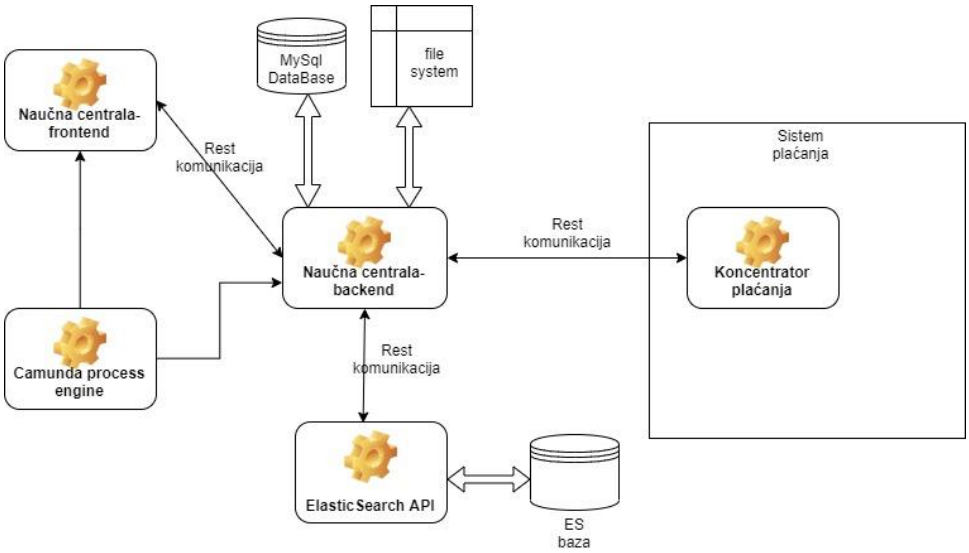
Upravljanje digitalnim dokumentima – kontrolna tačka

Arhitektura sistema

Arhitektura sistema za podršku rada naučne centrale je zamišljen na sledeći način:



Slika 1.

U principu sistem se sastoji od dva nezavisna sistema: Sistem naučne centrale i Koncetrator plaćanja. Pošto nam je za ovaj kurs koncentrator plaćanja irelevantan, preskočićemo njegov detaljniji opis.

Sistem naučne centrale (NC API) je zamišljena kao *web* aplikacija, sa troslojnom arhitekturom (sloj podataka, prezentacioni sloj, sloj poslovne logike). Kao što vidimo na slici 1, koristićemo različite tehnologije za realizaciju ovog projekta. Za realizaciju front-end aplikacije koristićemo *Angular 8* framework, baziran na TypeScript jeziku. Back-end će biti realizovan uz pomoć *Maven* build alata, kao *Java SpringBoot* aplikacija. Za pretragu radova koristićemo *ElasticSearch API.* Pretraživanje radova će biti moguće zadavanjem različitih parametara(naslov, autor, oblast itd.). Detaljniji opis pretrage i komunikacije između modula sledi u nastavku. *Camunda process engine* koristimo za potrebe kursa *"Upravljanje poslovnim procesima" za o*rekstraciju različitih procesa. Detaljniji opis je nebitan za ovaj kurs. Podatke ćemo čuvati na tri različita mesta. Same radove tj. *PDF* fajlove, ćemo čuvati u lokalnom fajl sistemu. Podaci o korisnicima, časopisima, radovima, metapodatke radova kao i putanju tj. lokaciju do samih radova (*PDF* fajlova) čuvaćemo u relacionoj bazi podataka gde ćemo kao server koristiti MySQL. Kako bi se olakšala komunikacija između relacione baze i back-end koristićemo JPA i Hibernate za relaciono mapiranje. Koordinaciju između svih ovi baza vršiće jedan DAO servis.

Konfiguracija ElasticSearch-a

ElasticSearch predstavlja *engine* za pretraživanje, razvijen u Java programskom jeziku. ElasticSearch je *RESTful API,* što znači putem HTTP zahteva možemo dodataviti nove podatke i zadatavati upite.

Pojam konfiguracije ElasticSearch-a, između ostalog, podrazumevao bi i konfigurisanje "elasticsearch.yml" fajla. Tu možemo da konfigurišemo različite stvari poput: klastera, čvorove (nodes), broj *shard*-ova i replika, veličine *heap-a,* port na kom API trči itd. Kada bi radili komercijalni projekat naučne cetrale, ovi parametri konfiguracije bi se svakako uzimali u razmatranje jer pravljanjem balansa između broja *shard-ova i* replika i drugih parametara dobijamo optimalne

performanse pretraživanja prema potrebama klijenta. U našem projektu naučne centrale, ovi parametri su podešeni na podrazumevane vrednosti, jer zadovoljavaju potrebe projekta. Potrbno je navesti da je podrazumevan port na kom ElasticSearch API trči je 9200.

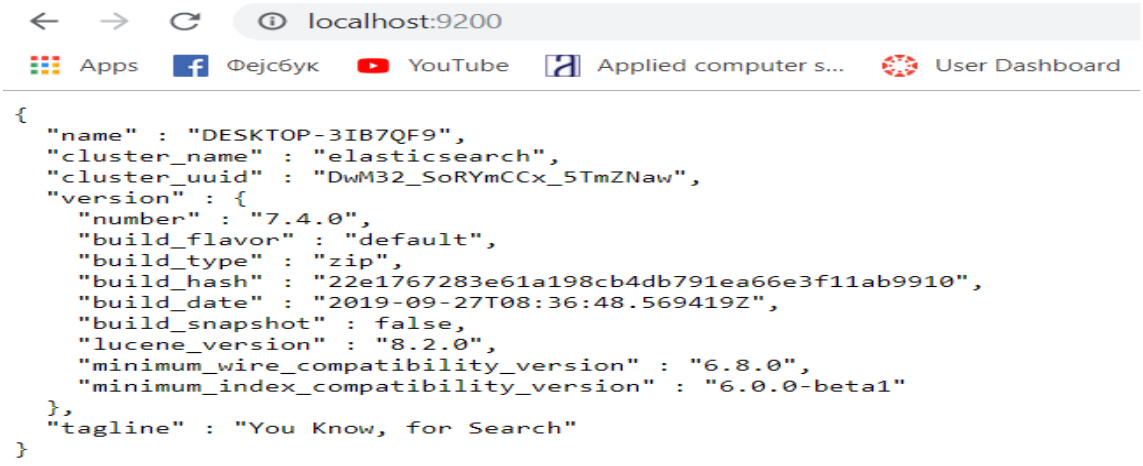
Jedine konkretne promene izvršene u konfiguraciju su u putanjama za podatke i logove(path.data i path.logs). Gledajući razne *YouTube* tutorijale, došli smo do zaključka da je ovo dobra praksa iz razloga čuvanja podatakau slučaju iznenadnog brisanja ElasticSearch-a.

Što se tiče pokretanja , uradili smo sledeće:

1)Sa oficijalnog sajta ElasticSearch-a skinuli smo verziju 7.4.0

2)Raspakovali .zip i iz *comand promt-a* pokrenuli *elasticsearch* datoteku unutar ./bin direktorijuma

3)Kucanjem adrese *localhost:9200* možemo proveriti da li nam je ElasticSearch pokrenut (slika 2)



Slika 2.

Komunikacija između komponenti

Front-end i back-end komuniciraju putem REST tehnologije (representational state transfer).Pošto je ElasticSearch RESTful API, takođe i ovde možemo uspostaviti klasičnu REST komunikaciju između ElasticSearch-a i back-end, gde bismo putem POST i GET metoda pravili nove dokumente i vršili upite nad njima. Međutim, kako bi se olakšala komunikacija sa ElasticSearch-om i pojednostavilo indeksiranje, pisanje upita i CRUD operacija koristićemo SpringData Elasticsearch. Ovo podrazumeva dodavanja nekoliko *dependency-ija* u pom fajl back-end projekta i dodatnu konfiguraciju. Detaljan prikaz konfiguracije pronašli smo prilikom istaživanja za ovu kontrolnu tačku na sledećim linkovima:

1)https://www.baeldung.com/spring-data-elasticsearch-tutorial

2)https://dzone.com/articles/spring-boot-elasticsearch

Uz oslonac na sadržaj ova dva linka, konfigurisaćemo i naš back-end i na ovakav način indeksirati dokumente i vršiti upite.

U sledećih nekoliko rečenica opisaćemo tok komunikacije priliko pretrage. Korisnik putem fron-end aplikacije unosi parametre za pretraživanje, koji se potom šalju na back-end aplikaciju. U back-end se formira upit koji se prosleđuje ElasticSearch-u. ElasticSearch vraća rezltate pretrage back-end-u a on front-end gde se rezultati pretrage prezentuju korisniku koji je zadao upit.

Konfiguracija SerbianAnalyzer

Skinuli smo i raspakovali SerbianAnalyzer, verziju 7.4.0. Potom se u komand promtu pocionirali u korenski direktorijum analajzera i izvršili sledeću komandu "*gradlew clean build*". Nakon toga pocionirali smo se u /bin direktorijum ElasticSearch-a i izvršili komadnu "*./elasticsearch-plugin install file:<absolute path of distribution archive>* "

Potom smo konfigurisali da nam SerbianAnalyzer bude podrazumevani tako što smo u elastisearch.yml fajlu uneli "*index.analysis.analyzer.default.type: serbian_analyzer*".

JSON index unit

Kako bi vršili pretraživanje radova, moramo ih nekako indeksirati. Najpre potrbeno je povući paralelu između terminologije kod relacione baze podataka i elasticserach-a.

Baza <---> Indeks

Tabela <---> Tip

Red(Torka) <---> Dokument

U našem slučaju potrebno je opisati koja će sve polja naš dokument(torka) imati. Prilikom odabira polja koje ćemo čuvati unutar dokumenta, oslonili smo se na specifikaciju projekta, koja nam govori po čemu sve ćemo vršiti pretragu. Na slici 3 prikazećemo našu JSON strukturu tj. odabir polja koja ćemo čuvati:

```
{
    "id":1,
    "magazineName":"elekroničar",
    "workTitle":"Frekventni regulatori",
    "workStatus":"approved",
    "sciencenArea":"elektrotehnika",
    "author":"Mirko Ivic",
    "authorId":1,
    "keyTerms": "kljucne reci su ovde",

    "reviewersIds":[1, 2 , 3],
    "text": "Lorem ipsum",
    "file":"rad.pdf"

}
```

Slika 3.

More like this funkcionalnost

U specifikaciji projekta, u odeljku 2.2.3. pod tačkom 11.b rečeno je da listu recenzenata možemo filtrirati, u ovom slučaju tako, gde će nam sistem dati listu recenzenata koji su recenzirali slične radove.

More like this funkcionalnost bismo realizovali uz pomoć ugrađenog “more_like_this” upita u elasticsearch. Primer upita možete videti na sledećoj slici:

```
GET /_search
{
    "query": {
        "more_like_this" : {
            "fields" : ["title", "description"],
            "like" : "Once upon a time",
            "min_term_freq" : 1,
            "max_query_terms" : 12
        }
    }
}
```

Link, gde smo čitali o *more like this*
https://www.elastic.co/guide/en/elasticsearch/reference/current/query-dsl-mlt-query.html

Ostala je dilema koju ćemo prilikom implementacije razrešiti, a to je kakav tip polja staviti u json strukturi po kom ćemo tražiti slične radove. Istražujući došli smo do 2 potencijalna rešenja.

1)Prvo rešenje zahteva da se iz pdf fajla podnetog rada izdvoji tekst, i smesti u json strukturu pod poljem naziva *text tipa String.*

2)Polje *file* nam je tipa *attachment. Zbog* toga smo u json index unit rada stavili polje *file* koje će biti tipa attachament, kako bismo mogli da tražimo slične radove.

Prvo ćemo pokušati s idejom 2 kako bi izbegli izvlačenje teksta. Ukoliko ne uspe onda 1, gde ćemo izvlačiti tekst iz pdf fajla i tražiti slične radove.

Geoprostorna pretraga

U specifikaciji projekta, u odeljku 2.2.3. pod tačkom 11.c rečeno je da listu recenzenata možemo filtrirati, u ovom slučaju tako, gde su svi recenzenti udaljeni od autora rada u radijusu od 100 kilometara.

Ovu funkcionalnost ćemo realizovati na sledeći način. Najpre ćemo napraviti još jedan tip, koji ćemo nazivati “reviewers”, unutar indeksne strukture. Polja koja će imati dokumenti u ovom tipu su sledeća:

1)id-predstavlja id recenzenta tipa *long*

2)name- ime recenzenta tipa *String*

3)jobMagazines- niz *long* koji predstavljaju id od časopisa u koijma recenzent radi

4)citiyCoordinates- struktura tipa *geopoint* koja sadrži polja *lat i lon* koja predstavljaju koordinate mesta u kom recenzent stanuje

Primer koji pokazuje definisanje polja tipa *geopint:*

```
PUT /my_locations
{
    "mappings": {
        "properties": {
            "pin": {
                "properties": {
                    "location": {
                        "type": "geo_point"
                    }
                }
            }
        }
    }
}

PUT /my_locations/_doc/1
{
    "pin" : {
        "location" : {
            "lat" : 40.12,
            "lon" : -71.34
        }
    }
}
```

Upit ćemo vršiti tako što zadamo radijus od 100km, časopis i koordinate autora rada, a elasticsearch će nam vraćati recenzente koji nisu u ovom radijusu. Primer upita možete videti na sledećoj slici(s tim što je ovde radijus 200km):

```
GET /my_locations/_search
{
    "query": {
        "bool" : {
            "must" : {
                "match_all" : {}
            },
            "filter" : {
                "geo_distance" : {
                    "distance" : "200km",
                    "pin.location" : {
                        "lat" : 40,
                        "lon" : -70
                    }
                }
            }
        }
    }
}
```

Geoprostrnu pretragu ćemo implementirati uz pomoć dokumnetacije koju smo pronašli na sledećem linku:https://www.elastic.co/guide/en/elasticsearch/reference/current/query-dsl-geo-distance-query.html