Upravljanje digitalnim dokumentima (UDD)

Kontrolna tačka, Zdravko Milinković E2-49/2023

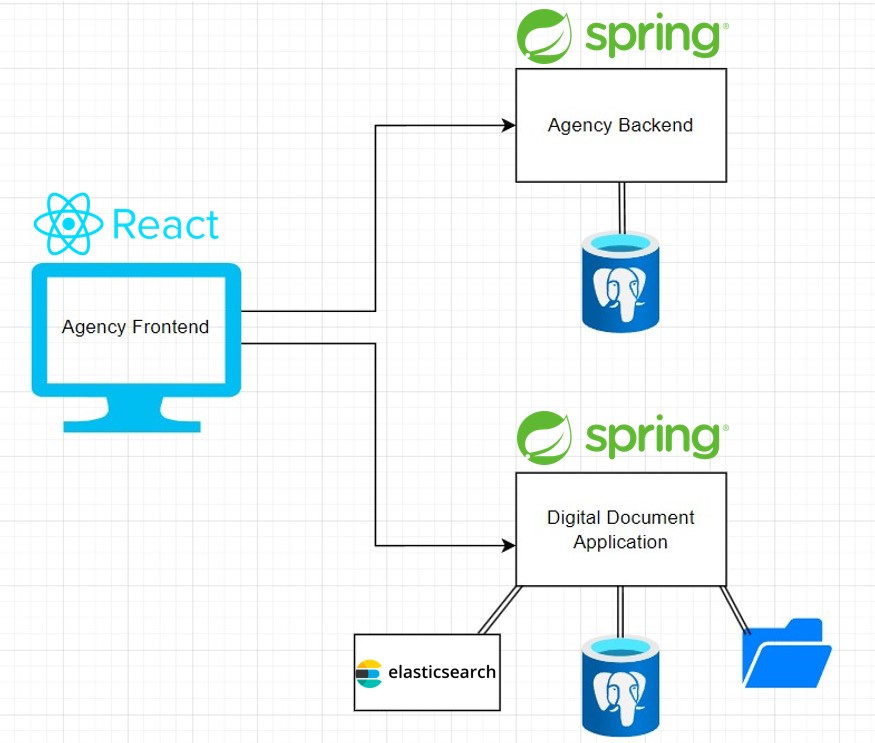
# Arhitektura sistema i High-Level pregled

Ovaj projekat predstavlja implementaciju sistema agencije koja služi za izdavanje zakona. Ključni procesi ovog sistema su:

* Registracija klijenata – tokom registracije, član vlade bira usluge koje želi da kupi od agencije. Pored toga, član vlade bira način pretplate, koji može biti mesečni ili godišnji, i način plaćanja.
* Kodifikacija zakona – proces izmene trenutnog zakona ili grupe zakona.
* Raskidanje ugovora – proces koji se automatski pokreće ukoliko je istekao rok od 5 dana za kodifikaciju zakona.

Kako bi korisničko iskustvo bilo pozitivno, na backendu aplikacije nalazi se modularni i plagabilni sistem za elektronsko plaćanje koji omogućava korisniku plaćanje usluga putem platne kartice, QR koda, PayPal-a i putem kriptovaluta. Poseban akcenat je stavljen na bezbednost ovog sistema i rukovanje novčanim sredstvima.

Implementacija sistema za efikasno skadištenje i pretraživanje digitalnih dokumenata, kao što su ugovori i zakoni, značajno doprinosi razvoju i performansama aplikacije, kao i poboljšanju korisničkog iskustva. Glavni akcenat sistema je brza i efikasna pretraga. Pregled arhitekture sistema za pretragu i upravljanje digitalnim dokumentima nalazi se na slici 1.



Slika 1. Prikaz arhitekture sistema

## Arhiktetura sistema za pretragu i upravljanje digitalnim dokumentima

Sistem predstavlja nezavisnu web aplikacija koja se sastoji od dva glavna dela:

* Serverska strana – deo sistema implementiran u *Spring Boot* radnom okviru, koji je zasnovan na REST API principima i služi za obradu digitalnih dokumenata putem HTTP zahteva. Ovaj deo sistema predstavljen je kao troslojna aplikacija i sastoji se od:
  + Prezentacionog sloja – predstavlja sloj najbliži klijentskoj strani čija je uloga primanje zahteva i njihovo mapiranje na određene metode u sistemu.
  + Biznis sloja – predstavlja sloj u kojem se odvija sama logika aplikacije. Sva rukovanja entitetima, kao i potencijalni izuzeci dešavaju se u okviru ovog sloja. U ovom sloju nalaze se *Indexer*, koji vrši operacije dodavanja, modifikacije i brisanja nad dokumentima unutar indeksne strukture, kao i *ResultRetriver* koji vrši upite nad tim dokumentima.
  + Sloja podataka – predstavlja najniži sloj ove strukture. Glavna uloga ovog sloja pored rada sa entitetima je i obavljanje komunikacije sa relacionom bazom podataka i *ElasticSearch* skladištem.
* Klijentska strana – deo sistema zadužen za interakciju sa korisnicima. Putem ovog dela aplikacije, korisicima je omogućena registracija na sistem, kao i pretraga ugovora i zakona putem odgovarajućih formi i odabranih parametara. Takođe, korisnici imaju mogućnost otpremljivanja i preuzimanja svih ugovora i zakona. Klijentska strana razvijena je upotrebom *React.JS* biblioteke za *JavaScript*.

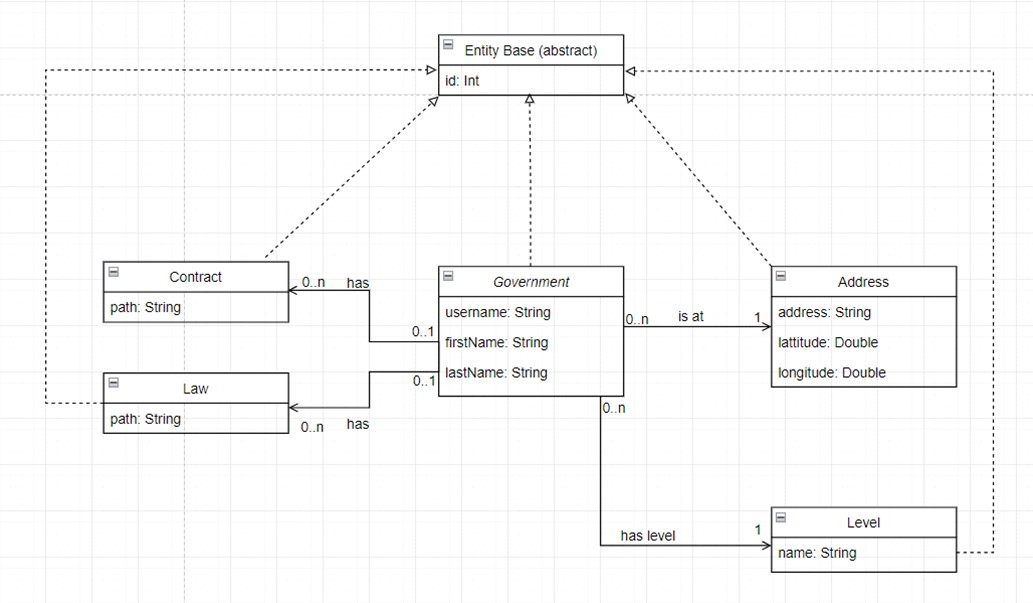
Za skladištenje entiteta koristi se PostgreSQL relaciona baza podataka. Digitalni dokumenti čuvaju se u pdf formatu na lokalnom računaru. Kako je neophodno podržati efikasnu i brzu pretragu, aplikacija se oslanja na rad *ElasticSearch* sistema i samim tim, deo podataka se skladišti u okviru indeksne strukture *ElasticSearch* sistema.

Radi lakše kolaboracije *Spring* *Boot* okruženja i *ElasticSearch* sistema koristi se paket *Spring* *Data* *ElasticSearch*. On omogućava razne prednosti kao što su:

* Jednostavna uspostava i održavanje komunikacije sa *ElasticSearch* serverom
* Anotaciju modela koristeći indekse strukture
* *ElasticSearchRestTemplate* – omogućava laku komunikaciju sa sistemom i olakšano kreiranje složenih upita.
* Generički repozitorijumi – interfejsi sa već generisanim skupovima upita.

## Dijagram klasa

Prethodno je pomenuto kako je neophodno podržati efikasnu pretragu po određenim atributima. Kako bi ovo bilo moguće, deo podataka će se čuvati u relacionoj PostgreSQL bazi, dok će se drugi deo podataka skladištiti u okviru *ElasticSearch* sistema. Na slici 2. prikazan je klasni dijagram sistema.



Slika 2. Klasni dijagram sistema

Razlikujemo sledeće klase:

* *Entity* *Base* - apstraktna klasa koja predstavlja bazu svih entita. U njoj se nalazi identifikaciona oznaka i svi entiteti sistema nasleđuju ovu klasu.
* *Contract* - klasa koja predstavlja ugovor i sastoji se od identifikacione oznake i atributa path koji ukazuje na njegovu lokaciju na lokalnom računaru.
* *Law* - klasa koja predstavlja zakon koja se takođe sastoji od identifikacione oznake i path atributa.
* *Government* - klasa koja predstavlja vladu i sastoji se od identifikacione oznake, atributa imena i prezime, kao i korisničkog imena. Vlada sadrži referencu klasa *Address* i *Level*, što znači da se vlada nalazi na tačno jednoj lokaciji i da ima tačno jedan nivo uprave.
* *Address* - klasa koja predstavlja lokaciju na kojoj se vlada nalazi. Pored identifikacione oznake sadrži adresu, kao i geografsku širinu i dužinu, za lakše predstavljanje na mapi.
* *Level* - klasa predstavlja nivo uprave vlade. Svaki nivo uprave sastoji se od svoje identifikacione oznake kao i naziva tog nivoa.

U zavisnosti od dalje implementacije zahteva projekta, sam model je sklon izmeni i dodavanju novih atributa ili izmeni hijerarhije klasa.

## Indexing Units

Indeksnu strukturu koja se koristi u okviru *Elasticsearch*-a modeluje se JSON objektima. Indeksna struktura sastoji se imena polja i njihovih tipova podataka, koja se dobijaju mapiranjem. U listingu 1 prikazan je primer indeksne strukture ugovora koja omogućava pretragu po imenu i prezimenu zaposlenog koji je potpisnik ugovora, po nivou uprave, po nazivu, po lokaciji kao i po sadržaju dokumenta.

Listing 1. Indeksa struktura ugovora

{

"mappings":{

"properties":{

"governmentUsername":{

"type": "text"

},

"governmentFirstName":{

"type": "text",

"analizer": "serbian"

},

" governmentLastName ":{

"type": "text",

"analizer": "serbian"

},

"governmentLevelId":{

"type": "integer"

},

"governmentAddress":{

"type": "geo\_point"

},

"contractContent":{

"type": "text",

"analizer": "serbian"

}

}

}

}

# Pretrage

U sistemu potrebno je implementirati:

* Pretraživanje ugovora po imenu i prezimenu zapolsenog koji je potpisnik ugovora
* Pretraživanje ugovora prema nazivu vlade i nivou uprave
* Pretraživanje dokumenata prema sadržaju ugovora
* Pretraživanje dokumenata prema sadržaju zakona
* Pretraživanje ugovora po geolokaciji vlade
* Pretraživanje kombinacijom prethodnih parametara

Kombinovano pretraživanje podrazumeva upotrebu *QueryBuilders* klase, koja nudi razne mogućnost za formiranje kompleksnih upita. Klasa *QueryBuilders* prevodi tj. mapira upit u odgovarajući *query*. Takođe, ova klasa podržava *BooleanQuery*, koji se dodaje pomoću *boolQuery* metode i omogućava logičko kombinovanje parametara koristeći standardne AND, OR i NOT operatore. Pretraga po frazama može se vršiti korišćenjem *MatchPhrazeQuery*-ja. Pretraga po geolokaciji vlade podrazumeva da korisnik zadaje ime grada ili adresu i radijus u okviru kojeg se vrši pretraga.

Za lepši prikaz rezultata potrebno je krerirati dinamički sažetak tj. *Highligher.* To se može učiniti korišćenjem klase *HighlighterBuilder* kojoj se dodaju polja za koja je potrebno kreirati kratak sažetak. Ovakvi složeni upiti mogu se vršiti nad *ElasticSearchRestTemplate*-om.

U sledećim listinzima biće prikazana implementacija nekih od prethodnih zahteva koristeći *ElasticSearch Data Repository*.

Listing 2. Pretraga ugovora po određenom ili višem nivou vlade

public interface ContractRepository extends ElasticsearchRepository<Contract, String> {

List<Contract> getContractsByLevelIdOrAbove(int levelId)

}

Listing 3. Pretraga ugovora po nazivu vlade koristeci @Query anotaciju

public interface ContractRepository extends ElasticsearchRepository<Contract, String> {

@Query("{\"bool\": {\"must\": [{\"match\": {\"contract.usename \": \"?0\"}}]}}")

List<Contract> getContractsByGovernmentUsername(String username);

}

Listing 5. Pretraga po frazi u okviru sadržaja ugovora

{

"query": {

"match\_phrase": {

"contractContent": {

"query": "ovaj ugovor sadrži ovu frazu"

}

}

}

}

Listing 3. i 4. JSON zapis za pretragu po imenu i prezimenu zaposlenog (AND i OR)

{

"query": {

"bool": {

"should": [

{"query": {"govermentFirstName": "first"}},

{"query ": {"govermentLastName ": "last"}}

]

}

}

}

{

"query": {

"bool": {

"must": [

{"query": {"govermentFirstName": "first"}},

{"query ": {"govermentLastName ": "last"}}

]

}

}

}

## Geoprostorna pretraga

Kako jedan od zahteva podrazumeva implementaciju geoprostorne pretrage tako što korisnik unosi ime grada ili adresu i radijus traženja, potrebno je u vladi čuvati i podatke o lokaciji. Za dobije tačnih koordinata putem adrese može se koristiti *Google Maps API*. Na osnovu tih koordinata kasnije se može veoma efikasno i precizno vršiti pretraga. Par numeričkih vrednosti (*latitude, longitude)*, koji se dobije konvertovanjem adrese u koordinate,čuva se u posebnom polju tipa geo\_point (*GeoPoint)*.

Listing 6. predstavlja primer geoprostorne pretrage ugovora u radijusu od 40km udaljenosti od Novog Sada.

"query": {

"bool": {

"must": {

"match\_all": {}

},

"filter": {

"geo\_distance": {

"distance": "40km",

"government.adress": {

"lat": 45.26,

"lon": 19.83

}

}

}

}

}

Thi

# Konfiguracije

## Konfiguracija *ElasticSearch*

Aplikacija se oslanja na *ElasticSearch* kao servis za efikasnu pretragu sadržaja. Odabrana verzija Elasticsearcha je 7.4.0. zbog kompatibilnosti sa *SerbianAnalyzer* plugin-om. *ElasticSearch* se pokreće kao zasebna aplikacija na lokalnoj mašini na portu 9200. Ovaj port je default-ni i ne mora se dodatno konfigurisati.

Komunikacija sa serverskom stranom aplikacije moguća je pozivanjem REST API koji nudi *ElasticSearch*. Za implementaciju je odabran pristup baziran na *Spring*-ovom projektu *Spring Data ElasticSearch*. Osnovna ideja jeste mapiranje metoda definisanih kroz repozitorijum nad određenim indeksnim strukturama na konkretne pozive REST API *endpoint*-a koje pruža *ElasticSearch* aplikacija. Ukoliko je potrebno, dodatne parametre kao što su lozinke, korisnička imena i sl. moguće je konfigurisati putem application.properties fajla.

U slučaju da repozitorijum ne budu mogao da pruži sve potrebne mogućnosti i da podrži sve potrebne upite, po potrebi se može koristiti i *ElasticSearchRestTemplate.*

## Konfiguracija SerbianAnalyzer Plugin-a

U okviru projekta je neophodno podržati pretprocesiranje ulaznog teksta napisanog na srpskom jeziku. Zbog specifičnosti jezika i određenih slova (ć, č, š, ž, đ), potrebno je preuzeti i upotrebiti navedeni plugin kako bi se omogućila efikasna i tačna pretraga teksta. Implementacija plugin-a može se pronaći na linku <https://github.com/chenejac/udd06> dok se način same integracije u projekat može detaljno ispratiti u README fajlu. Kasnije, ukoliko je potrebno koristiti ovaj plugin prilikom obrade teksta, to je potrebno eksplicitno označiti u indeksnoj strukturi (primer na listingu 1.).

## Konfiguracija ELK Stack-a

ELK Stack predstavlja grupu projekata dizajniranu da korisicnima omogući efikasnu i jednostavnu pretragu, analizu i vizuelizaciju podataka, prikupljenih sa raznih izvora u različitim formatima, u realnom vremenu. Sastoji se od 3 aplikacije:

* *ElasticSearch* – aplikacija za efikasnu pretragu koja omogućava visoke performanse.
* *Logstash* – aplikacija koja omogućava rukovanje logovima, pruža podršku za dobavljanje logova iz različitih izvora, njihovu obradu i prosleđivanje na željeno skladište. Često se koristi u kombinaciji sa *Filebeat* aplikacijom koja preuzima ulogu dostavljanja logova do *Logstash* aplikacije.
* *Kibaana* – aplikacija koja podržava različite vidove vizualizacije podataka i čija je integracija sa *ElasticSearch-*om podržana bez dodatne konfiguracije. Definisanjem *Index* *Patterns* moguće je preuzeti podatke skladištene u okviru *ElasticSearch*-a, koji kasnije mogu biti vizuelizovani u vidu grafikona, *chart*-ova i sl.

S obzirom na povezanost ovih aplikacija, neophodno je pažljivo izvrišiti konfigurisanje i integraciju, kako se ne bi narušila njihova međusobna kompatibilnost.