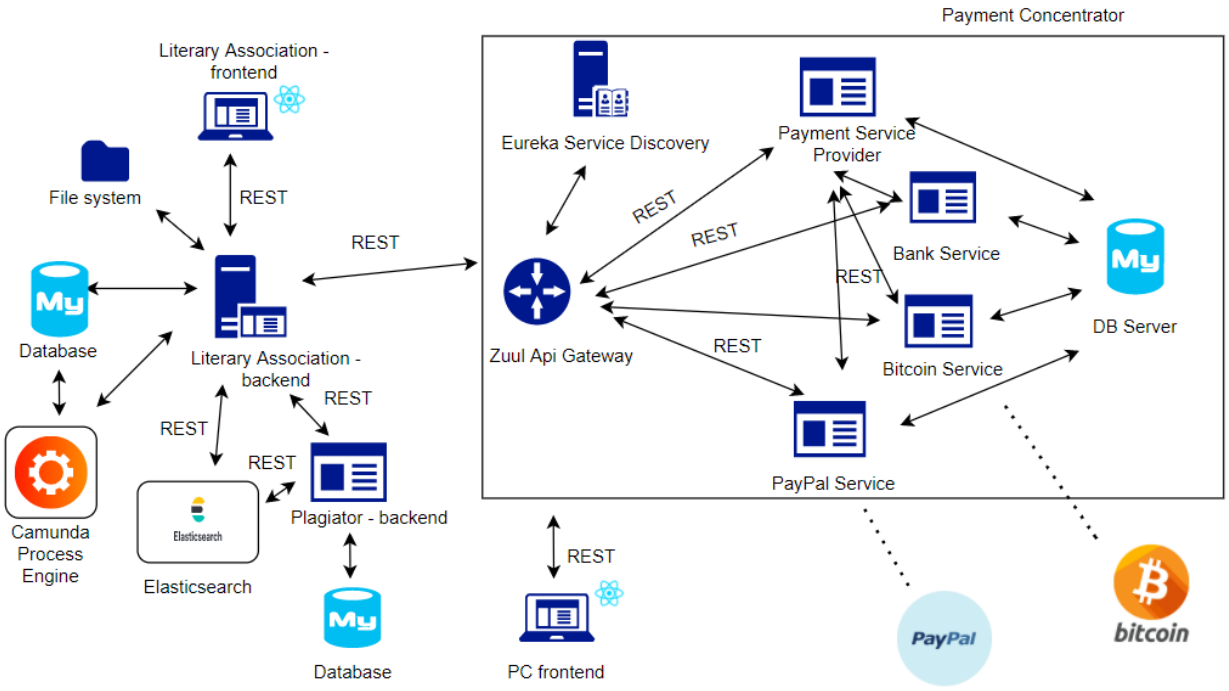
Upravljanje digitalnim dokumentima

Kontrolna tačka – Marina Bartulov E2 81/2020

1. Arhitektura aplikacija i komunikacija

Literarno udruženje je realizovano kao monolitna aplikacija koja se sastoji od frontend i backend aplikacije koji komuniciraju putem REST-a. Backend aplikacija je klasična Spring Boot aplikacija koja se sastoji od 3 sloja: sloj kontrolera, servisni sloj i sloj podataka koji ostvaruje konekciju ka bazi. Literarno udruženje komunicira sa drugim sistemom koji predstavlja koncentrator plaćanja. On je realizovan kao mikroservisna aplikacija i vrši usluge plaćanja za klijente kao što je Literarno udruženje. Koncentrator plaćanja i Literarno udruženje komuniciraju takođe putem REST-a. Backend obe aplikacije je Spring Boot, a frontend aplikacije su rađene u React-u. Ceo ovaj sistem prikazan je na Slici 1.

Knjige koje se mogu pretraživati, tj. PDF fajlovi će biti čuvani u lokalnom fajl sistemu. Podaci o entitetima sistema će se čuvati u relacionoj bazi, koja će biti MySql. Određeni podaci o entitetima koji se pretražuju biće ili čuvani samo u bazi, ili samo u indeksnoj strukturi, ili na oba mesta. Sloj podataka kojeg čine repozitorijumi će ostvarivati komunikaciju sa bazom. Literarno udruženja će putem REST-a da komunicira sa Elasticsearch-om prilikom indeksiranja i pretraživanja dokumenata, a takođe će da komunicira i sa sistemom za detekciju potencijalnih plagijarizama koji predstavlja posebnu web aplikaciju, i ta komunikacija će takođe da se ostvaruje putem REST-a.



Slika 1. Arhitektura sistema

2. Pokretanje i konfiguracija ElasticSearch-a

Elasticsearch možemo da pokrenemo tako što skinemo zip fajl sa zvaničnog sajta i otpakujemo, i zatim odatle pokrenemo. Drugi način je da ga pokrećemo zajedno sa Spring Boot aplikacijom. Zbog kasnije jednostavnosti instaliranja plugin-a ja sam se odlučila na prvi način. Skinula sam verziju 7.4.0 jer je ona kompatibilna sa pluginom za srpski jezik. Kako je Elasticsearch RESTful search engine komunikacija između backend aplikacije i njega će se odvijati takođe preko REST-a.

Da bi se pokrenuo Elasticsearch potrebno je pozicionirati se u bin folder otpakovanog zip fajla i ukucati komandu elasticsearch.bat. Elasticsearch se pokrene na portu 9200. Da bi se proverilo da li je Elasticsearch pokrenut može se poslati GET zahtev na http://localhost:9200 i dobiti odgovor kao na Slici 2.


```
{
    "name": "MARINA-PC",
    "cluster_name": "elasticsearch",
    "cluster_uuid": "KC_88jLKSZWNjKXBocP-4w",
    "version": {
        "number": "7.4.0",
        "build_flavor": "default",
        "build_type": "zip",
        "build_hash": "22e1767283e61a198cb4db791ea66e3f11ab9910",
        "build_date": "2019-09-27T08:36:48.569419Z",
        "build_snapshot": false,
        "lucene_version": "8.2.0",
        "minimum_wire_compatibility_version": "6.8.0",
        "minimum_index_compatibility_version": "6.0.0-beta1"
    },
    "tagline": "You Know, for Search"
}
```


Slika 2. Provera da li je ElasticSearch pokrenut

Elasticsearch ima dobre deafult opcije i zahteva jako malo konfiguracije. Što se tiče konfiguracije Elasticsearch-a ona se može obavljati u elasticsearch.yml fajlu koji se nalazi u config folderu otpakovanog zip fajla. Tu se mogu konfigurisati neke naprednije opcije, a za naše potrebe je za sada dovoljno da ostane default. Tu se mogu npr. specificirati putanje gde Elasticsearch upisuje podatke i logove. Polja kojima se to konfiguriše jesu path.data i path.logs i preporuka je da u produkciji ona budu podešena na lokacije izvan Elasticsearch foldera.

U Spring Boot projektu za komunikaciju sa Elasticsearch-om ću koristiti ElasticsearchRepository za indeksiranje, a za upite ElasticsearchTemplate jer on omogućava dinamički sažetak koji nam je potreban u projektu. Dependecy za to je prikazan na Slici 3.

```
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-data-elasticsearch</artifactId>
</dependency>
```

Slika 3. Dependency za komunikaciju sa Elasticsearch-om

3. Serbian Analyzer plugin

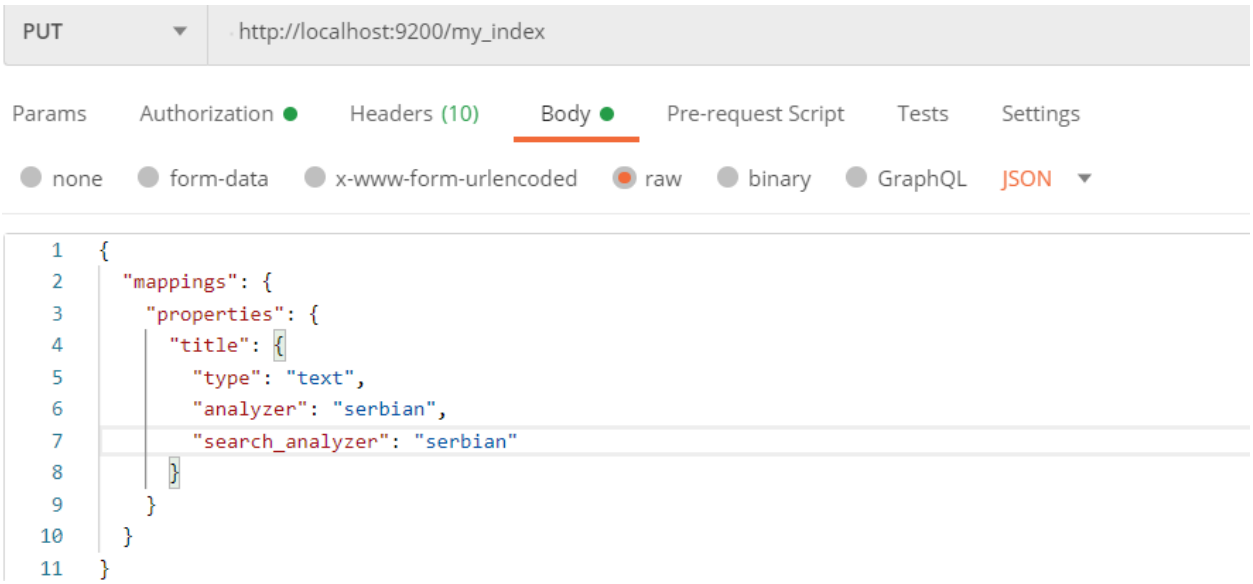
Serbian Analyzer plugin se koristi za analiziranje i pretprocesiranje unetog teksta na srpskom jeziku. Za njegovo instaliranje potrebno je da imamo instaliran build alat Gradle. Potrebno je da se pozicioniramo u root direktorijum od Serbian Analyzer plugin projekta i izvršimo komandu:

./gradlew clean build

Ova komanda će da kreira zip fajl i da ga smesti u build/distributions folder. Nakon toga da bismo podesili da ovaj plugin radi sa Elasticsearch-om, potrebno je da se pozicioniramo u njegov /bin direktorijum i da izvršimo komandu:

./elasticsearch-plugin install file:<absolute path of distribution archive>

Na različite načine se može reći Elasticsearch-u koji analyzer da gleda prilikom indeksiranja i pretraživanja. Za indeksiranje moguće je specificirati analyzer za polje ili default analyzer za indeks. Za pretraživanje se može još specificirati i za query. Najčešće specificiranje različitog analyzer-a za pretraživanje nije potrebno. U većini slučajeva jednostavan pristup je najbolji, a to je da se specificira analyzer za svako tekst polje posebno. Moj plan je da koristim taj pristup. Za svako tekst polje se kaže da je “analyzer” = “serbian” i “search_analyzer” = “serbian” (Slika 4). Mada nije čak ni potrebno navoditi za search, jer ukoliko se on ne navede podrazmeva se da je isti kao i za indeksiranje.



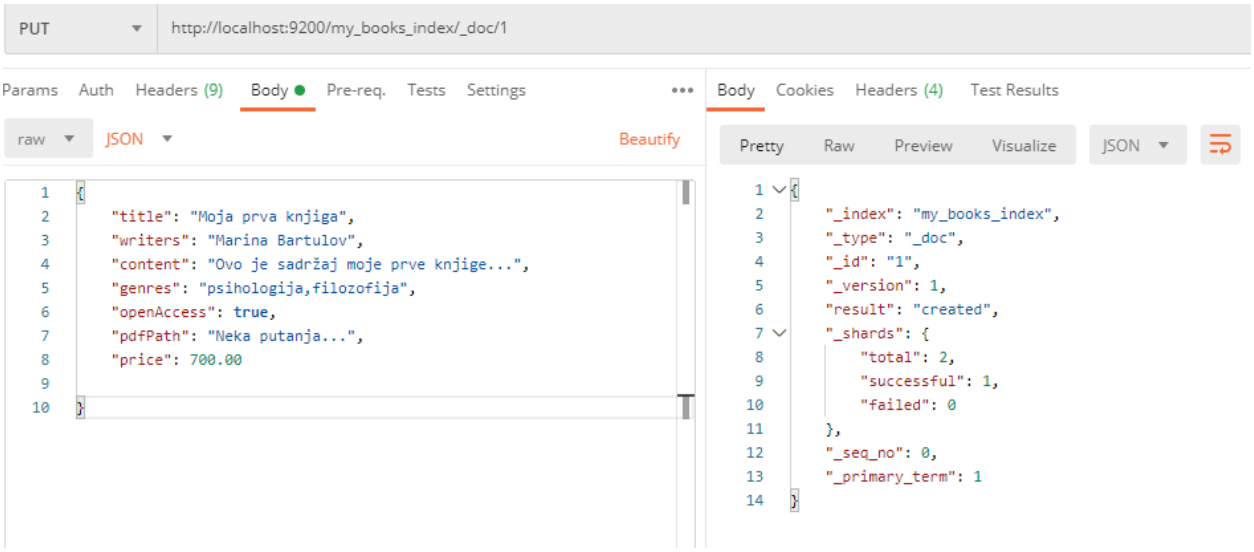
Slika 4. Način specificiranja analyzer-a

4. Book index unit

Što se tiče knjiga neki podaci će biti čuvani samo u bazi, jer neće biti korišćeni prilikom pretraživanja, niti će biti bitni da se prikažu u sklopu rezultata. Neki od tih podataka su ISBN, mesto i godina izdavanja i broj stranica. Dok će se u indeksoj strukturi pored podataka po kojima će se pretraživati knjige, nalaziti i oni podaci koji će biti bitni da se prikažu i na neki način koriste u rezultatu pretrage, a to su podatak da li je knjiga u open access mode-u, kao i putanja do pdf fajla kako bi se ta knjiga mogla download ukoliko jeste u open access mode-u, zatim sinopsis i cena. Takođe će se čuvati i sadržaj pdf fajla, kojeg neće biti u bazi. Na Slici 5 sam prikazala kako bi izgledalo kreiranje indeksa za knige.

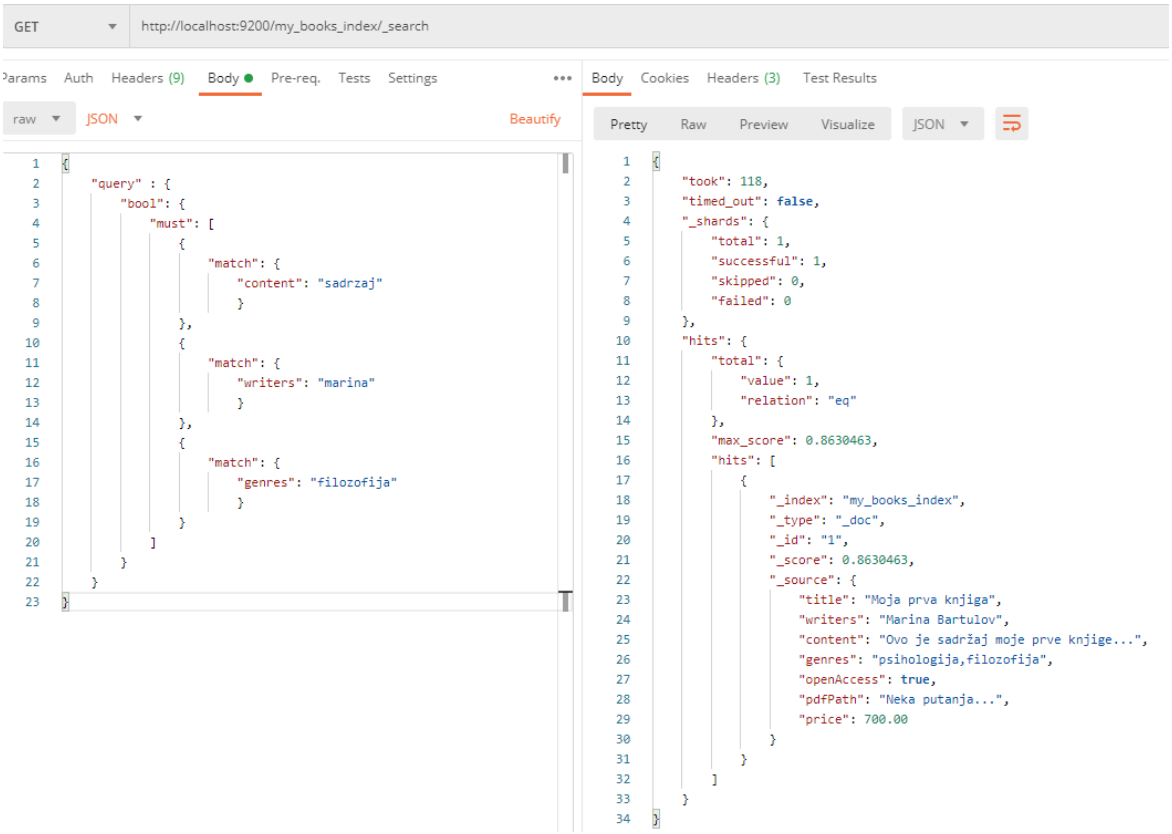
Slika 5. Kreiranje indeksa za knjige

Na sledećoj slici (Slika 6) je prikazano dodavanje jedne knjige u indeks i odgovor koji se dobija.



Slika 6. Dodavanje nove knjige u indeks

Na sledećoj slici (Slika 7) je prikazan jedan primer zadavanja upita nad ovom indeksom strukturom.



Slika 7. Primer upita za knjige

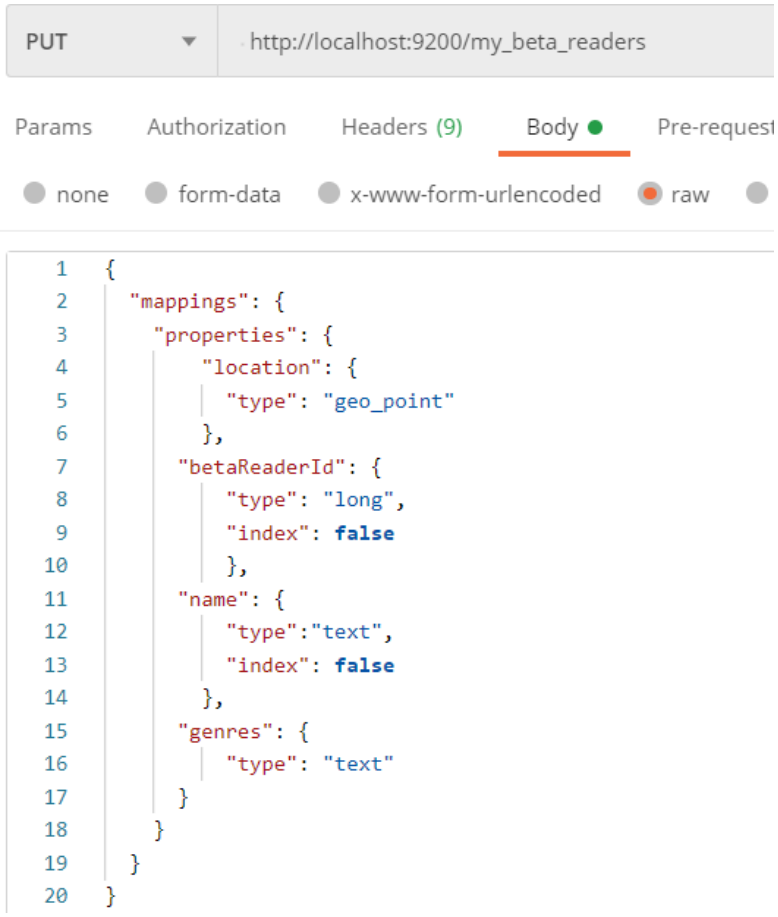
5. Geoprostorna pretraga

Geoprostorna pretraga u ElasticSearch-u se vrši pomoću geo-distance query koji u rezultat pretrage uključuje samo ono što postoji unutar određene udaljenosti od zadate geo tačke. Prilikom indeksiranja kao tip podatka se koristi geo_point. Primer kako bih kreirala indeks za beta-čitaoce koji treba da se geoprostorno pretražuju je prikazan na Slici 8. Id od beta-čitaoca i ime se neće indeksirati, nego će se oni koristiti u prikazu rezultata. Što se tiče žanrova za sada će to biti string koji će sadržati id-jeve žanrova koje je beta-čitalac izabrao, ali ću razmisliti još da li je ovo najbolje rešenje.

Podaci za beta-čitaoce kao što su username, password, grad, država, email i kazneni poeni će biti čuvani samo u bazi jer oni nisu važni za pretragu i prikaz rezultata pretrage.

Na Slici 9 sam prikazala kako bih dodala novog beta-čitaoca u postojeći indeks.

Na Slici 10 kako bih napravila upit, a na Slici 11 rezultat upita. U upitu govorim da se pronađu svi oni koji ne nalaze okolini od 100 km od zadatke tačke.



Slika 8. Indeksiranje beta-čitala

PUT http://localhost:9200/my_beta_readers/_doc/1

Params Authorization Headers (9) Body Pre-request Script Tests Settings

none form-data x-www-form-urlencoded raw binary GraphQL JSON

```
{
    "location": {
      "lat": 40.12,
      "lon": -71.34
    },
    "genres": "1,2,3",
    "name": "Marina Bartulov",
    "betaReaderId": 1
}
```

Slika 9. Dodavanje novog beta-čitaoca

GET http://localhost:9200/my_beta_readers/_search

Params Authorization Headers (9) Body Pre-reques

none form-data x-www-form-urlencoded raw

```
{
  "query": {
    "bool": {
      "must": {
        "match_all": {}
      },
      "filter": {
        "bool": {
          "must_not": {
            "geo_distance": {
              "distance": "100km",
              "location": {
                "lat": 50,
                "lon": -70
              }
            }
          }
        }
      }
    }
  }
}
```

Slika 11. Upit za geoprostornu pretragu

GET http://localhost:9200/my_beta_readers/_search

Params Authorization Headers (9) Body Pre-request Scr

Body Cookies Headers (3) Test Results

Pretty Raw Preview Visualize JSON

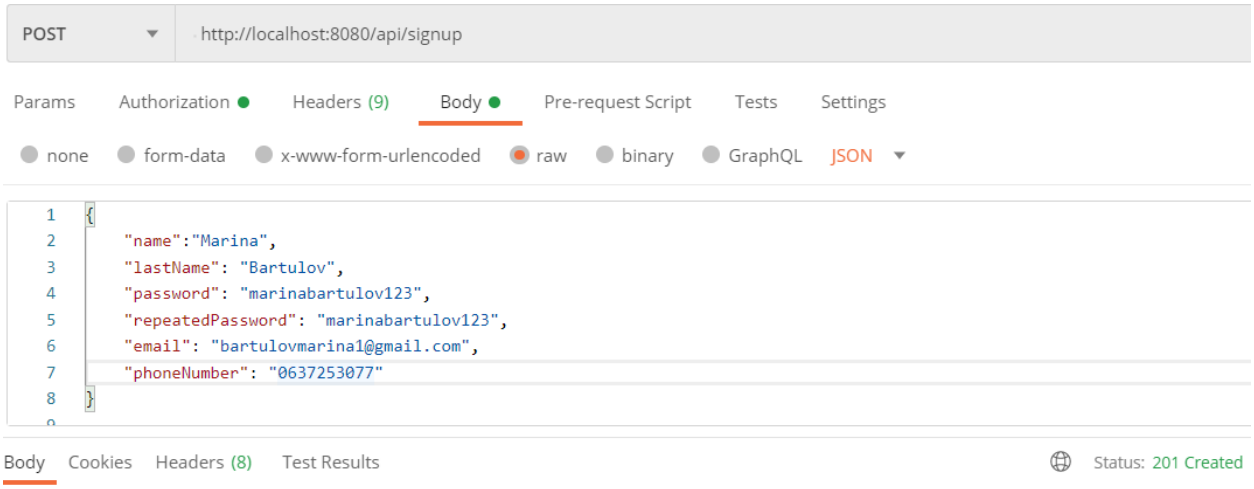
```
    "hits": {
        "total": {
            "value": 1,
            "relation": "eq"
        },
        "max_score": 1.0,
        "hits": [
            {
                "_index": "my_beta_readers",
                "_type": "_doc",
                "_id": "1",
                "_score": 1.0,
                "_source": {
                    "location": {
                        "lat": 40.12,
                        "lon": -71.34
                    },
                    "genres": "1,2,3",
                    "name": "Marina Bartulov",
                    "betaReaderId": 1
                }
            }
        ]
```

Slika 10. Rezultat pretrage

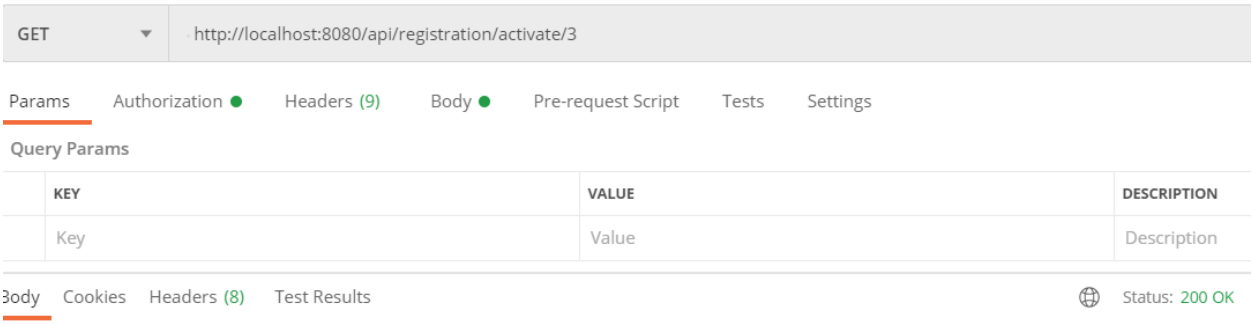
6. Sistem za detekciju potencijalnih plagijarizama

Sistem za detekciju potencijalnih plagijarizama predstavlja posebnu web aplikaciju, koja se sastoji od Spring Boot backend aplikacije i Angular frontend aplikacije. Za potrebe projekta ja ću koristiti samo backend aplikaciju sa kojom će backend aplikacija Literarnog udruženja da komunicira preko REST-a. Pre pokretanja Plagiatora neophodno je podesiti konekciju sa MySql bazom i promeniti mejl za slanje mejlova u application.properties fajlu.

Da bi se koristio Plagiator neophodno se prvo registrovati i aktivirati nalog (Slika 12 i Slika 13)

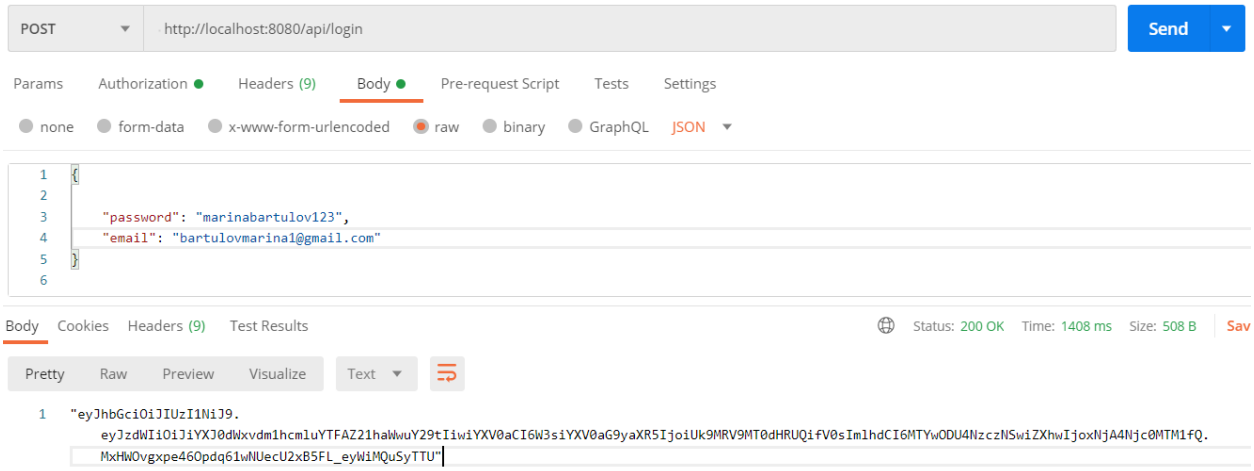


Slika 12. Registracija na Plagiator



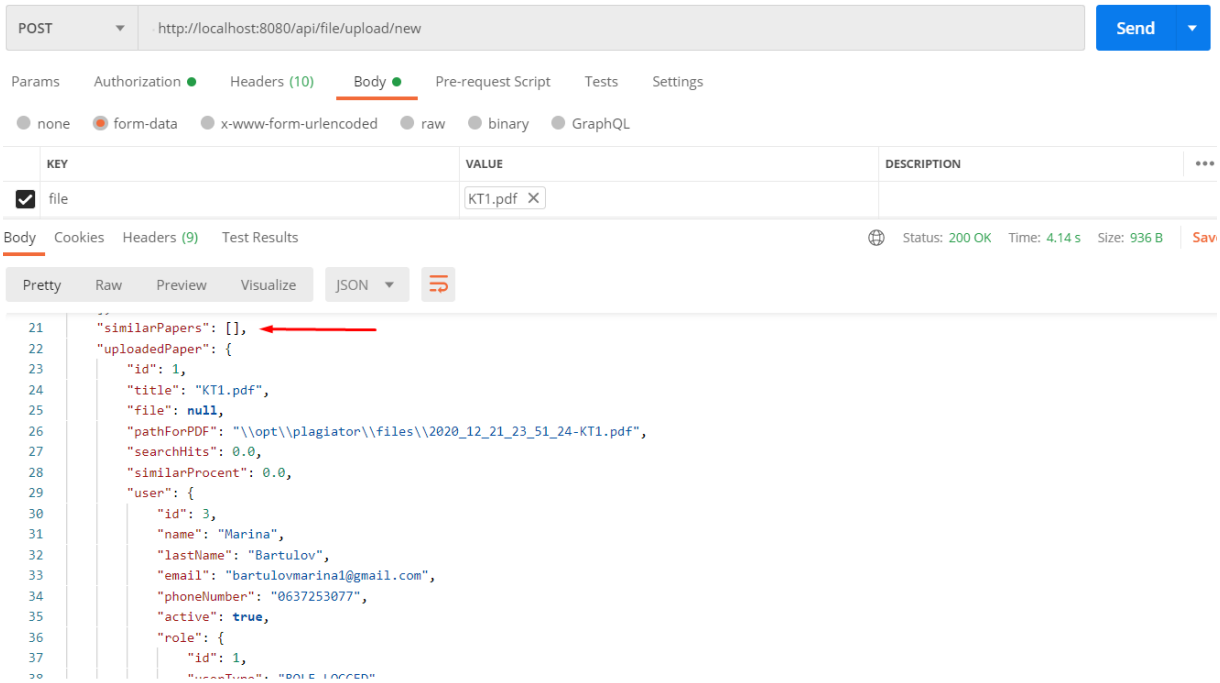
Slika 13. Aktivacija naloga

Nakon toga da bi se koristio sistem neophodno je da se korisnik uloguje. Pošto će backend Literarnog udruženja da komunicira sa ovom aplikacijom i šalje zahteve, pretpostavljam da ću morati da registrujem moju aplikaciju kao jednog od korisnika i prilikom slanja poziva moraće prvo da se šalje zahtev za logovanje na sistem kako bi se dobio jwt token (Slika 14).



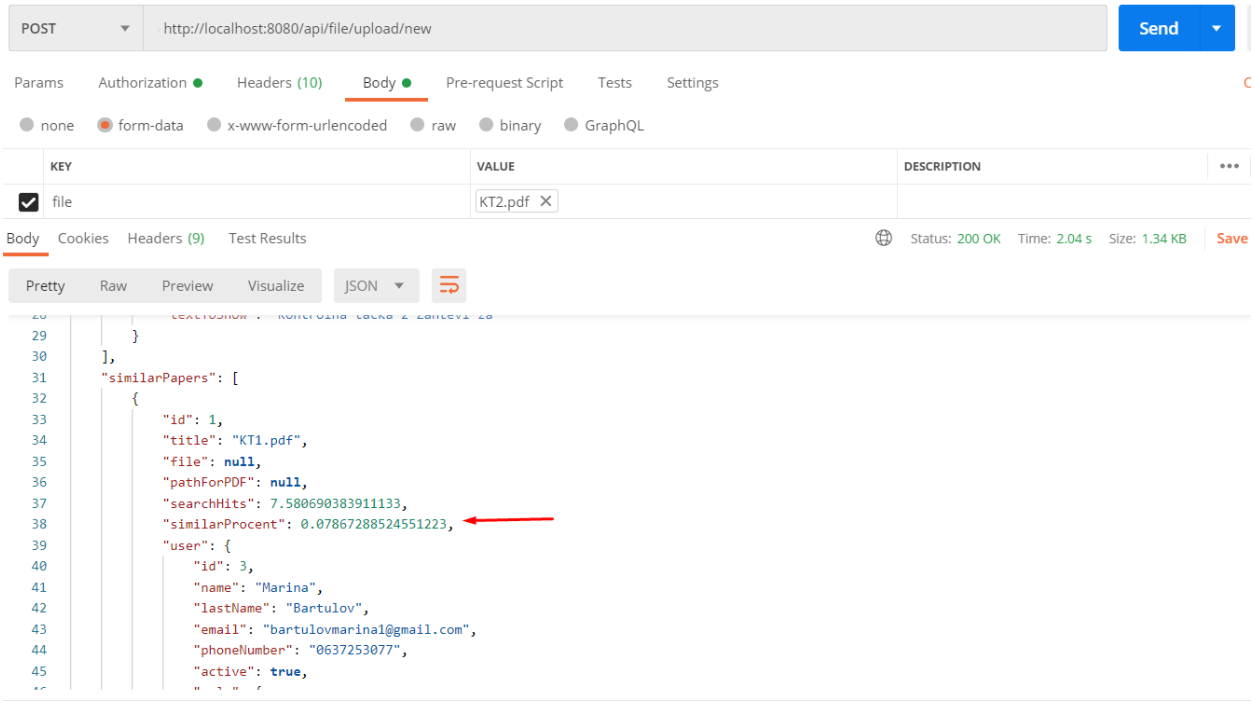
Slika 14. Logovanje na sistem

Moja aplikacija će da koristi endpoint za upload fajlova i dobijanje odgovora koji sadrži listu sličnih radova. Ovaj Plagiator ne govori da li je nešto plagijat ili ne, nego čovek to mora da odluči sam na osnovu dobijenih sličnih radova. Ukoliko želim da upload-ujemo fajl na sistem, šaljem POST zahtev na url http://localhost:8080/api/file/upload/new i podatak koji je neophodno poslati je file (Multipart), koji predstavlja knjigu. Primer jednog takvog zahteva prikazan je na Slici 15.



Slika 15. Upload fajla

Na prethodnoj slici se vidi da kada upload-ujem prvi fajl, da nema sličnih fajlova u odgovoru. Međutim kada ponovo pošaljem takav zahtev sa drugim fajlom dobijam odgovor kao što je na Slici 16.



Slika 16.Upload fajla

Sada se u similarPapers nalazi prethodno upload-ovan fajl i nalazi se podatak o tome koliko su slični (similarProcent).