A dream of 200

OSC: ADA

Cemu sluzi networking?

- Komunikaciji izmedu servera i mobilnog uredaja
- U mogucnosti smo slati i primati podatke sa servera

Sto je REST

- REST, or REpresentational State Transfer
- Arhitektura koja pruza standard izmedu racunalnih sistema (klijenta i servera)

Rest request zahtjeva:

- HTTP verb (GET, POST, PUT, DELETE)
- Header, koji sadrzi podatke o requestu
- Putanju do resursa
- Opcionalnu poruku koja sadrzi zeljeni info

HTTP Verbs

Postoje 4 osnovne vrste HTTP verbova koje se koriste u REST sustavima:

- GET - Dohvaca specificni resurs (po ID-ju) ili kolekciju
- POST - Kreira novi resurs
- PUT - Updatea specificni resurs (po ID-ju)
- DELETE - Brise specificni resurs (po ID-ju)

JSON

- JavaScript Object Notation
- Format za razmjenu podataka bez trosenja puno resurasa
- Jednostavan za pisati i citati

{"email": "mail@mail.com",
“name”: “name1”,
"password": "1234"}

Retrofit

- Retrofit pretvara HTTP API u Java interface
- Uz pomoc Retrofit klase generira se servis
- Retrofit sadrzava klasu Call koja radi sinkrone ili asinkrone pozive na server

OkHttp

- Efikasan HTTP klijent za rad s HTTP requestioma
- U pozadini rjesava stvari poput visestrukih requestova i socketinga
- Moguce ga je koristiti samoga umjesto Retrofita

Gson

- Googleov open source library za rad s Jsonom
- U mogucnosti je pretvoriti objekte u Json reprezentaciju
- Moze pretvoriti Json string u Java objekt
- Radi uz pomoc anotacija

Implementacija potrebnih librarija

```
implementation('com.squareup.retrofit2:retrofit:2.4.0') {
   exclude module: 'com.squareup.okhttp'
}
implementation 'com.squareup.okhttp3:logging-interceptor:3.10.0'
implementation 'com.squareup.retrofit2:converter-gson:2.4.0'
implementation 'com.google.code.gson:gson:2.8.2'
```

Manifest

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

UI za registraciju i login

RegisterActivity.java

Potrebna polja:

- Username
- Email
- Password
- Opis svakog polja
- Register button
- Login Button

LoginActivity.java

Potrebna polja:

- Username
- Password
- Opis svakog polja
- Login Button
- Register Button

Kreiranje Retrofita

```
public static final String BASE_URL =
"https://authenticationexample.herokuapp.com/";

public static Retrofit createRetrofit() {
   return new Retrofit.Builder()
           .baseUrl(BASE_URL)
           .addConverterFactory(GsonConverterFactory.create(getGson()))
           .client(okHttpClient())
           .build();
}
```

Kreiranje Retrofita

```
public static Gson getGson() {
   return getCommonBuilder()
           .create();
}

private static GsonBuilder getCommonBuilder() {
   return new GsonBuilder().excludeFieldsWithoutExposeAnnotation();
}
```

Kreiranje Retrofita

```
private static HttpLoggingInterceptor provideLoggingInterceptor() {
   return new
HttpLoggingInterceptor().setLevel(HttpLoggingInterceptor.Level.BODY);
}

private static OkHttpClient okHttpClient() {
   return new
OkHttpClient.Builder().addInterceptor(provideLoggingInterceptor()).build();
}
```

Service interface

```
public interface ApiService {

   @POST("api/register/")
   Call<RegistrationToken> registerUser(@Body JsonElement body);

}
```

Model poziva na server

```
public class RegistrationToken {
   @Expose
   @SerializedName("name")
   public String mUserName;

   @Expose
   @SerializedName("email")
   public String mEmail;

   @Expose
   @SerializedName("password")
   public String mPassword;
}
```

Kreiranje poziva

```
Retrofit retrofit = NetworkingUtil.createRetrofit();


ApiService apiService = retrofit.create(ApiService.class);
```

Kreiranje poziva

```
RegistrationToken registrationToken = new RegistrationToken();
registrationToken.mUserName = mUsernameEditText.getText().toString();
registrationToken.mEmail = mEmailEditText.getText().toString();
registrationToken.mPassword = mPasswordEditText.getText().toString();
```

Kreiranje poziva

```
Call registerCall = apiService.registerUser(json);
registerCall.enqueue(new Callback() {
    @Override
    public void onResponse(Call call, Response response) {

    }
    @Override
    public void onFailure(Call call, Throwable t) {

    }
});
```

Zadaci

- Kreiranje Login requesta
- Spremanje Tokena
- POSTanje novog taska na server
- Dohvacanje svih taskova sa servera

Zadaca

1. Prikazati registraciju samo prvi puta kada user pokrene aplikaciju. Svaki iduci puta prikazati login.
2. Omoguciti fejvanje taska, te kada user fejva task, isti prosljediti na server, te dati do znanja useru koji je task fejvan, a koji ne
3. Omoguciti brisanje taska sa servera
4. Prikazivati pageve Taskova. U mogucnosti ste querijati page po page sa servera. Napraviti UI koji ce listati stranice te raditi poziv na sever ovisno o selectanoj stranici.
5. Prije pozivanja svakog requesta provjeriti network state, te upozoriti usera ako nije spojen na net.

6. (Bonus zadatak)
Napraviti cache. Uz pomoc librarija po izboru spremiti podatke sa servera na uredaj. Kada je user offline, mora biti u mogucnosti vidjeti listu taskova.
7. (Bonus bonus zadatak)
Omoguciti useru da kreira novi task offline, te pri ponovnom spajanju na net, uploadati novokreirane taskove na server.