OSNOVE RAZVOJA WEB I MOBILNIH APLIKACIJA

LV4: Dohvaćanje online podataka

1. Laboratorijska vježba 4

1.1. Uvod

U ovoj vježbi koristiti ćemo Retrofit biblioteku kako bi dohvatili I serijalizirali online podatke u JSON formatu.

1.1.1. Potrebna predznanja

Osnove programiranja

- Kolegiji Programiranje I, Programiranje II, Algoritmi i strukture podataka

Osnove objektno orijentiranog programiranja

- Kolegij Objektno orijentirano programiranje

Osnove Java programskog jezika

- H. Schildt, Java, A Beginner's Guide, 5th Edition
- B. Eckel, Thinking in Java, 4th edition
- http://docs.oracle.com/javase/tutorial/

Osnove izrade Android aplikacija (activity, osnove UI-a)

- Predložak LV1
- Predložak LV2
- Predložak LV3
- Predavanja

1.1.2. Korisna predznanja

- JSON, http://www.json.org/
- Retrofit, https://square.github.io/retrofit/

1.2. Parsiranje online podataka

1.2.1. JSON

Efikasan način za predstavljanje informacija je JSON (JavaScript Object Notation). Radi se o laganom formatu za razmjenu podataka koji je lako čitljiv ljudima, a istovremeno jednostavan za generiranje, parsiranje i obradu na računalima.

Primjer izgleda JSON-a je:


```
"Users": [
    {
        "id": 1,
        "name": "Leanne Graham",
        "username": "Bret",
        "email": "Sincere@april.biz",
        "address": {
              "street": "Kulas Light",
              "suite": "Apt. 556",
              "city": "Gwenborough",
              "zipcode": "92998-3874",
              "geo": {
                   "lat": "-37.3159",
                   "lng": "81.1496"
              }
        },
        "phone": "1-770-736-8031 x56442",
        "website": "hildegard.org",
        "company": {
               "name": "Romaguera-Crona",
               "catchPhrase": "Multi-layered client-server neural-net",
               "bs": "harness real-time e-markets"
        }
    },
    ...
]
```


Slika 1.1. Primjer JSON-a

Slikom 1.1 prikazan je JSON objekt koji je vraćen prilikom upita na poslužitelj. Unutar JSON objekta nalazi se JSON array pod nazivom Users koji sadržava nove JSON objekte. Objekti unutar elemenata polja sadržavaju parove ključ/vrijednost koji opisuju pojedinog korisnika.

1.2.2. Retrofit

Retrofit je REST (engl. *Representational state transfer*) klijent građen nad OkHttp klijentom dizajniran za Android i Javu. Koristi anotacije kako bi opisao različite HTTP zahtjeve, pruža mogućnost kao što su dinamički parametri unutar URL-a, parametre upita, prilagođena zaglavlja, skidanje i učitavanje datoteka, pisanje lažnih odgovora (engl. *Mocking responses*). Omogućava sinkrone i asinkrone zahtjeve, a samostalno brine o sinkronizaciji, upravljaju nitima, keširanju, učitavanju, itd. Za upravljanje HTTP zahtjevima koristi OkHttp biblioteku koja je obavezna uz Retrofit, omogućuje jedinstveno definiranje krajnjih točaka komunikacije putem HTTP protokola (engl. *endpoint*).

Koristeći Retrofit u kombinaciji sa Gson pretvaračem za JSON ili Xml pretvaračem za XML prilično je lako podatke koji se nalaze na nekom Web servisu u JSON ili XML formatu dohvatiti i pretvoriti u POJO (engl. *Plain Old Java Object*). Retrofit automatski serijalizira JSON podatke koristeći Gson pretvarač i POJO koji mora biti unaprijed definiran po JSON strukturi, takav POJO objekt može sadržavati sve atribute koje ima JSON ili samo one koji se žele dohvatiti.

Za izvršavanje Retrofit zahtjeva potrebna je model klasa koja se koristi za serijalizaciju odgovora, sučelje u kojem je definiran tip HTTP zahtjeva te Retrofit instancu koja se koristi za izvršavanje HTTP zahtjeva definiranog u sučelju. Svako sučelje predstavlja kolekciju različitih API poziva, svaka metoda unutar sučelja koja se želi koristi kao HTTP zahtjev mora sadržavati anotaciju zahtjeva koji obavlja

1.3. Kreiranje Android projekta

Kreirati Android projekt kako je pokazano u predlošku LV1 s *Empty Activity* opcijom.

Sve projekte potrebno je u laboratoriju smjestiti u D:¥RMA¥ImePrezime

1.3.1. Kreiranje polaznog layouta

Izgled glavnog ekrana aplikacije definiran je u *activity_main.xml* datoteci. U nju je potrebno dodati *TextView* komponentu koja će prikazivati prikupljene online podatke. Primjer sastavljenog XML-a prikazan je na slici 1.2.

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

    <TextView
        android:id="@+id/tvUserInfo"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content" />
</ScrollView>
```

Slika 1.2. activity-main.xml

1.3.2. Dodavanje Retrofit biblioteke u projekt

```
dependencies {
    implementation fileTree(dir: 'libs', include: ['*.jar'])
    implementation 'com.android.support:appcompat-v7:28.0.0'
    implementation 'com.android.support.constraint:constraint-layout:1.1.3'

    implementation 'com.squareup.retrofit2:retrofit:2.5.0'
    implementation 'com.squareup.retrofit2:converter-gson:2.5.0'
}
```

Slika 1.3. build.gradle(app)

1.4. POJO objekt

Plain Old Java Object (POJO) predstavlja jednostavnu data klasu koja sadrži attribute te metode potrebne za postavljanje I pristupanje istim atributima.

1.4.1. Kreiranje POJO objekta

```
public class User {
    private int id;
    private String name;
    private Address address;

    public int getId() {
        return id;
    }

    public String getName() {
        return name;
    }

    public Address getAddress() {
        return address;
    }
}
```

Slika 1.4. User.class

```
public class Address {
    private String street;
    private String suite;
    private String city;
    private String zipcode;

    public String getStreet() {
        return street;
    }

    public String getSuite() {
        return suite;
    }

    public String getCity() {
        return city;
    }

    public String getZipcode() {
        return zipcode;
    }
}
```

Slika 1.5. Address.class

1.4.2. Kreiranje API sučelja

Unutar API sučelja navode se krajnje točke pojedinih HTTP zahtjeva unutar GET ili POST anotacija koje predstavljaju odgovarajuće HTTP metode te apstraktne metode čije će se implementacije generirati pri kompilaciji projekta. Osnovni tip koji ove metode mogu vraćati jest wrapper objekt *Call* oko tipa podatka koji dohvaćamo. HTTP zahtjev izvršava se *enqueue* metodom koja vraca Callback s podacima ili greškom. Moguće je dohvatiti i *Response* objekt koji sadrži podatke kao što su *response code*, *error message* i slično.

```
public interface APIInterface {
    @GET("users")
    Call<List<User>> getUsers();
}
```

Slika 1.6. APIInterface.java

1.4.3. Kreiranje Retrofit sučelja

Instanca Retrofit servisa kreira se i koristi kao *Singleton*. Ova klasa sadrži polje koje predstavlja instancu servisa. Ako instance ne postoji kreiramo ju, a ukoliko već postoji samo ju vraćamo. Instanca se kreira *Builder patternom* kojim postavljamo bazni URL, HTTP klijent, *JSON parser* i slično. Na kraju predajemo prije spomenuto sučelje s navedenim krajnjim točkama te tako stvaramo njegovu implementaciju.

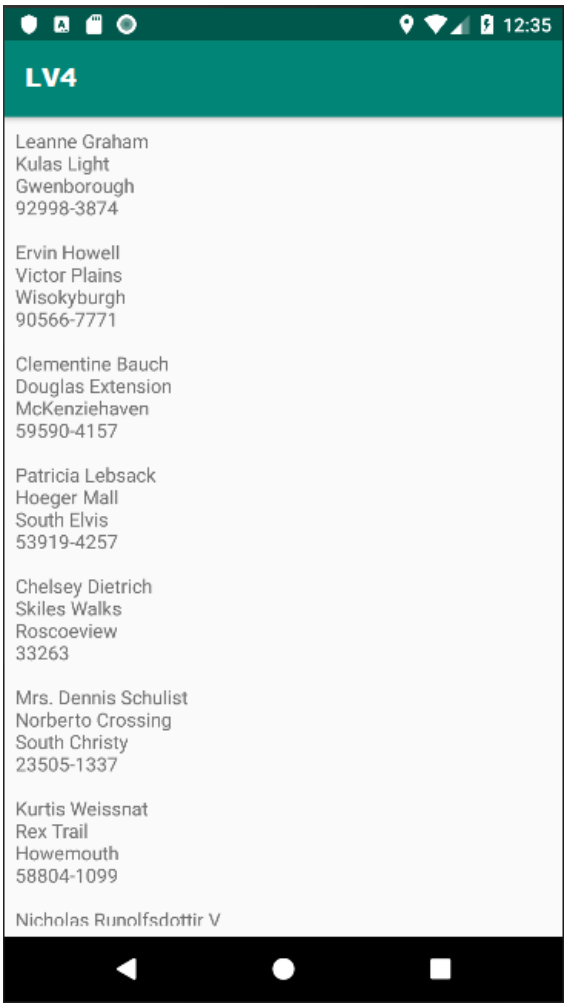
```
public class NetworkUtils {
    private static final String BASE_API =
"https://jsonplaceholder.typicode.com/";

    private static APIInterface apiInterface;

    public static APIInterface getApiInterface() {
        if (apiInterface == null) {
            Retrofit retrofit = new Retrofit.Builder()
                    .baseUrl(BASE_API)
                    .addConverterFactory(GsonConverterFactory.create())
                    .build();

            apiInterface = retrofit.create(APIInterface.class);
        }
        return apiInterface;
    }
}
```

Slika 1.7. NetworkUtils.class

1.4.4. Dohvaćanje online podataka

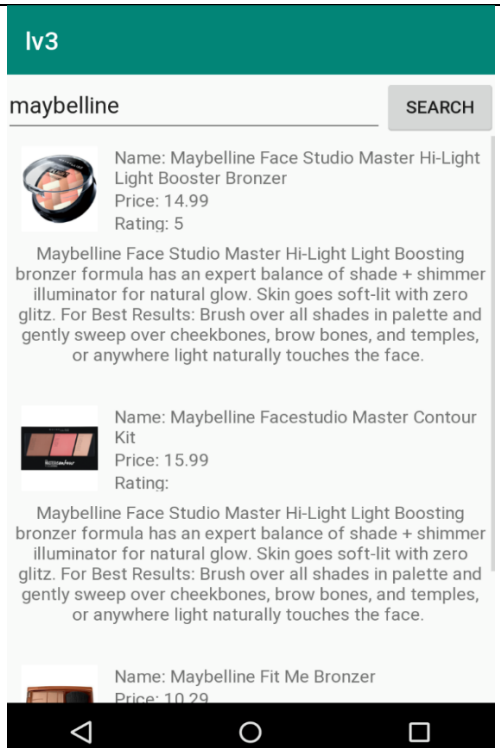
```
public class MainActivity extends AppCompatActivity {

    private TextView tvUserInfo;
    private Call<List<User>> apiCall;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        tvUserInfo = findViewById(R.id.tvUserInfo);
        setUpApiCall();
    }

    private void setUpApiCall() {
        apiCall = NetworkUtils.getApiInterface().getUsers();
        apiCall.enqueue(new Callback<List<User>>() {
            @Override
            public void onResponse(Call<List<User>> call, Response<List<User>>
response) {
                if (response.isSuccessful() && response.body() != null) {
                    showUsers(response.body());
                }
            }

            @Override
            public void onFailure(Call<List<User>> call, Throwable t) {
                Toast.makeText(MainActivity.this, "Error", Toast.LENGTH_SHORT).show();
            }
        });
    }

    private void showUsers(List<User> data) {
        StringBuilder stringBuilder = new StringBuilder();
        for (User tempUser : data) {
            stringBuilder.append(tempUser.getName()).append("\n")
                    .append(tempUser.getAddress().getStreet()).append("\n")
                    .append(tempUser.getAddress().getCity()).append("\n")
                    .append(tempUser.getAddress().getZipcode()).append("\n\n");

        }
        tvUserInfo.setText(stringBuilder.toString());
    }

    @Override
    protected void onDestroy() {
        super.onDestroy();
        if (apiCall != null)
            apiCall.cancel();
    }
}
```

Slika 1.8. MainActivity.class



Slika 1.9. Izgled gotove aplikacije

2. Zadaća

Zadaća 3. App
Korištenjem naučenoga na ovoj vježbi, kreirajte jednostavnu aplikaciju koja će pretraživati pružatelja podataka s različitim makeup proizvodima, primjer URL-a za dohvaćanje proizvoda brenda Maybelline je - **http://makeup-api.herokuapp.com/api/v1/products.json?brand=maybelline** Aplikacija mora imati mogućnost upisivanja brenda u EditText te na pritisak Buttona prikazati rezultate u RecyclerView-u. RecyclerAdapter će imati custom layout koji će se sastojati od četiri TextView komponente u kojima će se prikazivati ime, cijena, ocjena te opis proizvoda, te jednog ImageView-a koji će prikazivati sliku proizvoda. (Za prikaz slike možete koristiti neku od biblioteka kao što su Picasso, Glide, Fresco itd.) Ukoliko poslužitelj ne vrati rezultat prikazati poruku da rezultati pretraživanja nisu pronađeni. Na linku **http://makeup-api.herokuapp.com/** možete pogledati detalje pružatelja podataka (koje brendove podržava, te ostale mogućnosti).
• Kreirajte novi Android projekt s jednim *Activityem* • Napraviti sve potrebno za korištenje RecyclerViewa • Napraviti API sučelje s HTTP metodom • Napraviti Retrofit poziv koji će dohvatiti podatke s pružatelja • Prikazati podatke u RecyclerView-u


Napomena:
Svaka je nadogradnja poželjna i dobrodošla. Korištenje znanja koja nadilaze opseg vježbi se potiče.