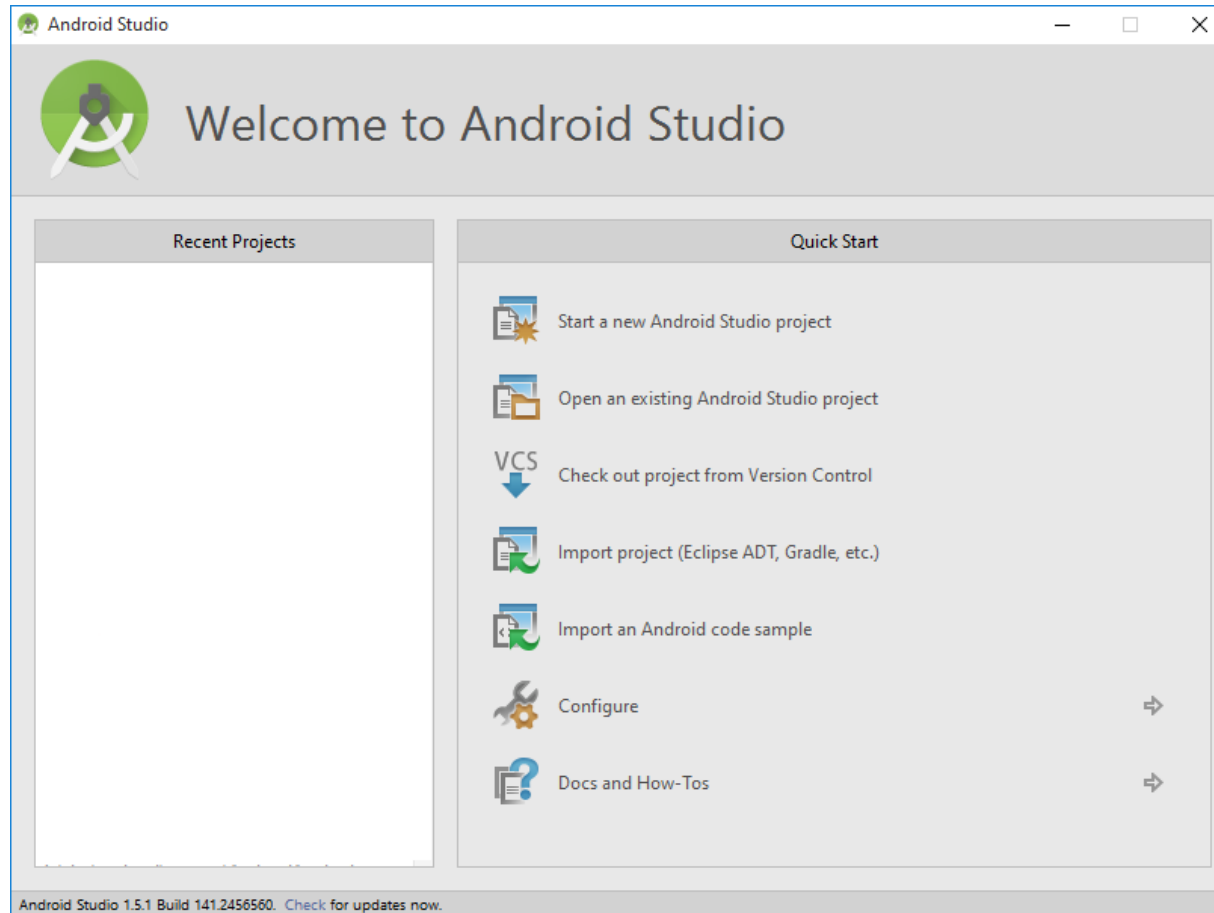
# Programowanie na platformę Android

Część 1

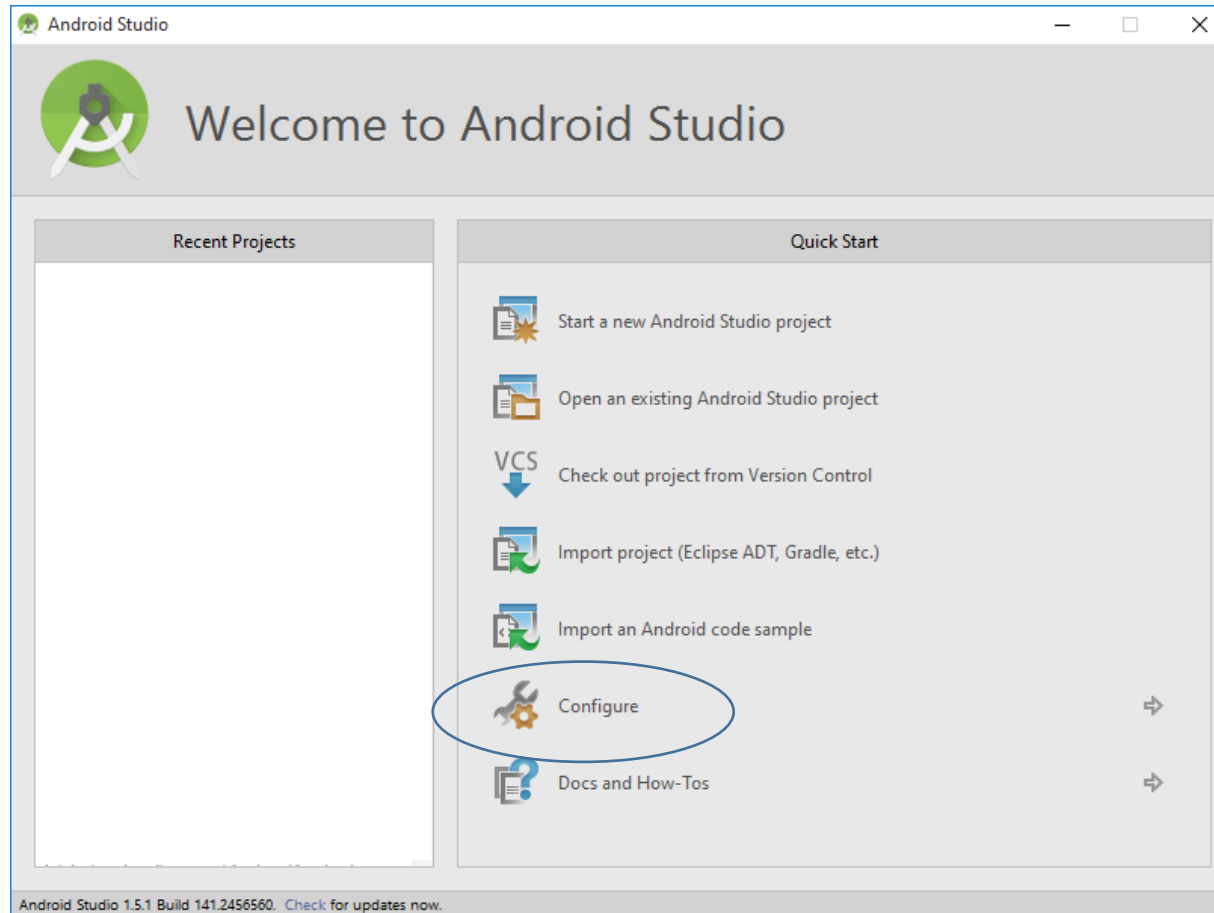
# Narzędzia potrzebne od zaraz

- Android Studio
- Android SDK

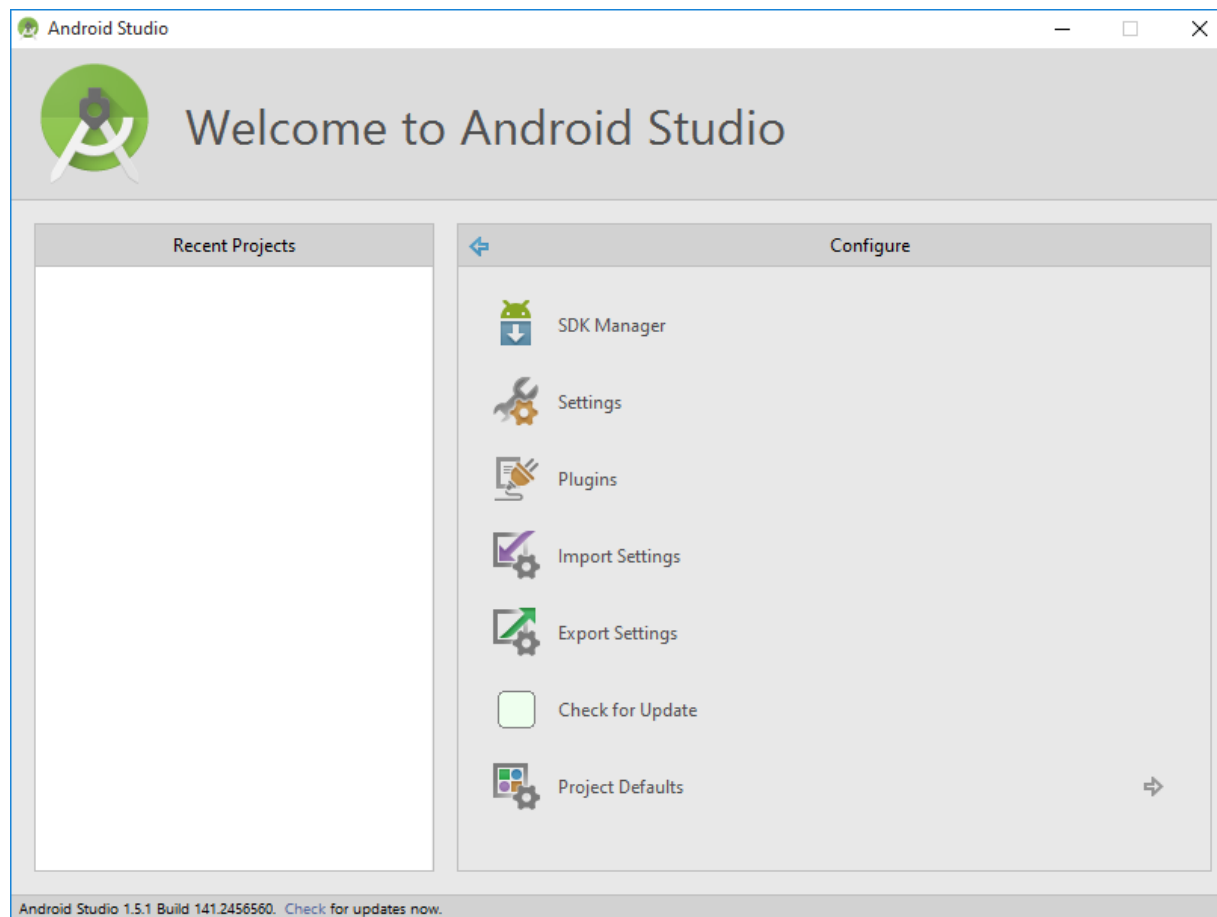
# Nowy projekt



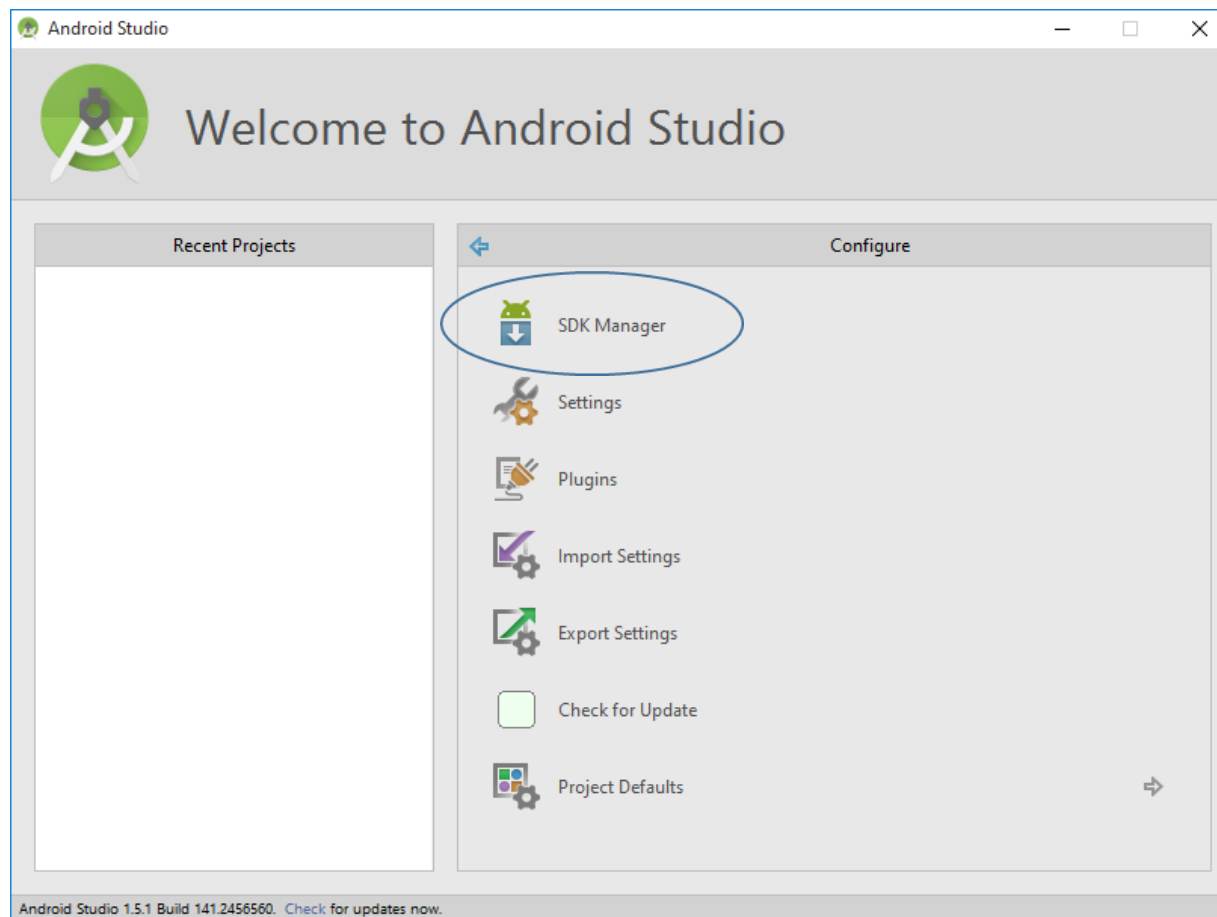
# Nowy projekt



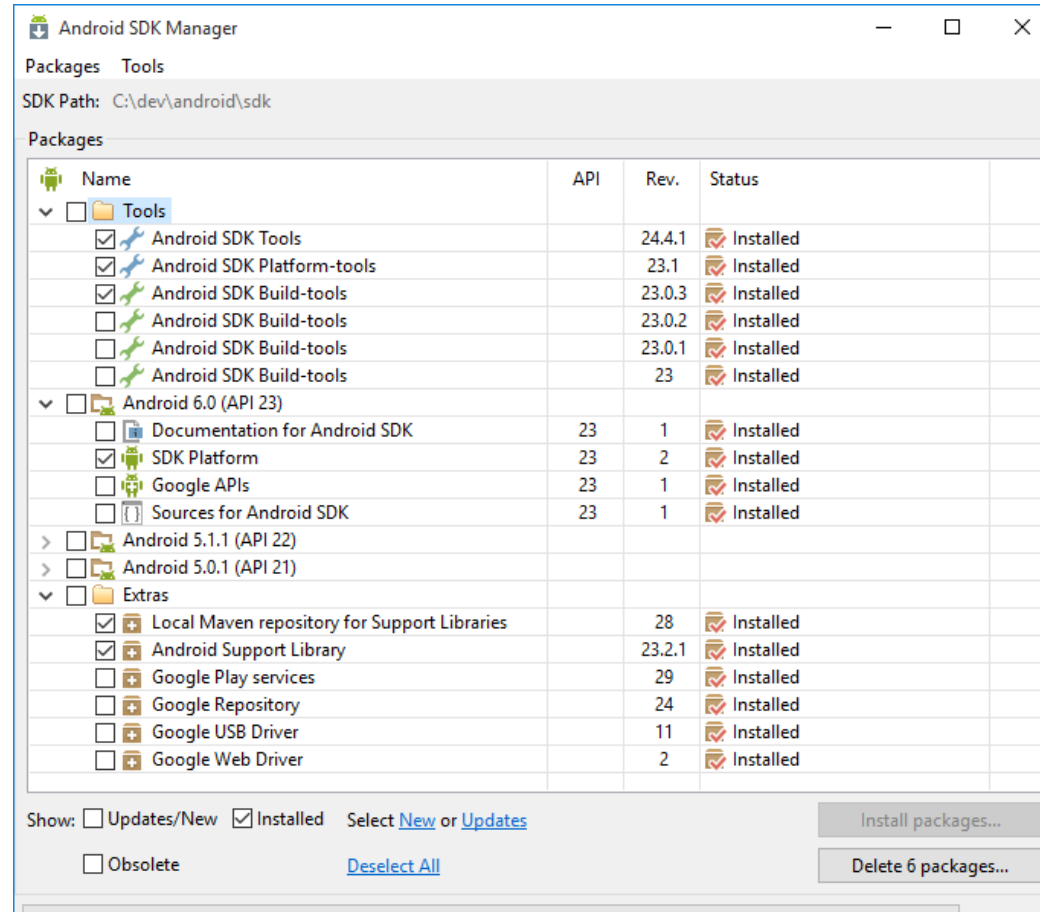
# Aktualizacja SDK



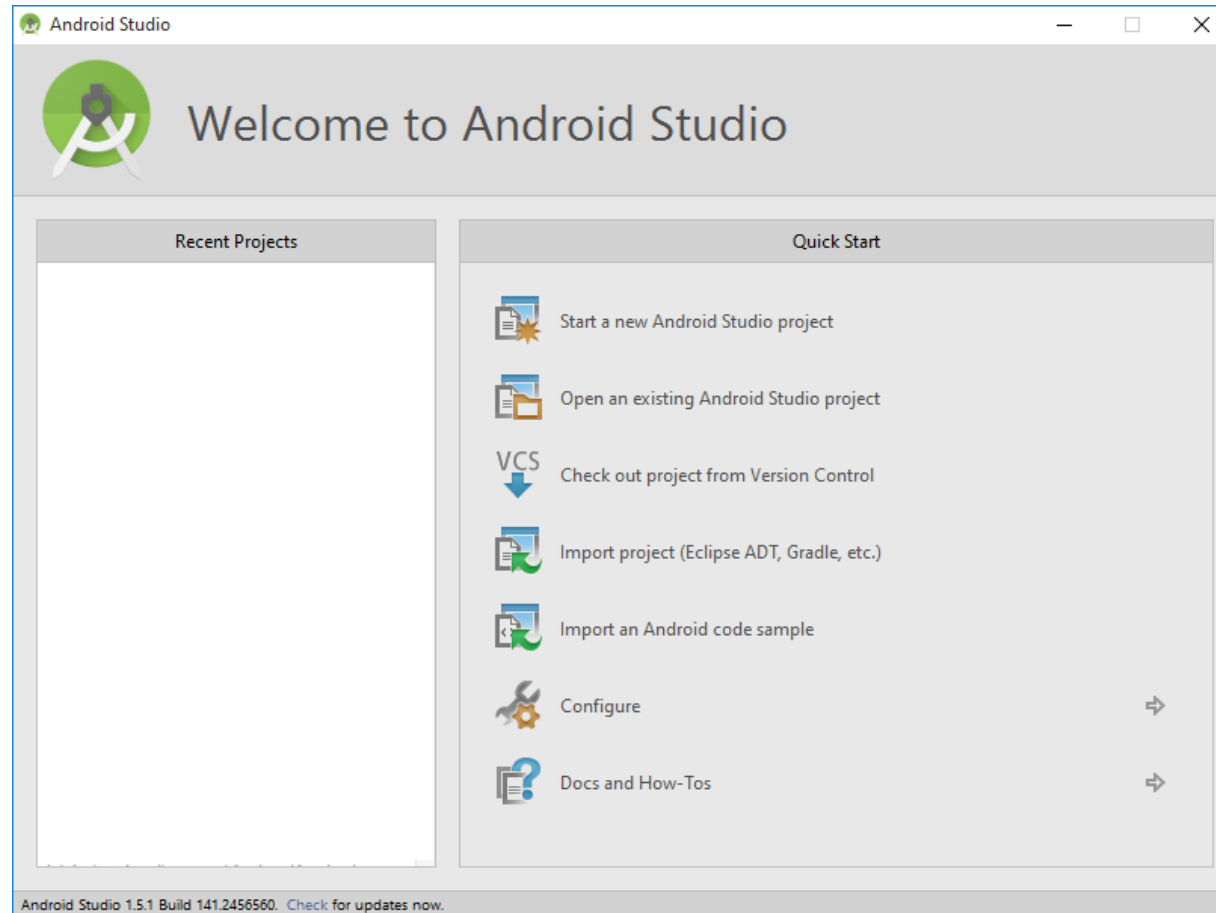
# Aktualizacja SDK



# Aktualizacja SDK

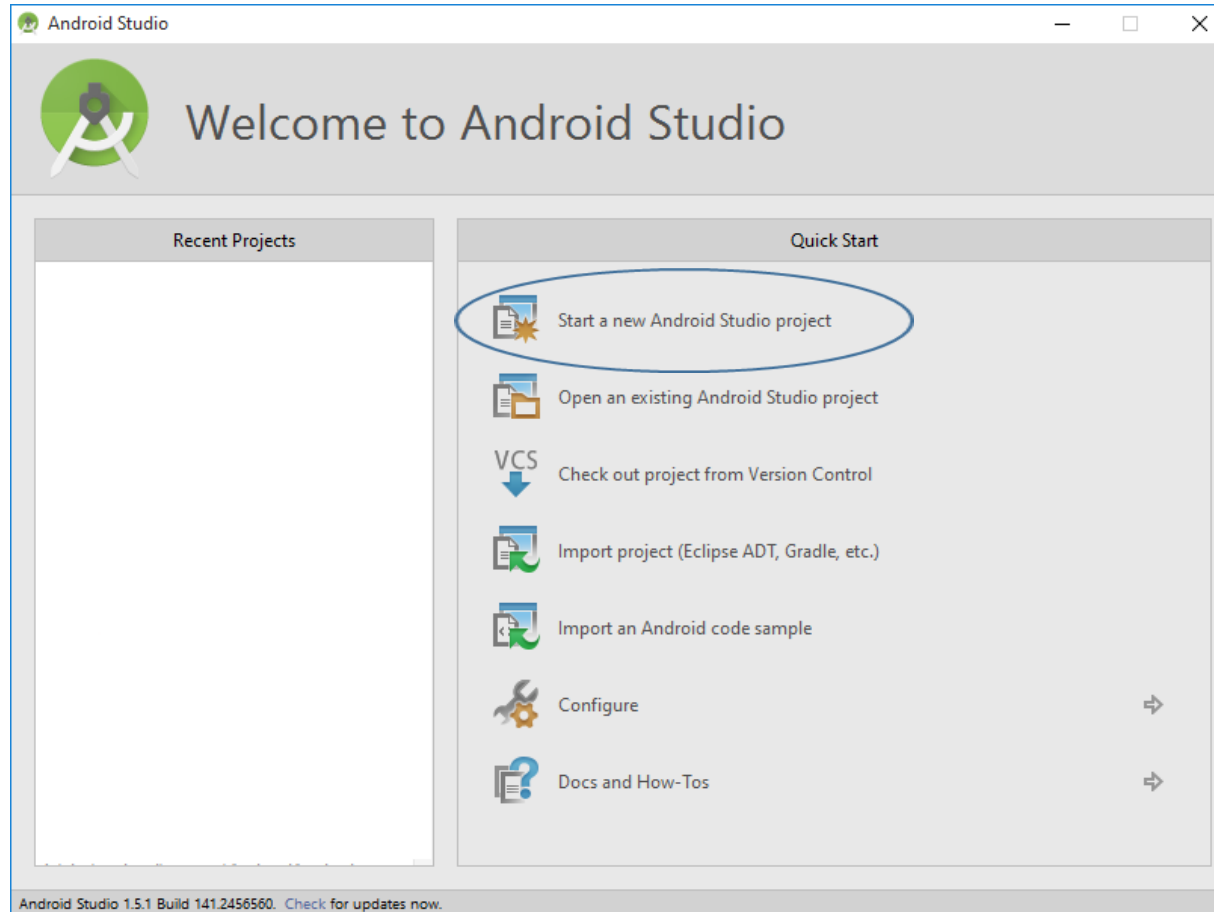


# Nowy projekt

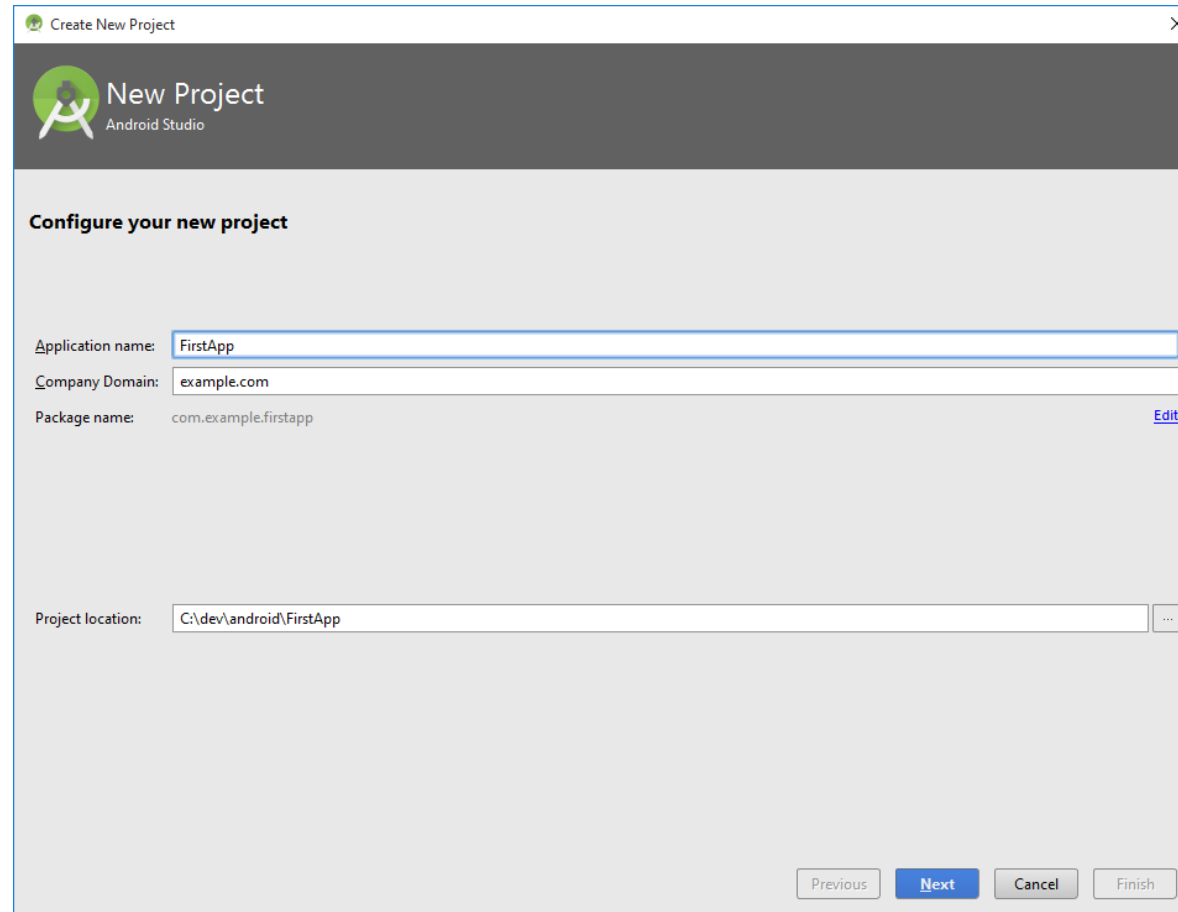





# Nowy projekt



# Nowy projekt - konfiguracja



Create New Project

 **New Project**  
Android Studio

**Configure your new project**

Application name:


Company Domain:

Package name:  [Edit](#)

Project location:

# Nowy projekt - wymagania

Create New Project

 Target Android Devices

**Select the form factors your app will run on**  
Different platforms may require separate SDKs

☒ Phone and Tablet

Minimum SDK

Lower API levels target more devices, but have fewer features available.  
By targeting API 15 and later, your app will run on approximately **97,3%** of the devices that are active on the Google Play Store.  
[Help me choose](#)

☐ Wear

Minimum SDK

☐ TV

Minimum SDK

☐ Android Auto

☐ Glass

Minimum SDK

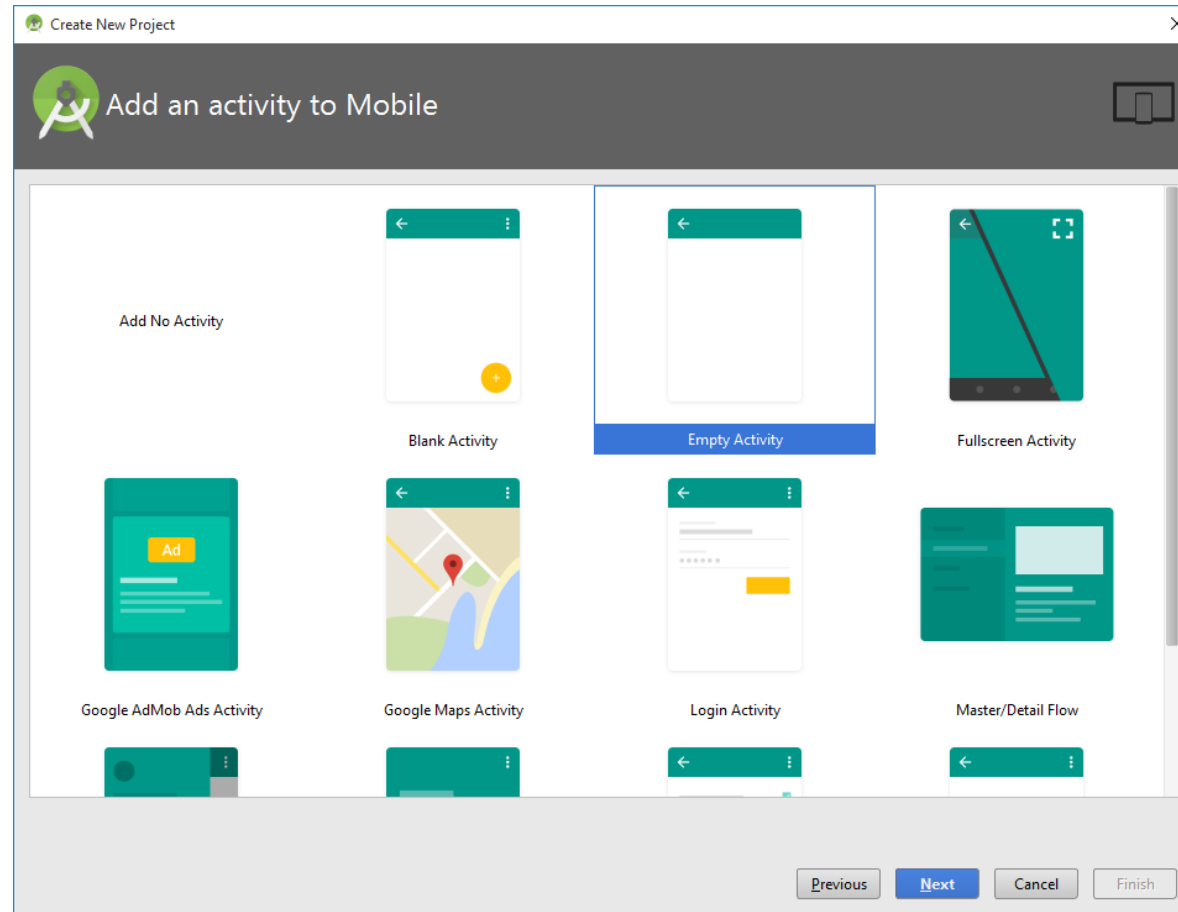
Previous

Next

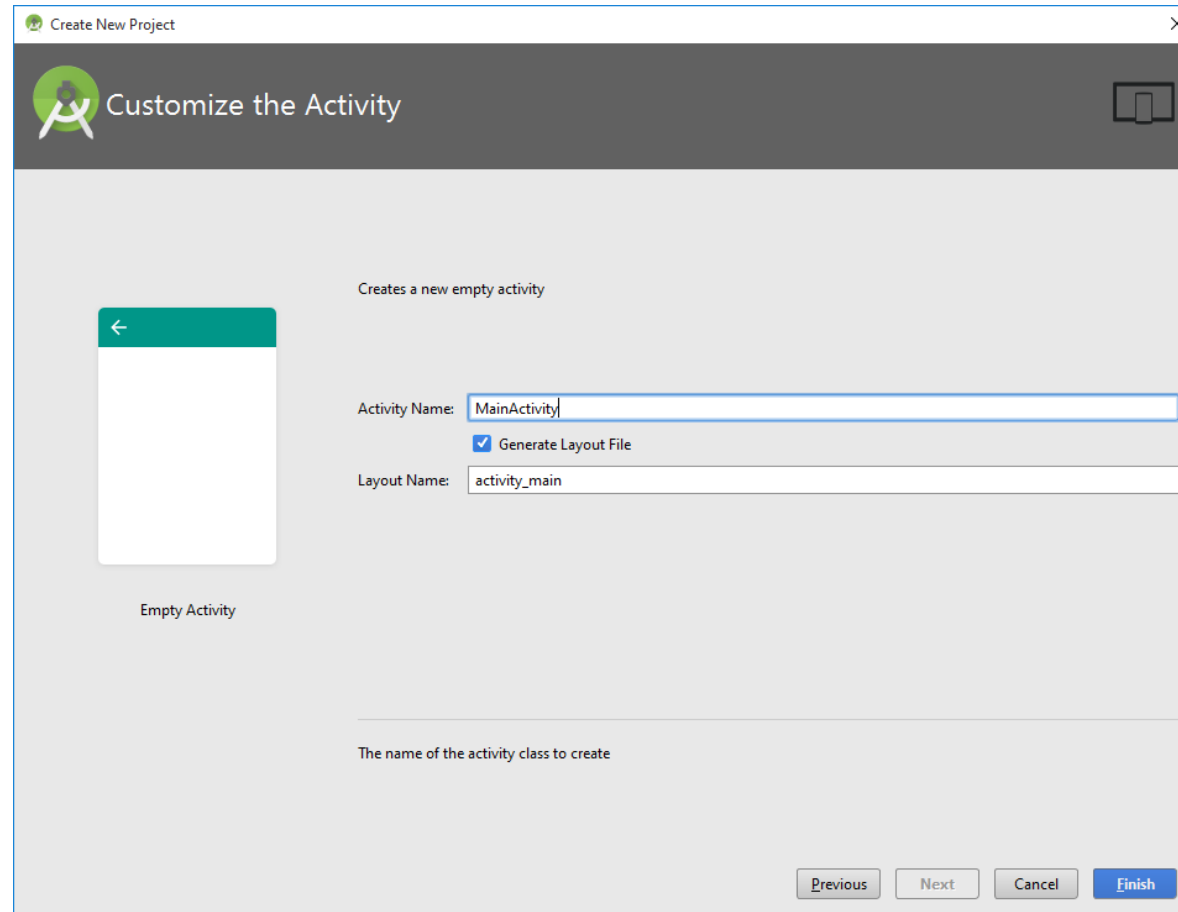
Cancel

Finish

# Nowy projekt – ekran startowy



# Nowy projekt – ekran startowy



The screenshot shows the 'Create New Project' dialog box in Android Studio. The title bar says 'Create New Project'. The main header is 'Customize the Activity' with an Android logo icon on the left and a mobile device icon on the right. The main content area is light gray. On the left, there is a preview of an 'Empty Activity' with a green header bar containing a white back arrow. To the right of the preview, the text 'Creates a new empty activity' is displayed. Below this, there are two text input fields: 'Activity Name' with the value 'MainActivity' and 'Layout Name' with the value 'activity\_main'. Between these fields is a checked checkbox labeled 'Generate Layout File'. At the bottom of the dialog, there is a horizontal line and the text 'The name of the activity class to create'. At the very bottom, there are four buttons: 'Previous', 'Next', 'Cancel', and 'Finish'.

Create New Project

Customize the Activity

Creates a new empty activity

Activity Name: MainActivity

☒ Generate Layout File

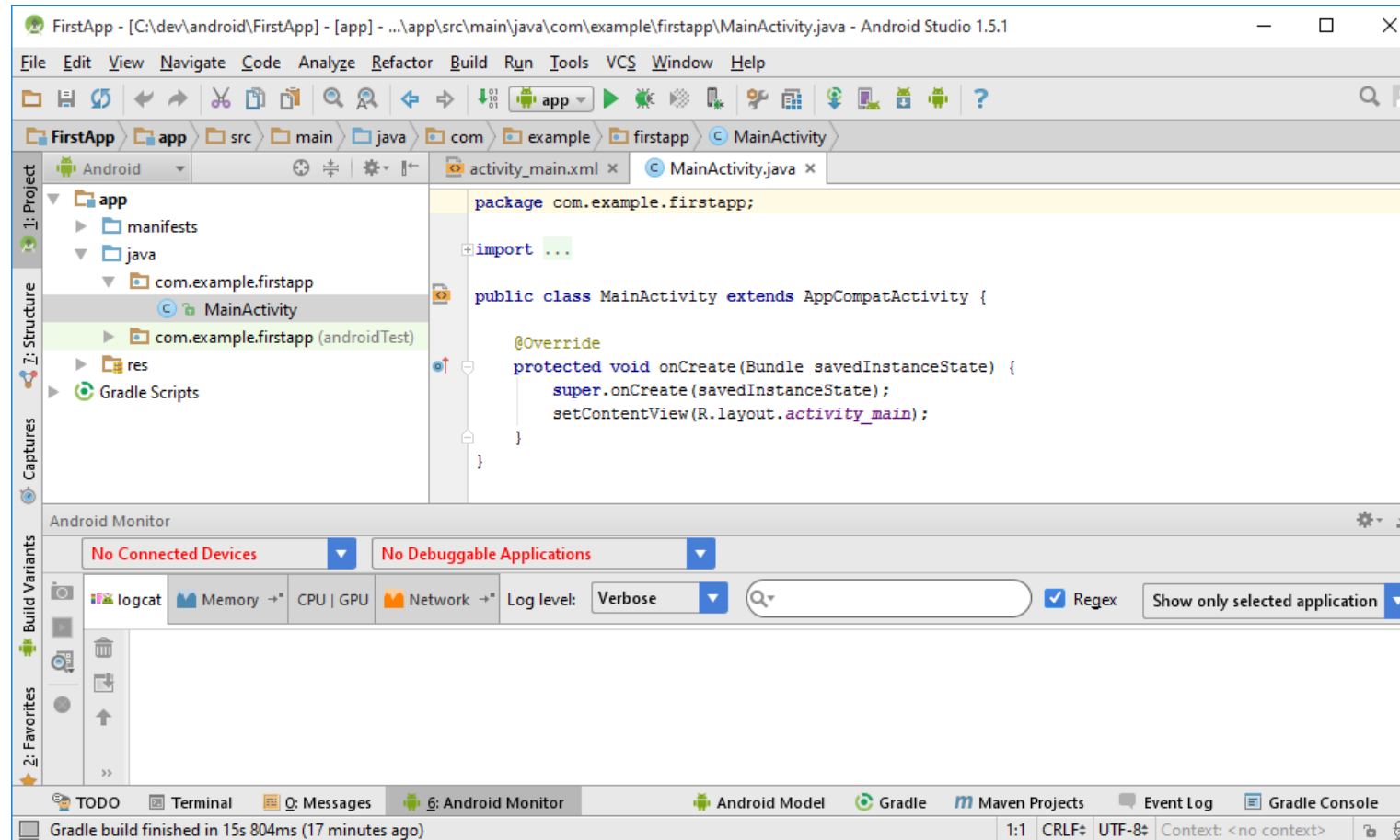
Layout Name: activity\_main

Empty Activity

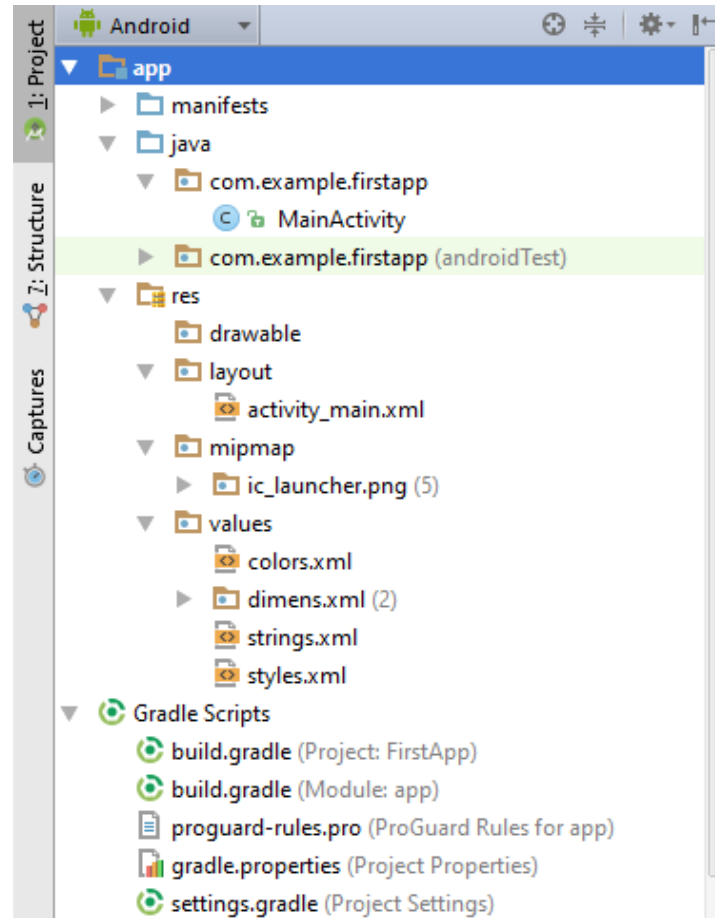
The name of the activity class to create

Previous Next Cancel Finish

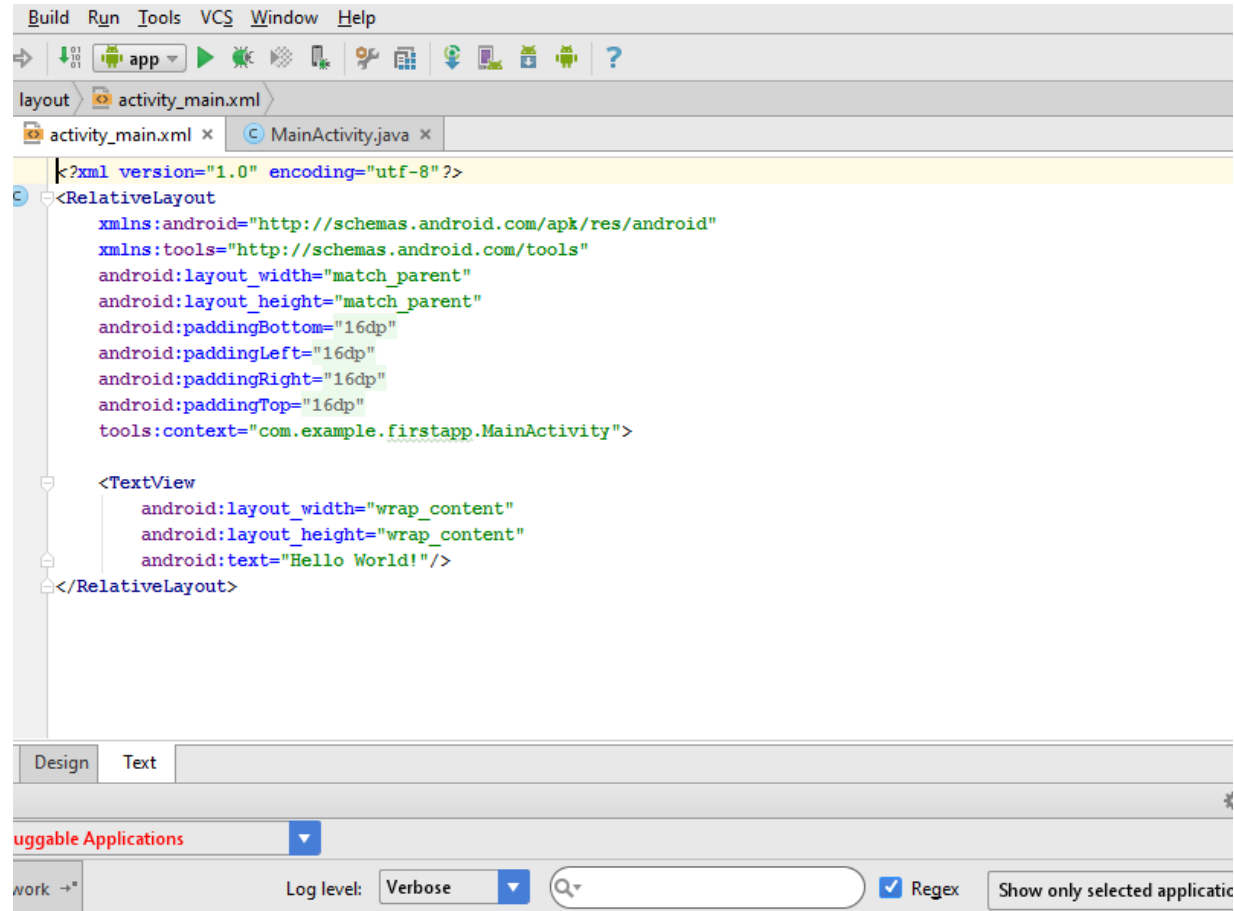
# Nowy projekt - gotowe



# Struktura projektu

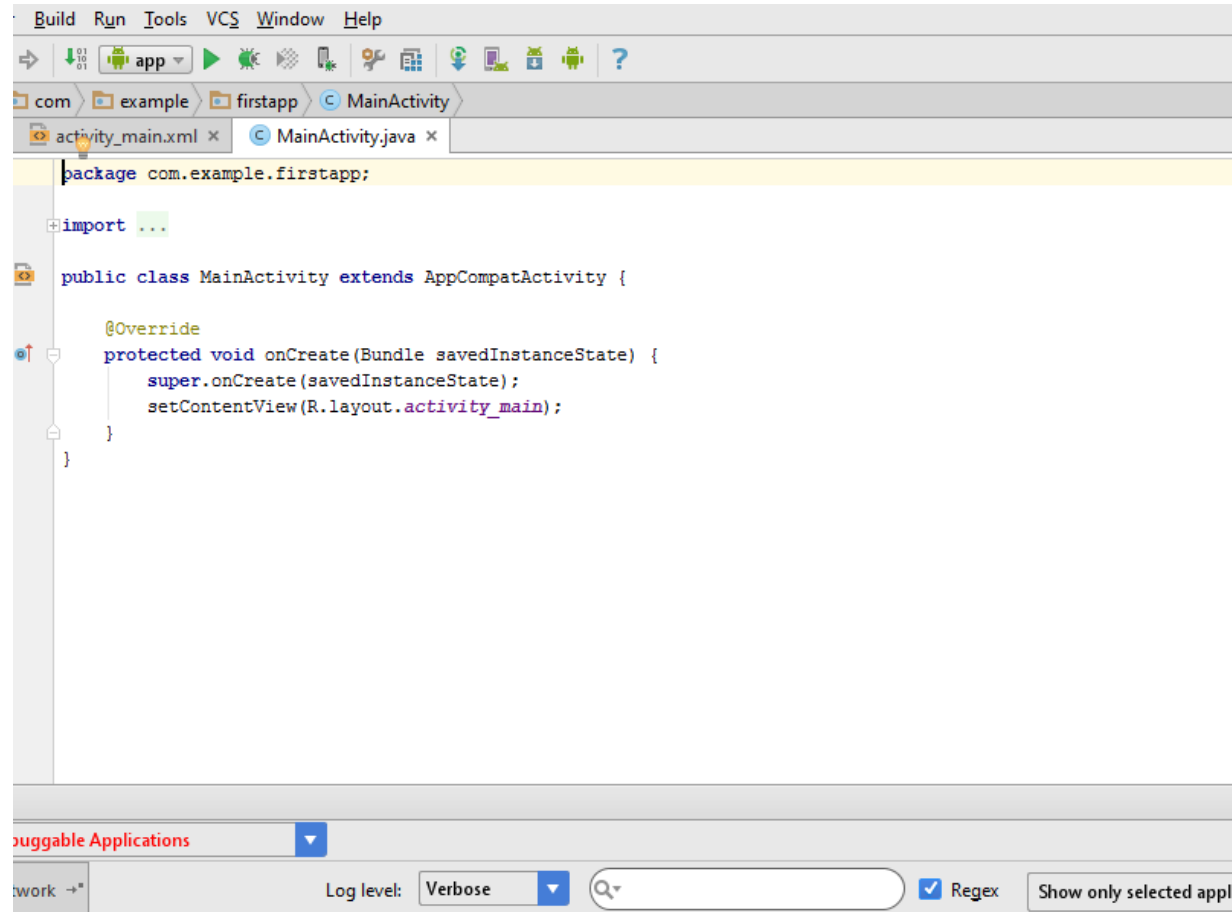


# Struktura projektu – Layout (widok)





# Struktura projektu – Activity (Kontroler)



```
Build Run Tools VCS Window Help
[Icons]
com > example > firstapp > MainActivity
activity_main.xml x MainActivity.java x
package com.example.firstapp;

import ...

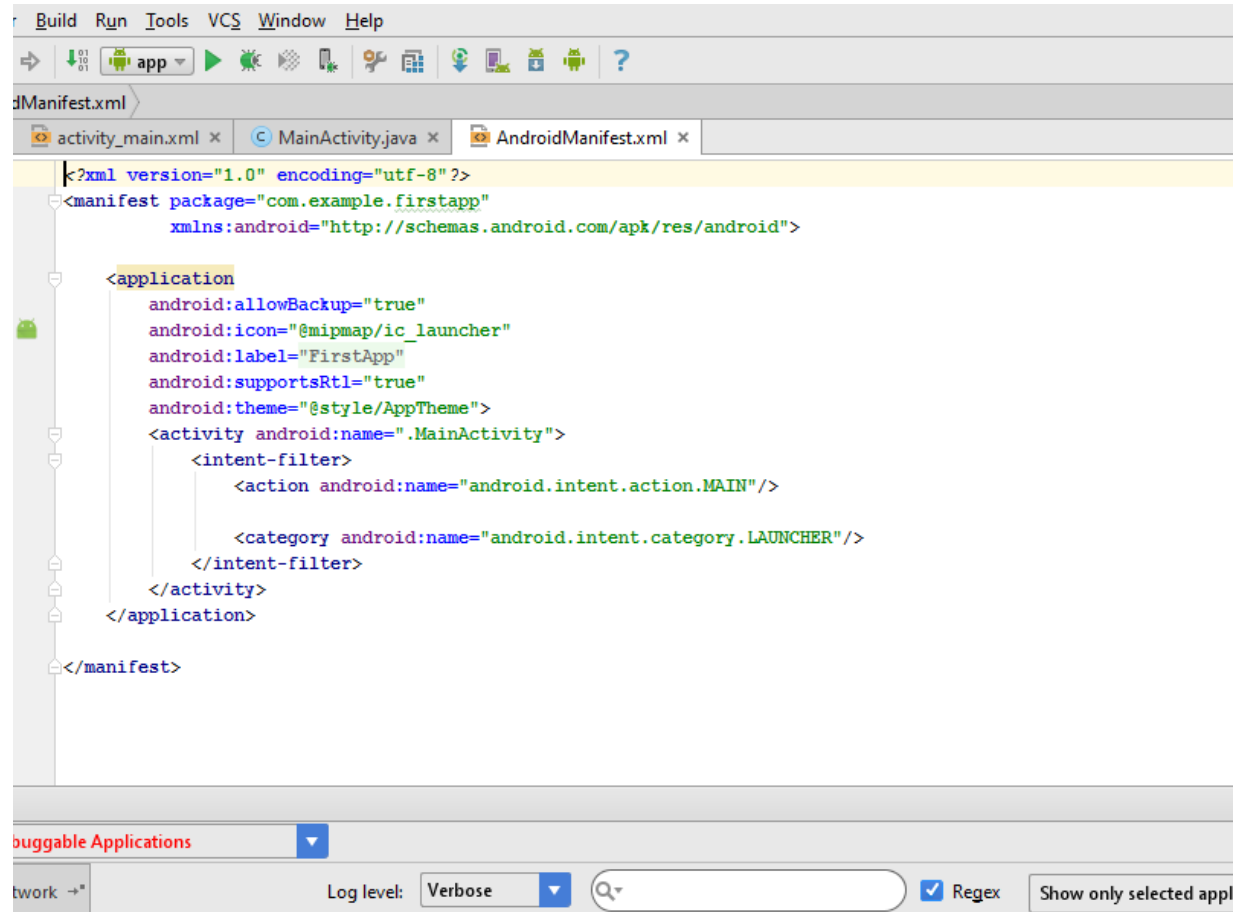
public class MainActivity extends AppCompatActivity {

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

Debuggable Applications

Log level: Verbose [Dropdown] [Search] [Regex] Show only selected appl

# Struktura projektu - Manifest



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

    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/first_app_label"
        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>
```

# Struktura projektu - Gradle

Narzędzie automatyzujące budowę oprogramowania dla języka Java.

## Download

---

Download [the latest JAR](#) or grab via Maven:

```
<dependency>
  <groupId>com.jakewharton</groupId>
  <artifactId>butterknife</artifactId>
  <version>7.0.1</version>
</dependency>
```

or Gradle:

```
compile 'com.jakewharton:butterknife:7.0.1'
```

## Download

---

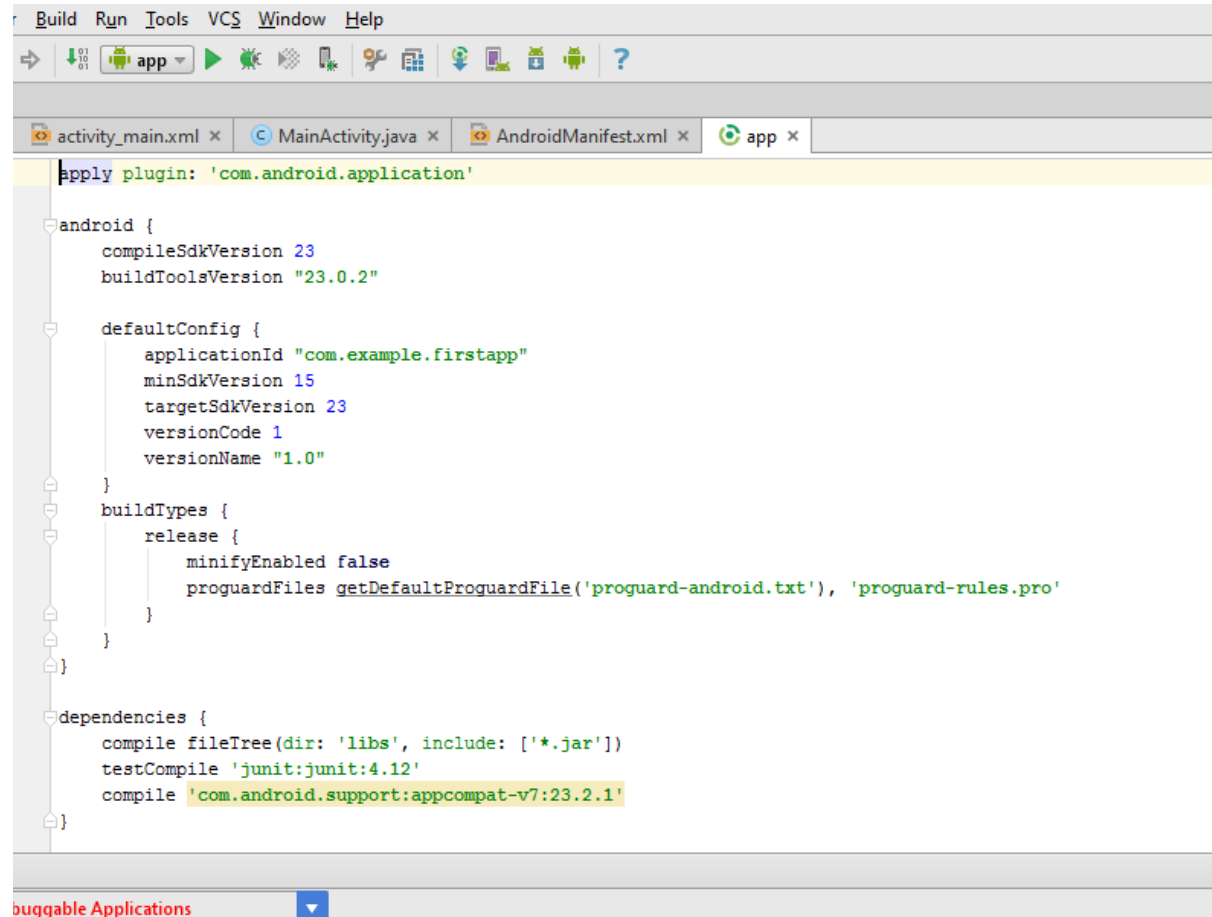
Download [the latest JAR](#) or grab via Gradle:

```
compile 'com.squareup.picasso:picasso:2.5.2'
```

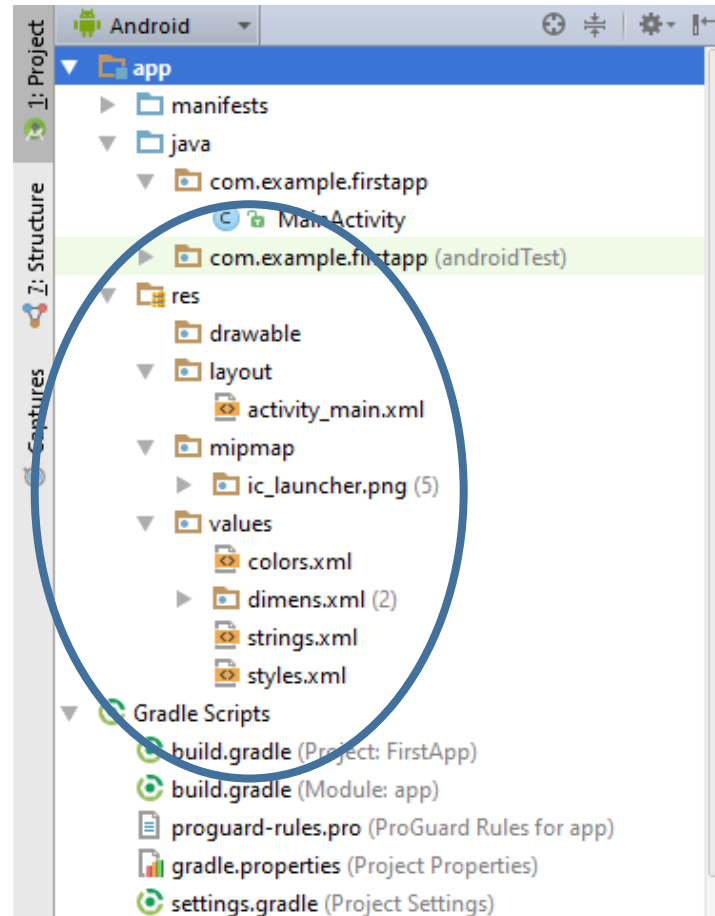
or Maven:

```
<dependency>
  <groupId>com.squareup.picasso</groupId>
  <artifactId>picasso</artifactId>
  <version>2.5.2</version>
</dependency>
```

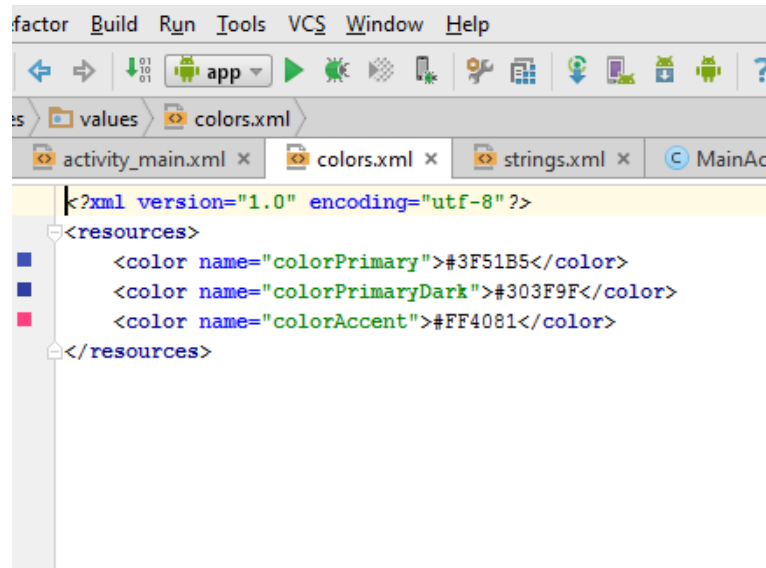
# Struktura projektu - Gradle



# Struktura projektu – res (resources)



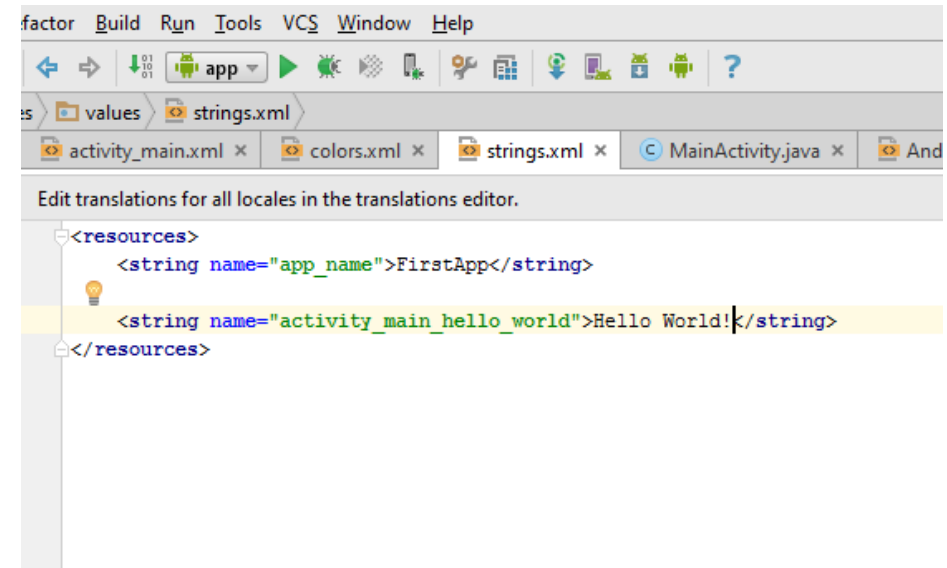
# Struktura projektu – res (resources)



The screenshot shows an IDE window with the file explorer on the left displaying the path 'res/values/colors.xml'. The main editor area shows the XML content of the file. The XML starts with a declaration of version '1.0' and encoding 'utf-8'. It then contains a root element '<resources>' which encloses three color definitions: 'colorPrimary' with hex value '#3F51B5', 'colorPrimaryDark' with hex value '#303F9F', and 'colorAccent' with hex value '#FF4081'. The file explorer also shows other files like 'activity\_main.xml', 'strings.xml', and 'MainActivity.java'.

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <color name="colorPrimary">#3F51B5</color>
    <color name="colorPrimaryDark">#303F9F</color>
    <color name="colorAccent">#FF4081</color>
</resources>
```

res/values/colors.xml



The screenshot shows an IDE window with the file explorer on the left displaying the path 'res/values/strings.xml'. The main editor area shows the XML content of the file. It starts with a root element '<resources>' which contains two string definitions: 'app\_name' with the value 'FirstApp' and 'activity\_main\_hello\_world' with the value 'Hello World!'. A lightbulb icon is visible next to the second string definition. The file explorer also shows other files like 'activity\_main.xml', 'colors.xml', 'MainActivity.java', and 'AndroidManifest.xml'.

```
<resources>
    <string name="app_name">FirstApp</string>
    <string name="activity_main_hello_world">Hello World!</string>
</resources>
```

res/values/strings.xml

# Uruchomienie aplikacji – sterowniki (krok 0)

Samsung

<http://developer.samsung.com/technical-doc/view.do?v=T000000117>

HTC

<http://www.htc.com/us/support/software/htc-sync-manager.aspx>

LG

<http://www.mylgphones.com/lg-android-usb-device-drivers>

Sony

<http://developer.sonymobile.com/downloads/drivers/>

# Uruchomienie aplikacji – sterowniki (krok 0)

Samsung

<http://developer.samsung.com/technical-doc/view.do?v=T000000117>

HTC


<http://www.htc.com/us/support/software/htc-sync-manager.aspx>

LG

<http://www.mylgphones.com/lg-android-usb-device-drivers>

Sony

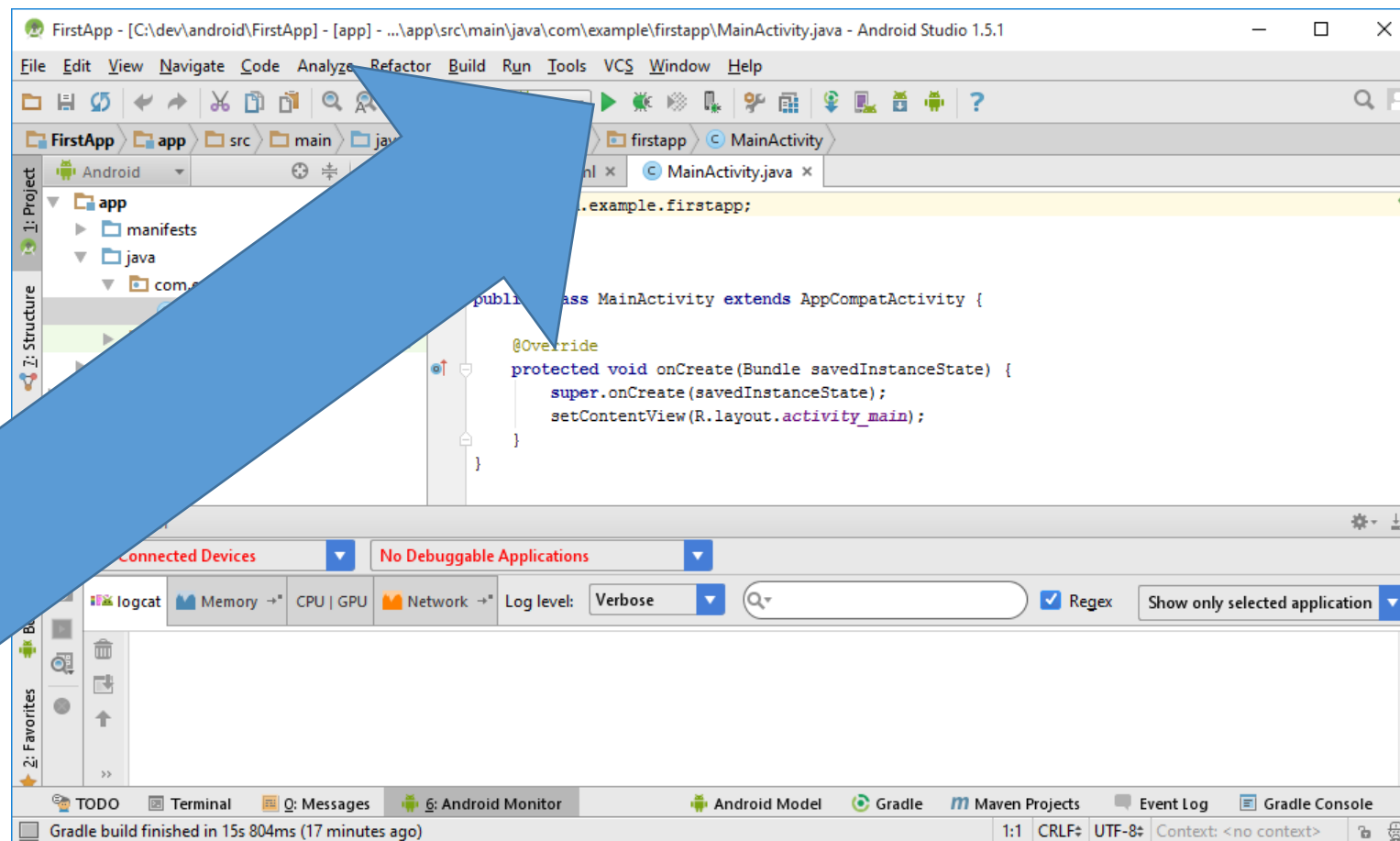
<http://developer.sonymobile.com/downloads/drivers/>



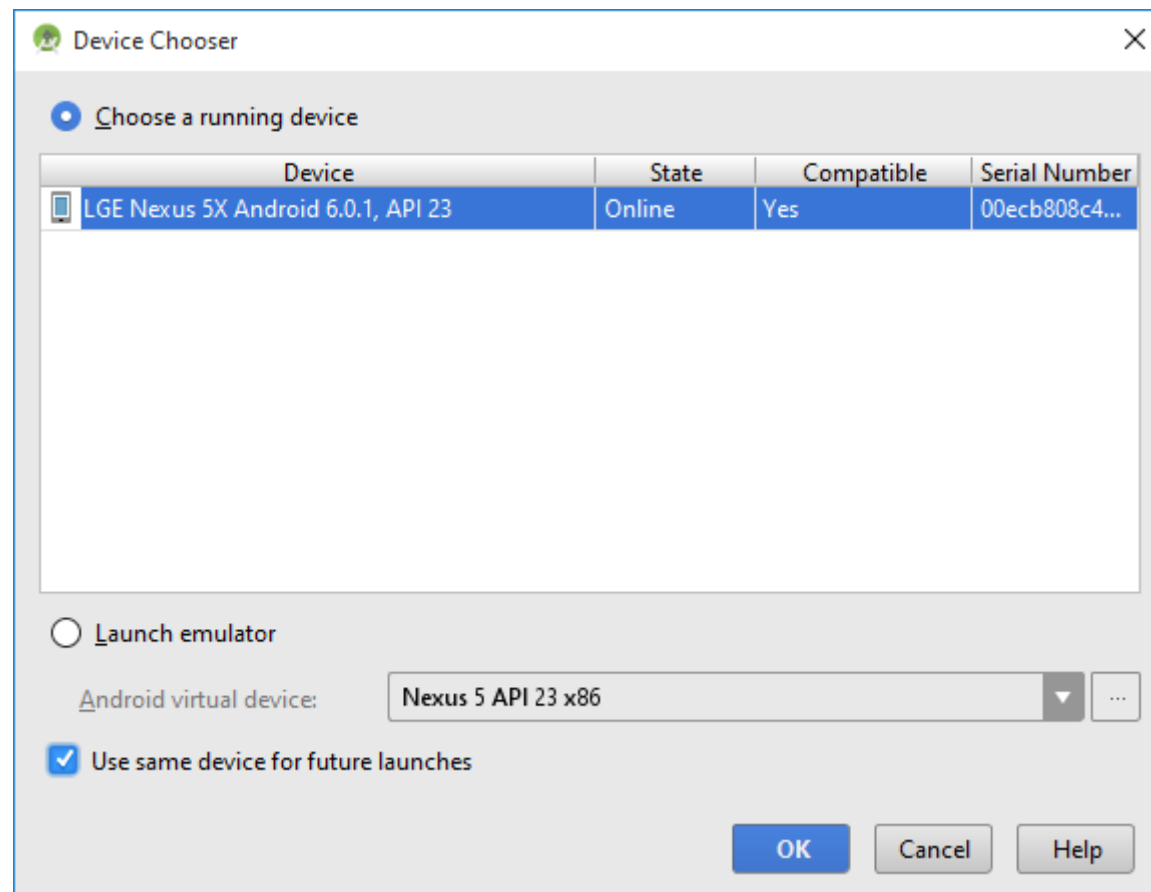
Google.pl -> „... usb drivers”



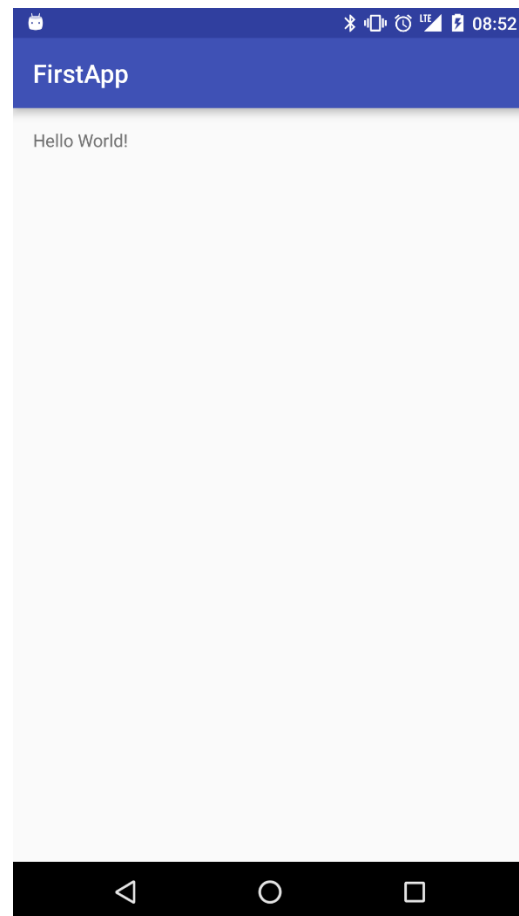
# Uruchomienie aplikacji – krok 1



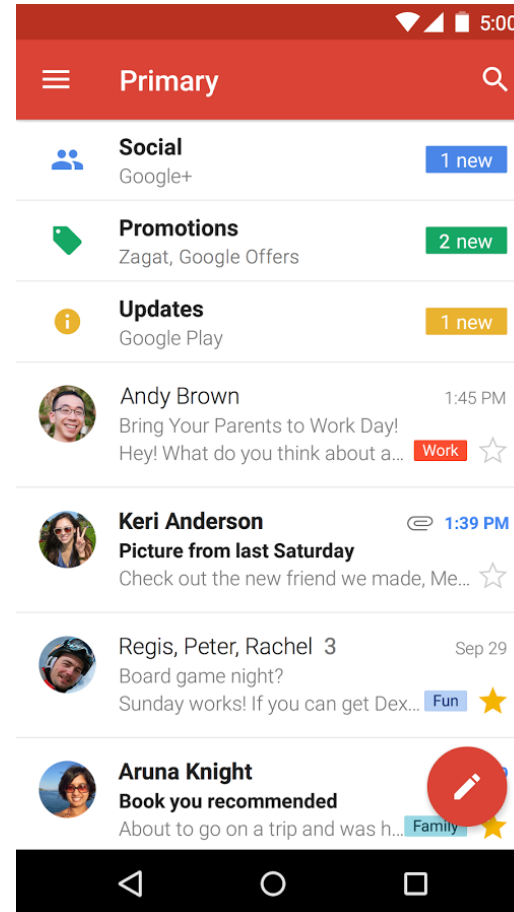
## Uruchomienie aplikacji – krok 2



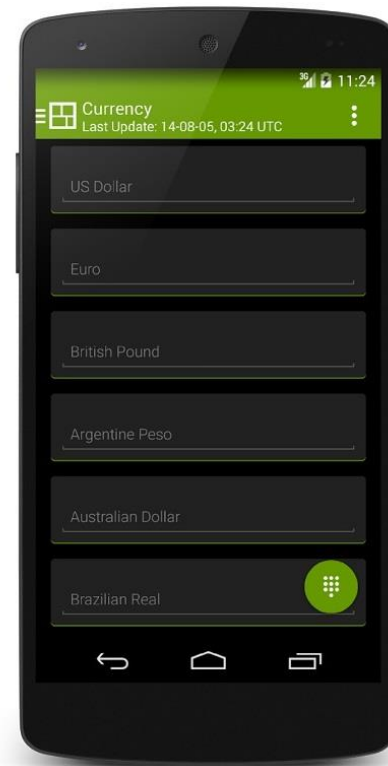
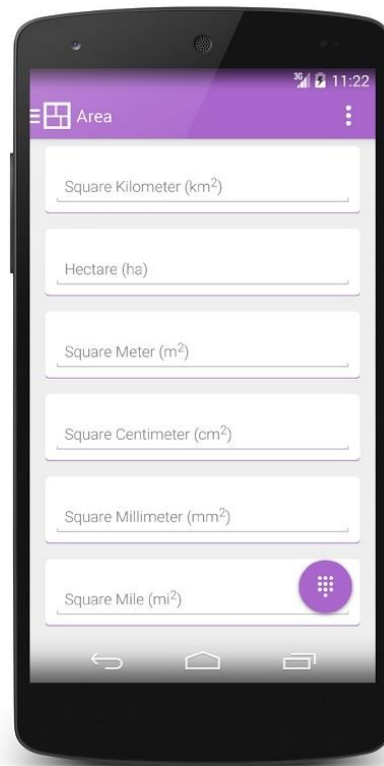
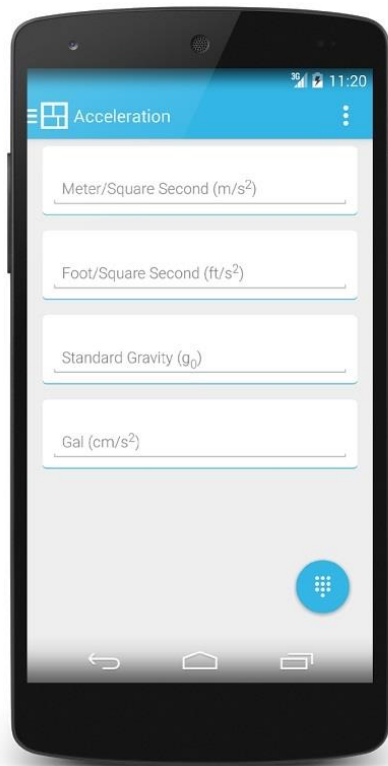
# Aplikacja uruchomiona na urządzeniu



# Material Design



# Material Design



# Kontrolki

```
<TextView
    android:id="@+id/edit_text_test"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:text="@string/activity_main_button_log_in"
    android:textColor="@color/colorPrimary"
    android:textSize="16sp"/>

<EditText
    android:id="@+id/edit_text_test"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:hint="@string/activity_main_edit_text_username_hint"
    android:inputType="numberDecimal"
    android:text="@string/activity_main_button_log_in"
    android:textColor="@color/colorPrimary"
    android:textSize="16sp"/>

<Button
    android:id="@+id/button_test"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="@string/activity_main_button_log_in"
    android:textColor="@color/colorPrimary"
    android:textSize="16sp"/>
```

# Kontrolki

```
<TextView
    android:id="@+id/edit_text_test"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:text="@string/activity_main_button_log_in"
    android:textColor="@color/colorPrimary"
    android:textSize="16sp"/>

<EditText
    android:id="@+id/edit_text_test"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:hint="@string/activity_main_edit_text_username_hint"
    android:inputType="numberDecimal"
    android:text="@string/activity_main_button_log_in"
    android:textColor="@color/colorPrimary"
    android:textSize="16sp"/>

<Button
    android:id="@+id/button_test"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="@string/activity_main_button_log_in"
    android:textColor="@color/colorPrimary"
    android:textSize="16sp"/>
```

POLE TEKSTOWE (Output)

# Kontrolki

```
<TextView
    android:id="@+id/edit_text_test"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:text="@string/activity_main_button_log_in"
    android:textColor="@color/colorPrimary"
    android:textSize="16sp"/>
```

POLE TEKSTOWE (Output)

```
<EditText
    android:id="@+id/edit_text_test"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:hint="@string/activity_main_edit_text_username_hint"
    android:inputType="numberDecimal"
    android:text="@string/activity_main_button_log_in"
    android:textColor="@color/colorPrimary"
    android:textSize="16sp"/>
```

POLE DO WPROWADZANIA (Input)

```
<Button
    android:id="@+id/button_test"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="@string/activity_main_button_log_in"
    android:textColor="@color/colorPrimary"
    android:textSize="16sp"/>
```



# Kontrolki

```
<TextView
    android:id="@+id/edit_text_test"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:text="@string/activity_main_button_log_in"
    android:textColor="@color/colorPrimary"
    android:textSize="16sp"/>
```

POLE TEKSTOWE (Output)

```
<EditText
    android:id="@+id/edit_text_test"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:hint="@string/activity_main_edit_text_username_hint"
    android:inputType="numberDecimal"
    android:text="@string/activity_main_button_log_in"
    android:textColor="@color/colorPrimary"
    android:textSize="16sp"/>
```

POLE DO WPROWADZANIA (Input)

```
<Button
    android:id="@+id/button_test"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="@string/activity_main_button_log_in"
    android:textColor="@color/colorPrimary"
    android:textSize="16sp"/>
```

PRZYCISK

# RelativeLayout

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

    <TextView
        android:id="@+id/text_view_test"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentTop="true"/>

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_below="@id/text_view_test"/>

    <TextView
        android:id="@+id/text_view_center_test"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_centerInParent="true"/>

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_toRightOf="@id/text_view_center_test"/>

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentBottom="true"/>

</RelativeLayout>
```

# LinearLayout

```
<?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:orientation="vertical">

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"/>

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="center"/>

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

        <TextView
            android:layout_width="0dp"
            android:layout_height="wrap_content"
            android:layout_weight="0.5"/>

        <TextView
            android:layout_width="0dp"
            android:layout_height="wrap_content"
            android:layout_weight="0.5"/>

    </LinearLayout>
</LinearLayout>
```

Widok – ekran logowania

CHWILA CISZY...

# Widok – ekran logowania

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingBottom="16dp"
    android:paddingLeft="16dp"
    android:paddingRight="16dp"
    android:paddingTop="@dimen/activity_vertical_margin">

    <TextView
        android:id="@+id/text_view_header"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentTop="true"
        android:layout_centerHorizontal="true"
        android:layout_marginTop="32dp"
        android:text="Logowanie"
        android:textSize="26sp"/>
```

```
<LinearLayout
    android:id="@+id/linear_layout_user_data"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_below="@id/text_view_header"
    android:layout_marginTop="32dp"
    android:orientation="vertical">

    <EditText
        android:id="@+id/edit_text_username"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:hint="nazwa użytkownika"
        android:singleLine="true"/>

    <EditText
        android:id="@+id/edit_text_password"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:hint="hasło"
        android:inputType="textPassword"
        android:singleLine="true"/>

</LinearLayout>

<Button
    android:id="@+id/button_log_in"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_below="@id/linear_layout_user_data"
    android:layout_centerHorizontal="true"
    android:layout_marginTop="32dp"
    android:background="@color/colorAccent"
    android:text="Zaloguj"
    android:textColor="@android:color/white"/>

</RelativeLayout>
```

# Widok od strony kodu

```
public class MainActivity extends AppCompatActivity {

    EditText editTextUsername;
    EditText editTextPassword;
    Button buttonLogIn;

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

        initComponents();
    }

    private void initComponents() {
        editTextUsername = (EditText) findViewById(R.id.edit_text_username);
        editTextPassword = (EditText) findViewById(R.id.edit_text_password);
        buttonLogIn = (Button) findViewById(R.id.button_log_in);
    }
}
```

# Widok od strony kodu - lepiej

```
public class MainActivity extends AppCompatActivity {  
  
    @Bind(R.id.edit_text_username)  
    EditText editTextUsername;  
    @Bind(R.id.edit_text_password)  
    EditText editTextPassword;  
    @Bind(R.id.button_log_in)  
    Button buttonLogIn;  
  
    @Override  
    protected void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
        setContentView(R.layout.activity_main);  
    }  
}
```

# Widok od strony kodu

```
public class MainActivity extends AppCompatActivity {

    EditText editTextUsername;
    EditText editTextPassword;
    Button buttonLogIn;

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

        initUiComponents();
    }

    private void initUiComponents() {
        editTextUsername = (EditText) findViewById(R.id.edit_text_username);
        editTextPassword = (EditText) findViewById(R.id.edit_text_password);
        buttonLogIn = (Button) findViewById(R.id.button_log_in);

        buttonLogIn.setOnClickListener(getOnButtonLogInClick());
    }

    private View.OnClickListener getOnButtonLogInClick() {
        return new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                // do something
            }
        };
    }
}
```



# Widok od strony kodu - lepiej

```
public class MainActivity extends AppCompatActivity {

    @Bind(R.id.edit_text_username)
    EditText editTextUsername;
    @Bind(R.id.edit_text_password)
    EditText editTextPassword;
    @Bind(R.id.button_log_in)
    Button buttonLogIn;

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

    @OnClick(R.id.button_log_in)
    public void onButtonLogInClick() {
        // do something
    }
}
```

# Jak pisać kod?

```
public static final int NICE_CONSTANT = 42;
```

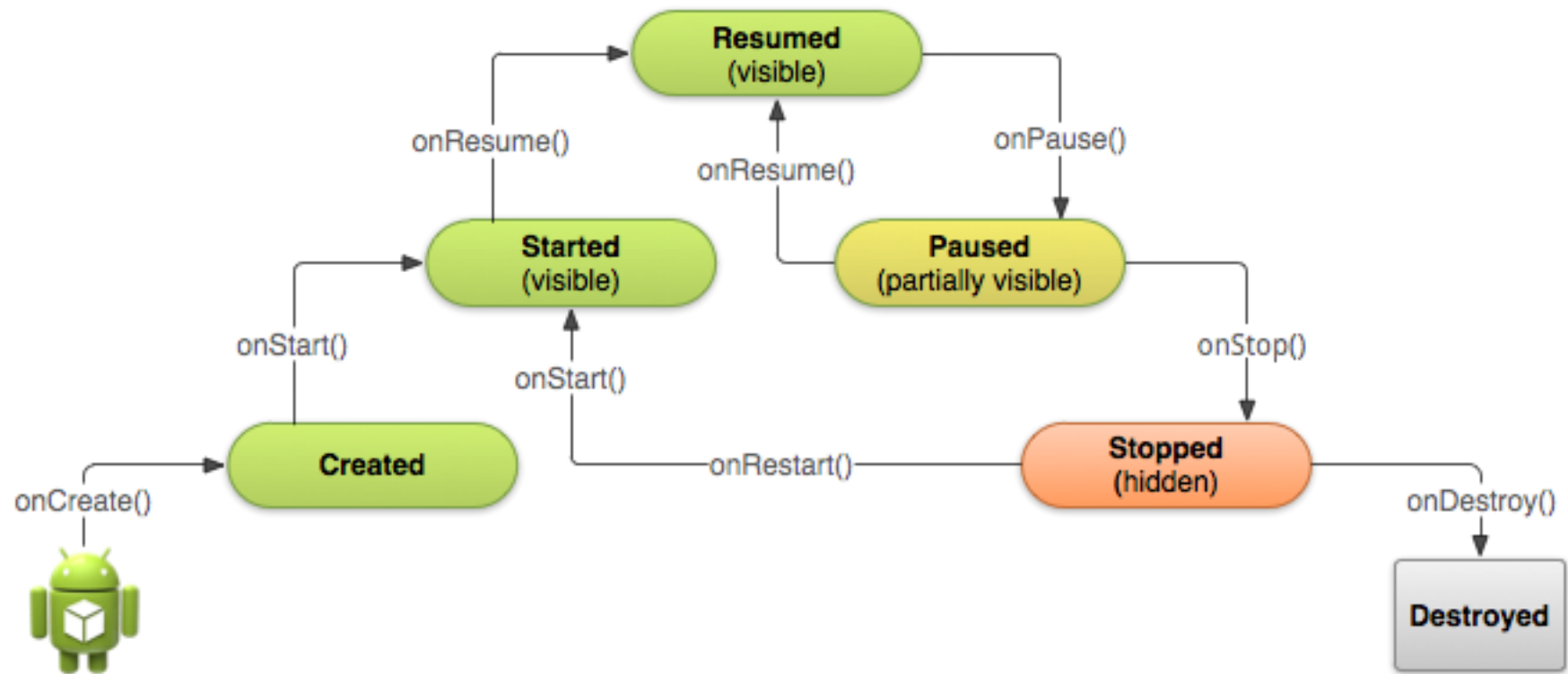
```
public int publicField;
```

```
private static MyClass sSingleton;
```

```
int mPackagePrivate;
```

```
private(protected) int mPrivate;
```

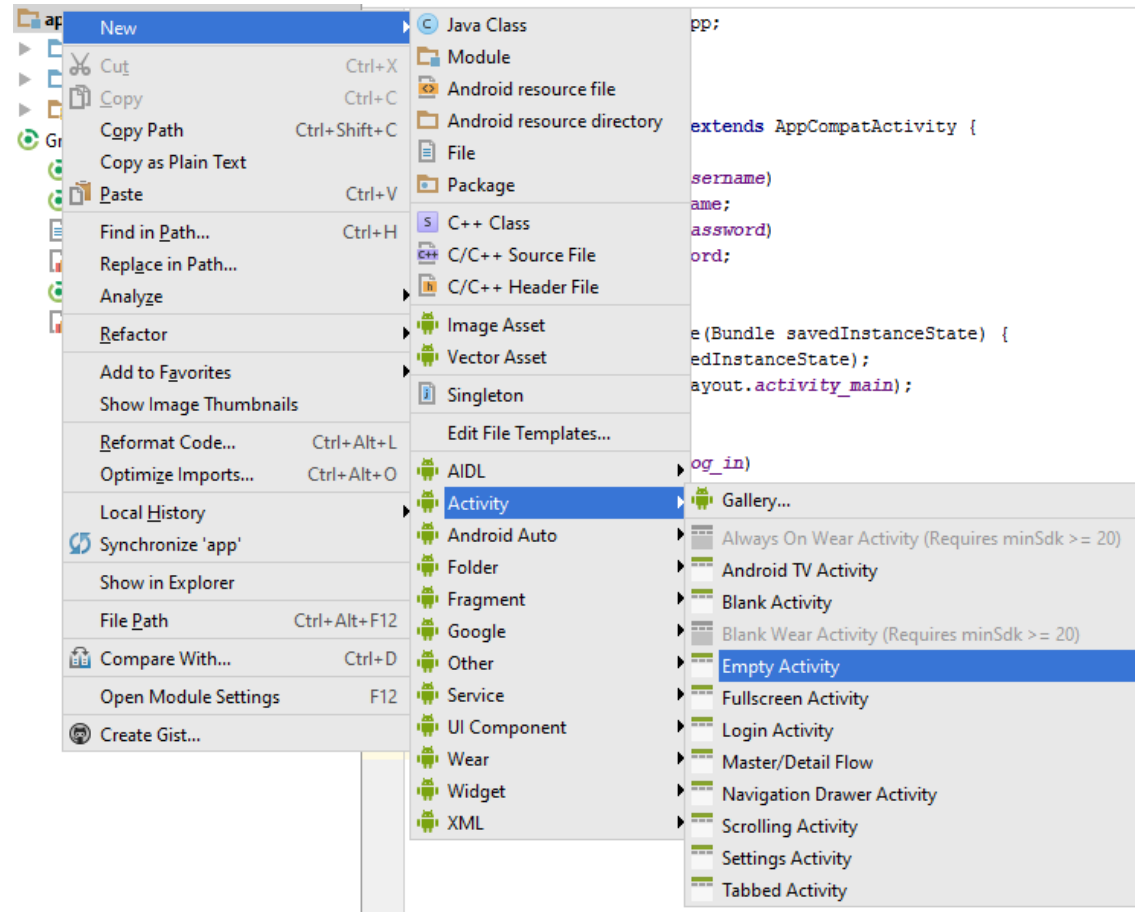
# Activity – cykl życia



# Activity – cykl życia

```
public class LifecycleActivity extends AppCompatActivity {  
  
    @Override  
    protected void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
        setContentView(R.layout.activity_life_cycle);  
    }  
  
    @Override  
    protected void onStart() {  
        super.onStart();  
    }  
  
    @Override  
    protected void onResume() {  
        super.onResume();  
    }  
  
    @Override  
    protected void onPause() {  
        super.onPause();  
    }  
  
    @Override  
    protected void onStop() {  
        super.onStop();  
    }  
  
    @Override  
    protected void onDestroy() {  
        super.onDestroy();  
    }  
}
```

# Tworzenie nowego Activity



# Tworzenie nowego Activity - konfiguracja

New Android Activity

Customize the Activity

Creates a new empty activity

Activity Name:

☒ Generate Layout File

Layout Name:

☐ Launcher Activity

Package name:

Target Source Set:

Empty Activity

Previous Next Cancel Finish

# Przejdźcie do nowego Activity

```
public class MainActivity extends AppCompatActivity {

    @Bind(R.id.edit_text_username)
    EditText mEditTextUsername;
    @Bind(R.id.edit_text_password)
    EditText mEditTextPassword;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        ButterKnife.bind(this);
    }

    @OnClick(R.id.button_log_in)
    public void onButtonLogInClick() {
        String username = mEditTextUsername.getText().toString();
        String password = mEditTextPassword.getText().toString();

        Intent menuIntent = new Intent(this, MenuActivity.class);
        menuIntent.putExtra(IntentExtras.USERNAME, username);
        menuIntent.putExtra(IntentExtras.PASSWORD, password);
        startActivity(menuIntent);
        finish();
    }
}
```

# Przykład odbioru danych w Activity

```
public class MenuActivity extends AppCompatActivity {

    private String mUsername;
    private String mPassword;

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

        getData();

        Toast.makeText(this, mUsername, Toast.LENGTH_LONG).show();
    }

    private void getData() {
        Bundle bundle = getIntent().getExtras();
        if (bundle != null) {
            mUsername = bundle.getString(IntentExtras.USERNAME);
            mPassword = bundle.getString(IntentExtras.PASSWORD);
        }
    }
}
```



# ListView (Lista)

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout
    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="com.example.firstapp.ClientListActivity">

    <ListView
        android:id="@+id/list_view_clients"
        android:layout_width="match_parent"
        android:layout_height="match_parent"/>

</RelativeLayout>
```

```
public class ClientListActivity extends AppCompatActivity {

    @Bind(R.id.list_view_clients)
    ListView mListViewClients;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_client_list);
        ButterKnife.bind(this);
    }
}
```

# Adapter dla ListView

```
public class ClientListActivity extends AppCompatActivity {

    @Bind(R.id.list_view_clients)
    ListView mListViewClients;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_client_list);
        ButterKnife.bind(this);

        initUiComponents();
    }

    private void initUiComponents() {
        String[] clients = {"Pierwszy klient", "Drugi klient"};
        ArrayAdapter<String> clientsAdapter = new ArrayAdapter<>(this, android.R.layout.simple_list_item_1, clients);
        mListViewClients.setAdapter(clientsAdapter);
    }
}
```

# Własny Adapter

```
public class SimpleAdapter extends BaseAdapter {

    @Override
    public int getCount() {
        return 0;
    }

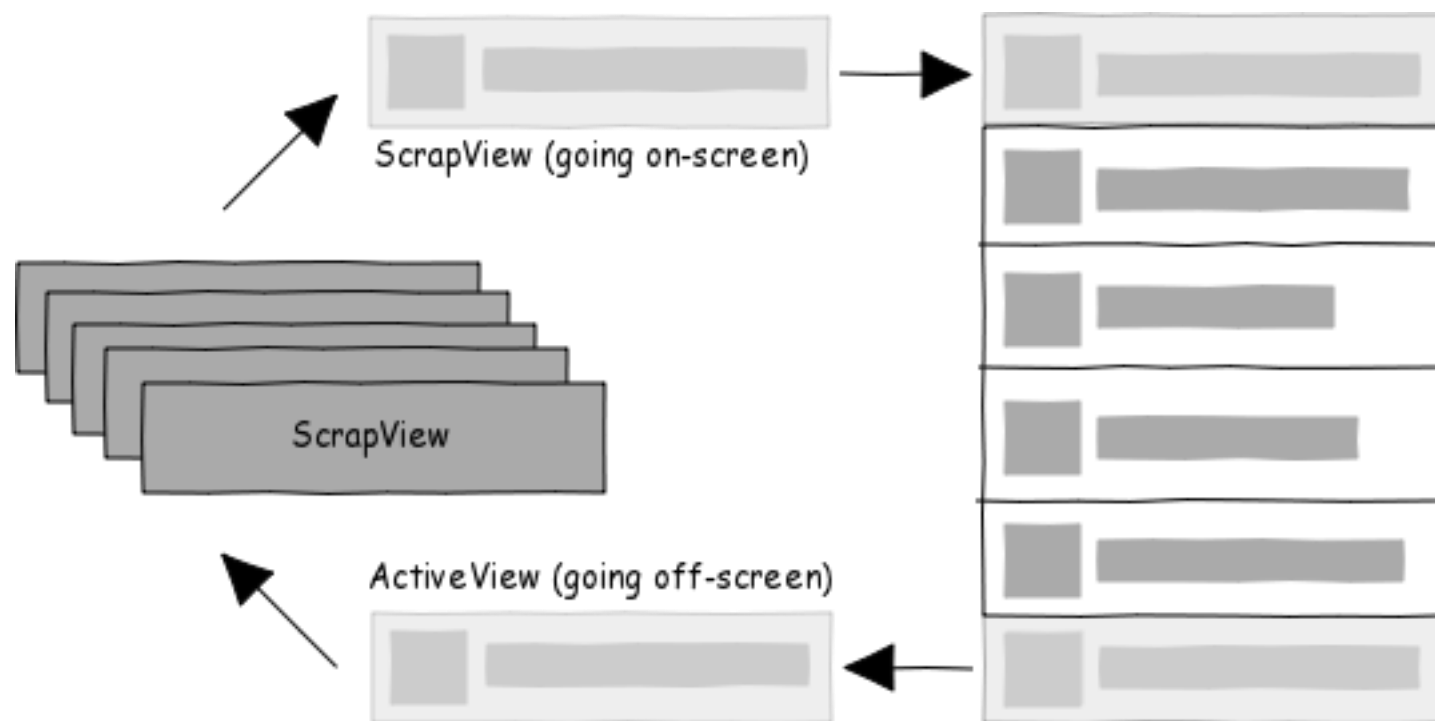
    @Override
    public Object getItem(int position) {
        return null;
    }

    @Override
    public long getItemId(int position) {
        return 0;
    }

    @Override
    public View getView(int position, View convertView, ViewGroup parent) {
        if (convertView == null) {
            convertView = LayoutInflater.from(parent.getContext()).inflate(R.layout.row_clients_list, parent, false);
        }

        return convertView;
    }
}
```

# Działanie ListView



Własny Adapter

Chwila ciszy...

# Własny Adapter

```
public class ClientsListAdapter extends BaseAdapter {

    List<Client> mClientsList;

    public ClientsListAdapter(List<Client> clientsList) {
        mClientsList = clientsList;
    }

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

    @Override
    public Client getItem(int position) {
        return mClientsList.get(position);
    }

    @Override
    public long getItemId(int position) {
        Client client = mClientsList.get(position);
        return client.getId();
    }
}
```

# Własny Adapter

```
public class ClientsListAdapter extends BaseAdapter {

    @Override
    public View getView(int position, View convertView, ViewGroup parent) {
        if (convertView == null) {
            convertView
                = LayoutInflater
                    .from(parent.getContext())
                    .inflate(R.layout.row_clients_list, parent, false);
        }

        Client client = mClientsList.get(position);
        // find view by id etc

        return convertView;
    }
}
```

# ViewHolder do Adaptera

```
static class ViewHolder {  
  
    @Bind(R.id.text_view_client_name)  
    TextView clientName;  
  
    public ViewHolder(View view) {  
        ButterKnife.bind(this, view);  
    }  
}
```



# Własny Adapter z ViewHolder'em

```
@Override
public View getView(int position, View convertView, ViewGroup parent) {
    ViewHolder holder;
    if (convertView != null) {
        holder = (ViewHolder) convertView.getTag();
    } else {
        convertView
            = LayoutInflater.from(parent.getContext())
                .inflate(R.layout.row_clients_list, parent, false);
        holder = new ViewHolder(convertView);
        convertView.setTag(holder);
    }

    Client client = mClientsList.get(position);
    holder.clientName.setText(client.getName());

    return convertView;
}
```

# Programowanie na platformę Android

Część 2

# IntentExtras – co to za magia?

```
public class IntentExtras {  
  
    public static final String USERNAME = "username";  
    public static final String PASSWORD = "password";  
    public static final String CLIENT = "client";  
}
```

## Adapter dla kolekcji obiektów – nasz obiekt klienta

```
public class Client {  
  
    private Long mId;  
    private Long mExternalId;  
    private String mName;  
    private String mAddress;  
    private String mPhone;  
    private String mEmail;  
  
    public Client() {  
  
    }  
  
    // gettery/settery  
}
```

# Adapter dla kolekcji obiektów

```
public class ClientsListAdapter extends BaseAdapter {

    List<Client> mClientsList;

    public ClientsListAdapter(List<Client> clientsList) {
        mClientsList = clientsList;
    }

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

    @Override
    public Client getItem(int position) {
        return mClientsList.get(position);
    }

    @Override
    public long getItemId(int position) {...}
}
```

# Adapter dla kolekcji obiektów

```
@Override
public View getView(int position, View convertView, ViewGroup parent) {
    ViewHolder holder;
    if (convertView != null) {
        ...
    } else {
        ...
    }

    Client client = mClientsList.get(position);
    holder.clientName.setText(client.getName());
    holder.clientAddress.setText(client.getAddress());

    return convertView;
}
```

## Wypełnienie listy kolekcją obiektów

```
private void initComponents() {  
    mClientsList = new ArrayList<>();  
  
    for (long i = 1; i <= 30; i++) {  
        Client client = new Client();  
        client.setId(i);  
        client.setName(String.format("Klient %s", i));  
        client.setAddress("1 Maja 133 Katowice");  
        client.setEmail(String.format("test%s@o2.pl", i));  
        client.setPhone(String.format("123456%s", i));  
        mClientsList.add(client);  
    }  
  
    mClientsListAdapter = new ClientsListAdapter(mClientsList);  
    mListViewClients.setAdapter(mClientsListAdapter);  
}
```

# String.format()

```
public static String format(String format, Object... args)
```

```
String.format("Uwaga %s", "Wszyscy");
```

```
String.format("Uwaga %s, %s", "Wszyscy", "uwaga");
```

Zamiast:


```
"Uwaga" + " Wszyscy, " + "uwaga"
```



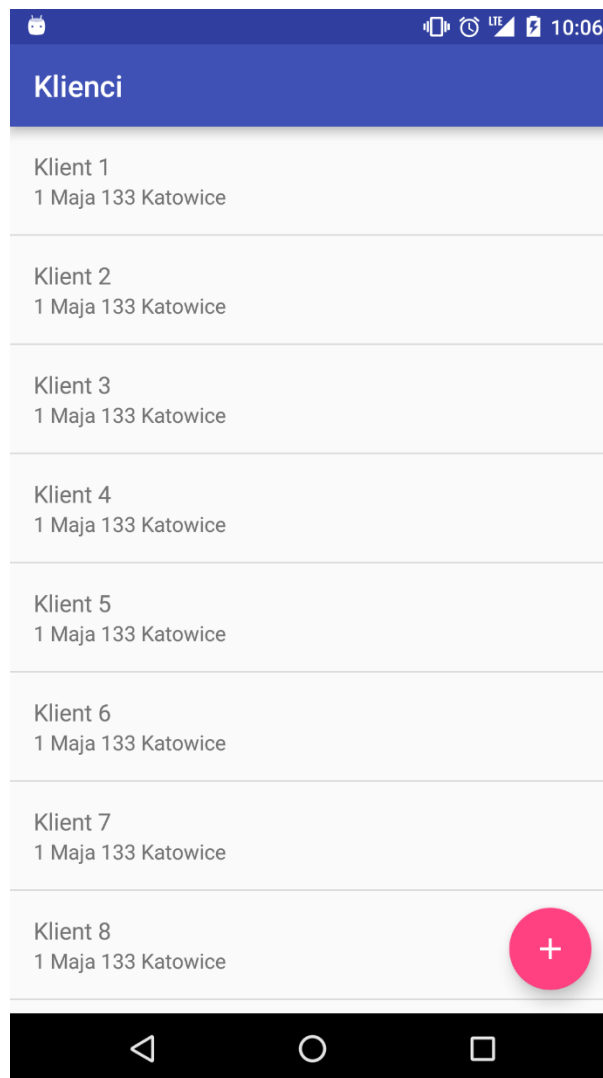
## Adapter – jak odświeżyć listę?

```
mClientsListAdapter.setClientsList(mClientsList);  
mClientsListAdapter.notifyDataSetChanged();
```

```
public setClientsList(List<Client> clientsList) {  
    mClientsList = clientsList;  
}
```



# Wypełnienie listy kolekcją obiektów



# CoordinatorLayout

compile 'com.android.support:design:[wersja] (aktualnie 23.2.1)

```
<android.support.design.widget.CoordinatorLayout  
    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">
```

Często używany przy tworzeniu ciekawych layout'ów z wykorzystaniem Material Design. Również umożliwia kontrolę zależności pomiędzy kontrolkami.

# Floating Action Button - kontrolka

```
<?xml version="1.0" encoding="utf-8"?>
<android.support.design.widget.CoordinatorLayout
    ...>

    <ListView
        android:id="@+id/list_view_clients"
        android:layout_width="match_parent"
        android:layout_height="match_parent"/>

    <android.support.design.widget.FloatingActionButton
        android:id="@+id/floating_action_button_add_client"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_margin="16dp"
        android:padding="0dp"
        android:src="@mipmap/ic_add"
        app:elevation="8dp"
        app:layout_anchor="@id/list_view_clients"
        app:layout_anchorGravity="bottom|right|end"/>

</android.support.design.widget.CoordinatorLayout>
```

# Floating Action Button - klik

```
@OnClick(R.id.floating_action_button_add_client)
public void onFABAddClientClick() {
    Intent intent = new Intent(this, AddClientActivity.class);
    startActivityForResult(intent, REQUEST_CODE);
}
```

## Nowe Activity – startActivity() vs startActivityForResult()

```
Intent intent = new Intent(this, ClientDetailsActivity.class);  
startActivity(intent);
```

**VS**

```
Intent intent = new Intent(this, AddClientActivity.class);  
startActivityForResult(intent, REQUEST_CODE);
```

## Nowe Activity – startActivityForResult()

```
Intent resultIntent = new Intent();  
resultIntent.putExtra(IntentExtras.CLIENT, client);  
setResult(Activity.RESULT_OK, resultIntent);  
finish();
```

```
@Override  
protected void onActivityResult(int requestCode, int resultCode, Intent data) {  
    if (REQUEST_CODE == requestCode && RESULT_OK == resultCode && data != null) {  
        Client client = data.getExtras().getParcelable(IntentExtras.CLIENT);  
        mClientsList.add(client);  
        mClientsListAdapter.notifyDataSetChanged();  
    }  
}
```

# Parcelable

```
public class Client implements Parcelable {  
    ...  
    protected Client(Parcel in) {  
    }  
  
    @Override  
    public int describeContents() {  
        return 0;  
    }  
  
    @Override  
    public void writeToParcel(Parcel dest, int flags) {  
    }  
    ...  
}
```



# Parcelable

```
public class Client implements Parcelable {  
    ...  
    public static final Creator<Client> CREATOR = new Creator<Client>() {  
        @Override  
        public Client createFromParcel(Parcel in) {  
            return new Client(in);  
        }  
  
        @Override  
        public Client[] newArray(int size) {  
            return new Client[size];  
        }  
    };  
}
```

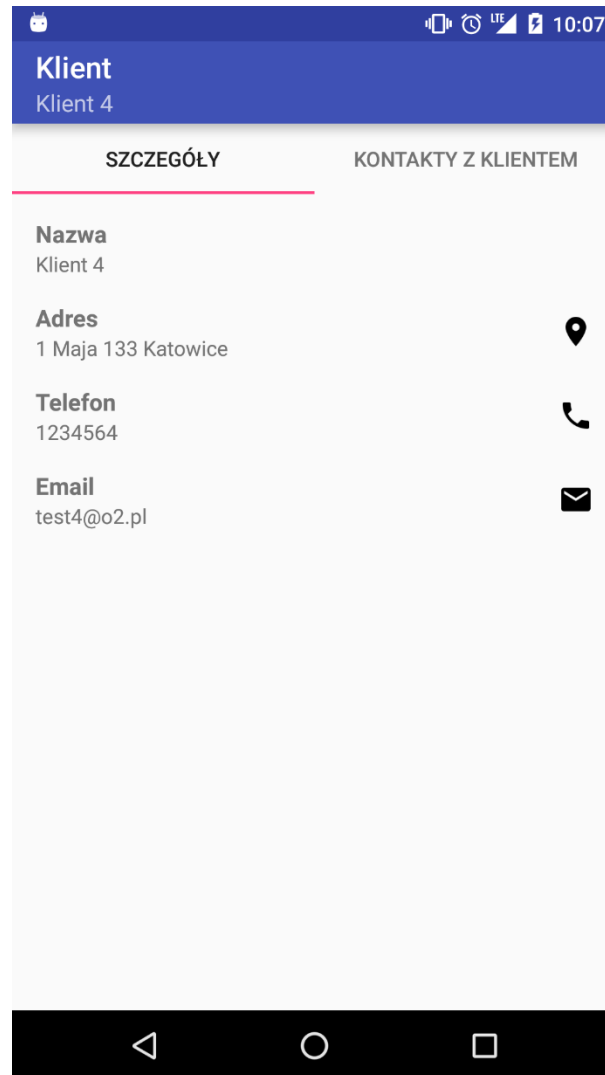
# Parcelable

```
public class Client implements Parcelable {  
  
    private String mName;  
    ...  
    protected Client(Parcel in) {  
        mName = in.readString();  
    }  
  
    @Override  
    public void writeToParcel(Parcel dest, int flags) {  
        dest.writeString(mName);  
    }  
    ...  
}
```

## ListView – klik na elemencie listy

```
@OnClick(R.id.list_view_clients)  
public void onListViewClientsItemClick(int position) {  
    Client client = mClientsListAdapter.getItem(position);  
  
    Intent intent = new Intent(this, ClientDetailsActivity.class);  
    intent.putExtra(IntentExtras.CLIENT, client);  
  
    startActivity(intent);  
}
```

# Nowe Activity – szczegóły klienta



# ViewPager - kontrolka

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

# Adapter dla ViewPager'a

```
public class ClientDetailsPagerAdapter extends FragmentStatePagerAdapter

@Override
public Fragment getItem(int position) {
    switch (position) {
        case 1:
            return ... (fragment);

        default:
            return null;
    }
}

@Override
public int getCount() {
    return 2;
}
```

Fragment

# FRAGMENT NIE ISTNIEJE BEZ ACTIVITY!

Do dynamicznej zmiany ekranu przy pozostaniu w obrębie jednego Activity.

Np. dla przewijanych kartek (page'y)

# Fragment

```
public class ClientDetailsFragment extends Fragment {  
    ...  
    @Override  
    public void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
  
        Bundle bundle = getArguments();  
        if (bundle != null) {  
            mClient = bundle.getParcelable(IntentExtras.CLIENT);  
        }  
    }  
    ...  
}
```



# Fragment

```
public class ClientDetailsFragment extends Fragment {  
    ...  
    @Override  
    public View onCreateView(LayoutInflater inflater, ViewGroup container,  
                             Bundle savedInstanceState) {  
  
        View fragmentView  
            = inflater.inflate(R.layout.fragment_client_details, container, false);  
  
        ButterKnife.bind(this, fragmentView);  
        initComponents();  
  
        return fragmentView;  
    }  
    ...  
}
```

# Fragment

```
public class ClientDetailsFragment extends Fragment {  
    ...  
    public static ClientDetailsFragment getInstance(Client client) {  
        ClientDetailsFragment fragment = new ClientDetailsFragment();  
        Bundle bundle = new Bundle();  
        bundle.putParcelable(IntentExtras.CLIENT, client);  
        fragment.setArguments(bundle);  
  
        return fragment;  
    }  
    ...  
}
```

Fragment – widok szczegółów klienta

Chwila ciszy...

# ViewPager - kontrolka

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

# Adapter dla ViewPager'a

```
public class ClientDetailsPagerAdapter extends FragmentStatePagerAdapter

public ClientDetailsPagerAdapter(FragmentManager fragmentManager, Client client) {
    super(fragmentManager);

    mClient = client;
}

@Override
public int getCount() {
    return PAGES;
}
```

# Adapter dla ViewPager'a

```
@Override
public Fragment getItem(int position) {
    switch (position) {
        case CLIENT_DETAILS_PAGE:
            return ClientDetailsFragment.getInstance(mClient);

        case CONTACTS_LIST_PAGE:
            return ContactsListFragment.getInstance(mClient);

        default:
            return null;
    }
}
```

# Podpięcie Adaptera do ViewPager'a

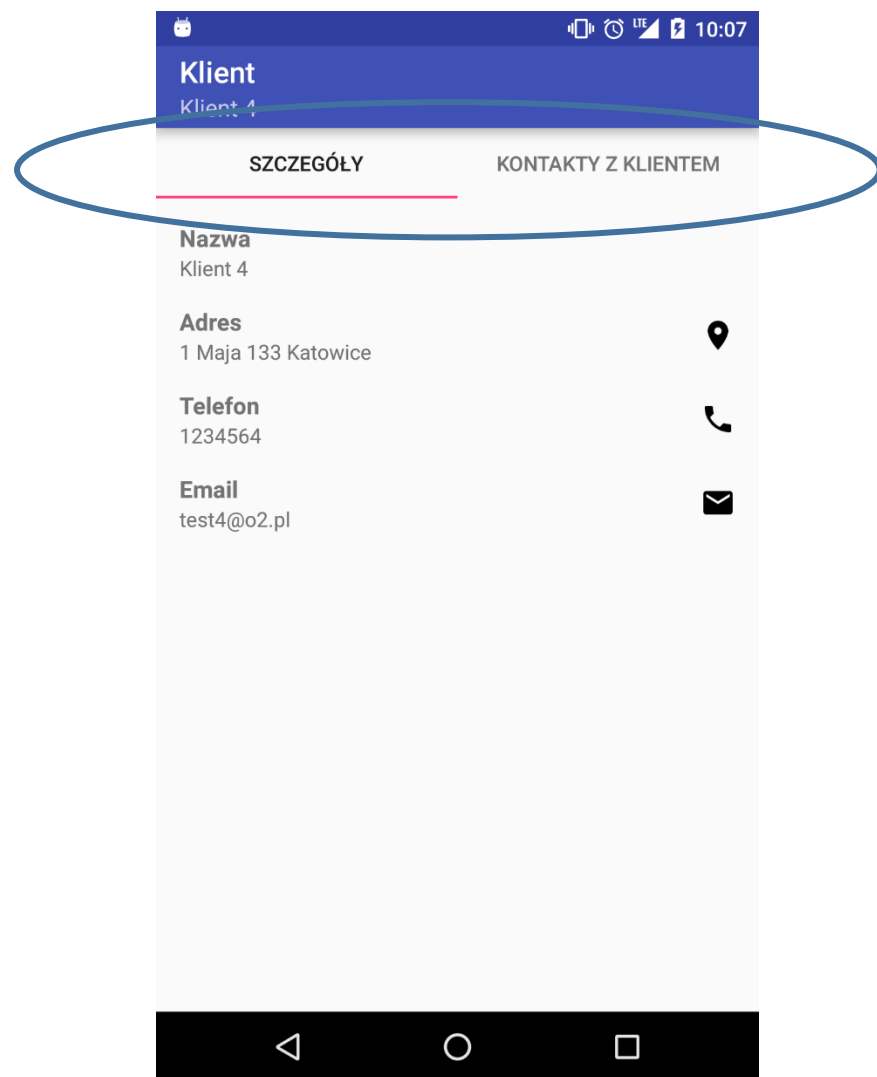
```
ClientDetailsPagerAdapter clientDetailsPagerAdapter  
    = new ClientDetailsPagerAdapter(getSupportFragmentManager(), mClient);  
  
mViewPagerClientDetails.setAdapter(clientDetailsPagerAdapter);
```

Podpięcie Adaptera do ViewPager'a

Chwila ciszy...



Ale skąd te tab'y?



# TabLayout - kontrolka

```
<android.support.design.widget.TabLayout  
    android:id="@+id/tab_layout_client_details"  
    android:layout_width="match_parent"  
    android:layout_height="wrap_content"/>
```

```
<android.support.v4.view.ViewPager
```

```
...
```

## TabLayout – podpięcie do ViewPager'a

```
ClientDetailsPagerAdapter clientDetailsPagerAdapter  
    = new ClientDetailsPagerAdapter(getSupportFragmentManager(), mClient);  
  
mViewPagerClientDetails.setAdapter(clientDetailsPagerAdapter);  
  
mTabLayoutClientDetails.setupWithViewPager(mViewPagerClientDetails);
```

# Skąd TabLayout zna tytuł danego Fragmentu?

```
public class ClientDetailsPagerAdapter extends FragmentStatePagerAdapter {  
    ...  
    @Override  
    public CharSequence getPageTitle(int position) {  
        switch (position) {  
            case CLIENT_DETAILS_PAGE:  
                return mContext.getString(R.string...);  
  
            case CONTACTS_LIST_PAGE:  
                return mContext.getString(R.string...);  
  
            default:  
                return null;  
        }  
    }  
    ...  
}
```

# Czym jest magiczny mContext?

```
private final Context mContext;  
private final Client mClient;  
  
public ClientDetailsPagerAdapter(FragmentManager fragmentManager, Context context,  
                                Client client) {  
    super(fragmentManager);  
  
    mContext = context;  
    mClient = client;  
}
```

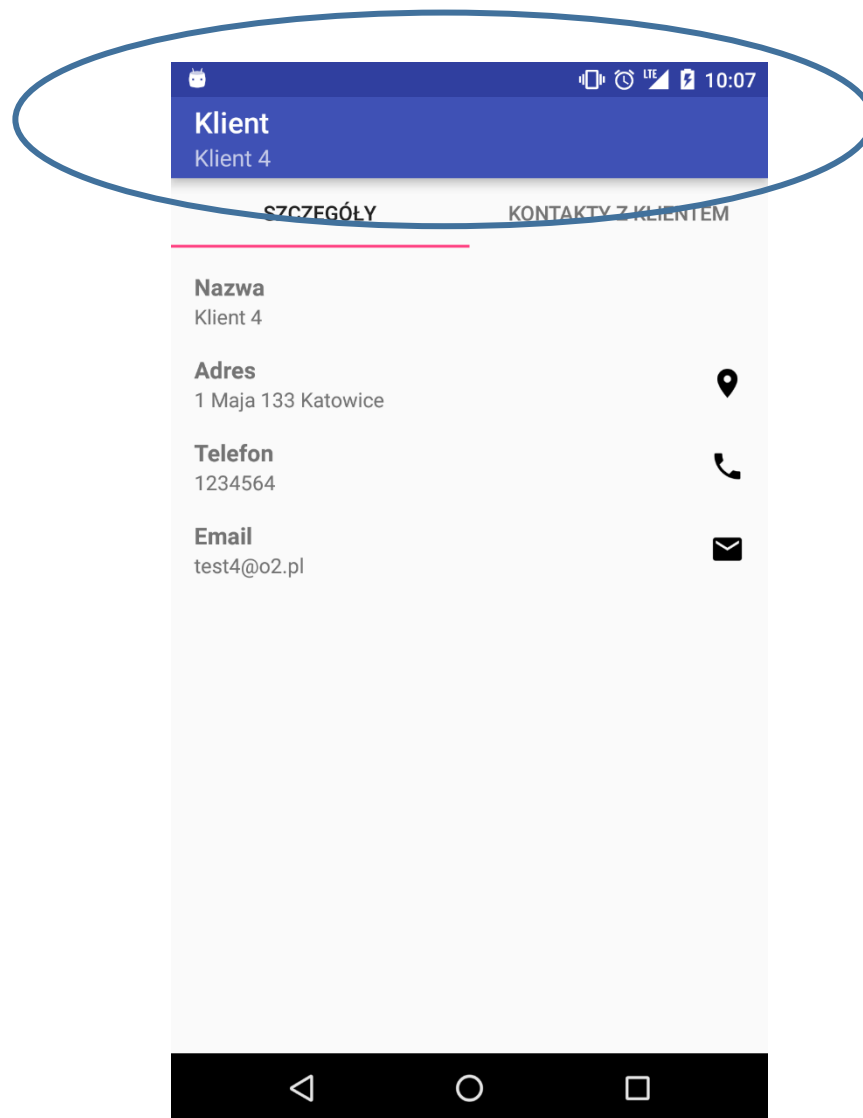
## TabLayout – podpięcie do ViewPager'a w wersji ostatecznej

```
ClientDetailsPagerAdapter clientDetailsPagerAdapter  
    = new ClientDetailsPagerAdapter(getSupportFragmentManager(), this, mClient);  
  
mViewPagerClientDetails.setAdapter(clientDetailsPagerAdapter);  
  
mTabLayoutClientDetails.setupWithViewPager(mViewPagerClientDetails);
```

TabLayout

Chwila ciszy...

# ActionBar





## ActionBar - subtitle

```
getSupportActionBar().setSubtitle („Super tytuł”);
```

# Wykorzystanie możliwości urządzenia - dzwonienie

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

```
String url = String.format("tel:%s", „123456789");  
Intent callIntent = new Intent("android.intent.action.CALL", Uri.parse(url));  
startActivity(callIntent);
```

# Wykorzystanie możliwości urządzenia - email

```
String url = String.format("mailto:%s", „test@test.pl");  
Intent emailIntent = new Intent("android.intent.action.VIEW", Uri.parse(url));  
startActivity(emailIntent);
```

# Wykorzystanie możliwości urządzenia - nawigacja

```
String uri = String.format("geo:0,0?q=%s", „1 Maja 133 Katowice");
Intent mapsIntent = new Intent(android.content.Intent.ACTION_VIEW, Uri.parse(uri));
mapsIntent.setClassName("com.google.android.apps.maps",
                        "com.google.android.maps.MapActivity");
startActivity(mapsIntent);
```

Wykorzystanie możliwości urządzenia

Chwila ciszy...

# Nowe Activity – dodawanie nowego klienta

The screenshot shows an Android application interface for adding a new client. The title bar is blue with the text "Dodaj klienta" and a "ZAPISZ" (Save) button. The form contains four input fields: "Nazwa" (Name) with the placeholder "Nazwa firmy", "Adres" (Address) with the placeholder "1 Maja 133", "Telefon" (Phone) with a red border, and "Email". At the bottom, there is a numeric keypad with digits 1-9, \*, #, 0, and a green arrow button.

Field	Value
Nazwa	Nazwa firmy
Adres	1 Maja 133
Telefon	
Email	

1 2 ABC 3 DEF -  
4 GHI 5 JKL 6 MNO .  
7 PQRS 8 TUV 9 WXYZ X  
\* # 0 + >

## Co nowego? – TextInputLayout

Nazwa

**Nazwa firmy**

---

Adres

**1 Maja 133**

---

Telefon

|

---

Email

---

# TextInputLayout - kontrolka

```
<android.support.design.widget.TextInputLayout
    android:layout_width="match_parent"
    android:layout_height="wrap_content">

    <EditText
        android:id="@+id/edit_text_client_name"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:hint="@string/activity_add_client_name"
        android:inputType="text" />
</android.support.design.widget.TextInputLayout>
```



## Co nowego? - Menu

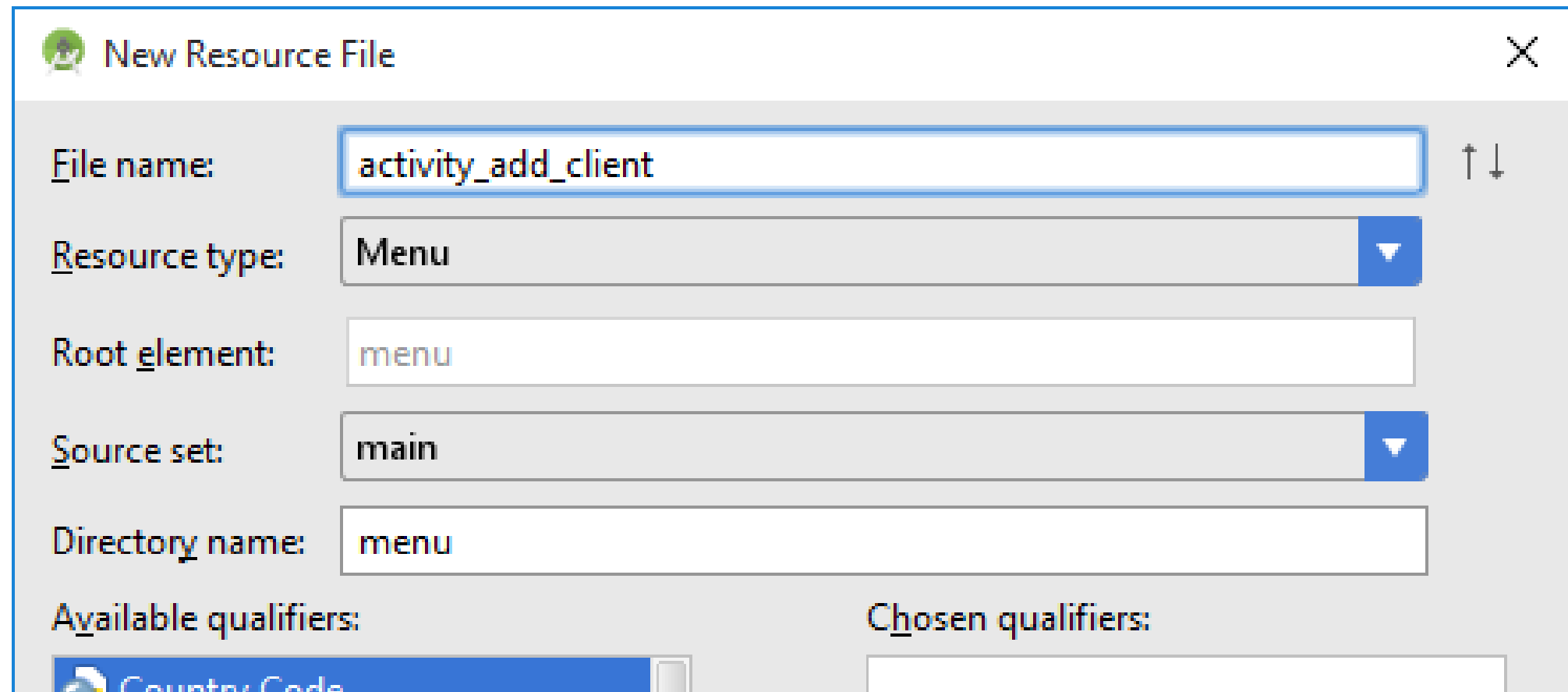



The screenshot shows an Android application interface. At the top is a dark blue header bar with a white Android robot icon on the left and status icons (signal, alarm, LTE, battery) and the time '12:15' on the right. Below the header is a white area with a blue title bar containing the text 'Dodaj klienta' in white and a 'ZAPISZ' button in blue. Below the title bar is a form with a light gray background. The first label is 'Nazwa' in gray, followed by a text input field containing 'Nazwa firmy' in bold black text.

Nazwa

**Nazwa firmy**

## Dodajemy menu



 New Resource File ✕

File name:  ↑ ↓

Resource type:  ▼

Root element:

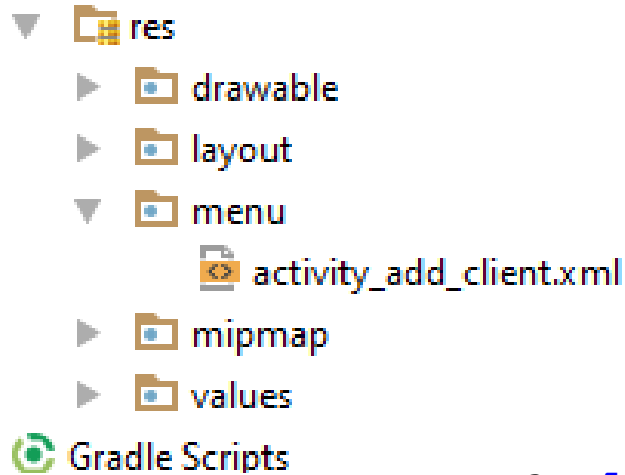
Source set:  ▼

Directory name:

Available qualifiers:

Chosen qualifiers:

# Dodajemy menu



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

    <item
        android:id="@+id/action_save"
        android:title="@string/activity_add_client_menu_save"
        app:showAsAction="always"/>

</menu>
```

# Dodajemy menu

```
@Override
public boolean onCreateOptionsMenu(Menu menu) {
    getMenuInflater().inflate(R.menu.activity_add_client, menu);
    return true;
}
```

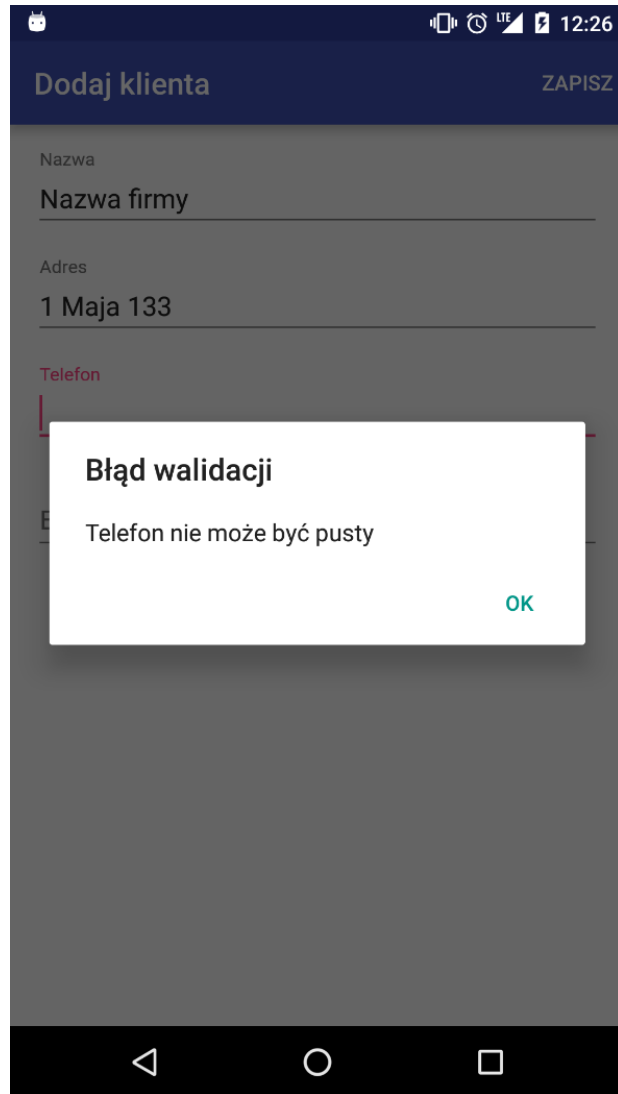
```
@Override
public boolean onOptionsItemSelected(MenuItem item) {
    switch (item.getItemId()) {
        case R.id.action_save:
            doSaveAction();
            break;
    }

    return true;
}
```

Trochę nowości

Chwila ciszy...


# Dialog



# Tworzymy własny dialog

```
AlertDialog.Builder builder = new AlertDialog.Builder(this);  
builder.setTitle(„Tytuł”);  
builder.setMessage(„Wiadomość”);  
builder.setPositiveButton(„OK”, null);  
builder.show();
```

```
new View.OnClickListener() {  
    new DialogInterface.OnClickListener() {  
        @Override  
        public void onClick(DialogInterface dialog, int which) {  
            ...  
        }  
    }  
});
```



## Obsługa dodawania nowego klienta

Po wybraniu opcji zapisz:

- Pobranie wprowadzonych wartości z kontrolek EditText
- Walidacja (czy nie puste)
- Jeżeli puste -> dialog
- Jeżeli wszystko ok -> stworzony obiekt zwracamy do ClientsListActivity



# SQLite - baza danych

- **INTEGER**
- **REAL** (float)
- **TEXT**
- **BLOB**

# Obsługa bazy danych w aplikacjach

```
public class DBHelper extends SQLiteOpenHelper {  
  
    private static final String NAME = "firstapp.db";  
    private static final int VERSION = 1;  
  
    private Context mContext;  
  
    public DBHelper(Context context) {  
        super(context, NAME, null, VERSION);  
        mContext = context;  
    }  
    ...  
}
```

# Obsługa bazy danych w aplikacji

```
public class DBHelper extends SQLiteOpenHelper {  
    ...  
    public DBHelper(Context context) {  
        ...  
    }  
  
    @Override  
    public void onCreate(SQLiteDatabase db) {  
        db.execSQL(getSql(R.string.table_test));  
    }  
  
    @Override  
    public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {  
  
    }  
    ...  
}
```

## Metoda getSql(int)

```
public String getSql(int sqlId, Object... args) {  
    String sqlFormat = getSql(sqlId);  
    return String.format(sqlFormat, args);  
}
```

```
public String getSql(int sqlId) {  
    return mContext.getString(sqlId);  
}
```

```
String.format("SELECT * FROM %s", "Test");
```



Mam swój DBHelper i co dalej?

- getReadableDatabase();
- getWritableDatabase();

```
new DBHelper(this).getReadableDatabase().execSql("SELECT * FROM Test");
```

# Cursor

```
Cursor cursor = new DBHelper(this)
                .getReadableDatabase()
                .rawQuery("SELECT * FROM Test", null);

while (cursor.moveToNext()) {
    ...
}
```

# Cursor

```
Cursor cursor = new DBHelper(this)
                    .getReadableDatabase()
                    .rawQuery("SELECT * FROM Test", null);
```

```
while (cursor.moveToNext()) {
    ...
}
```

```
while (cursor.moveToNext()) {
    int columnIndex = cursor.getColumnIndex("NAZWA_KOLUMNY");
    String rowColumnValue = cursor.getString(columnIndex);
}
```

# DAO

## **Data Access Object**

Komponent dostarczający jednolity interfejs do komunikacji między aplikacją a źródłem danych (np. bazą danych czy plikiem).

Źródło: [wikipedia.org](https://pl.wikipedia.org/wiki/Data_Access_Object)



Chwila na zdobycie niezbędnych klas

BaseDao - [pastebin.com/X5Qm9EzR](https://pastebin.com/X5Qm9EzR)

IDatabaseObject - [pastebin.com/PsiX4aUj](https://pastebin.com/PsiX4aUj)

## Problem z BaseDao?

```
SQLiteDatabase database = ...getDbHelper().getReadableDatabase();
```

```
SQLiteDatabase database = FirstAppApplication  
    .getInstance()  
    .getDbHelper()  
    .getReadableDatabase();
```

# Czym jest Application

```
public class FirstAppApplication extends Application {  
  
    private static FirstAppApplication sInstance;  
  
    @Override  
    public void onCreate() {  
        super.onCreate();  
        sInstance = this;  
    }  
  
    public static FirstAppApplication getInstance() {  
        return sInstance;  
    }  
}
```

# Skąd obiekt DBHelper w Application?

```
public class FirstAppApplication extends Application {  
  
    private static FirstAppApplication sInstance;  
  
    private DBHelper mDbHelper;  
  
    @Override  
    public void onCreate() {  
        ...  
    }  
  
    public DBHelper getDbHelper() {  
        if (mDbHelper == null) {  
            mDbHelper = new DBHelper(this);  
        }  
        return mDbHelper;  
    }  
}
```

# Niezbędna konfiguracja własnej implementacji Application

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

    ...

    <application
        android:name=".FirstAppApplication"

    ...
```

# Jak korzystać z BaseDao

```
public class ClientDao extends BaseDao<Client> {  
  
    private static final String TABLE_NAME = "Client";  
  
    private static final String COLUMN_ID = "id";  
    private static final String COLUMN_NAME = „name”;  
  
    public ClientDao() {  
        super(TABLE_NAME);  
    }  
}
```

# Jak korzystać z BaseDao

```
public class ClientDao extends BaseDao<Client> {  
    ...  
    @Override  
    protected Client getObjectFromCursor(Cursor cursor) {  
        Client client = new Client();  
        client.setId(getLong(cursor, COLUMN_ID));  
        client.setName(getString(cursor, COLUMN_NAME));  
        return client;  
    }  
  
    @Override  
    protected ContentValues getObjectContentValues(Client object) {  
        ContentValues contentValues = new ContentValues();  
        contentValues.put(COLUMN_NAME, object.getName());  
        return contentValues;  
    }  
}
```

## BaseDao – operacje na bazie (CRUD)

```
new ClientDao().insertObject(client);
```

```
new ClientDao().updateObject(client);
```

```
new ClientDao().deleteObject(client);
```



## BaseDao – operacje na bazie (CRUD)

```
public long insertObject(T object) {  
    SQLiteDatabase database  
        = FirstAppApplication.getInstance().getDbHelper().getWritableDatabase();  
  
    ContentValues values = getObjectContentValues(object);  
  
    long objectId = database.insert(mTableName, null, values);  
    object.setId(objectId);  
  
    return objectId;  
}
```

## BaseDao – operacje na bazie (CRUD)

```
public boolean updateObject(T object) {  
    SQLiteDatabase database  
        = FirstAppApplication.getInstance().getDbHelper().getWritableDatabase();  
  
    ContentValues values = getObjectContentValues(object);  
  
    String whereClause  
        = String.format(Locale.getDefault(), "id = %d", object.getId());  
  
    return database.update(mTableName, values, whereClause, null) > 0;  
}
```

## BaseDao – operacje na bazie (CRUD)

```
public boolean deleteObject(T object) {  
    SQLiteDatabase database  
        = FirstAppApplication.getInstance().getDbHelper().getWritableDatabase();  
  
    String whereClause  
        = String.format(Locale.getDefault(), "id = %d", object.getId());  
  
    return database.delete(mTableName, whereClause, null) > 0;  
}
```

Skąd metoda getId() w obiekcie typu T?

```
public abstract class BaseDao<T extends IDatabaseObject>
```

```
public interface IDatabaseObject {
```

```
    Long getId();
```

```
    void setId(Long id);
```

```
    Long getExternalId();
```

```
    void setExternalId(Long id);
```

```
}
```

```
public class Client implements IDatabaseObject
```

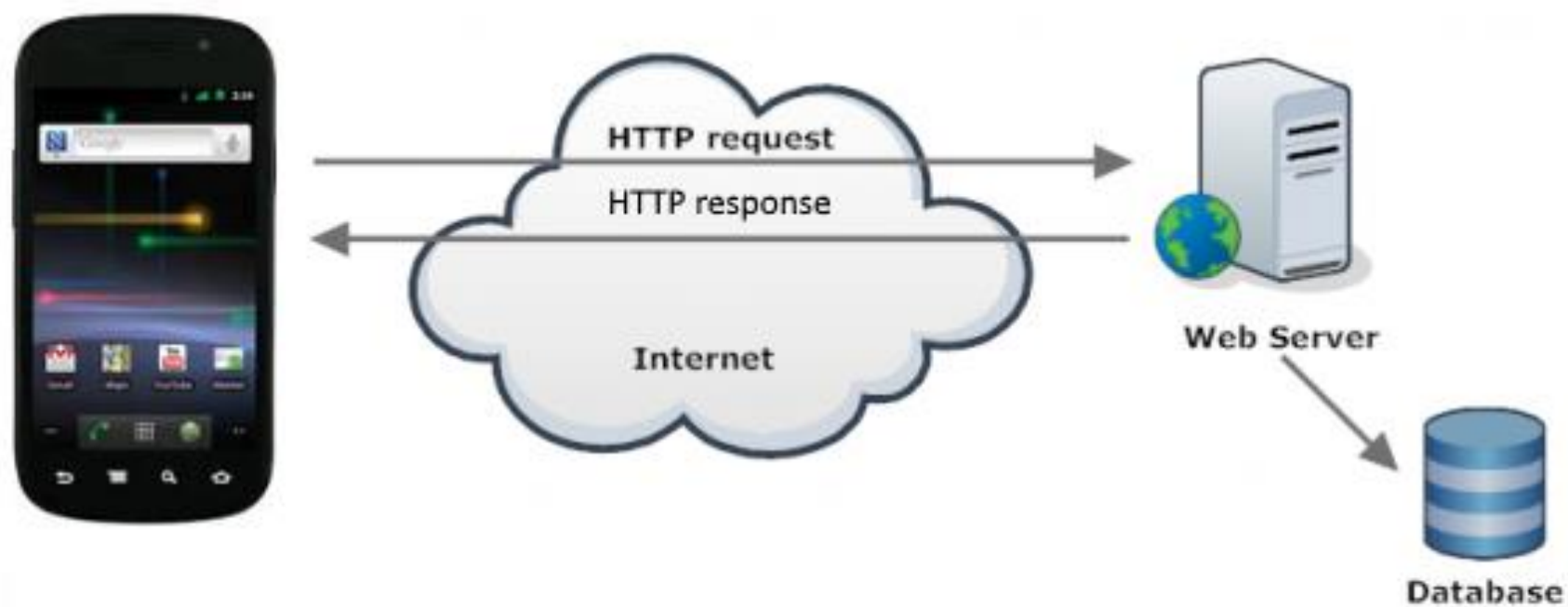
## BaseDao – pozostał ostatni CRUD (Read)

```
public List<Client> getClientsList() {  
    String sql = [...].getDbHelper().getSql(R.string.sql_dao_client_get_clients);  
    Cursor cursor = openRawQueryCursor(sql);  
    try {  
        List<Client> clientsList = new ArrayList<>();  
  
        while (cursor.moveToNext()) {  
            Client client = getObjectFromCursor(cursor);  
            clientsList.add(client);  
        }  
  
        return clientsList;  
    } catch (Exception e) {  
        return new ArrayList<>();  
    } finally {  
        closeCursor(cursor);  
    }  
}
```

# Programowanie na platformę Android

Część 3

# Połączenie z usługami zewnętrznymi (typu REST)



# REST

Styl architektury oprogramowania oparty o specyfikację protokołu HTTP.

Bezstanowy.

Umożliwia dostęp do zasobów.

Źródło: [wikipedia.org](https://pl.wikipedia.org/wiki/REST)



# REST

GET – każdorazowe wyświetlenie strony w przeglądarce

POST – wysyłanie formularza ze strony (czasem realizowane przez GET)

PUT, DELETE

REST

GET <http://mobile1.osoz.pl:8080/crm/rest/version>

REST

GET <http://mobile1.osoz.pl:8080/crm/rest/version>

```
{  
  "version": "2016.04.22.1"  
}
```

Status 200

# HTTP – najpopularniejsze kody statusów

Success:

200 OK

201 CREATED

204 NO CONTENT

# HTTP – najpopularniejsze kody statusów

Client error:

400 BAD REQUEST

401 UNAUTHORIZED

403 FORBIDDEN

404 NOT FOUND

408 REQUEST TIMEOUT

HTTP – najpopularniejsze kody statusów

Server error:

500 INTERNAL SERVER ERROR

503 SERVICE UNAVAILABLE

# JSON

JavaScript Object Notation – lekki format wymiany danych.

```
{  
    "klucz": "wartość",  
}
```

# JSON – obiekty

```
{  
  "obiekt": {  
    "klucz": "wartość",  
    "klucz": "wartość"  
  },  
  "obiekt": {  
    "klucz": "wartość",  
    "klucz": "wartość"  
  },  
}
```



## JSON – tablice

```
{  
  "tablica": [  
    { "klucz": "wartość" },  
    {  
      "klucz": "wartość",  
      "klucz": "wartość"  
    }  
  ]  
}
```

# REST

POST <http://mobile1.osoz.pl:8080/crm/rest/login>

Request body:

```
{  
  "username": "użytkownik",  
  "password": "haslo1"  
}
```

Możliwe statusy odpowiedzi:

- 204 NO CONTENT
- 401 UNAUTHORIZED

# Obsługa usług REST w urządzeniu mobilnym

Przykładowe dostępne biblioteki:

- Volley
- HttpOk
- **Retrofit**

... i wiele innych

# Retrofit - gradle

```
compile 'com.squareup.retrofit2:retrofit:2.0.2'
```

```
compile 'com.squareup.retrofit2:converter-gson:2.0.2'
```

Jak wygląda nasza przykładowa usługa

Adres: <http://mobile1.osoz.pl:8080/crm/rest/>

GET /version

GET /clients

GET /client/(external\_id)

Jak wygląda nasza przykładowa usługa

Adres: <http://mobile1.osoz.pl:8080/crm/rest/>

POST /login

```
{  
    "username": "uzytkownik",  
    "password": "haslo1"  
}
```

Jak wygląda nasza przykładowa usługa

Adres: <http://mobile1.osoz.pl:8080/crm/rest/>

POST /client

```
{  
    "name": "...",  
    "address": "...",  
    "phone": "...",  
    "email": "..."  
}
```

## Retrofit – tworzymy niezbędne klasy

```
@GET("client/{id}")  
Call<Client> getClient(@Path("id") long id);
```



# Retrofit – tworzymy niezbędne klasy

```
@POST ("login")
```

```
Call<Void> login(@Body UserCredentials userCredentials);
```

```
public class UserCredentials {
```

```
    public static final String USERNAME = "username";  
    public static final String PASSWORD = "password";
```

```
    private String mUsername;  
    private String mPassword;
```

```
    public UserCredentials() {  
    }
```

```
    public UserCredentials(String username, String password) {  
        mUsername = username;  
        mPassword = password;  
    }
```

```
    // GETTER, SETTER
```

```
}
```

Jak wygląda nasza przykładowa usługa

Nagłówki żądania:

username: uzytkownik

password: haslo1

## Retrofit – tworzymy niezbędne klasy

```
@GET("client/{id}")  
Call<Client> getClient(@Header("username") String username,  
                       @Header("password") String password,  
                       @Path("id") long id);
```

Retrofit – tworzymy niezbędne klasy

Chwila ciszy...

# Gson

```
public class Client {  
  
    private Long mId;  
    @SerializedName("id")  
    private Long mExternalId;  
    @SerializedName("name")  
    private String mName;  
  
    public Client() {  
    }  
}
```

# Retrofit – tworzymy niezbędne klasy

```
public interface CrmService {

    @POST(CrmServiceEndpoint.LOGIN)
    Call<Void> login(@Body UserCredentials userCredentials);

    @GET(CrmServiceEndpoint.CLIENT_LIST)
    Call<List<Client>> clientList(@Header(UserCredentials.USERNAME) String username,
                                @Header(UserCredentials.PASSWORD) String password);

    @POST(CrmServiceEndpoint.ADD_CLIENT)
    Call<Client> addClient(@Header(UserCredentials.USERNAME) String username,
                           @Header(UserCredentials.PASSWORD) String password,
                           @Body Client client);

    @GET(CrmServiceEndpoint.GET_CLIENT)
    Call<Client> getClient(@Header(UserCredentials.USERNAME) String username,
                           @Header(UserCredentials.PASSWORD) String password,
                           @Path("id") long id);

}
```

# Retrofit – tworzymy niezbędne klasy

```
public class CrmServiceEndpoint {  
  
    public static final String LOGIN = "login";  
    public static final String CLIENT_LIST = "clients";  
    public static final String ADD_CLIENT = "client";  
    public static final String GET_CLIENT = "client/{id}";  
}
```

## Retrofit – tworzymy niezbędne klasy

```
Retrofit retrofit = new Retrofit.Builder()  
    .baseUrl("http://mobile1.osoz.pl:8080/crm/rest/")  
    .addConverterFactory(GsonConverterFactory.create())  
    .build();
```

```
CrmService crmService = retrofit.create(CrmService.class);
```



# Retrofit – tworzymy niezbędne klasy

```
public class ServiceManager {  
  
    private static final String CRM_SERVICE = "http://mobile1.osoz.pl:8080/crm/rest/";  
  
    public static CrmService getCrmService() {  
        Retrofit retrofit = getRetrofitInstance(CRM_SERVICE);  
        return retrofit.create(CrmService.class);  
    }  
  
    private static Retrofit getRetrofitInstance(String url) {  
        Retrofit retrofit = new Retrofit.Builder()  
            .baseUrl(url)  
            .addConverterFactory(GsonConverterFactory.create())  
            .build();  
        return retrofit;  
    }  
}
```

# Retrofit – użycie

```
@OnClick(R.id.button_log_in)
public void onButtonLogInClick() {
    String username = mEditTextUsername.getText().toString();
    String password = mEditTextPassword.getText().toString();
    UserCredentials userCredentials = new UserCredentials(username, password);

    CrmService crmService = ServiceManager.getCrmService();
    Call<Void> loginCall = crmService.login(userCredentials);
    loginCall.enqueue(getLoginCallback(userCredentials));
}
```

# Retrofit – użycie

```
private Callback<Void> getLoginCallback() {  
    return new Callback<Void>() {  
        @Override  
        public void onResponse(Call<Void> call, Response<Void> response) {  
            if (response.isSuccessful()) {  
                // Status 2XX  
            } else {  
                // Status 4XX lub 5XX (prawdopodobnie 401 UNAUTHORIZED)  
            }  
        }  
    }  
  
    @Override  
    public void onFailure(Call<Void> call, Throwable t) {  
        // Connection error  
    }  
};  
}
```

Retrofit – użycie

Chwila ciszy...

# Retrofit – użycie

```
private Callback<Void> getDoLoginCallback() {  
    return new Callback<Void>() {  
        @Override  
        public void onResponse(Call<Void> call, Response<Void> response) {  
            if (response.isSuccessful()) {  
                Intent menuIntent = new Intent(MainActivity.this, MenuActivity.class);  
                startActivity(menuIntent);  
                finish();  
            } else {  
                Toast.makeText(MainActivity.this, R.string.[message], Toast.LENGTH_LONG).show();  
            }  
        }  
    }  
  
    @Override  
    public void onFailure(Call<Void> call, Throwable t) {  
        Toast.makeText(MainActivity.this, R.string.[message], Toast.LENGTH_LONG).show();  
    }  
};  
}
```

# Retrofit – użycie

```
public class FirstAppApplication extends Application {  
  
    //...  
    private UserCredentials mUserCredentials;  
  
    //...  
    public UserCredentials getUserCredentials() {  
        return mUserCredentials;  
    }  
  
    public void setUserCredentials(UserCredentials userCredentials) {  
        mUserCredentials = userCredentials;  
    }  
}
```

Retrofit – użycie

Chwila ciszy – pobierzcie listę klientów

# Retrofit – użycie

```
UserCredentials userCredentials
    = FirstAppApplication.getInstance().getUserCredentials();

CrmService crmService = ServiceManager.getCrmService();

Call<List<Client>> clientListCall
    = crmService.clientList(userCredentials.getUsername(), userCredentials.getPassword());

clientListCall.enqueue(new ClientListCallback(this));
showProgressDialog();
```



# Retrofit – użycie

```
public class ClientListCallback implements Callback<List<Client>> {

    private final Context mContext;
    private final ClientDao mClientDao;

    public ClientListCallback(Context context) {
        mContext = context;
        mClientDao = new ClientDao();
    }

    @Override
    public void onResponse(Call<List<Client>> call, Response<List<Client>> response) {
        if (response.isSuccessful()) {
            List<Client> downloadedClientList = response.body();
            saveClientList(downloadedClientList);

            EventBus.getDefault().post(new ClientListDownloadedEvent());
        } else {
            Toast.makeText(mContext, R.string.[message], Toast.LENGTH_LONG).show();
        }
    }

    @Override
    public void onFailure(Call<List<Client>> call, Throwable t) {
        Toast.makeText(mContext, R.string.[message], Toast.LENGTH_LONG).show();
    }
}
```

# Retrofit – użycie

```
public class ClientListCallback implements Callback<List<Client>> {  
  
    //...  
  
    private void saveClientList(List<Client> downloadedClientList) {  
        if (downloadedClientList == null || downloadedClientList.size() == 0) {  
            return;  
        }  
  
        for (Client client : downloadedClientList) {  
            mClientDao.insertObject(client);  
        }  
    }  
  
    //...  
}
```

# EventBus – kto to? co to?

```
EventBus.getDefault().post(new SomeEvent());
```

Biblioteka. Umożliwia komunikację między różnymi komponentami aplikacji – Activity, Fragment, Thread, Service.

Gradle:

```
compile 'org.greenrobot:eventbus:3.0.0'
```

# EventBus – jak używać na przykładzie Activity?

```
@Override
protected void onResume() {
    EventBus.getDefault().register(this);
    super.onResume();
}
```

```
@Override
protected void onPause() {
    super.onPause();
    EventBus.getDefault().unregister(this);
}
```

```
@Subscribe
public void onSomeEvent(SomeEvent someEvent) {
    // action...
}
```

## EventBus – jak wywołać akcję?

```
EventBus.getDefault().post(new SomeEvent());
```

# EventBus – co z klasą SomeEvent?

```
public class SomeEvent {  
  
}
```

```
public class SomeEvent {  
  
    private Object someObject;  
  
    public SomeEvent(Object someObject) {  
        this.someObject = someObject;  
    }  
  
    public Object getSomeObject() {  
        return someObject;  
    }  
  
    public void setSomeObject(Object someObject) {  
        this.someObject = someObject;  
    }  
}
```

# ShowProgressDialog()?

```
private ProgressDialog mProgressDialog;

private void showProgressDialog() {
    if (mProgressDialog == null) {
        mProgressDialog = new ProgressDialog(this);
        mProgressDialog.show();
    }
}

private void hideProgressDialog() {
    if (mProgressDialog != null) {
        mProgressDialog.dismiss();
        mProgressDialog = null;
    }
}
```

Title of progress dialog.



Loading.....