# Управљање дигиталним документима – контролна тачка

Драгана Грбић R2 8/2021

## Архитектура система

Архитектура система приказана је на [слици 1](#слика_1). У систему се може уочити 9 логичких целина:

* Frontend апликација која служи као кориснички интерфејс апликантима да достављају пријаве за одређен оглас и запосленима у HR служби да претражују достављене пријаве. Апликација је имплементирана коришћењем Angular радног оквира и TypeScript језика. Поседује страницу за пријаву, страницу за приказ свих активних огласа за посао, страницу на којој кандидат може да аплицира за одређен оглас за посао и страницу на којој запослени у HR служби може да претражује достављене пријаве.
* Backend апликација која нуди REST API Frontend апликацији и тако омогућава пријаву регистрованих корисника, складиштење огласа за посао, складиштење достављених пријава за посао и њихову претрагу. Приликом складиштења и претраге достављених пријава за посао, Backend апликација комуницира са Elasticsearch REST API-ем и тако обезбеђује да се достављене пријаве индексирају и накнадно могу брзо претражити. Апликација је имплементирана коришћењем Spring радног оквира и Maven Build алата. Поседује endpoint-е за пријаву корисника, добављање огласа за посао, доставу нових пријава за посао, претрагу достављених пријава и endpoint-е за преузимање PDF фајлова достављених приликом пријаве за посао. Endpoint-и који креирају нову пријаву за посао и претражују постојеће пријаве комуницирају са Elasticsearch-ом, како би се пријаве индексирале, односно претраживале коришћењем индексне претраге.
* Backend релациона база података у којој се складиште сви подаци који се индексирају – регистроване корисничке улоге, регистровани корисници и регистровани огласи за посао. Додатно, у овој бази се чувају и сви подаци о достављеним пријавама за посао које је прикладно складиштити у релационој бази података (поља која имају јаку структуру), како би се у случају губитка индекса или његовог оштећења могло урадити реиндексирање.
* Backend File System у којем се чувају достављени PDF фајлови и фајл са генерисаним логовима.
* Elasticsearch који нуди REST API преко којег Backend апликација складишти и претражује индексиране пријаве за посао, Logstash складишти логове од значаја, а Kibana визуализује и анализира складиштене индексе.
* Elasticsearch NoSQL база података која служи за складиштење достављених пријава за посао и логова од значаја, како би се накнадно могли претраживати/анализирати коришћењем Kibana-e.
* Filebeat који служи за читање логова са Backend File System-a и њихово слање Logstash-у на даљу обраду.
* Logstash који служи за обраду/парсирање логова генерисаних од стране Backend апликације и њихово смештање у одговарајуће индексе унутар Elasticsearch-a.
* Kibana која служи за визуелизацију и анализу података индексираних унутар Elasticsearch-a, првенствено логова достављених од стране Logstash-a.

Diagram

Description automatically generated

Слика 1. Архитектура система

## Конфигурација Elasticsearch-a

Elasticsearch служи за индексирање података и њихову брзу претрагу, како би се проналажење докумената учинило што ефикаснијим, а User Experience корисника побољшао. У овом пројекту користиће се за индексирање пријава за посао и њихову брзу претрагу, као и за складиштење логова од значаја, како би се накнадно могли анализирати коришћењем Kibana-e.

Верзија Elasticsearch-a коју сам преузела са официјалног сајта је 7.4.0. Неопходно је преузети баш ту верзију, будући да се SerbianAnalyzer који ћу користити може инсталирати само унутар те верзије. Такође, SerbianAnalyzer захтева и коришћење Java верзије 13, тако да ће приликом покретања свих алата (Elasticsearch, SerbianAnalyzer, Filebeat, Logstash, Kibana) бити неопходно претходно подесити *JAVA\_HOME* варијаблу да показује на Java верзију 13.

Распаковала сам зип са скинутим Elasticsearch-ом и отворила *./config/elasticsearch.yml* фајл. Унутар њега направила сам само једну измену - додала сам на крај фајла линију *indices.query.bool.max\_clause\_count: 3*. Ова вредност ће се користити код креирања динамичког сажетка приликом претраге по садржају PDF фајла, односно кориснику се неће приказивати читав садржај PDF фајла када је претрага успешна, већ само највише 3 реченице. Затим сам се позиционирала унутар *./bin* директоријума и покренула команду *elasticsearch*. Сачекала сам да се сервер покрене и након тога отишла на адресу *localhost:9200.* Добивши одговор од сервера, потврђено ми је да је Elasticsearch успешно подигнут.

Следећи корак био је повезати Backend апликацију са конфигурисаном Elasticsearch-ом. Будући да је Elasticsearch RESTful API, сва комуникација између њега и Backend-a могла се успоставити коришћењем стандардних Spring компонената за REST комуникацију, као што је RestTemplate. Међутим, како постоје готове компоненте које енкапсулирају комуникацију са Elasticsearch-ом, одлучила сам се њих користити. Зато сам унутар Spring Backend пројекта додала dependency са [слике 2](#слика_2) и тиме омогућила коришћење компонената ElasticsearchRepository и ElasticsearchRestTemplate. ElasticsearchRepository има сличну намену као и JpaRepository, с тим да он омогућава комуникацију са Elasticsearch-ом, а не релационом базом. Слично, ElasticsearchRestTemplate има сличну намену као и RestTemplate, с тим да он омогућава комуникацију искључиво са Elasticsearch-ом, а не произвољним REST API-ем. ElasticsearchRepository сам користила за складиштење достављених пријава за посао, а ElasticsearchRestTemplate за њихову претрагу. Иако је претрагу могуће радити и коришћењем ElasticsearchRepository интерфејса, одлучила сам се за ElasticsearchRestTemplate, будући да је код њега лакше конфигурисати прављење динамичког сажетка приликом претраге.

Text

Description automatically generated

Слика 2. Elasticsearch dependency

## Конфигурација SerbianAnalyzer-a

Након што је Elasticsearch конфигурисан и повезан са Backend апликацијом, неопходно је унутар њега инсталирати plugin који ће омогућити да се приликом индексирања и претраге достављених пријава за посао користи аналајзер за спрски језик. У складу са тим, са линка <https://github.com/chenejac/udd06> преузела сам SerbianAnalyzer, позиционирала се у његов коренски директоријум и покренула команду *gradlew clean build*. Након што је JAR фајл направљен, привремено сам угасила Elasticsearch сервер, позиционирала се у његов *./bin* директоријум и покренула команду *elasticsearch-plugin install file:<serbian\_analyzer\_jar\_file\_location>*. Команда врши инсталацију plugin-a унутар Elasticsearch-a и након што је plugin инсталиран, поново сам покренула сервер. Након овога, омогућено ми је конфигурисање да се приликом индексирања и претраге пријава за посао користи аналајзер за спрски језик. Више о томе у наредном поглављу.

## Index Unit

Према спецификацији, кандидат приликом пријаве за одређен оглас за посао уноси име, презиме, email, адресу, образовање и прикаже CV и пропратно писмо у виду PDF фајлова. Накнадно, запослени у HR служби може да претражује достављене пријаве према имену, презимену, образовању, садржају PDF фајла и по геолокацији. У складу са тим, Index Unit који складишти једну пријаву за посао унутар Elasticsearch-a дефинисан је са 10 поља означених на [слици 3](#слика_3). Поље *firstName* је име апликанта, *lastName* је презиме, *email* је email адреса, *address* је адреса, *education* је образовање које апликант поседује, *educationLevel* је степен стручне спреме, *cvText* је садржај CV PDF фајла, *cvLocation* је локација CV фајла на диску, *letterText* је садржај PDF фајла који представља пропратно писмо, *letterLocation* је локација пропратног писма на диску, *adTitle* је наслог огласа за који је кандидат аплицирао и *location* је геолокација апликанта, односно његове адресе.

Text

Description automatically generated

Слика 3. Index Unit за пријаву за посао

Дефиниција поља *firstName*, *lastName*, *education*, *educationLevel*, *cvText* и *letterText* приказана је на сликама [4](#слика_4) и [5](#слика_5). Свим пољима атрибут index има вредност true, будући да према њима претражују документи, те је вредност поља неопходно узети у обзир приликом претраге (у ранијим верзијама атрибут index Је био енумерација која је имала вредност analyzed уколико се према пољу вршила претрага и not\_analyzed у супротном, у последњој верзији енумерација је замењена са boolean). Такође, атрибут store им исто има вредност true, будући да је неопходно чувати њихову оригиналну вредност. Сва поља осим *educationLevel* за атрибут type имају вредност Text, јер се у њима чува текстуални садржај. Поље *educationLevel* за атрибут type има вредност Integer, јер се у њему чува број који представља степен стручне спреме. То поље ће се моћи да претражује према опсегу вредности, односно корисник ће моћи приликом претраге да унесе почетну и крајњу вредност стручне спреме, а систем ће му вратити све пријаве којима степен стручне спреме упада у задати опсег.

A picture containing company name

Description automatically generated

Слика 4. Дефиниција поља firstName, lastName, education, cvText и letterText

Text

Description automatically generated with low confidence

Слика 5. Дефиниција поља educationLevel

Дефиниција поља *email, address, cvLocation, letterLocation* и *adTitle* приказана је на [слици 6](#слика_6). Свим пољима атрибут index има вредност false, будући да се према њима не претражују документи, те се вредност поља не узима у обзир приликом претраге. Међутим, атрибут store им има вредност true, јер је неопходно складиштити њихове оригиналне вредности, како би се могле приказати у резултатима претраге. На тај начин се избегава беспотребно слање додатних API позива који би преузимали податке из релационе базе и повећавају се перформансе система. Поља *email* и *address* се складиште јер садрже информације на који начин се апликант може контактирати. Када запослени претражи пријаве и одлучи се да му је нека од њих интересанта, може га занимати како би контактирао тог апликанта. Зато је те податке корисно приказати у резултатима претраге. Поља *cvLocation* и *letterLocation* се складиште како би се у резултатима претраге могли приказати линкови за преузимање PDF фајлова. Кликом на линк, слаће се API позив на Backend који ће на основу прослеђене путање фајла вратити сам фајл и омогућити његов download. Поље *adTitle* се складишти како би се повећао User Experience запосленог који претражује пријаве, односно како би запослени за пријаве које задовољавају његову претрагу могао да види на који оглас за посао се односе. Сва поља за атрибут type имају вредност Text, будући да се у њима чува текстуални садржај.

A picture containing graphical user interface

Description automatically generated

Слика 6. Дефиниција поља email, address, cvLocation, letterLocation и adTitle

Након што је дефинисан Index Unit и подаци који се складиште унутар њега, потребно је подесити да се приликом индексирања и претраге достављених пријава за посао користи претходно инсталиран аналајзер за српски језик. Ова конфигурација постигнута је анотирањем класе која се мапира на Index Unit као на [слици 7](#слика_7). Садржај означеног фајла settings.json приказан је на [слици 8](#слика_8). На тај начин, у тренутку када се покрене мапирање класе на индекс у Elasticsearch-у, конфигурисаће се да се за складиштење и претрагу података у том индексу користи инсталиран SerbianAnalyzer.

Graphical user interface, text

Description automatically generated

Слика 7. Конфигурисање индекса са пријавама за посао



Слика 8. Конфигурациони фајл за коришћење SerbianAnalyzer-a

## BooleanQuery, PhraseQuery, прављење динамичког сажетка

Након што су подаци индексирани, може се вршити њихова претрага. Према спецификацији, неопходно је омогућити претрагу по имену, презимену, образовању и садржају PDF фајла, као и комбинацију тих параметара претраге коришћењем и AND и OR оператора. У складу са тим, захтев за претрагом пријава могао би да има тело облика као на [слици 9](#слика_9).

Graphical user interface, text, application, email

Description automatically generated

Слика 9. Тело захтева за претрагом пријава за посао

Дакле, прослеђује се листа query-a која одређује за које поље се очекује која вредност и између њих се врши комбинација коришћењем логичког оператора (AND или OR). Уколико би се текстуална вредност унутар поља *value* појединачног query-а нашла под наводницима, позивала би се метода *matchPhraseQuery* из класе *QueryBuilders*. На тај начин омогућило би се задавање фраза приликом претраге. Уколико се текст не нађе између наводника, врши се стандардна претрага. За поље *educationLevel* позива се метода *rangeQuery* из класе *QueryBuilders*, будући да је идеја претрага према опсегу вредности. Након што се креирају query упити за свако поље, врши се њихова комбинација задатим оператором. За то се користи *BoolQueryBuilder* и његове методе *must* и *should*. Крајњи query прослеђује се *NativeSearchQueryBuilder*-у који подешава *highlight* ([слика 10](#слика_10)). Одговор који би се могао добити на захтев са [слике 9](#слика_9) приказан је на [слици 11](#слика_11).

Text

Description automatically generated

Слика 10. Подешавање highlight-a

Graphical user interface, text, email

Description automatically generated

Слика 11. Одговор на захтев за претрагом

## Претрага по геолокацији

Према спецификацији неопходно је омогућити геопросторну претрагу достављених пријава за посао. Код те претраге уноси се назив града и радијус, а очекивани резултат треба да буду све пријаве код којих адреса/град апликанта упада у задату зону. За потребе те функционалности дефинише се поље *location* са [слике 3](#слика_3). Дефиниција тог поља приказана је на [слици 12](#слика_12).

Text

Description automatically generated with medium confidence

Слика 12. Дефиниција location поља

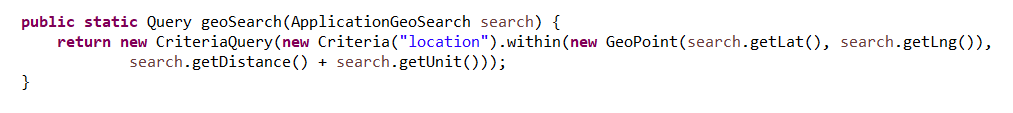
Поље је типа *GeoPoint*, што значи да се у њему чувају координате локације – географска ширина и дужина. Користећи њега, касније се могу претраживати документи којима та локација упада у зону дефинисану другом геолокацијом и радијусом. То значи да је приликом креирања нове пријаве и претраге креираних пријава потребно проследити Backend-у не само назив града, већ и његове позиције на мапи. За то се могу користити различити API-и, ја сам се одлучила за Algolia API (<https://www.algolia.com>). Он омогућава да се за одређено input поље на Frontend апликацији региструје autocomplete са градовима/адресама који ће додатно враћати географску ширину и дужину за одабрано место. На тај начин, кандидат када шаље нову пријаву, унеће своју адресу, али ће се на Backend послати и његова геолокација. Исто важи и за претрагу пријава, запослени у HR служби уносиће назив града, а на Backend ће се слати и геолокација. Изглед Algolia autocomplete-a приказана је на [слици 13](#слика_13).

Graphical user interface, application

Description automatically generated

Слика 13. Algolia autocomplete

Када се на Backend пошаље захтев за геопросторном претрагом, у чијем телу се налази геолокација задатог града и радијус, претрага се може извршити на начин дефинисан сликама [14](#слика_14) и [15](#слика_15).



Слика 14. Креирање Query-a за геопросторну претрагу

Text

Description automatically generated

Слика 15. Прослеђивање Query-a ElasticsearchRestTemplate-у и преузимање резултата

На [слици 14](#слика_14) приказано је креирање Query-a који ће вршити претрагу по геолокацији. Из захтева за претрагом преузимају се географска ширина и дужина, радијус и мерна јединица у којој је радијус изражен и од њих се креира *CriteriaQuery* који дефинише геопросторну претрагу. Потом се креиран Query прослеђује ElasticsearchRestTemplate-у ([слика 15](#слика_15), атрибут template је инстанца класе ElasticsearchRestTemplate). ElasticsearchRestTemplate мапира прослеђен Query на REST позив који тражи од Elasticsearch-a да добави све пријаве којима локација апликанта упада у задату зону. Резултат претраге се преузима и шаље назад кориснику на преглед.

## Складиштење у релационој бази и на File System-у

У релационој бази Backend апликације складиште се подаци који се не индексирају, односно за које није неопходна брза претрага. То су табеле са регистрованим корисничким улогама, регистрованим корисницима и регистрованим огласима за посао. Корисничка улога дефинисана је id-ом и називом. Корисник је дефинисан id-ом, email-ом и лозинком којом се пријављује и страним кључем која показује на корисничку улогу коју поседује. Оглас за посао је дефинисан id-ом, насловом, описом и датумом објаве. Поред ових података, у релационој бази се чувају и подаци о пријавама за посао. Поред тога што се они индексирају ради брзе претраге, чувају се и у релационој бази, како би се омогућило реиндексирање ради потребе. Поља која се индексирају а не чувају у релационој бази су *cvText*, *letterText* и *adTitle*. Поља *cvText* и *letterText* имају слабу структуру те их није прикладно чувати у релационој бази. Поље *adTitle* није потребно чувати у табели, будући да постоји страни кључ за оглас за посао.

У релационој бази и индексу се не чувају достављени PDF фајлови, већ само њихове локације на диску (поља *cvLocation* и *letterLocation*). Сами PDF фајлови складиште се на File System-у, унутар resources фолдера Spring апликације. У индексу се, поред локације на диску, чува и садржај фајлова (поља *cvText* и *letterText*), како би се омогућила претрага по садржају. Екстраковање садржаја PDF фајлова урађено је коришћењем готових библиотека. Уколико се деси потреба за реиндексирањем, садржај се може поново извући из фајлова складиштених на диску. Осим PDF фајлова, на диску се чува и фајл са логовима. У њега Backend апликација уписује логове који ће се накнадно читати, обрађивати и смештати у Elasticsearch ради даље анализе.

## Конфигурација Filebeat-a

Према спецификацији неопходно је омогућити увид у две значајне статистике:

1. Преглед из ког града је било највише приступа форми за пријаву апликанта
2. Преглед у које време (доба дана) је било највише пријава

Будући да је при имплементацији претходно наведених статистика неопходно користити ELK Stack, решење мора поседовати генерисање логовање, читање и обрада тих логова и напослетку смештање у Elasticsearch. Након што се логови обраде и нађу у одговарајућем индексу, могуће их ја визуализовати и анализирати коришћењем Kibana-e.

Пошто је неопходно анализирати две значајне статистике у систему, то ће значити да ће се генерисати две значајне групе логова. Логове ће генерисати Backend апликација сваки пут када се деси одговарајућа акција – када се приступи форми за пријаву апликанта или се достави нова пријава. Логови ће се смештати у одговарајући фајл, и то у формату приказаном на [слици 16](#слика_16).

A picture containing calendar

Description automatically generated

Слика 16. Фајл са генерисаним логовима

Са слике се може уочити да се у фајл смештају две врсте логова – логови који се генеришу када се приступи форми за пријаву и логови који се генеришу када се нова пријава достави. Генерисане логове неопходно је прво прочитати, а након тога на одговарајући начин обрадити и сместити у Elasticsarch.

Filebeat је алат који служи за читање логова складиштених у File System-у и њихово слање на одговарајуће одредиште. У конкретном случају, логови ће се читати из фајла дефинисаног на [слици 16](#слика_16) и слати на даљу обраду Logstash-у.

Верзија Filebeat-a коју сам преузела са официјалног сајта је 7.4.0. Неопходно је преузети баш ту верзију, како би била компатибилна са верзијом Logstash-a, који ће исто морати да има верзију 7.4.0 како би био компатибилан са верзијом Elasticsearch-a. Распаковала сам зип фајл и позиционирала се у коренски директоријум. Отворила сам *filebeat.yml* фајл и изменила га тако да му садржај буде као на [слици 17](#слика_17).

Graphical user interface, text

Description automatically generated

Слика 17. Конфигурациони фајл за Filebeat

У *inputs* делу фајла са [слике 17](#слика_17) дефинише се извор са којег се читају логови. У конкретном случају, у питању је фајл у који Backend апликација генерише логове од значаја. Мора се навести апсолутна путања фајла како би читање радило исправно. У *outputs* делу дефинише се где се прочитани логови даље шаљу. У конкретном случају, логови се шаљу у Logstash на даљу обраду.

## Конфигурација Logstash-a

Након што су логови прочитани, неопходно их је обрадити и сачувати у Elasticsearch-у. За то се користи Logstash. Logstash је алат који такође служи за читање логова, али, за разлику од Filebeat-a, омогућава и њихови обраду, парсирање и сл. пре него што их сачува на одговарајуће одредиште. У складу са тим, логови складиштени на File System-у се углавном читају коришћењем Filebeat-a и потом шаљу Logstash-у да их додатно обради пре слања Elasticsearch-у. У конкретном случају, из логова је неопходно извући податке од значаја – град/локација из којег се приступило форми за пријаву и време/доба дана када је пријава достављена. Када се ти подаци извуку из логова, могу се послати Elasticsearch-у да их сачува у одговарајући индекс и након тога се могу анализирати коришћењем Kibana-e.

Верзија Logstash-a коју сам преузела са официјалног сајта је 7.4.0. Као што је већ речено, мора се преузети баш та верзија како би била компатибилна са верзијом Elasticsearch-a који се користи. Распаковала сам зип и позиционирала се у *./bin* директоријум. Унутар њега креирала сам фајл *logstash.conf* са садржајем као на [слици 18](#слика_18).

Text

Description automatically generated

Слика 18. Конфигурациони фајл за Logstash

У *inputs* делује са [слике 18](#слика_18) дефинише се извор са којег се примају логови. У конкретном случају, логови се примају од Filebeat-a који их шаље на порт 5044 (на слици 17 може се уочити да Filebeat шаље логове Logstash-у на порт 5044). У *filter* дефинише се како се логови обрађују. Будући да постоје две значајне групе логова који се анализирају, тако у *filter* делу постоје две секције за парсирање логова. Прва секција проверава да ли лог има формат који се генерише када се приступи форми за пријаву, а друга секција проверава да ли лог има формат који се генерише када се достави нова пријава. Формат тих логова дефинисан је на [слици 16](#слика_16). Обе групе логова садрже унутар себе датум догађаја, време догађаја, нит унутар које је генерисан лог, тип лога, извор односно метода унутар које је генерисан лог и додатни садржај који се разликује у зависности од типа лога. Уколико је у питању лог који се генерише приликом приступа форми, додатан садржај је назив града из којег је апликант приступио форми и његове координате. Уколико је у питању лог који се генерише када се достави нова пријава, додатан садржај је само порука која наглашава да је пријава достављена. У зависности од тога, Logstash на различите начине парсира другачије логове, односно различите податке извлачи из њих. У случају логова прве групе, подаци од значаја су сам град и његове координате, зато се они извлаче. У случају логова друге групе, подаци од значаја су доба дана када се догађај догодио, зато сам се одлучила да извлачим датум догађаја и у колико се сати догодио догађај тог дана као један податак. Тај податак је означен са *date\_time* у другом *match* делу на [слици 18](#слика_18). Након што се из логова извуку подаци од значаја, врши се одбацивање вишка поља генерисаних од стране Filebeat-a и самог Logstash-a. То су поља *ecs, host, agent, log, type, @version, tags, input, source, lat, thread, lon* и *level*.

Након што се логови прочитају и обраде, врши се њихово складиштење у Elasticsearch-у. Логови прве групе уписују се у индекс са називом *logstash-1*, логови друге групе уписују се у индекс са називом *logstash-2*. На тај начин, логови се накнадно могу независно анализирати и визуализовати унутар Kibana-e. Додатно, логови се исписују на стандардни излаз, како би се сам debug режим олакшао.

## Конфигурација Kibana-e

Након што сам конфигурисала Filebeat и Logstash, генерисала сам пар логова прве и друге групе и сачекала да се прочитају и обраде. Након тога отишла сам на линк <http://localhost:9200/_cat/indices> како бих проверила да ли су индекси *logstash-1* и *logstash-2* успешно креирани. Одговор који сам добила приказан је на [слици 19](#слика_19).

Text

Description automatically generated

Слика 19. Списак индекса креираних унутар Elasticsearch-a

Као што се може видети, индекси су успешно креирани и унутар њих се налази 7 односно 3 докумената. То се слаже са бројем логова генерисаних унутар фајла са [слике 16](#слика_16). Како је потврђено да су логови успешно смештени у Elasticsearch, могуће их је даље анализирати и визуализовати коришћењем Kibana-e.

Верзија Kibana-e коју сам скинула са официјалног сајта је 7.4.0. Из сличног разлога као и пре, мора се преузети баш та верзија, како би била компатибилна са верзијом Elasticsearch-a. Распаковала сам зип и позиционирала се у *./bin* директоријум. Покренула сам *kibana* команду и сачекала да се сервер подигне. Након што се сервер покренуо, отишла сам на *localhost:5601* адресу и добила одговор да је Kibana је успешно покренута и да се може прећи на анализу складиштених логова.

Креирала сам два Index Pattern-a – pattern *logstash-1\** за прву групу логова и pattern *logstash-2\** за другу групу логова. Након тога сам на страници за приказ Index Pattern-a одабрала први pattern и приказао ми се садржај као на [слици 20](#слика_20).

Graphical user interface, text, application, email

Description automatically generated

Слика 20. Анализа логова прве групе

Са леве стране може се очитати који број/проценат приступа форми се односи на који град (у конкретном случају колико приступа је било из Новог Сада, а колико из Париза, будући да сам тестирала достављање пријава од стране два различита корисника, једног из Новог Сада, другог из Париза). Такође, може се уочити да подаци садрже поље *geoip.location* које је типа *geo\_point*, што значи да се може визуализовати на мапи. Кликом на дугме Vizualize, отвара се страница као на [слици 21](#слика_21). На њој је сваки град из којег се десио приступ форми за пријаву означен на мапи и то кругом чија величина је сразмерна броју приступа из тога града. У конкретном случају, број приступа из Новог Сада је већи него што је то случај код Париза, у складу са тим је и круг око тог града на мапи већи. Преласком миша преко круга приказује се тачан број приступа из тог града као и сама локација града.

Map

Description automatically generated

Слика 21. Приказ логова прве групе на мапи

На графику са [слике 20](#слика_20) приказан је број приступа форми у зависности од времена, односно *@timestamp* поља. Ово је омогућено, будући да сам приликом креирања Index Pattern-a одабрала да Time Filter Field буде управо *@timestamp* поље. Такође, могуће је одабрати преглед докумената унутар *logastash-1* Index Pattern-a у JSON формату ([слика 20](#слика_20), доле десно). Може се уочити да логови складиштени унутар Elasticsearch-a имају поља *message, @timestamp, date, time, city* и *geoip.location*.

Анализа логова друге групе ради се слично као и код прве групе логова. На страници за преглед Index Pattern-a одабрала сам *logstash-2\** pattern и добила приказ као на [слици 22](#слика_22). Слично као и раније, са леве стране може се видети у које време/доба дана је било највише достављених пријава. На графику се може анализирати број пријава у зависности од времена, а испод њега се може одабрати да се документ прикаже у JSON формату и тако прегледа списак свих његових поља. Са слике се види да складиштени логови имају поља *message, @timestamp, date\_time* и *time\_extra*. Поље *time\_extra* представља додатну информацију о минутима и секундима када је пријава достављена.

Graphical user interface, text, application, Word

Description automatically generated

Слика 22. Анализа логова друге групе