UNIVERZITET U NOVOM SADU
FAKULTET TEHNIČKIH NAUKA U NOVOM SADU

Milica Pranjkić E2 11/2020 – *Inteligentni sistemi*

Opis osnovne strukture i funkcionalnosti projektnog zadatka Literarno udruženje.

Novi Sad, 2020

1. Kratak uvod

Napomena: Kako sam student sa modula „inteligentni sistemi“ aplikacija će sadržati samo funkcionalnosti vezane za predmet „Upravljanje digitalnim dokumentima“.

Model podataka i osnovne funkcionalnosti prikazane unutar ovog dokumenta biće podložni promenama u procesu razvoja aplikacije. Dokument prikazuje opis pomenutih funkcionalnosti i korišćenih alata.

Aplikacija će biti višeslojna. Kao ulazna tačka biće korišćen jednostavan interfejs koji korisnicima omogućava izvršavanje funkcionalnosti vezanih za dodavanje i različite vrste pretrage knjiga. Na aplikativnom sloju biće implementirani kontroleri koji će vršiti obradu korisničkih zahteva. Aplikativni sloj će komunicirati sa *Elasticsearch*-om u cilju izvršavanja zahteva. Za određene zahteve koristiće *SerbianAnalyzer*, za pretprocesiranje upita. Ukoliko se ukaže potreba, biće dodat i sloj za perzistenciju podataka.

2. Model podataka

Kako je fokus predmeta na realizaciji različitih pretraga, kao primarni model podataka biće prezentovana samo klasa knjiga. Prema projektnoj specifikaciji, klasa bi trebala da se sastoji od dole navedenih polja. Međutim, moguće su određene izmene u toku razvoja aplikacije, ukoliko se ukaže potreba ili se ustanovi da bi promena doprinela unapređenju performansi.

Knjiga

- Naslov
- Pisci
- Žanr
- ISBN
- Ključni pojmovi
- Izdavač
- Godina izdavanja
- Mesto izdavanja
- Broj stranica
- Sinopsis
- Podaci o lektorisanju
- Urednici
- Sadržaj

3. Konfiguracija *elasticsearch*-a

Elasticsearch može biti pokrenut i korišćen na nekoliko različitih načina. U ovom dokumentu razmotrićemo dva načina pokretanja. U konkretnoj implementaciji biće upotrebljen onaj koji se ukaže kao najpodobniji u kombinaciji sa *SerbianAnalyzer* plugin-om.

Razmatrani način implementacije u konkretnoj aplikaciji jeste kombinacija *ElasticSearchTemplate* i *ElasticSearchRepository*.

3.1 Pokretanje u okviru aplikacije

Prilikom pokretanja *elasticsearch*-a u okviru aplikacije u dokumentu *properties* biće naznačene pojedinosti, neophodne za pokretanje servisa.

```
@Configuration
public class ElasticSearchConfiguration {
        @Value("${elasticsearch.host}")
        private String EsHost;
        @Value("${elasticsearch.port}")
        private int EsPort;
        @Value("${elasticsearch.clustername}")
        private String EsClusterName;
        public Client nodeClient() {
            Settings settings = Settings.builder()
                    .put("path.home", "data")
                    .build();
            final Node node = new NodeBuilder()
                    .settings(settings)
                    .local(true)
                    .build().start();
            return node.client();
        }
        @Bean
        public ElasticsearchOperations elasticsearchTemplate() {
            return new ElasticsearchTemplate(nodeClient());
        }
}
```

```
spring.data.elasticsearch.properties.http.enabled=true
elasticsearch.clustername = uns
elasticsearch.host = localhost
elasticsearch.port = 9200
```

3.2 Pokretanje kao nezavisne aplikacije

U slučaju nezavisnog pokretanja *elasticsearch*-a, pojedinosti definišemo prilikom svakog pokretanja, ili prilikom instalacije servisa. Moguće je proveriti aktivnost servisa gađanjem odabranog porta preko *browser*-a.


```
{
  "name" : "DESKTOP-0QBV8EA",
  "cluster_name" : "elasticsearch",
  "cluster_uuid" : "Pjhj1H_RRFq_sfPXzzJ2jQ",
  "version" : {
    "number" : "7.10.1",
    "build_flavor" : "unknown",
    "build_type" : "unknown",
    "build_hash" : "1c34507e66d7db1211f66f3513706fdf548736aa",
    "build_date" : "2020-12-05T01:00:33.671820Z",
    "build_snapshot" : false,
    "lucene_version" : "8.7.0",
    "minimum_wire_compatibility_version" : "6.8.0",
    "minimum_index_compatibility_version" : "6.0.0-beta1"
  },
  "tagline" : "You Know, for Search"
}
```

4. Pretraga knjiga

Na samom početku neophodno je kreirati indeks. Naime, indeks se može uporediti sa relacionom bazom podataka. Ima mapiranje, koje definiše više različitih tipova podataka. Ilustracija 1. prikazuje kreiranje indeksa *books*.

```
POST /project_index/books
{
  "mappings":
  {
    "books":
    {
      "properties":
      {
        "Title": {"type": "text"},
        "Writer": {"type": "text"},
        "Genre": {"type": "int"},
        "ISBN": {"type": "text"},
        "KeyWords": {"type": "text"},
        "PublishingYear": {"type": "date"},
        "PublishingPlace": {"type": "text"},
        "PageNumber": {"type": "int"},
        "Synopsis": {"type": "text"},
        "LecturingData": {"type": "text"},
        "Editor": {"type": "text"},
        "Content": {"type": "text"}
      }
    }
  }
}
```

Ilustracija 1.

Zatim sledi postupak dodavanje podataka, prikazan na Ilustraciji 2.

```
POST /project_index/books
{
  "Title": "Some title",
  "Writer": "William Shakespeare",
  "Genre": ["Genre1", "Genre2"],
  "ISBN": "1-23-456789-1",
  "KeyWords": ["Key word 1", "Key word 2", "Key word 3"],
  "PublishingYear": 1990,
  "PublishingPlace": "New York",
  "PageNumber": 982,
  "Synopsis": "A brief summary.",
  "LecturingData": "Some lecturing data.",
  "Editor": ["Edithor1", "Edithor 2"],
  "Content": "The fool doth think he is wise, but the wise man knows himself to be a fool."
}
```

Ilustracija 2

Ilustracija 3. prikazuje zahteve pretrage elasticsearch algoritmom. Naime, prva i druga linija prikazuju postupak pretrage podataka prema navedenom imenu ili prezimenu pisca. Druga linija prikazuje postupak pretrage po željenom žanru knjige.

```
GET /project_index/books/_search?q=Writer:Agatha
GET /project_index/books/_search?q=Writer:Christie

GET /project_index/books/_search?q=Genre:Genre1
```

Ilustracija 3

Ilustracije 4. i 5. prikazuju pretragu podataka kombinacijom gore navedenih parametara pretrage. Formiran je tzv. *BooleanQuery* koji omogućava upotrebu logičkih operacija *AND* ili *OR* nad parametrima pretrage.

Na ilustraciji 4. prikazan je zahtev koji predstavlja primenu logičke operacije *AND* nad unetim parametrima pretrage. Podaci koji pristignu kao rezultat pretrage zadovoljavaju oba parametra pretrage koje korisnik unese.

```
GET /project_index/books/_search
{
  "query":
  {
    "bool":
    {
      "must": [
        {"query_string": {"query": "Writer:Agatha"}},
        {"query_string": {"query": "Genre:Genre2"}}
      ]
    }
  }
}
```

Ilustracija 4

Na ilustraciji 5. prikazan je zahtev koji predstavlja primenu logičke operacije *OR* nad unetim parametrima pretrage. Podaci koji pristignu kao rezultat pretrage zadovoljavaju barem jedan od unetih parametra pretrage, a mogu i oba.

```
GET /project_index/books/_search
{
  "query":
  {
    "bool":
    {
      "should": [
        {"query_string": {"query": "Writer:Agatha"}},
        {"query_string": {"query": "Genre:Genre2"}}
      ]
    }
  }
}
```

Ilustracija 5

Ilustracija 6. prikazuje realizaciju mogućnosti zadavanja *PhrazeQuery*-a za ime i prezime pisca. Izlistani rezultati će biti samo oni koji sadrže zadato ime i prezime, tačno u tom redosledu.

```
GET /project_index/books/_search
{
  "query":
  {
    "match_phrase":
    {
      "Writer":
      {
        "query": "Agatha Christie"
      }
    }
  }
}
```

Ilustracija 6

Ilustracija 7. prikazuje kreiranje dinamičkog zažetka prilikom prikaza rezultata, tzv. *Highlighter*-a. Markeri omogućavaju dobavljanje istaknutih isečaka iz jednog ili više polja u rezultatima pretrage, u svrhu prikazivanja korisniku mesto gde se podudaraju s upitom.

```
GET /project_index/books/_search
{
  "_source": ["highlight"],
  "query":
  {
    "match_phrase":
    {
      "Genre":
      {
        "query": "Genre1"
      }
    }
  },
  "highlight":
  {
    "pre_tags": ["<em>"],
    "post_tags": ["</em>"],
    "tags_schema": "styled",
    "fields": {"Genre": {} }
  }
}
```

Ilustracija 7

Ilustracija 8. prikazuje postupak indeksiranja novog *PDF* fajla. Pamte se osnovne informacije vezane za objetak, a zatim se konstruiše *DocumentHandler,* konkretno *PDFDocHandler* (ilustracija 9), koji vrši parsiranje dokumenta na odgovarajući način.

```
private void indexUploadedFile(UploadModel model) throws IOException{

    for (MultipartFile file : model.getFiles()) {

        if (file.isEmpty()) {
            continue; //next please
        }
        String fileName = saveUploadedFile(file);
        if(fileName != null){
            IndexUnit indexUnit = indexer.getHandler(fileName).getIndexUnit(new File(fileName));
            indexUnit.setTitle(model.getTitle());
            indexUnit.setKeywords(model.getKeywords());
            indexer.add(indexUnit);
        }
    }
}
```

Ilustracija 8

```
public DocumentHandler getHandler(String fileName){
    if(fileName.endsWith(".txt")){
        return new TextDocHandler();
    }else if(fileName.endsWith(".pdf")){
        return new PDFHandler();
    }else if(fileName.endsWith(".doc")){
        return new WordHandler();
    }else if(fileName.endsWith(".docx")){
        return new Word2007Handler();
    }else{
        return null;
    }
}
```

Ilustracija 9

Ilustracija 10. prikazuje kako se iz *PDF* dokumenta preuzima tekstualni sadržaj a zatim se smešta u promenljivu u okviru podatka za indeksiranje.

```
public String getText(File file) {
    try {
        PDFParser parser = new PDFParser(new RandomAccessFile(file, "r"));
        parser.parse();
        PDFTextStripper textStripper = new PDFTextStripper();
        String text = textStripper.getText(parser.getPDDocument());
        return text;
    } catch (IOException e) {
        System.out.println("Greksa pri konvertovanju dokumenta u pdf");
    }
    return null;
}
```

Ilustracija 10

Kao posledica obrade ulaznog dokumenta, korišćenjem koda sa prethodne tri ilustracije, dobija se sadržaj koji može da se pretražuje. Ilustracija 11. prikazuje zahtev za pretragu dokumenata prema sadržaju.

```
GET /project_index/books/_search?q=Content:knows
```

Ilustracija 11

5. Pretprocesiranje upita pomoću *SerbianAnalyzer*-a

Plugin koji obavlja pretprocesiranje srpshih tekstova biće preuzet sa *github*-a i integrisan unutar projekta kao na ilustraciji 8.

```
import rs.ac.uns.ftn.informatika.udd.vezbe01.lucene.indexing.analysers.SerbianAnalyzer;
```

Ilustracija 12

Njegova primena biće implementirana slično kao u pokaznim primerima sa vežbi. Prilikom realizacije *PhrazeQuery* pretrage. Naime, *QueryParser* će koristiti *SerbianAnalyzer* prilikom parsiranja ulaznog parametra pretrage.

```
@PostMapping(value="/search/queryParser", consumes="application/json")
public ResponseEntity<List<ResultData>> search(@RequestBody SimpleQuery simpleQuery) throws Exception {
    QueryParser qp=new QueryParser("title", new SerbianAnalyzer());
    Query query=qp.parse(simpleQuery.getValue());
    List<RequiredHighlight> rh = new ArrayList<RequiredHighlight>();
    List<ResultData> results = ResultRetriever.getResults(query, rh);
    return new ResponseEntity<List<ResultData>>(results, HttpStatus.OK);
}
```

Ilustracija 13

6. Geoprostorna pretraga

Geoprostorna pretraga u okviru projekta koristi se za proveru ispunjenosti uslova udaljenosti beta-čitaoca od pisca koji izdaje knjigu. Prilikom kreiranja klase pisca, biće dodata polja koja će naznačavati njegovu konkretnu lokaciju (*latitude* i *longitude*). Zatim, kada se bude ispitivala ispunjenost uslova udaljenosti, izvršiće se provera u kojoj će biti ispitano da li je uslov udaljenosti od 100km zadovoljen. Upit koji vraća pisce na manjoj udaljenosti od 100km od zadate lokacije prikazan je ilustracijom 14.

```
GET zipgeo/_search
{
  "query":
  {
    "bool":
    {
      "filter":
      {
       "geo_distance": {
        "distance": 100,
        "distance_unit": "km",
        "FIELD": {
         "lat": 40.73,
         "lon": -74.1
        }
       }
      }

    }
  }
}
```

Ilustracija 14

7. Sistem za detekciju potencijalnih plagijarizama

Pre samog izdavanja knjige, one se šalju beta-čitaocima na pregled kako bi utvrdili potencijalni plagijarizam, a zatim ga i prijavili. Kako bi se to moglo implementirati sa *github*-a će biti preuzet projekat plagiator, biće pokrenut kao zasebna aplikacija i primarna aplikacija za dodavanje i pretragu knjiga će ga kontaktirati po potrebi.

Ukoliko beta-čitalac posumnja da je neki dokument potencijalni plagijarizam, on ga šalje amdinistratorima na pregled. Administrator proverava sve, potencijalno plagirane, dokumente (ilustracija 15) a zatim donosi konačnu odluku, tj. da li dokument jeste ili nije plagijat.

```
public ResponseEntity<List<PaperDTO>> getPapers(Pageable page, HttpServletRequest request) {
    List<Paper> papers = paperService.findAll();
    List<PaperDTO> papersDTO = new ArrayList<PaperDTO>();

    Principal principal = request.getUserPrincipal();
    String email = principal.getName();
    User logged = userService.findByEmail(email);

    for(Paper paper: papers) {
        PaperDTO paperDTO = objectMapper.map(paper, PaperDTO.class);
        PaperResultPlagiator plagiator = paperResultPlagiatorService.findByUploadedPaperId(paper.getId());

        if(plagiator != null) {
            //nije od tog korisnika pa ne moze da ga vidi i korisnikova uloga nije admin
            if(!logged.getRole().getUserType().equals(RoleConstants.ROLE_ADMIN) //ako je admin nek prodje
                    && !plagiator.getUploadedPaper().getUser().getEmail().equals(email)) { //nisi ga ti uplodovao
                paperDTO.setPlagiatorId(null);
            }
            else { //inace moze da mu pristupi da vidi pregled. To znaci da postoji i da ima privilegiju.
                paperDTO.setPlagiatorId(plagiator.getId());
            }
        }
        else {
            paperDTO.setPlagiatorId(null);
        }
        papersDTO.add(paperDTO);
    }
    return new ResponseEntity<>(papersDTO, HttpStatus.OK);
}
```

Ilustracija 15