Управљање дигиталним документима – контролна тачка

Немања Јанковић, R2-13/2020

Архитектура апликације

Апликација ће бити развијена пратећи често имплементирану трослојну архитектуру система. Заснована је на презентационом, апликативном и слоју података који међусобно комуницирају.

Презентациони слој ће бити рађен као засебна, *client side rendered web* апликација, урађена ослањајући се на *Angular* радни оквир. Сва корисничка интеракција ће ићи путем овог слоја.

Апликативни слој ће бити рађен као *Java* апликација, ослањајући се на *Spring Boot* радни оквир. Овај слој ће имплементирати пословну логику потребну за исправан рад система.

Слој података ће користи *MySQL* базу за потребне складиштења и добављања података. Складиштење самих књига у изворној дигиталној форми ће бити рађено на диску, при чему ће њихова путања бити запамћена у бази. Претрага сачуваних књига ће бити одрађена користећи *Elasticsearch* софтверско решење. Пре саме претраге, сви елементи битни за њу ће бити индексирани[1].

Слојеви међусобно комуницирају ослањајући се на REST[2] принципе. За интеграцију система са деловима везаним за Elasticsearch, биће коришћена *Spring Data Elasticsearch* [3] библиотека, добијена путем *Apache Maven* алата који се уједно користи и за управљање *build* процесом апликације. За потребе пројекта, ова библиотека ће омогућити лакшу REST API [4] комуникацију са покренутом инстанцом Elasticsearch софтвера, пружајући олакшице као што су анотације приликом индексирања докумената [5], јасно дефинисан интерфејс приликом операција [6], репозиторијуме у којима ће бити складиштена индексирана обележја [7] и друго.

Конфигурација ElasticSearch-а

Пројекат ће користи ElasticSearch верзију 7.4.0 као што се захтева у *SerbianAnalyzer* плугину о којем ће више речи бити касније. Инстанца Elasticsearch-а ће бити покренута на портовима 9200 и 9300. Први се користи за комуникацију путем HTTP API захтева а други за бинарну, *Transport*, комуникацију између *node*-ова у кластеру [8]. Унутар апликативног слоја, ова комуникација може да

[1] https://www.elastic.co/guide/en/elasticsearch/reference/7.4/getting-started-index.html#getting-started-index
[2] https://restfulapi.net/
[3] https://docs.spring.io/spring-data/elasticsearch/docs/current/reference/html/#reference
[4] https://docs.spring.io/spring-data/elasticsearch/docs/current/reference/html/#elasticsearch.clients.rest
[5] https://docs.spring.io/spring-data/elasticsearch/docs/current/reference/html/#elasticsearch.mapping.meta-model.annotations
[6] https://docs.spring.io/spring-data/elasticsearch/docs/current/reference/html/#elasticsearch.operations.usage
[7] https://docs.spring.io/spring-data/elasticsearch/docs/current/reference/html/#elasticsearch.repositories
[8] https://discuss.elastic.co/t/what-are-ports-9200-and-9300-used-for/238578

се врши позивањем ElasticsearchRestTemplate [9] за REST комуникацију или ElasticsearchTemplate [10] за бинарну комуникацију. Комуникација ће бити одрађена на први начин.

Node [11] представља покренуту инстанцу Elasticsearch-а, при чему свака инстанца припада једном кластеру [12]. Индексирани подаци се дистрибуирано деле кроз ове инстанце како би перформансе операција над тим подацима биле што боље. Сваки индекс се састоји из једног или више *shard*-ова, инстанци *Lucene* индекса. Такође постоје реплике ових shard-ова, који нису алоцирани на исте машине одакле је изворни, *primary shard* [13].

За основну изведбу пројектног задатка, једна инстанца ће бити довољна али је потребно имати у виду да је ово потребно скалирати како расте број података које је потребно индексирати и претраживати као и која хардверска захтевност је потребна за комфоран рад [14].

Да би се успоставила правилна комуникација између Elasticsearch-а и Spring Boot апликације, горепоменуту Spring Data Elasticsearch библиотеку је, након скидања са Maven репозиторијума, потребно конфигурисати користећи *ClientConfiguration* класу те библиотеке [15]. Унутар ње се уносе адресе преко којих се комуницира, задају сигурносни параметри аутентификације, наводи да ли се користи *TLS* протокол и слично.

Након тога је потребно додати анотације на класе модела, који ће бити коришћени за потребе мапирања мета модела Java објеката, ентитета домена пројекта, у JSON документе који ће бити складиштени при Elasticsearch инстанцама[16]. Овако додате анотације специфицирају начине репрезентација ових докумената, типове атрибута које ће они имати и начине на које ће се индексирати. За потребне пројекта, ове анотације се додају за кориснике (писце и читаоце) и књиге које су издате или су у процесу издавања.

Приступ складиштењу и извршавање операција над документима унутар њих омогућено је постављањем Spring Data Elasticsearch репозиторијума. Комуникацијом са њима, даље ће бити упућени сви упити, направљени имплицитно (на пример „List<Book> findByName(String name);“) или експлицитно, коришћењем *@Query* анотације [17].

---

[9] https://docs.spring.io/spring-data/elasticsearch/docs/current/reference/html/#elasticsearch.operations.resttemplate
[10] https://docs.spring.io/spring-data/elasticsearch/docs/current/reference/html/#elasticsearch.operations.template
[11] https://www.elastic.co/guide/en/elasticsearch/reference/7.4/modules-node.html
[12] https://www.elastic.co/guide/en/elasticsearch/reference/7.4/modules-cluster.html
[13] https://www.elastic.co/guide/en/elasticsearch/reference/7.4/scalability.html
[14] https://www.elastic.co/blog/how-many-shards-should-i-have-in-my-elasticsearch-cluster
[15] https://docs.spring.io/spring-data/elasticsearch/docs/current/reference/html/#elasticsearch.clients.configuration
[16] https://docs.spring.io/spring-data/elasticsearch/docs/current/reference/html/#elasticsearch.mapping.meta-model.annotations
[17] https://docs.spring.io/spring-data/elasticsearch/docs/current/reference/html/#elasticsearch.repositories

SerbianAnalyzer plugin

Ово је додатак на постојеће сервисе Elasticsearch-а који додаје могућност обраде текстова на српском језику. Налази се на репозиторијуму: https://github.com/chenejac/udd06 . Писан је за Java верзију 13.0.1, Gradle верзију 6.0 и Elasticsearch верзију 7.4.0.

Кораке његове инсталације могуће је испратити са истоименог репозиторијума. Захтевају *build*-овање пројекта помоћу *Gradle* алата, након чега га је могуће инсталирати локално, користећи Elasticsearch метод инсталирања plugin-ова задајући апсолутну путању на којој се софтвер налази.

Индексне структуре (*Indexing units)*

Жељена индексна структура за писце је представљена листингом 1.

```
{
   "id": text,
   "name": text,
   "surname": text,
   "location": geo_point,
   "genresInterestedIn": text[]
}
```

Листинг 1 - Индексна структура писца

Жељена индексна структура за читаоце је представљена листингом 2. У поређењу са писцем додата су поља која означавају да ли је читалац уједно и бета читалац и које за које жанрове је заинтересован да оцењује књиге. Укинута су поља имена и презимена јер претрага по овим пољима не доприноси никаквим битним ставкама за пројектни задатак.

У тренутној ставци задатка једина битна претрага по читаоцима је она везана за бета читаоце. При изради решења остављено је индексирање свих читалаца јер су ово подаци који ипак могу бити касније од употребе. Пример претпостављене претраге би био: „регуларном читаоцу предложи писце који потичу из његовог места из жанрова за који је заинтересован“.

```
{
   "id": text,
   "location": geo_point,
   "genresInterestedIn": text[],
   "betaReader": boolean,
   "betaReaderGenresInterestedIn : text[]
}
```

Листинг 2 - Индексна структура читалаца

Индексна структура књига дата је листингом 3. Поља “author” и “coauthors” су аутор структуре, приказани раније. Поље „price” има облик *scaled_float* како би се избегле непрецизности. [18]

```
{
   "id": text,
   "title": text,
   "author": object,
   "coauthors": object[],
   "genres": text[],
   "filePath": text,
   "content": text,
   "openAccess": boolean,
   "price": scaled_float,
   "currency": text
}
```

Листинг 3 - Индексна структура књиге

Имплементација обавезних ставки пројекта

Претраживање књига по наслову

Може се извршити директно прављењем имплицитног упита на нивоу репозиторијума. Листинг 4 приказује пример таквог упита[19].

```
public interface BookElasticSearchRepository extends ElasticsearchRepository<Book, String> {
   Page <Book> findByTitleLike(String title, Pageable pageable);
}
```

Листинг 4 - Пример имплицитног упита на нивоу репозиторијума за претрагу по наслову

Претраживање књига по именима и презименима писаца

Може се извршити директно прављењем експлицитног упита на нивоу репозиторијума. Такви експлицитни упити су означени са кључном речи @Query која анотира неку методу[17].

[18] https://www.elastic.co/guide/en/elasticsearch/reference/current/number.html#_which_type_should_i_use
[19] https://docs.spring.io/spring-data/elasticsearch/docs/current/reference/html/#elasticsearch.query-methods.criterions

Претраживање књига по садржају (из PDF фајла)

Може се извршити директно прављењем имплицитног упита на нивоу репозиторијума. Листинг 5 приказује пример таквог упита.

```
public interface BookElasticSearchRepository extends ElasticsearchRepository<Book, String> {

   Page<Book> findByContentContains(String query, Pageable pageable);

}
```

Листинг 5- Пример имплицитног упита на нивоу репозиторијума за претрагу по садржају

Претраживање књига по жанровима

Може се извршити директно прављењем имплицитног упита на нивоу репозиторијума. Листинг 6 приказује пример таквог упита.

```
public interface BookElasticSearchRepository extends ElasticsearchRepository<Book, String> {

   Page<Book> findByGenreContains(String genre, Pageable pageable);

}
```

Листинг 6 - Пример имплицитног упита на нивоу репозиторијума за претрагу по жанровима

Комбинација претходних параметара претраге (BooleanQuery)

Може се извршити ручно, коришћењем *CriteriaQuery* [20] или применом решења попут *QueryDSL* [21] путем кога ће се динамички генерисати потребан упит, у зависности од комбинација које су унете од стране корисника. У свим случајевима посебно пажњу је потребно посветити корисничком интерфејсу, где опције претраге морају бити представљене на једноставан начин, без тога да превише закомпликују интерфејс.

Претрага фраза (PhrazeQuery) по свим пољима

Потребно је користити *match_phrase* [22] упит. Овај упит проналази документа који садрже унету фразу у оном реду у коме су њени упити унети након примене анализатора, за разлику од класичног "query_string" упита[23]. Може да се контролише са параметром „slop" који одређује која удаљеност термина може да буде присутна да би ти термини и даље чинили једну фразу.

Такође се може градити ручно, коришћењем CriteriaQuery-ја или применом QueryDSL решења. И овде је посебну пажњу потребно посветити корисничком интерфејсу.

[20] https://docs.spring.io/spring-data/elasticsearch/docs/current/reference/html/#elasticsearch.operations.queries
[21] https://www.elastic.co/guide/en/elasticsearch/reference/current/query-dsl.html
[22] https://www.elastic.co/blog/phrase-Queries-a-world-without-stopwords
[23] https://www.elastic.co/guide/en/elasticsearch/reference/7.4/query-dsl-query-string-query.html

Претпроцесирање упита помоћу SerbianAnalyzer-а

Након инсталације, коришћење овог analyzer-а је, у пројекту могуће користити анотирајући текстуална поља ентитета да користе овај анализатор текста при раду са Spring Data Elasticsearch библиотеком. Пример такве анотације био би листинг 7.

```
@Field(type = FieldType.Text, searchAnalyzer = "serbian", analyzer = "serbian")
private String name;
```

Листинг 7 - Пример анотираног текстуалног поља који користи SerbianAnalyzer

Приказивање динамичког сажетка приликом приказа резултата

За потребе приказивања динамичког сажетка, решавање овог проблема могуће је коришћењем Elasticsearch highlighting опције[24] која ће вратити сажетак на основу задате претраге. Ово је могуће урадити јер се при индексирању докумената унутар Elasticsearch-а чува и њихов текстуални садржај, добијен прерадом кориснички унетог PDF документа.

Приказивање различитих линкова за преузимање књиге, директно преузимање у open-access моду или линк ка сајту за куповину ако није

При индексирању књига, памти се ово поље. Из тог разлога, за било какву претрагу, и овај потребан податак ће бити враћен. Након тога, довољно је проверити на презентационом слоју вредност која је враћена и на основу тога презентовати опцију која је погодна. Наравно, потребно је на апликативном нивоу одрадити проверу да ли корисник има дозволу скидања фајла или не, како би се избегао сценарио у ком корисници директно позивају API сервис који им фајл враћа без икакве провере.

Имплементација додатних ставки пројекта

Геопросторна претрага

Задатак, у ставци 2.2.3 тачки 7, налаже да се, у процесу издавања књиге, прихваћени рукопис још увек неиздате књиге шаље бета читаоцима, који рукопис треба да коментаришу. Додатан услов који треба решити налаже да се рукопис може проследити само бета читаоцима који су удаљени више од 100 километара у односу на аутора рукописа. Уреднику је потребно доставити листу бета читалаца који су заинтересовани за жанр рукописа и који су довољно удаљени.

Ову геопросторну претрагу је могуће реализовати користећи механизме пружене Elasticsearch програмом. Претрага се ослања на два уграђена типа поља: [25]

1) *geo_point* – поље које представља парове географских ширина и дужина, чинећи основу приказа координата
2) *geo_shape* – поља која представљају произвољне географске облике приказане помоћу различитих геометријских облика, попут линија и полинома

[24] https://www.elastic.co/guide/en/elasticsearch/reference/current/highlighting.html
[25] https://www.elastic.co/guide/en/elasticsearch/reference/7.4/geo-queries.html

За потребе пројекта, могуће је искористи *geo-distance* упите који раде над овим типовима поља, који претражују вредности горе приказаних поља у простору око произвољно задатих координати. [26] Приказ оваквог овако задатог упита дат је на листингу 8.

```
GET /readers/_search

{
  "query" : {
    "bool" : {
      "must_not" : {
        "geo_distance" : {
          "distance" : "100km",
          "author.location" : { "lat" : number, "lon" : number }
        }
      }
    }
  }
}
```

Листинг 8 – Геопросторна претрага

Претрага се заснива на коришћењу "must_not" bool упита који ће вратити све читаоце изван траженог опсега од 100 километара у односу на аутора, који се налази на неким произвољним координатама означеним као number (произвољна, нумеричке вредности, права вредност у реалном раду).

Овакво коришћење упита захтеве претходно индексирање података. Ови подаци се односе на локацију корисника система, читалаца и аутора. Добијање ових вредности вршиће се током регистрације корисника, по уносу њихове адресе. Комуникацијом са неким од доступних геопросторних сервиса попут Geocoding API-ја компаније Google[27], апликација ће натраг добити информације од координатама унесене корисничке адресе у виду JSON структуре које садржи пар вредности географску ширину и дужину.

Интеграција са „Plagiator" системом

Апликација користи услуге пружене Plagiator системом како би се детектовали плагијати рукописа књига у процесу издавања. Систем се налази на линку: https://github.com/chenejac/plagiator . Ово је *backend* система који је довољан за интеграцију. Одрађен је у виду Java Spring Boot апликације.

Пре покретања изградње пројекта, потребно је унети следеће измене у *application.properties* фајлу:

1) Унети потребне податке за приступање базе (адресу, корисничко име, шифру)

[26] https://www.elastic.co/guide/en/elasticsearch/reference/7.4/query-dsl-geo-distance-query.html
[27] https://developers.google.com/maps/documentation/geocoding/start

2) По потреби заменити H2 базу за неку другу
3) Ако је H2 база остала, по потреби заменити адресу у фајл система са кога се покреће
4) По потреби такође заменити адресу инстанце Elasticsearch-а и порт на коме ослушкује саобраћај
5) Заменити емаил адресу на коју се шаљу поруке
6) По потреби, заменити порт на којој апликација ради

За покретање система потребно је га је скинути са репозиторијума и користећи Maven алат изградити потребан систем из изворног кода. Овим ће се уједно скинути и сви dependency пакети потребни за пројекат. Овај процес је могуће одрадити коришћењем команди “mvn clean package”, након чега ће се креирати *.jar* архива унутар *target* директоријума који је потребно покренути.

Након покретања апликације, потребно јој је приступити на адреси на којој је покренута, и путем HTTP захтева, послати захтеве за регистрацију (“api/signup”) и верификацију корисника након ње (“api/registration/activate/{userId}”, при чему се идентификатор корисника добија на унети емаил налог при регистрацији). Овим се добија кориснички налог преко кога је могуће извршити остале функционалности.

Plagiator систем се користи тако што се при већању о прихватању нове књиге њен садржај пошаље на “api/file/upload/new” API endpoint, након чега ће бити извршено снимање садржаја на диску и његово индексирање кроз Elasticsearch-а. Након тога, покреће се процес евалуације плагијаризма који враћа податке о томе у којој мери је садржај плагијаризован, односно, враћа листу сличних, претходно снимљених, радова и делова текста који су слични. За покретање овог сервиса, потребно је бити улогован на претходно креирани налог (“api/login”).

Процес одлуке о плагијаризму зависи од броја претходно унетих радова те ће се са већом количином унетих радова добијати боља прецизности.