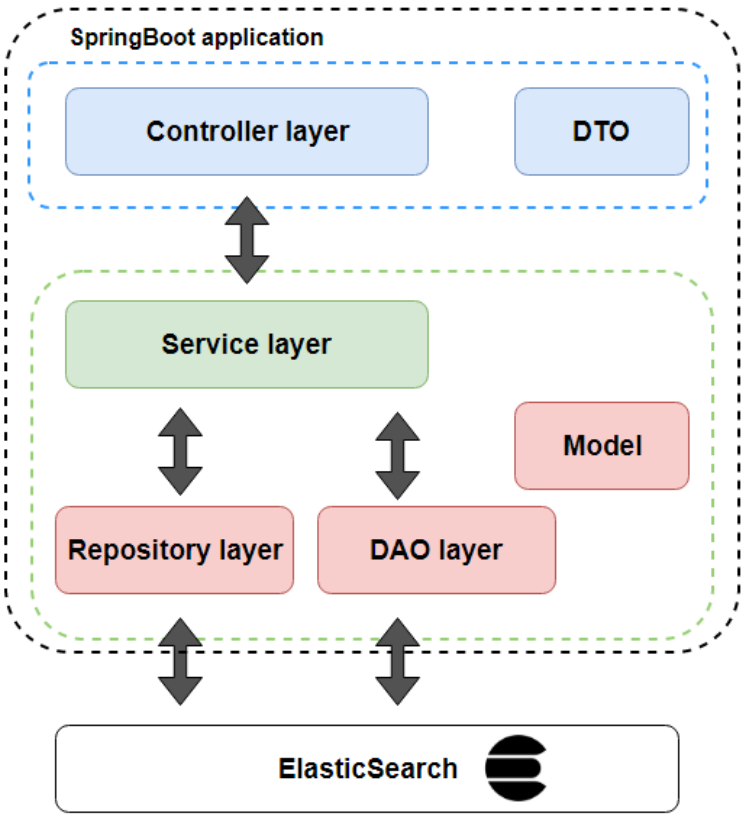
UDD - prva kontrolna tačka
Opis arhitekture projektne aplikacije

1. Uvod

U ovom radu će biti opisana arhitektura projektne aplikacije iz predmeta Upravljanje digitalnim dokumentima. Osnovna namjena aplikacije je čuvanje i pretraga radova iz naučnih časopisa. Aplikacija će biti izgrađena upotrebom SpringBoot radnog okvira. Čuvanje i pretraživanje podataka će se vršiti pomoću ElasticSearch-a. Integracija SpringBoot aplikacije i ElasticSearch engine-a će se vršiti pomoću Spring Data ElasticSearch modula Spring radnog okvira.

2. Arhitektura aplikacije

Na Slici 1 je prikazana slojevita arhitektura projektne aplikacije. Controller sloj prima HTTP zahtjeve i predstavlja sloj putem koga se pristupa aplikacije. U Service sloju se vrši sva poslovna logika i zahtijevane izmjene podataka, upotrebom Repository sloja. Model sloj sadrži klase koje modeluju entitete iz domena rješavanog problema, u našem slučaju entiteti su: autori, urednici, recenzenti, radovi, časopisi itd. DTO sloj sadrži klase koje su nastale na osnovu klasa Model sloja. DTO klase sadrže podskup atributa Model klasa i služe za razmjenu podataka između Controller-a i klijenata



Slika 1 Arhitektura projektne aplikacije

ElasticSearch engine u sebi sadrži NoSQL bazu u koju će biti smještani podaci. Podaci se smještaju u bazu u vidu JSON dokumenata, koji su nastali na osnovu klasa modela i njihovih atributa Povezivanje SpringBoot aplikacije ElasticSearch engine-om se vrši preko Repository i DAO sloja.

Repository sloj sadrži interfejse koji nasleđuju ElasticsearchRepostory interfejs iz Spring Data Elasticsearch modula. Na Slici 2 je prikazan interfejs UserRepositoy koji je definisan za klasu User. Ovako definisan interfejs nam automatski omogućava CRUD operacije nad dokumentima tip User.

```
@Repository
public interface UserRepository extends ElasticsearchRepository<User, String> {
}
```

Slika 2 Interfejs UserRepository

Klasa User mora biti anotirana na odgovarajući način kao što je prikazano na Slici 3.

```
@Document(indexName = "my_index", type = "user")
public class User {

    @Id
    @Field(type = FieldType.Long, index = FieldIndex.analyzed, store = true)
    private long id;
    @Field(type = FieldType.String, index = FieldIndex.analyzed, store = true)
    private String name;
    @Field(type = FieldType.String, index = FieldIndex.analyzed, store = true)
    private String surname;

    //getters and setters

}
```

Slika 3 Klasa User

Pored CRUD operacija, omogućene su i jednostavne pretrage kao što je prikazano na Slici 4.

```
public class UserService {

        @Autowired
        private UserRepository userRepository;

        public List<User> getAllUsers() {
            List<User> users = new ArrayList<>();
            userRepository.findAll().forEach(users::add);
            return users;
        }

}
```

Slika 4 Klasa UserService

Pored Repository sloja za komunikaciju sa ElasticSearch-om se koristi i klasa ElasticsearchTemplate koja naslijeđuje klasu ElasticsearchOperations. ElasticsearchTemplate klasa je moćnija od Repository interfejsa jer može da kreira i briše indekse, ali i da vrši agregacione pretrage. Za svaku od klasa modela se kreira jedna klasa DAO sloja koja, upotrebom ElasticsearchTemplate-a,

vrši složene pretrage nad dokumentima nastalim nad pomenutom klasom modela. Na Slici 5 je prikazana upotreba ElasticsearchTemplate -a.

```
public List<User> getAllUsers() {
    SearchQuery getAllQuery = new NativeSearchQueryBuilder()
            .withQuery(matchAllQuery()).build();
    return esTemplate.queryForList(getAllQuery, User.class);
}
```

Slika 5 Upotreba ElasticsearchTemplate-a

Da bi mogli koristiti ElasticsearchTemplate i Repository interfejse potrebno je definisati adresu i port na kojoj je pokrenut ElasticSearch engine i ime indeksa kome pristupamo unutar application.properties fajla.

3. JSON indexing unit i skladištenje podataka

Jedna od osnovnih funkcionalnosti projektne aplikacije je pretraga skupa prihvaćenih naučnih radova. Pored pretrage radova po meta podacima potrebno je omogućiti i full-text pretragu sadržaja radova. Da bi takva pretraga bila moguća neophodno je izvršiti indeksiranje nad cjelokupnim sadržajem naučnog rada.

```
{
    "metaData": [
      {"author":"Marko Marković"},
      {"language":"Serbian"},
      {"title":"Naučni rad"}
    ],
    "content":"Ovo je sadržaj naučnog rada!",
    "fileLocation":"/someFolder/file.pdf"
}
```

Slika 6 Pojednostavljeni indexing unit

Na Slici 6 je prikazan pojednostavljeni JSON indexing unit-a. Za svaki rad koji je potrebno sačuvati kreiraće se ovakav JSON objekat. JSON indexing unit ima tri polja. Polje metaData sadrži niz u kome se nalaze meta podaci rada kao što su: naziv, podaci o autoru, kategorija u koju rad spada, datum objavljivanja itd. Polje content sadrži tekstualni sadržaj rada, a polje fileLocation sadrži lokaciju na kojoj je sačuvana PDF verzija rada. Ovakav JSON objekat se zatim prosljeđuje Elasticsearch-u, koji ga čuva u svojoj NoSQL bazi, i vrši indeksiranje nad njime, radi omogućavanja pretrage. ElasticSearch-u se ne prosljeđuje PDF fajl rada, nego se PDF fajl čuva na lokaciji na koju pokazuje polje fileLocation. Nakon uspješne pretrage, ukoliko je korisniku potrebno poslijediti rad, rad se čita sa lokacije navedene u fileLocation polju i prosljeđuje korisniku.

Pored podataka o naučnim radovima u ElasticSearch-ovoj NoSQL bazi će se čuvati i podaci o: naučnim časopisima, urednicima, autorima, recenzentima i kategorijama časopisa. Za pretragu recenzenata po udaljenosti biti će iskorištene ugrađene ElasticSearch-ove operacije za geo pretrage, o čemu će viš riječi biti u nastavku.

4. MoreLikeThis funkcionalnost

Funkcionalnost MoreLikeThis podrazumijeva da prilikom izbora recenzenata za novi naučni rad uredniku budu ponuđeni recenzenti koji su recenzirali slične radove. Dakle, na osnovu teksta novog naučnog rada potrebno je pronaći slične naučne radove i njihove recenzente predložiti uredniku.

Funkcionalnost MoreLikeThis će biti realizovana pomoću ElasticSearch-ovog upita "more_like_this". Ovaj upit preuzima ulazni dokument ili tekst analizira ga i izabira termove koji najbolje opisuju taj ulazni dokument ili tekst. Zatim, te termove koristi u or upitu kako bi pronašao dokumente slične ulaznom dokumentu ili tekstu.

Na Slici 7 se nalazi JSON primjer "more_like_this" upita, a na linku je detaljno opisana upotreba ovog upita. Ovaj upit se može izvršavati po poljima koja su tipa text ili keyword , tako da u našem slučaju možemo iskoristiti ključne riječi ulaznog teksta kao upitni tekst i tražiti slične dokumente po polju ključne riječi. Nakon što pronađemo dokumente potrebno je još da formiramo listu recenzenata koji su recenzirali pronađene dokumente.

```
{
  "query": {
      "more_like_this" : {
          "fields" : ["title", "description"],
          "like" : [
          {
              "_index" : "imdb",
              "_type" : "movies",
              "_id" : "1"
          }
          ],
          "min_term_freq" : 1,
          "max_query_terms" : 12
      }
  }
}
```

Slika 7 Primjer "more_like_this" upita

5. Geoprostorna pretraga

Prilikom izbora recenzenata za novi naučni rad uredniku je potrebno omogućiti da filtrira recenzente, tako da ostanu samo oni čiji je grad udaljen najmanje 100km od gradova svih autora. Zbog potreba ove funkcionalnosti pored tekstualnog naziva adrese čuvaće se i geografska lokacija adrese(par geografska širina i geografska dužina) za svakog korisnika sistema.

Ova funkcionalnost će biti omogućena upotrebom ElasticSearch-ovog geo upita. Geografska lokacija recenzenata i autora će biti predstavljena poljima tipa „geo-point". A zatim će se koristiti „geo-distance" upit. Ovaj upit pronalazi dokumente sa tačkama koje se nalaze unutar kružnice, zadatog prečnika, opisane oko ulazne tačke. U našem slučaju ulaz će biti lokacija autora, a potrebno je pronaći recenzente koji su udaljeni najmanje 100km od autora. Dakle, prethodno navedeni upit bi nam vratio sve recenzente koji su udaljeni manje od 100km od autora, a nama treba upravo suprotno. Prethodni upit možemo staviti u „must_not" klauzu „bool" upita da bismo dobili tražene

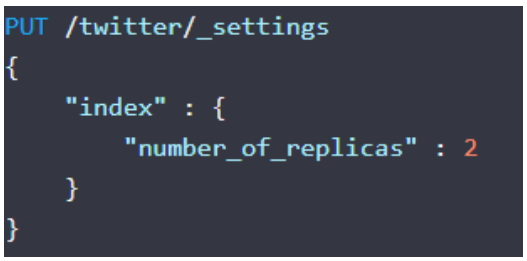
recenzente. Zatim, za svakog preostalog autora ponovimo upit i pronađemo presjek dobijenih rezultata. Taj presjek predstavlja skup recenzenata koji su udaljeni minimalno 100km od svakog od autora. Na Slici 8 se nalazi primjer „geo-distance“ upita a na linku su opisani svi mogući geo upiti i geo tipovi podataka.

Slika 8 Primjer "geo-distance" upita

6. Instaliranje analyzer-a za srpski jezik i konfiguracija ElasticSearch-a

Da bismo mogli pretraživati dokumente na srpskom jeziku pomoću ElasticSearch-a neophodno je instalirati specijalizovani analyzer za srpski jezik. Srpski analzyer vrši obradu ulaznog teksta: pretvara velika slova u mala, mijenja ćirilicu u latinicu, izbacuje stop riječi, stemuje riječi i zamjenjuje dj sa d. Analzyer koji će biti korišten se može preuzeti sa GitHub repozitorijuma dostupnog na linku, a instalacija će se vrši prema uputstvu iz README fajla navedenog repozitorijuma.

ElasticSearch će biti pokrenut nezavisno od SpringBoot aplikacije. Konfigurisanje ElasticSearch-a, odnosno podešavanje imena klastera, broja node-ova, broja shard-ova i broja replica-a će se vršiti izmjenom elasticsearch.yml fajla. Na linku je detaljno opisan pristup konfiguracionim fajlovima. Podešavanja ElasticSearch-a se mogu vršiti i preko REST API-ja. Na Slici 9 je prikazan zahtjev kojim se podešava broj replica-a twitter index-a.



Slika 9 Podešavanje putem REST API-ja