**Android – Szkolenie Podstawowe**

**Future Processing**

**FP Mobile Division**

***„Projekt Jaszczur”* 2015**

**Android – Szkolenie Podstawowe**

[Zad 1 - Utworzenie nowego projektu 3](#_Toc413450302)

[Cykl życia aktywności 3](#_Toc413450303)

[Struktura projektu 5](#_Toc413450304)

[Gradle 6](#_Toc413450305)

[AndroidManifest 7](#_Toc413450306)

[Zad 2 – Tworzymy pierwszą aktywność 8](#_Toc413450307)

[Zad 3 – Odkrywamy magię cyklu życia 10](#_Toc413450308)

[Zad 4 - Tworzymy drugą aktywność z listą 11](#_Toc413450309)

[Zad 5 – Zapisanie wybranej waluty do pamięci trwałej 18](#_Toc413450310)

[Zad 6 – Odczyt z pamięci 19](#_Toc413450311)

[Zad 7 – Asynchroniczne ładowanie bitmap 20](#_Toc413450312)

[Zad 8 – Pobieranie walut z internetu i parsowanie json’a 21](#_Toc413450313)

[Zad 9 – Dodawanie menu kontekstowego oraz akcji do ActionBar’a 23](#_Toc413450314)

[Zad 10 – Dialog do edycji aktualnej waluty 24](#_Toc413450315)

[Zad 11 – Przeliczanie walut 24](#_Toc413450316)

[Przydatne linki 25](#_Toc413450317)

### Zad 1 - Utworzenie nowego projektu

### Cykl życia aktywności



**onCreate()** – wywoływana gdy aktywność została utworzona po raz pierwszy. To jest mniejsce gdzie należy wykonywać podstawowe czynności, jak tworzenie widoków, bindowanie danych itp.

**onRestart()** – wywoływana gdy aktywność została zatrzymana i uruchomiona ponownie

**onStart()** – wywoływana, gdy aktywność staje się widoczna dla użytkownika.

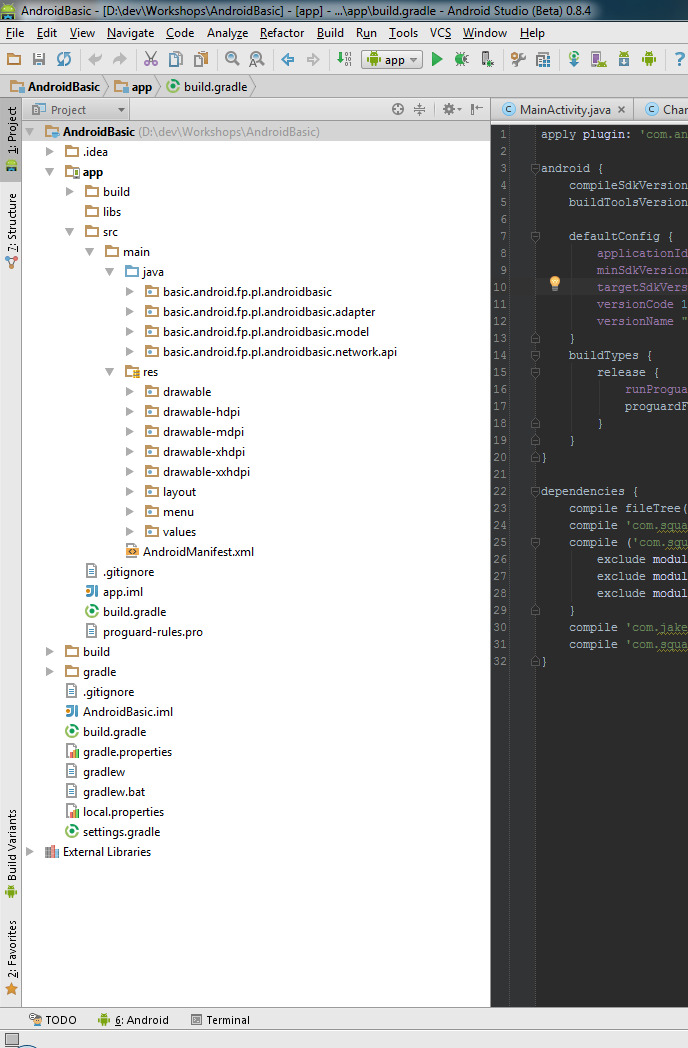
**onResume()** – wywoływana gdy aktywność rozpoczyna interakcję z użytkownikiem. W tym omencie aktywność jest na szczycie sotsu aktywności.

**onPause()** - wywoływana, gdy system rozpoczyna przywracanie poprzedniej aktywności. Jest zazwyczaj używana do zapisania trwałych danych, zatrzymania animacji i innych rzeczy, które mogą obciążać procesor.

**onStop()** – wywoływana gdy aktywność przestaje być widoczna dla użytkownik. Dzieje się tak zazwyczaj gdy otwierana jest nowa aktywność lub aktualna zostaje zniszczona.

**onDestroy()** – Ostatnia metoda wywoływana przed zniszczeniem aktywności.

### Struktura projektu



### Gradle

**Plik gradle.build w module:**

*apply plugin: 'com.android.application'*

*android {*

*compileSdkVersion 21*

*buildToolsVersion "21.1.2"*

*defaultConfig {*

*applicationId "basic.android.fp.pl.androidbasic"*

*minSdkVersion 15*

*targetSdkVersion 21*

*versionCode 1*

*versionName "1.0"*

*}*

*buildTypes {*

*release {*

*minifyEnabled false*

*proguardFiles getDefaultProguardFile('proguard-android.txt'), 'proguard-rules.pro'*

*}*

*}*

*}*

*dependencies {*

*compile fileTree(dir: 'libs', include: ['\*.jar'])*

*}*

### AndroidManifest

*<?xml version="1.0" encoding="utf-8"?>*

*<manifest xmlns:android="http://schemas.android.com/apk/res/android"*

*package="basic.android.fp.pl.androidbasic">*

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

*<application*

*android:allowBackup="true"*

*android:icon="@drawable/ic\_launcher"*

*android:label="@string/app\_name"*

*android:theme="@style/AppTheme">*

*<activity*

*android:name=".MainActivity"*

*android:label="@string/app\_name">*

*<intent-filter>*

*<action android:name="android.intent.action.MAIN" />*

*<category android:name="android.intent.category.LAUNCHER" />*

*</intent-filter>*

*</activity>*

*<activity android:name=".ListCurrenciesActivity" />*

*</application>*

*</manifest>*

### Zad 2 – Tworzymy pierwszą aktywność

**Pamiętaj:** Każda aktywność musi być zadeklarowana w manifeście!!

**MainActivity:**

import android.os.Bundle;

import android.widget.Button;

public class MainActivity extends Activity {

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_main);

Button changeCurrencyButton = (Button) findViewById(R.id.listCurrenciesButton);

Button changeCurrencyDialogButton = (Button) findViewById(R.id.rateChangeButton); }

}

**Activity\_main.xml:**

<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"

android:layout\_width="match\_parent"

android:layout\_height="match\_parent"

android:orientation="vertical"

android:gravity="center">

<Button

android:id="@+id/listCurrenciesButton"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:text="@string/open\_exchange\_rates\_list" />

<Button

android:id="@+id/rateChangeButton"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:text="@string/change\_rate\_manually" />

</LinearLayout>

**Do pliku strings.xml należy dodać dwa teksty:**

<string name="change\_rate\_manually">Zmien kurs ręcznie</string>

<string name="open\_exchange\_rates\_list">Otwórz listę kursów</string>

### Zad 3 – Odkrywamy magię cyklu życia

public class MyActivity extends Activity {

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

Log.i("LIFECYCLE", "-OnCreate");

}

@Override

protected void onStart() {

super.onStart();

Log.i("LIFECYCLE", "--OnStart");

}

@Override

protected void onResume() {

super.onResume();

Log.i("LIFECYCLE", "---OnResume");

}

@Override

protected void onPause() {

super.onPause();

Log.i("LIFECYCLE", "---OnPause");

}

@Override

protected void onStop() {

super.onStop();

Log.i("LIFECYCLE", "--OnStop");

}

@Override

protected void onDestroy() {

super.onDestroy();

Log.i("LIFECYCLE", "-OnDestroy");

}

@Override

protected void onRestart() {

super.onRestart();

Log.i("LIFECYCLE", "=====>OnRestart");

}

}

### Zad 4 - Tworzymy drugą aktywność z listą

**Activity\_change\_currency.xml:**

<ListView xmlns:android="http://schemas.android.com/apk/res/android"

android:id="@+id/list"

android:layout\_width="match\_parent"

android:layout\_height="match\_parent"/>

**ListCurrenciesActivity:**

public class ListCurrenciesActivity extends Activity {

@InjectView(R.id.list)

protected ListView currencyListView;

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_change\_currency);

ButterKnife.inject(this);

}

}

**Dodajemy do build.gradle w tagu dependencies:**

compile 'com.jakewharton:butterknife:+'

**Dodajemy do MainActivity otwieranie nowej aktywności po naciśnięciu buttona.**

**W metodzie onCreate() dodajemy:**

button.setOnClickListener(new View.OnClickListener() {

@Override

public void onClick(View v) {

Intent i = new Intent(getApplicationContext(), ListCurrenciesActivity.class);

startActivity(i);

}

});

**Tworzymy layout dla pojedyńczego elementu listy:**

**item\_currency\_list.xml:**

<?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="horizontal">

<ImageView

android:id="@+id/flag"

android:layout\_width="36dp"

android:layout\_height="36dp"/>

<LinearLayout

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:orientation="vertical">

<TextView

android:id="@+id/currencyName"

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:paddingLeft="6dp"

android:text="Name"

android:textAppearance="?android:attr/textAppearanceMedium"/>

<TextView

android:id="@+id/averageRate"

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:paddingLeft="6dp"

android:text="1 EUR 4 PLN"

android:textAppearance="?android:attr/textAppearanceSmall"/>

</LinearLayout>

</LinearLayout>

**JSON otrzymywany z serwisu:**

{

"date" : "2015-03-06",

"base" : "PLN",

"rates" : [

{"currency" : "USD", "rate" : 0.2662},

{"currency" : "IDR", "rate" : 3444.17}

]

}

**Na podstawie json’a tworzymy klasy modelu:**

RatesList**:**

public class RatesList {

private String date;

private Currency base;

private List<ExchangeRate> rates;

public RatesList(String date, Currency base, List<ExchangeRate> exchangeRates) {

this.date = date;

this.base = base;

this.rates = exchangeRates;

}

public String getDate() {

return date;

}

public void setDate(String date) {

this.date = date;

}

public Currency getBase() {

return base;

}

public void setBase(Currency base) {

this.base = base;

}

public List<ExchangeRate> getExchangeRates() {

return rates;

}

public void setExchangeRates(List<ExchangeRate> exchangeRates) {

this.rates = exchangeRates;

}

}

**ExchangeRate:**

public class ExchangeRate implements Serializable {

private final Currency currency;

private Double rate;

public ExchangeRate(Currency currency, Double rate) {

this.currency = currency;

this.rate = rate;

}

public Currency getCurrency() {

return currency;

}

public Double getRate() {

return rate;

}

public void setRate(Double rate) {

this.rate = rate;

}

}

**Tworzymy adapter dla listy:**

**CurrencyListAdapter:**

public class CurrencyListAdapter extends BaseAdapter {

private final Context context;

private final List<ExchangeRate> exchangeRates;

private final LayoutInflater inflater;

public CurrencyListAdapter(Context context, RatesList currencyTable) {

this.context = context;

this.currencyTable = currencyTable;

inflater = (LayoutInflater) context.getSystemService(Context.LAYOUT\_INFLATER\_SERVICE);

}

@Override

public int getCount() {

return exchangeRates.size();

}

@Override

public ExchangeRate getItem(int position) {

return exchangeRates.get(position);

}

@Override

public long getItemId(int position) {

return position;

}

@Override

public View getView(int position, View convertView, ViewGroup parent) {

return convertView;

}

}

**Dodajemy klasę wewnętrzną.**

**ViewHolder:**

protected class ViewHolder {

@InjectView(R.id.currencyName)

TextView currencyName;

@InjectView(R.id.averageRate)

TextView averageRate;

private ViewHolder(View rootView) {

ButterKnife.inject(this, rootView);

}

protected void populate(ExchangeRate exchangeRate) {

currencyName.setText(exchangeRate.getCurrency().getCountry() + " " + exchangeRate.getCurrency().getCurrencyName());

averageRate.setText(exchangeRate.getRate().toString());

}

}

**Definiujemy jak wypełniany ma być element listy.**

**W metodzie getView() dodajemy:**

@Override

public View getView(int position, View convertView, ViewGroup parent) {

ViewHolder vh;

if (convertView == null) {

convertView = inflater.inflate(R.layout.item\_currency\_list, parent, **false**);

vh = new ViewHolder(convertView);

convertView.setTag(vh);

} else {

vh = (ViewHolder) convertView.getTag();

}

ExchangeRate exchangeRate = getItem(position);

vh.populate(exchangeRate);

return convertView;

}

**Ważne, by przy inflatowaniu podać jako parametr attacheToRoot „false”(pogrubione w powyższym listingu), ponieważ adapter pod spodem robi to za nas, więc podpięcie samodzielnie spowalnia cały proces.**

**Tworzymy dane testowe i dodajemy adapter do listy w ListCurrenciesActivity w metodzie onCreate():**

**MockData:**

public class MockData {

static RatesList ratesList;

public static RatesList getListOfRates() {

if (ratesList == null) {

List<ExchangeRate> rates = new ArrayList<ExchangeRate>();

rates.add(new ExchangeRate(Currency.AUD, 0.3431));

rates.add(new ExchangeRate(Currency.BGN, 0.4724));

rates.add(new ExchangeRate(Currency.BRL, 0.7974));

rates.add(new ExchangeRate(Currency.CAD, 0.3326));

rates.add(new ExchangeRate(Currency.CHF, 0.2584));

rates.add(new ExchangeRate(Currency.CNY, 1.676));

rates.add(new ExchangeRate(Currency.CZK, 6.6242));

rates.add(new ExchangeRate(Currency.DKK, 1.8007));

rates.add(new ExchangeRate(Currency.GBP, 0.1752));

rates.add(new ExchangeRate(Currency.HKD, 2.0737));

rates.add(new ExchangeRate(Currency.HRK, 1.85));

rates.add(new ExchangeRate(Currency.HUF, 73.764));

rates.add(new ExchangeRate(Currency.IDR, 3469.59));

rates.add(new ExchangeRate(Currency.ILS, 1.0694));

rates.add(new ExchangeRate(Currency.INR, 16.646));

rates.add(new ExchangeRate(Currency.JPY, 32.152));

rates.add(new ExchangeRate(Currency.KRW, 294.43));

rates.add(new ExchangeRate(Currency.MXN, 4.0236));

rates.add(new ExchangeRate(Currency.MYR, 0.9763));

rates.add(new ExchangeRate(Currency.NOK, 2.0644));

rates.add(new ExchangeRate(Currency.NZD, 0.357));

rates.add(new ExchangeRate(Currency.PHP, 11.8));

rates.add(new ExchangeRate(Currency.RON, 1.0738));

rates.add(new ExchangeRate(Currency.RUB, 16.332));

rates.add(new ExchangeRate(Currency.SEK, 2.2258));

rates.add(new ExchangeRate(Currency.SGD, 0.3661));

rates.add(new ExchangeRate(Currency.THB, 8.6685));

rates.add(new ExchangeRate(Currency.TRY, 0.6924));

rates.add(new ExchangeRate(Currency.USD, 0.2674));

rates.add(new ExchangeRate(Currency.ZAR, 3.1443));

rates.add(new ExchangeRate(Currency.EUR, 0.2416));

ratesList = new RatesList("2015-03-07", Currency.PLN, rates);

}

return ratesList;

}

**ListCurrenciesActivity -> onCreate():**

CurrencyListAdapter adapter = new CurrencyListAdapter(this, MockData. getListOfRates());

currencyListView.setAdapter(adapter);

### Zad 5 – Zapisanie wybranej waluty do pamięci trwałej

**Dodajemy klasę pomocniczą do zapisu i odczytu z pamięci trwałej.**

public class SharedPreferencesSupporter {

private static final String CURRENCY\_MAIN\_KEY = SharedPreferencesSupporter.class.getName() + ".currency";

private static final String NAME = ".name";

private static final String AVERAGE\_RATE = ".averageRate";

public static ExchangeRate loadCurrentRate(Context context) {

SharedPreferences preferences = PreferenceManager.getDefaultSharedPreferences(context);

double averageRate = preferences.getFloat(CURRENCY\_MAIN\_KEY + AVERAGE\_RATE, 1f);

String name = preferences.getString(CURRENCY\_MAIN\_KEY + NAME, Currency.PLN.toString());

return new ExchangeRate(Currency.valueOf(name), averageRate);

}

public static void saveCurrentRate(ExchangeRate exchangeRate, Context context) {

SharedPreferences preferences = PreferenceManager.getDefaultSharedPreferences(context);

SharedPreferences.Editor editor = preferences.edit();

editor.putFloat(CURRENCY\_MAIN\_KEY + AVERAGE\_RATE, (float) exchangeRate.getRate().doubleValue());

editor.putString(CURRENCY\_MAIN\_KEY + NAME, exchangeRate.getCurrency().toString());

editor.apply();

}

}

**Następnie obsługujemy kliknięcie na elemencie listy.**

**W klasie ListCurrenciesActivity w metodzie onCreate dodajemy:**

currencyListView.setOnItemClickListener(new AdapterView.OnItemClickListener() {

@Override

public void onItemClick(AdapterView<?> parent, View view, int position, long id) {

CurrencyListAdapter currencyAdapter = (CurrencyListAdapter) parent.getAdapter();

ExchangeRate exchangeRate = currencyAdapter.getItem(position);

SharedPreferencesSupporter.saveCurrentRate(exchangeRate, ListCurrenciesActivity.this);

Toast.makeText(this, "Currency saved to SharedPreferences", Toast.LENGTH\_SHORT).show(); }

});

**Lub używając Butterknifa:**

@OnItemClick(R.id.list)

void onListItemClick(AdapterView<?> parent, View view, int position){

CurrencyListAdapter currencyAdapter = (CurrencyListAdapter) parent.getAdapter();

ExchangeRate exchangeRate = currencyAdapter.getItem(position);

SharedPreferencesSupporter.saveCurrentRate(exchangeRate, ListCurrenciesActivity.this);

Toast.makeText(this, "Currency saved to SharedPreferences", Toast.LENGTH\_SHORT).show();

}

### Zad 6 – Odczyt z pamięci

**By zobaczyć rezultat zapisu w pamięci dodamy pole tekstowe w MainActivity z aktualnie wybraną walutą.**

**Edytujemy activity\_main.xml:**

<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"

android:layout\_width="match\_parent"

android:layout\_height="match\_parent"

android:orientation="vertical"

android:gravity="center">

<TextView

android:id="@+id/currentCurrency"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:textSize="20sp"

android:textStyle="bold" />

<Button

android:id="@+id/listCurrenciesButton"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:text="@string/open\_exchange\_rates\_list" />

<Button

android:id="@+id/rateChangeButton"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:text="@string/change\_rate\_manually" />

</LinearLayout>

**Oraz obsługujemy pobieranie z pamięci w MainActivity:**

**W metodzie onCreate:**

currentCurrency = (TextView) findViewById(R.id.currentCurrency);

**Dodajemy metodę:**

@Override

protected void onResume() {

super.onResume();

currentExchangeRate = SharedPreferencesSupporter.loadCurrentRate(this);

final Currency currency = currentExchangeRate.getCurrency();

currentCurrency.setText("Twoja waluta to: " + currency.getCountry() + " " + currency.getCurrencyName() + "\nKurs: " + currentExchangeRate.getRate());

}

### Zad 7 – Asynchroniczne ładowanie bitmap

**Do pliku build.gradle w tagu dependecsies dodajemy:**

compile 'com.squareup.picasso:picasso:+'

**Następnie dodajemy klasę pomocniczą dostarczającą adresy poszczególnych flag:**

public class FlagAddress {

public static String obtainAddress(Currency currency) {

return "http://192.168.0.11:8087/" + currency.getCountry().toLowerCase().replace(" ", "") + ".png";

}

}

**Uaktualniamy ViewHolder**

protected class ViewHolder {

@InjectView(R.id.currencyName)

TextView currencyName;

@InjectView(R.id.averageRate)

TextView averageRate;

@InjectView(R.id.flag)

ImageView flag;

private ViewHolder(View rootView) {

ButterKnife.inject(this, rootView);

}

}

**W metodzie populate dodajemy pobranie obrazków za pomocą Picasso:**

Picasso.with(context).load(FlagAddress.obtainAddress(exchangeRate.getCurrency())).placeholder(R.drawable.money).into(flag);

### Zad 8 – Pobieranie walut z internetu i parsowanie json’a

**Tworzymy API które definiuje punkty dostępu do serwisu.**

public interface JsonRatesService {

@GET("/list/USD")

RatesList getCurrencyTable();

}

**W ListCurrenciesActivity w metodzie onCreate() konfigurujemy adapter serwisu.**

Gson gson = new GsonBuilder().

setFieldNamingPolicy(FieldNamingPolicy.LOWER\_CASE\_WITH\_UNDERSCORES).

registerTypeAdapter(Currency.class, new CurrencyTypeAdapter()).

create();

RestAdapter restAdapter = new RestAdapter.Builder().

setEndpoint(getString(R.string.webservice\_url)).

setConverter(new GsonConverter(gson)).

build();

service = restAdapter.create(JsonRatesService.class);

**Używamy biblioteki GSON do parsowania JSON.**

**Dodajemy klasę adaptera dla GSON’a wspomagającą parsowanie:**

public class CurrencyTypeAdapter extends TypeAdapter<Currency> {

@Override

public void write(JsonWriter out, Currency value) throws IOException {

if (value == null) {

out.nullValue();

return;

}

out.value(value.toString());

}

@Override

public Currency read(JsonReader in) throws IOException {

if (in.peek() == JsonToken.NULL) {

in.nextNull();

return null;

}

return Currency.valueOf(in.nextString());

}

}

**Pobieranie z internet wywoływane musi być na osobnym wątku, inaczej rzucony zostanie NetworkOnMainThreadException. Dlatego oddelegujemy wywołanie operacji sieciowej do osobnego wątku. Zdefiniujemy wewnętrzną klasę rozszerzającą AsyncTask.**

private class GetCurrencyTableTask extends AsyncTask<Currency, Void, RatesList> {

private final ProgressDialog dialog;

public GetCurrencyTableTask(Context context) {

dialog = new ProgressDialog(context);

dialog.setMessage(getString(R.string.please\_wait));

}

@Override

protected void onPreExecute() {

super.onPreExecute();

dialog.show();

}

@Override

protected RatesList doInBackground(Currency... params) {

return service.getCurrencyTable(params[0]);

}

@Override

protected void onPostExecute(RatesList currencies) {

super.onPostExecute(currencies);

dialog.dismiss();

currencyListView.setAdapter(new CurrencyListAdapter(ListCurrenciesActivity.this, currencies));

}

}

**Dodajemy metodę:**

private void loadData() {

new GetCurrencyTableTask(this).execute();

}

**Pozostaje w metodzie onCreate wywołać metodę loadData:**

### Zad 9 – Dodawanie menu kontekstowego oraz akcji do ActionBar’a

**By zdefiniować menu kontekstowe oraz akcje w ActionBarze, należy utworzyć nowy plik w folderze menu.**

**Change\_currency.xml:**

<menu xmlns:android="http://schemas.android.com/apk/res/android">

<item

android:id="@+id/action\_refresh"

android:title="@string/action\_refresh"

android:showAsAction="always"

android:icon="@drawable/ic\_menu\_refresh"/>

<item

android:id="@+id/menu\_refresh"

android:title="@string/action\_refresh"

android:showAsAction="never"

android:icon="@drawable/ic\_menu\_refresh"/>

</menu>

**Wracamy do ChangeCurrencyActivity i dodajemy metody:**

@Override

public boolean onCreateOptionsMenu(Menu menu) {

getMenuInflater().inflate(R.menu.change\_currency, menu);

return true;

}

@Override

public boolean onOptionsItemSelected(MenuItem item) {

int id = item.getItemId();

if (id == R.id.action\_refresh || id == R.id.menu\_refresh) {

loadData();

return true;

}

return super.onOptionsItemSelected(item);

}

### Zad 10 – Dialog do edycji aktualnej waluty

**Dodajemy klasę definiującą dialog do edycji waluty:**

**...**

**W MainActivity dodajemy obsługę klikania na drugi button:**

changeCurrencyDialogButton.setOnClickListener(new View.OnClickListener() {

@Override

public void onClick(View v) {

RateChangeDialogFragment.getInstance(currentExchangeRate).show(getFragmentManager(), "tag");

}

});

**Oraz implementacje interfejsu OnCurrencychangedListener**

@Override

public void onRateChanged(ExchangeRate exchangeRate) {

currentExchangeRate = exchangeRate;

final Currency currency = exchangeRate.getCurrency();

currentCurrency.setText("Twoja waluta to: " + currency.getCountry() + " " + currency.getCurrencyName() + "\nKurs: " + exchangeRate.getRate());

SharedPreferencesSupporter.saveCurrentRate(exchangeRate, this);

}

### Zad 11 – Przeliczanie walut

### Przydatne linki

* <http://square.github.io/retrofit/>
* <http://jakewharton.github.io/butterknife/>
* <http://square.github.io/picasso/>
* <http://gradleplease.appspot.com/>
* <http://developer.android.com/index.html>