**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 19](#_Toc413450310)

[Zad 6 – Odczyt z pamięci 20](#_Toc413450311)

[Zad 7 – Asynchroniczne ładowanie bitmap 21](#_Toc413450312)

[Zad 8 – Pobieranie walut z internetu i parsowanie json’a 22](#_Toc413450313)

[Zad 9 – Dodawanie menu kontekstowego oraz akcji do ActionBar’a 24](#_Toc413450314)

[Zad 10 – Dialog do edycji aktualnej waluty 25](#_Toc413450315)

[Zad 11 – Przeliczanie walut 27](#_Toc413450316)

[Przydatne linki 29](#_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.

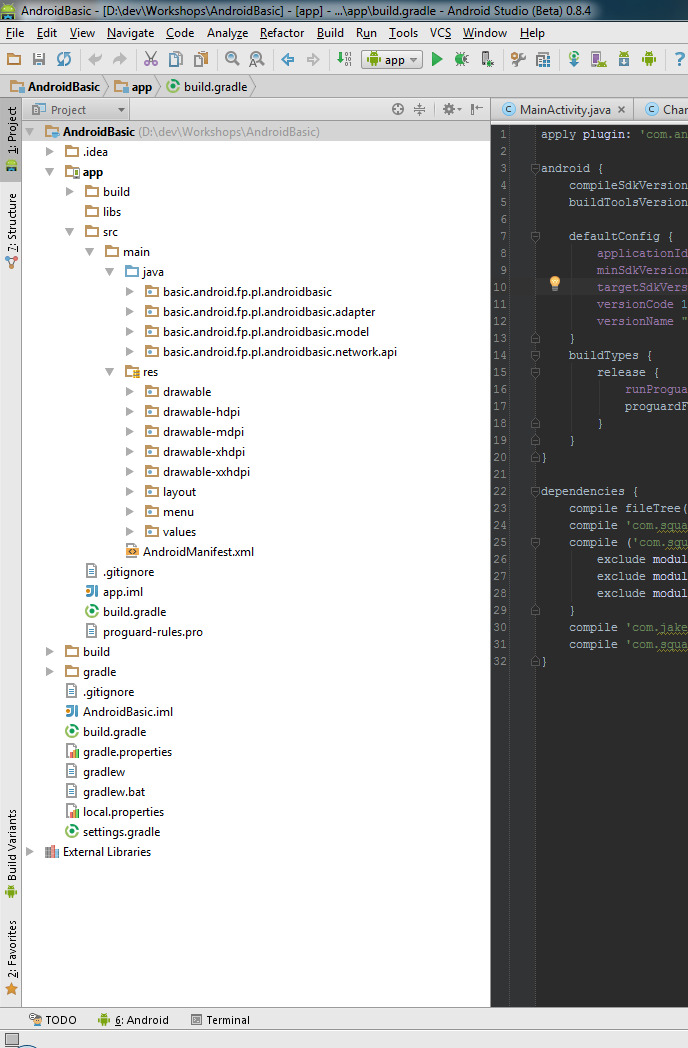
**onResume()** – wywoływana gdy aktywność rozpoczyna interakcję z użytkownikiem. W tym omencie aktywność jest na szczycie sotsu aktywności.

**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 22*

*buildToolsVersion "22.0.1"*

*defaultConfig {*

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

*minSdkVersion 15*

*targetSdkVersion 22*

*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 – Modyfikujemy pierwszą aktywność

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

**MainActivity:**

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>

**Zmieniamy styl aplikacji w styles.xml na:**

<style name="AppTheme" parent="android:Theme.Holo.Light.DarkActionBar">

### Zad 3 – Odkrywamy magię cyklu życia

public class MainActivity extends Activity {

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_main);

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

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

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

}

@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:**

changeCurrencyButton.setOnClickListener(new View.OnClickListener() {

@Override

public void onClick(View v) {

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

startActivity(i);

}

});

**Dodajemy nową aktywność do manifestu.**

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

**Tworzymy layout dla pojedyńczego elementu listy:**

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

<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"

android:src="@drawable/money" />

<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>

**Grafika money dostępna jest na serwerze z zadaniami.**

**JSON otrzymywany z serwisu:**

{

"date" : "2015-04-02",

"base" : {

"currency" : "Dollar",

"country" : "United States"

},

"rates" : [{

"currency" : "Rupiah",

"rate" : 13001.44,

"country" : "Indonesia"

}

]

}

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

**ExchangeRate:**

public class ExchangeRate implements Serializable {

private final String currency;

private final String country;

private Float rate;

public ExchangeRate(String currency, String country, Float rate) {

this.currency = currency;

this.country = country;

this.rate = rate;

}

public String getCurrency() {

return currency;

}

public String getCountry() {

return country;

}

public Float getRate() {

return rate;

}

public void setRate(Float rate) {

this.rate = rate;

}

}

**RatesList:**

public class RatesList {

private final String date;

private final ExchangeRate base;

private final List<ExchangeRate> rates;

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

this.date = date;

this.base = base;

this.rates = exchangeRates;

}

public List<ExchangeRate> getExchangeRates() {

return rates;

}

}

**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 ratesList) {

this.context = context;

exchangeRates = ratesList.getExchangeRates();

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.getCountry() + " " + exchangeRate.getCurrency()); 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 {

private static RatesList ratesList;

public static RatesList getListOfRates() {

if (ratesList == null) {

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

rates.add(new ExchangeRate("Dollar", "Australia", 0.3431f));

rates.add(new ExchangeRate("Lev", "Bulgaria", 0.4724f));

rates.add(new ExchangeRate("Real", "Brazil", 0.7974f));

rates.add(new ExchangeRate("Dollar", "Canada", 0.3326f));

rates.add(new ExchangeRate("Franc", "Switzerland", 0.2584f));

rates.add(new ExchangeRate("Yuan Renminbi", "China", 1.676f));

rates.add(new ExchangeRate("Koruna", "Czech Republic", 6.6242f));

rates.add(new ExchangeRate("Krone", "Denmark", 1.8007f));

rates.add(new ExchangeRate("Pound", "United Kingdom", 0.1752f));

rates.add(new ExchangeRate("Dollar", "Hong Kong", 2.0737f));

rates.add(new ExchangeRate("Kuna", "Croatia", 1.85f));

rates.add(new ExchangeRate("Forint", "Hungary", 73.764f));

rates.add(new ExchangeRate("Rupiah", "Indonesia", 3469.59f));

rates.add(new ExchangeRate("Shekel", "Israel", 1.0694f));

rates.add(new ExchangeRate("Rupee", "India", 16.646f));

rates.add(new ExchangeRate("Yen", "Japan", 32.152f));

rates.add(new ExchangeRate("Won", "Korea (South)", 294.43f));

rates.add(new ExchangeRate("Peso", "Mexico", 4.0236f));

rates.add(new ExchangeRate("Ringgit", "Malaysia", 0.9763f));

rates.add(new ExchangeRate("Krone", "Norway", 2.0644f));

rates.add(new ExchangeRate("Dollar", "New Zealand", 0.357f));

rates.add(new ExchangeRate("Peso", "Philippines", 11.8f));

rates.add(new ExchangeRate("New Leu", "Romania", 1.0738f));

rates.add(new ExchangeRate("Ruble", "Russia", 16.332f));

rates.add(new ExchangeRate("Krona", "Sweden", 2.2258f));

rates.add(new ExchangeRate("Dollar", "Singapore", 0.3661f));

rates.add(new ExchangeRate("Baht", "Thailand", 8.6685f));

rates.add(new ExchangeRate("Lira", "Turkey", 0.6924f));

rates.add(new ExchangeRate("Rand", "South Africa", 3.1443f));

rates.add(new ExchangeRate("Euro", "Euro Member", 0.2416f));

ratesList = new RatesList("2015-03-07", new ExchangeRate("Zloty", "Poland", 0f), rates);

}

return ratesList;

}

}

**Dodajemy dane do adaptera.**

**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();

private static final String CURRENCY = ".currency";

private static final String COUNTRY = ".country";

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

public static ExchangeRate loadCurrentRate(Context context) {

SharedPreferences preferences = PreferenceManager.getDefaultSharedPreferences(context);

float averageRate = preferences.getFloat(CURRENCY\_MAIN\_KEY + AVERAGE\_RATE, 3.73f);

String currency = preferences.getString(CURRENCY\_MAIN\_KEY + CURRENCY, "Dollar");

String country = preferences.getString(CURRENCY\_MAIN\_KEY + CURRENCY, "United States");

return new ExchangeRate(currency, country, 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, exchangeRate.getRate());

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

editor.putString(CURRENCY\_MAIN\_KEY + COUNTRY, exchangeRate.getCountry());

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(ListCurrenciesActivity.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, 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 dodajemy wyszukanie nowego widoku:**

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

**Dodajemy metodę:**

@Override

protected void onResume() {

super.onResume();

currentExchangeRate = SharedPreferencesSupporter.loadCurrentRate(this);

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

}

**Tworzymy pola:**

private TextView currentCurrency;

private ExchangeRate currentExchangeRate;

### 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 FlagAddressBuilder {

public static String obtainAddress(Context context, ExchangeRate rate) {

String ws\_url = context.getString(R.string.webservice\_url);

String port = context.getString(R.string.static\_webservice\_port);

return ws\_url + ":" + port + "/" + rate.getCountry().toLowerCase().replace(" ", "") + ".png";

}

}

**Do zasobów strings.xml dodajemy adres i port webserwisu:**

<string name="webservice\_url">http://... ip zostanie podane na warsztacie</string>

<string name="static\_webservice\_port">8087</string>

**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);

}

protected void populate(ExchangeRate exchangeRate) {

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

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

}

}

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

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

**By móc komunikować się z internetem należy nadać aplikacji pozwolenie w manifeście.**

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

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

**W Build gradle nalezy dodac linijke:**

compile 'com.squareup.retrofit:retrofit:+'

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

public interface JsonRatesService {

@GET("/list/USD")

RatesList getCurrencyTable();

}

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

RestAdapter restAdapter = new RestAdapter.Builder().

setEndpoint(getString(R.string.webservice\_url) + ":" + getString(R.string.webservice\_port)).

build();

service = restAdapter.create(JsonRatesService.class);

**Niejawnie używamy biblioteki GSON do parsowania JSON.**

**Musimy jeszcze dodać pole:**

private JsonRatesService service;

**Oraz nowy zasób tekstowy:**

<string name="webservice\_port">8086</string>

**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<Void, 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(Void... param) {

return service.getCurrencyTable();

}

@Override

protected void onPostExecute(RatesList currencies) {

super.onPostExecute(currencies);

dialog.dismiss();

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

}

}

**Dodajemy zasób tekstowy:**

<string name="please\_wait">Proszę czekać</string>

**Dodajemy metodę:**

private void loadData() {

new GetCurrencyTableTask(this).execute();

}

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

loadData();

**Usuwamy dane testowe z metody onCreate:**

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

currencyListView.setAdapter(adapter);

### 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>

**Dodajemy zasób tekstowy:**

<string name="action\_refresh">Odśwież</string>

**Wracamy do ListCurrenciesActivity 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:**

public class RateChangeDialogFragment extends DialogFragment {

private static final String CURRENCY\_BUNDLE\_KEY = "CURRENCY\_BUNDLE\_KEY";

private OnCurrencyChangedListener onCurrencyChangedListener;

private ExchangeRate currencyRate;

private EditText inputEditText;

public static RateChangeDialogFragment getInstance(ExchangeRate rate) {

Bundle bundle = new Bundle();

bundle.putSerializable(CURRENCY\_BUNDLE\_KEY, rate);

RateChangeDialogFragment fragment = new RateChangeDialogFragment();

fragment.setArguments(bundle);

return fragment;

}

@Override

public void onAttach(Activity activity) {

super.onAttach(activity);

// This makes sure that the container activity has implemented

// the callback interface. If not, it throws an exception

try {

onCurrencyChangedListener = (OnCurrencyChangedListener) activity;

} catch (ClassCastException e) {

throw new ClassCastException(activity.toString() + " must implement OnCurrencyChangedListener");

}

currencyRate = (ExchangeRate) getArguments().getSerializable(CURRENCY\_BUNDLE\_KEY);

}

@Override

public Dialog onCreateDialog(Bundle savedInstanceState) {

inputEditText = createInputEditText();

return new AlertDialog.Builder(getActivity()) //

.setIcon(R.mipmap.ic\_launcher) //

.setTitle(R.string.dialog\_title) //

.setMessage(R.string.dialog\_message) //

.setPositiveButton(R.string.ok, new PositiveOnClickListener()) //

.setNegativeButton(R.string.cancel, null) //

.setView(inputEditText) //

.create();

}

private EditText createInputEditText() {

EditText input = new EditText(getActivity());

input.setInputType(InputType.TYPE\_CLASS\_NUMBER | InputType.TYPE\_NUMBER\_FLAG\_DECIMAL);

input.addTextChangedListener(new CurrencyTextWatcher());

input.setText(String.valueOf(currencyRate.getRate()));

return input;

}

private boolean isValid(String text) {

try {

Float.parseFloat(text);

return true;

} catch (NumberFormatException e) {

return false;

}

}

private class PositiveOnClickListener implements DialogInterface.OnClickListener {

@Override

public void onClick(DialogInterface dialog, int which) {

changeExchangeRate();

}

}

private void changeExchangeRate() {

if (isValid(inputEditText.getText().toString())) {

onCurrencyChangedListener.onRateChanged(currencyRate);

} else {

Toast.makeText(getActivity(), R.string.invalid, Toast.LENGTH\_SHORT).show();

}

}

private class CurrencyTextWatcher implements TextWatcher {

@Override

public void onTextChanged(CharSequence s, int start, int before, int count) {

}

@Override

public void beforeTextChanged(CharSequence s, int start, int count, int after) {

}

@Override

public void afterTextChanged(Editable s) {

if (isValid(s.toString())) {

currencyRate.setRate(Float.parseFloat(s.toString()));

}

}

}

public interface OnCurrencyChangedListener {

void onRateChanged(ExchangeRate currency);

}

}

**Dodajemy zasoby tekstowe:**

<string name="dialog\_title">Zmiana waluty</string>

<string name="dialog\_message">Zmieńmy wartość przelicznika</string>

<string name="ok">OK</string>

<string name="cancel">Cancel</string>

<string name="invalid">Nieprawidłowa liczba</string>

**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 implementujemy interfejs OnCurrencychangedListener w MainActivity**

public class MainActivity extends Activity implements RateChangeDialogFragment.OnCurrencyChangedListener

@Override

public void onRateChanged(ExchangeRate exchangeRate) {

currentExchangeRate = exchangeRate;

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

SharedPreferencesSupporter.saveCurrentRate(exchangeRate, this);

}

### Zad 11 – Przeliczanie walut

**Tworzymy nowe activity:**

public class ExchangeActivity extends Activity {

public static final String CURRENCY\_BUNDLE\_KEY = "CURRENCY\_BUNDLE\_KEY";

@InjectView(R.id.newCurrency)

protected TextView newCurrencyTextView;

@InjectView(R.id.currency)

protected TextView currencyTextView;

private ExchangeRate exchangeRate;

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_exchange);

ButterKnife.inject(this);

exchangeRate = (ExchangeRate) getIntent().getSerializableExtra(CURRENCY\_BUNDLE\_KEY);

currencyTextView.setText("Aktualny kurs to:\t" + exchangeRate.getRate());

}

@OnTextChanged(value = R.id.currencyEditText, callback = OnTextChanged.Callback.AFTER\_TEXT\_CHANGED)

protected void onTextChange(Editable text) {

if (isValid(text.toString())) {

float value = Float.parseFloat(text.toString()) \* exchangeRate.getRate();

newCurrencyTextView.setText("To\t" + value + "\t" + exchangeRate.getCurrency());

} else {

newCurrencyTextView.setText(R.string.invalid);

}

}

private boolean isValid(String text) {

try {

Float.parseFloat(text);

return true;

} catch (NumberFormatException e) {

return false;

}

}

}

**Dodajemy layout activity\_exchange.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:layout\_margin="30dp">

<TextView

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:text="USD:"

android:textSize="12sp"

android:textStyle="bold" />

<EditText

android:id="@+id/currencyEditText"

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:inputType="number|numberDecimal" />

<TextView

android:id="@+id/newCurrency"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:layout\_marginTop="30dp" />

<TextView

android:id="@+id/currency"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:layout\_marginTop="30dp" />

</LinearLayout>

**Do manifestu dodajemy nową aktywność.**

<activity android:name=".ExchangeActivity" />

**Edytujemy metodę onCreate() w MainActivity – dodajemy:**

Button calculateCurrencyButton = (Button) findViewById(R.id.calculateCurrency);

calculateCurrencyButton.setOnClickListener(new View.OnClickListener() {

@Override

public void onClick(View v) {

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

i.putExtra(ExchangeActivity.CURRENCY\_BUNDLE\_KEY, currentExchangeRate);

startActivity(i);

}

});

**Pozostaje dodanie nowego przycisku do activity\_main.xml**

<Button

android:id="@+id/calculateCurrency"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:text="@string/convert\_currency" />

**Oraz brakujący zasób tekstowy**

<string name="convert\_currency">Przelicz waluty</string>

### 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>