Literarno udruženje UDD

Stefan Mirilović, E2 77/2020

1. Uvod

Projekat literarno udruženje je realizovan mikroservisnom arhitekturom u Java Spring Boot okruženju. Eureka se koristi kao service discovery i Zuul služi kao gateway između različitih mikroservisa. Klijentski deo projekta je razvijen u Angular okruženju.

Pored mikroservisa koji se koriste za plaćanje za predmet SEP, biće jedan centralni mikroservis *LiterarySociety* koji će predstavljati projekat iz predmeta UPP.

Iako bismo mogli da implementiramo pretragu u istom mikroservisu, zbog toga što se ovaj projekat radi individualno za razliku od ostalih, najlogičniji pristup bi bio napraviti odvojeni mikroservis, baš za pretragu, koji bi se zvao *SearchModule*.

2. Arhitektura aplikacije

SearchModule bi se sastojao od više slojeva ili paketa:

- Model – sloj čine klase modela
- Service – sloj čine servisi koji vrše radnu logiku
- Controller – sloj čine endpoint-i koji primaju zahteve, prosleđuju servisnom sloju i pretvaraju rezultat u odgovor
- Repository – sloj čine interfejsi koji direktno upravljaju bazom podataka i omogućavaju pravljenje upita nad Elasticsearch-om
- DTO (Data Transfer Object) – sloj čine objekti preko kojih se komunicira sa klijentskim delom umesto klasa modela
- Config – sloj čine klase za konfiguraciju

2.1. Model

Dve klase bi činile sloj modela:

- *Book*
- *BetaReader*

Ova dva modela predstavljaju indexing unit za Elasticsearch.

Izgled klase *Book* se može videti na Slici 1, a klasa *BetaReader* na Slici 2. Ovo ne predstavlja njihov konačan izgled, ali dobro predstavlja ideju.

```
@Document(indexName = "book")
public class Book {
 @Id
@Field(type = FieldType.Long, store = true)
private Long LUid;  //id iste knjige u LU aplikaciji
@Field(type = FieldType.Keyword, store = true)
private String filename;    //putanja do fajla
@Field(type = FieldType.Text, store = true)
private String title;
@Field(type = FieldType.Text, store = true)
private String text;    //sadrzaj knjige
@Field(type = FieldType.Text, store = true)
private String authorFirstName;
@Field(type = FieldType.Text, store = true)
private String authorLastName;
@Field(type = FieldType.Keyword, store = true)
private String genre;
}
```

Slika 1 - WIP izgled Book klase

```
@Document(indexName = "betaReader")
public class BetaReader {
    @Id
    @Field(type = FieldType.Long, store = true)
    private Long LUid;  //id istog beta citaoca u LU aplikaciji
    @Field(type = FieldType.Keyword, store = true)
    private String name;
    @Field(type = FieldType.Keyword, store = true)
    private String surname;
    @GeoPointField
    private GeoPoint point;
}
```

Slika 2 - WIP izgled BetaReader klase

Kao što se vidi na Slikama 1 i 2, ove dve klase imaju id tipa Long koji će odgovarati njihovom id-u iz *LiterarySociety* mikroservisa. Ovo je zato što *LiterarySociety* drži bazu gde će takođe biti sačuvane knjige i beta čitaoci. Dodavanje knjiga i beta čitaoca na *LiterarySociety* će sinhrono dodati i u *SearchModule* pomoću *FeignClient*-a. *Upload* i *download* knjiga je zamišljen da bude funkcionalnost *LiterarySociety*-a, zbog veza sa npr. kupcima koje ovom mikroservisu (za sad) ne trebaju. U bazi će se čuvati putanja do knjige, dok će se sam fajl nalaziti na nekom direktorijumu.

Geoprostorna pretraga će biti opisana u daljem tekstu.

2.2. Repository

Sloj repozitorijuma će sastojati interfejse koji odgovaraju klasama iz modela:

- BookRepository
- BetaReaderRepository

Ovi interfejsi će *extend*-ovati *ElasticsearchRepository* i time će omogućiti lako pravljenje upita.

2.3. Elasticsearch klijent

Kao klijent za Elasticsearch će se koristiti *Java High Level Rest Client*, koji je ionako *default*-ni. Konfiguracija ovog klijenta će se nalaziti u RestClientConfig klasi u *config* paketu. *RestClientConfig* će izgledati kao na Slici 3.

```
@Configuration
public class RestClientConfig extends AbstractElasticsearchConfiguration {

   @Override
   public RestHighLevelClient elasticsearchClient() {
      ClientConfiguration clientConfiguration
            = ClientConfiguration.builder()
            .connectedTo("localhost:9200")
            .build();
      return RestClients.create(clientConfiguration).rest();
   }
}
```

Slika 3 - WIP izgled RestClientConfig klase

3. Konfiguracija Elasticsearch-a

Elasticsearch verzija 7.4.0. će se pokretati na portu 9200. Zašto ova verzija, a ne neka novija? To je zato što je *SerbianPlugin* (sa https://github.com/chenejac/udd06) kompatibilan sa tom verzijom.

Pored ove instance Elasticsearch-a, zato što *Plagiator* ne radi sa verzijom 7.4.0., koristiće se neka starija verzija na drugom portu. Koja tačno verzija nije još sigurno u ovoj fazi.

3.1. Analyzer za Srpski jezik

Za pretprocesiranje srpskih tekstova će se koristiti već spomenuti *SerbianPlugin*.

SerbianPlugin će biti podešen po uputstvu sa istog github-a. Plugin će onda biti instaliran i pokretaće se zajedno sa instancom Elasticsearch-a. Moraće se i namestiti pravilan *mapping* da bi analizirao pravilna polja za oba naša indexing unit-a. U ovoj fazi *mapping* još nije namešten.

3.2. Geoprostorna pretraga

Na Slici 2 se vidi polja *point* koje je tipa *GeoPoint*. Ovo je u suštini kombinacija dva polja, *latitude* i *longitude*, koji zajedno predstavljaju lokaciju *BetaReader*-a. Da bi dobili ova dva polja iz naziva grada koji korisnik unese na formi za registraciju za *LiterarySociety*, moramo koristiti API za pretvaranje adrese u *latitude* i *longitude*. Ima mnoštvo ovakvih API-a koji ovo nude, najpopularniji od kojih je *Google*-ov

Geocoding API. Nažalost, ovaj, kao i većina dostupnih API-a, deluje da se plaća. U ovoj fazi projekta još nije izabran specifičan API, ali neki od kandidata su:

- PositionStack (https://positionstack.com),
- LocationIQ (https://locationiq.com),
- MapQuest Developer (https://developer.mapquest.com/documentation/open/geocoding-api/),
- Geocod.io (https://www.geocod.io),
- HERE Developer (https://developer.here.com/c/geocoding), među ostalima.

Ideja je da nakon što se beta čitalac uspešno registruje na *LiterarySociety*-u, pita se jedan od navedenih API-a da vrati *longitude* i *latitude* od njegovog grada koji je uneo u formi, i nakon što dobije rezultat, pošalje preko *FeignClient*-a POST metodu *SearchModule*-u sa kojom se napravi indexing unit *BetaReader*-a.

Za dobijanje čitalaca koji su van 100km od pozicije autora, uzme se grad autora iz baze *LiterarySociety*-a, šalje se upit jednom od geocoding API-a i kad vrati rezultat, *latitude* i *longitude* se onda ubacuju u upit za Elasticsearch preko klase BetaReaderRepository koji bi izgledao nešto poput na Slici 4.

```
GET /my_locations/_search
{
    "query": {
        "bool" : {
            "must_not" : {
                "match_all" : {}
            },
            "filter" : {
                "geo_distance" : {
                    "distance" : "100km",
                    "pin.location" : {
                        "lat" : 40,
                        "lon" : -70
                    }
                }
            }
        }
    }
}
```

Slika 4 - WIP izgled geoprostornog upita nad Elasticsearch-om

4. Sistem za detekciju potencijalnih plagijarizama

Za detekciju potencijalnih plagijarizama će se koristiti *Plagiator* sa https://github.com/chenejac/plagiator, ali sa nekim izmenama:

- baza će biti promenjena u PostreSQL da odgovara ostatku sistema
- aplikacija će biti podešena da se tretira kao mikroservis, što uključuje njeno podešavanje kao Eureka client i rutiranje preko Zuul-a

- biće izbačeno prijavljivanje, registrovanje i bezbednost aplikacije i fokus će biti samo na detekciju plagijarizama[1]
- biće dodata integracija sa *LiterarySociety* mikroservisom, preko *FeignClient*-a, ako bude bilo potrebe

4.1. Integracija sa ostatkom sistema

Plagiator će biti mikroservis koji će primati radne verzije rukopisa i da nađe slične radove. Ovo se dešava prilikom izdavanje knjige od strane autora. Autor izvrši upload na *LiterarySociety* koji to prosledi *Plagiator*-u preko *FeignClient* (ili će se direktno vršiti na klijentskoj strani *Plagiator*-a). *Plagiator* vrati rezultat koji uključuje sve radove koji su previše slični primljenom rukopisu. Ove rezultate *LiterarySociety* ispiše uredniku i proces izdavanja knjige se nastavlja.

5. Zaključak

Sve što je ovde rečeno je početna ideja sistema i sve je podložno izmenama.

[1] Ako izbacivanje spomenutih delova izazove nepredviđene poteškoće, ostaviće se kako je bilo – ali će se registracija i prijavljivanje autora morati vršiti i na *Plagiator*-u, možda oni sami da se registruju i prijavljuju ili sistem da ih nekako automatski registruje pri registraciji na *LiterarySociety*.