Zbornik radova Fakulteta tehničkih nauka, Novi Sad

UDK: 004.738.1:766
DOI: https://doi.org/10.24867/20BE52Mosorinski

IMPLEMENTACIJA PROŠIRENJA INTELLIJ RAZVOJNOG OKRUŽENJA ZA PODRŠKU RADA SA ELASTICSEARCH PLATFORMOM

PLUGIN IMPLEMENTATION OF THE INTELLIJ DEVELOPMENT ENVIRONMENT TO SUPPORT WORK WITH THE ELASTICSEARCH PLATFORM

Veljko Mošorinski, *Fakultet tehničkih nauka, Novi Sad*

Oblast – PRIMENJENE RAČUNARSKE NAUKE I INFORMATIKA

Kratak sadržaj –*U radu je predstavljeno proširenje IntelliJ razvojnog okruženja za podršku rada sa ElasticSearch platformom. Implementirano je rešenje koje pruža grafički korisnički interfejs i komunicira sa ElasticSearch serverom koristeći njegov REST API. U ovom radu se mogu pronaći potrebne tehnologije za sistem poput opisanog, specifikacija, implementacija kao i korisničko uputstvo. Na samom kraju, prodiskutovane su i prednosti i mane, kao i moguća proširenja koja bi donela dodatne benefite korisnicima.*

Ključne reči: *IntelliJ, ElasticSearch, REST API, specifikacija, implementacija.*

Abstract – *This paper describes and presents a plugin for IntelliJ development environment using a Elasticsearch platform. Implemented solution provides a graphical user interface and communicates with ElastcSearch server using his REST API. In this paper are presented needed technology for a system like this, specification, implementation and user manual as well. At the very end, advantages and disadvantages were discussed, as well as possible extensions that would bring additional benefits to users.*

Keywords: *IntelliJ, ElasticSearch, REST API, specification, implementation.*

1. UVOD

Razvitak informacionih tehnologija i sve češća upotreba u različitim sferama života i mnogim strukama koje i nisu srodne pomenutoj, dovodi do stvaranja velikih količina podataka i velikih sistema koji obrađuju te podatke. Efikasnost, egzaktnost, sigurnost su svakako zahtevi koje treba ispuniti. Ograničen broj računarskih resursa i visoke cene istih dovode do razmišljanja kako optimizovati rešenja i dati brze i validne odgovore korisniku u realnom vremenu.

Danas, kada se inženjeri svakodnevno susreću sa robusnim projektima koji sadrže velike količine koda ili bilo kakve tekstualne fajlove sa različitim semantikama, poželjno je olakšati razvitak i održavanje sa različitim vrstama softvera ili delova softvera.

Pretraga fajl sistema, grupacije poruka, logova sistema, delova koda, obrada i odgovori na određene upite su neizostavni deo razvoja. Rešenja prikazano u ovom radu jeste ekstenzija IntelliJ okruženja koja olakšava i ubrzava razvoj aplikacija čiji zahtevi pripadaju skupu prethodno navedenih problema koristeći ElasticSearch platformu.

Elastcsearch je server koji služi za pretragu i analizu podataka. Napisan je u Java programskom jeziku, što omogućava pokretanje na svim platformama. Baziran je na Lucene indeksima i omogućava korisnicima da pretraže veliku količinu podataka vrlo brzo. Može se koristiti i za čuvanje podataka ali je njegova glavna uloga indeksiranje i pretraga podataka u realnom vremenu.

2. SLIČNA REŠENJA

Za sistem poput implementirane ekstenzije je očekivano da će biti jedno od rešenja i da će postojati još mnoštvo sličnih implementacija. Obzirom da je Elasticsearch jedno od najpoznatijih i najkorišćenijih rešenja za pretragu velikih količina podataka, to se nije moglo izbeći. Postoje razne ekstenzije za Intellij IDEA okruženje pomoću kojih se može vršiti komunikacija sa ELK stekom. Ukoliko je reč o drugim ekstenzijama, neke od njih su:

- **Elasticsearch** [1]– omogućava konektovanje sa Elasticsearch i Kibana serverom, pretraživanje i modifikaciju podataka, kao i izvršavanje različitih zahteva ka REST API-u
- **Cap-Elasticsearch-Client** [2]– slično prethodnom, razlikuje ga to što ne može da komunicira sa Kibana serverom i omogućava izvršavanje SQL upita
- **Elasticsearch Query** [3]– omogućava izvršavanje upita i tabelarni prikaz podataka
- **Cap-Elasticsearch** [4]– vrlo sličan Kibana klijentskom delu

Bitna razlika između napomenutih ekstenzija i opisane u ovom radu je to što sve navedene zahtevaju detaljno poznavanje REST API-a Elasticsearch-a, dok to u ELP-u nije slučaj. Većina poziva je apstrahovana i prilagođena bilo kom korisniku, pa čitanje dokumentacije i upoznavanje sa radom nije neophodno. Takođe, postoji mnoštvo funkcionalnosti koje predstavljaju obradu podataka na klijentskoj strani koje server pruža.

Ukoliko je reč o rešenjima koja nisu u direktnoj vezi sa Intellij okruženjem, najbitnije je spomenuti Luke. Ova aplikacija većinski obuhvata funkcionalnosti ekstenzije.

NAPOMENA:
Ovaj rad proistekao je iz master rada čiji mentor je bio dr Dragan Ivanović, red. prof.

Kao i ELP ekstenzija, Luke sadrži funkcionalnosti:

- Prikaz metapodataka indeksa
- Manipulisanje dokumentima
- Razni tipovi pretrage
- Analiza teksta

3. OPIS KORIŠĆENIH TEHNOLOGIJA

Za dostignuće krajnjeg cilja bilo je neophodno koristiti više različitih tehnologija. Pre svega, kod je napisan u *Java* programskom jeziku, verzije 18 sa Swing, AWT i Java Http Client paketima i bibliotekama, koristeći *IntelliJ Platform Plugin SDK* koji pruža sve neophodne biblioteke za razvoj ekstenzija *IntelliJ IDEA* okruženja. Analiza podataka i podrška svih funkcionalnosti ove ekstenzije se vrši pomoću *Elasticsearch* pretraživača koji je zasnovan na *Lucene* biblioteci [7].

3.1 Java

Java programski [5] jezik je objektno orijentisani jezik visokog nivoa sa širokom primenom. Razvijen je početkom 1990-ih od strane Sun Microsystems kompanije. Ono što Javu čini specifičnom jeste nezavisnost od hardvera i olakšano upravljanje memorijom u odnosu na C-ovske jezike. Pretpostavlja se da je upravo to dovelo do razvoja jezika sa totalno dugačijim principom – želelo se nešto drugačije od popularnog C-a.
Java je, uz Kotlin, zvanično podržan jezik za implementaciju Android mobilnih aplikacija.

Neke od najznačajnijih karakteristika su:

- Jednostavnost – sistem u kome se lako programira bez potrebe za komplikovanim uhodavanjem
- Objektno orijentisan – u Javi je sve predstavljeno kao sadržaj klasa
- Distribuiranost – poseduje biblioteke za rad sa TCP/IP protokolima. Aplikacije mogu da pristupaju objektima preko mreže i preko URL-a sa, podjednako kao lokalnom sistemu datoteka
- Robusnost - namenjena za pisanje programa koji moraju biti pouzdani sa dosta aspekata
- Neutralnost – kompajler stvara objektne datoteke nezavisnog sadržaja u odnosu na operativni sistem
- Dinamičnosti – prilagođena okruženju koje se stalno menja, implementacije su lako proširive novim funkcionalnostima

Izvršavanje aplikacija pisanih u Java programskom jeziku obezbeđuje Java virtuelna mašina. Osim njih, moguće je izvršavanje i programa pisanih u drugim programskih jezicima koji su iskompajlirani u Java bajt kodu. Primeri jezika koji mogu biti izvršavani u JVM-u su:

- Javaskript
- Paskal
- PHP
- Pajton
- Rubi

3.2 ElasticSearch

Elasticsearch [6] je pretraživač baziran na *Lucene* biblioteci. Pruža distribuirano pretraživanje teksta od strane više korisnika, sa HTTP veb interfejsom u dokumentima bez šeme.

Prva verzija je objavljena februara 2014. godine, a verzija 7 je korištena u svrhe testiranja projekta, objavljena aprila 2019. godine. Pretraživač je napisan u Javi, a njegovi klijenti su dostupni u jezicima Python, .NET, PHP, Java. Po DB-Engines rangiranju, Elasticsearch je najpopularniji pretraživač.
Elasticsearch je jedan od elemenata ELK steka, kome pripadaju:

- **Elasticsearch**
- **Kibana** – interfejs za vizuelizaciju podataka
- **Logstash** – interfejs za preprocesiranje podataka na strani servera

Bilo kakva vrsta dokumenata može biti pretražena putem Elasticsearch pretraživača. Omogućava skalabilnu pretragu u realnom vremenu koristeći njegov REST API. Pristup funkcionalnostima je omogućen različitim grupacijama API-a, podeljenih prema osnovnim elementima podataka i funkcijama na koje se pretraživač oslanja, kao što su Index API, Document API, Search API, itd.
Za bolje razumevanje samog rada ovog pretraživača i strukture koja čuva semantiku i vrednost podataka, neophodno je objasniti osnovne elemente.

4. SPECIFIKACIJA SISTEMA

Ekstenzija za rad sa indeksima je grafičko rešenje koje omogućava olakšani rad sa REST API-em Elasticsearch-a i pruža razne funkcionalnosti koje možemo podeliti u dve osnovne grupacije:

1. Indeksiranje fajlova u zadatom direktorijumu
2. Rad sa indeksiranim podacima

Indeksiranje određenog direktorijuma omogućava crawler implementiran unutar ekstenzije. Prolazi rekurzivno kroz sve poddirektorijume zadatog direktorijuma i indeksira svaki pronađeni dokument, očitavanjem vrednosti njegovih atributa i sadržaja fajla.

Prva funkcionalnost nije preduslov druge funkcionalsti, već predstavlja jednu od mogućnosti korišćenja.

Takođe, moguće je parametrizovati sistem da koristi indeks koji nije kreiran korišćenjem internog crawlera. Funkcionalnosti su implementirane generički i moguće je prikazivati podatke i slati zahteve ka bilo kom indeksu kreiranom unutar Elasticsearch-a, što proširuje namenu ovog sistema i otvara mogućnosti pretrage različitih skupova podataka.

5. ARHITEKTURA SISTEMA

Struktura i celine ovog sistema lako su uočljive posmatrajući pakete u kojima je implementacija raspoređena, a to su:

- communication
- configuration
- crawler
- factory
- gui
- handler
- model

5.1 Paket communication

Zbog različitosti komunikacije sa različitim Elasticsearch servisima i podacima koji se šalju, neophodno je bilo enkapsulirati kompletnu komunikaciju i izdvojiti u jednu celinu. Svaki HTTP poziv usmeren ka serveru je enkapsuliran u okviru jedne Java metode i omogućen univerzalni način korišćenja zasebnih funkcionalnosti. Cilj ovog paketa jeste apstrahovanje i odvajanje logike same komunikacije od ostatka sistema, a sa druge strane, stvaranje univezalnog načina razmene poruka sa serverom iz svih delova aplikacije, kao i pojednostavljena obrada grešaka, ukoliko do njih dođe.

5.2 Paket crawler

U paketu crawler se nalazi sve neophodno za kreiranje novog indeksa i indeksiranje određenog dela fajl sistema. Predstavlja jednu odvojenu celinu u okviru sistema koja većinski odvojeno od sistema funkcioniše, koristeći paket za komunikaciju.

5.3 Paket factory

Da bi se ekstenzija kreirala, neophodna je bila fabrika koja će inicijalizovati početno stanje aplikacije i proizvesti njene objekte za egzistenciju. Upravo to se nalazi u paketu factory.

5.4 Paket gui

Grafički korisnički paket je neizostavni element sistema poput implementirane ekstenzije. Sadrži sve grafičke elemente koje aplikacija poseduje.
Paket je podeljen na 3 podpaketa:

- dialog – sadrži modalne dijagole(prozori koji imaju konkretnu funkcionalnost i pokreću se kao reakcija na određene akcije u okviru glavnog prozora)
- panels – svi strukturno najzahtevniji paneli u okviru Sistema
- window – glavni prozor aplikacije

5.5 Paket handler

Sistem koji komunicira sa serverom i na zahteve dobija odgovore od servera zahteva posebnu pažnju pri obradi rezultata akcija(odgovora), generisanjem obaveštenja za uspešnost posla zahtevanog od korisnika, kao i obradu određenih izuzetaka. Paket handler upravo ima tu ulogu, da pruži funkcionalnosti za obradu rezultata zahteva.

5.6 Paket model

Paketu modela podataka pripada "standardizacija" informacija koje sistem koristi. Podeljen je u dva podpaketa:

- data – model podataka koji se razmenjuju tokom komunikacije
- table – model podataka tabela koje se prikazuju u glavnom prozoru

6. IMPLEMETANCIJA SISTEMA

Izvorni kod se sastoji od više skupova istorodnih klasa, poredivši po osnovnoj ideja biznis logike njihove implementacije. Da objašnjenja ne bi bila redudantna i ponavljala se, biće prikazan po jedan primer iz svakog skupa.

6.1 Komunikacija sa ElasticSearch serverom

Krajnja tačka ka Elasticsearch serveru iz perspektive sistema je Elasticsearch klasa u okviru *communication* modula. U ovoj klasi su implementirane metode http zahteva kao što su:

- GET
- POST
- DELETE
- PUT

Svaka od navedenih metoda koristi funkcionalnosti *HttpClient* klase iz *java.net.http* paketa da bi kreirala http zahtev i poslala ga.

6.2 Konfiguracija

Da bi korisnik mogao da menja podešavanja vezana za Elasticsearch i da ta podešavanja ne bi bila vezana isključivo za određenu sesiju, neophodno je bilo uvesti konfiguraciju. Implementirana konfiguracija parametrizuje 3 stvari:

- Adresu Elasticsearch servera
- Port Elasticsearch servera
- Indeks koji se trenutno koristi

ConfigProvider klasa omogućava sve navedeno implementirajući *ApplicationComponent* i *PersistentStateComponent* interfejse iz Intellij openapi paketa namenjenog za razvitak ekstenzija

6.3 Pretraživač fajl sistema

Jedna od komponenti koja se semantički potpuno razlikuje od ostatka sistema jeste pretraživač implementiran u klasi Crawler. Predstavlja zasebnu celinu jer je njegov cilj kreiranje indeksa i popunjavanje podacima, dok ostatak sistema koristi taj indeks i analizira podatke. Suština ove komponente se ogleda u rekurzivnom prolasku kroz određeni deo fajl sistema i indeksiranju atributa i sadržaja svakog fajla na koji se naiđe. Indeksiranje sadržaja svakog ne

garantuje i prikaz sadržaja u okviru sistema jer bi to dovelo do zauzimanja velike količine memorije u RAM-u i usporavanje sistema ili indeksiranje samo manjih kolekcija fajlova.

7. ZAKLJUČAK

U ovom radu je opisana ideja implementacija ekstenzije okruženja za razvoj softvera Intellij IDEA. Konačan broj funkcionalnosti svakako otvara mogućnost za daljim razvitkom i dodavanjem novih opcija koje bi bile korisne i interesatne korisnicima. Cilj ovakvog sistema jeste da pruži olakšanu pretragu velikih kolekcija podataka, što ne isključuje i pretrage podataka manjeg obima. Tokom razvitka, posebna pažnja se posvećivala strukturiranoj implementaciji i kreiranjem što manje povezanih komponenti, radi olakšanog proširivanja i impementacije novih funkcionalnosti. Određenim delovima, poput komponente za komunikaciju sa Elasticsearch serverom, se moglo posvetiti više pažnje i naći drugačije rešenje za kreiranje tela upita. Trenutno rešenje je vezano isključivo za određene verzije API-a, što iziskuje promenu implementacije prilikom prelaska na noviji Elasticsearch. Izbor neke od biblioteka za komunikaciju bi svakako bilo bolje rešenje od generisanja upita na način implemetiran u aplikaciji, ali bi uvelo nove probleme zbog specifičnosti zahteva u samoj aplikaciji. Potpuno održivo rešenje, sa kojim promena implementacije ne bila potrebna pri prelasku na drugačiji API nije poznata.
Poboljšanje implementacije je moglo biti ostvareno u još jednom smeru. Nezavisnost komponenti je postojana, ali je poželjna u većoj meri od trenutne. Proširenje trenutnih dodavanjem novih funkcionalnosti je poprilično izvodljivo, ali dodavanje novih komponenti zahteva veći broj promena. Jedan primer jeste veza između modela podataka i grafičkog korisničkog interfejsa. Ukoliko se podaci promene, potrebno je direktno pozvati određene metode da bi korisnik primetio novine. Jedna od ideja za poboljšanje jeste korišćene nekih od dizajn paterna, kao što je Obzerver. Na ovaj način komponente bi dobijale notifikacije o promenama u ostalim komponentama, a same bi bile zadužene za reakciju na date promene. Što omogućuje da svaka komponenta ne mora da poznaje biznis logiku ostalih, već ima samo jedan zadatak, a to je da javno obavesti ostale da je došlo do promene unutar nje. Samim tim, zavisnost u sistemu se smanjuje i izmene i dodavanje novih fukncionalnosti se pojednostavljuje.
Potrebe prilikom pretrage skupa informacija su raznovrsne, samim tim postoje funkcionalnosti koje će određeni korisnici više koristiti, određeni manje, ali postoje i one koje možda nisu implementirane.
Iz tog razloga, korisno bi bilo proširiti sistem da korisnik može da generiše sam upite i šalje ih serveru, da se pruži funkcionalnost sa kojom bi moguće bilo pozvati bilo koji Elasticsearch API.
Ako se sve informacije sumiraju, ova ekstenzija će pružiti korisniku osnovnu i naprednu pretragu dokumenata po različitim kriterijumima, kao i izmenu kolekcije. Svakako da ne može pokriti sve potrebe svih korisnika, ali je podobna promenama i doradama da bi se adaptirala dokumentima posebnog sadržaja i posebnim analizama.

8. LITERATURA

[1] ElasticSearch Intellij IDEA plugin
https://plugins.jetbrains.com/plugin/14512-elasticsearch
[2] Cap-ElasticSearch Client plugin
https://plugins.jetbrains.com/plugin/16111-cap-elasticsearch-client
[3] Query plugin
https://plugins.jetbrains.com/plugin/16364-elasticsearch-query--edql
[4] Cap-ElasticSearch plugin
https://plugins.jetbrains.com/plugin/17692-cap-elasticsearch
[5] Java
https://docs.oracle.com/javase/11/docs/api/
[6] ElasticSearch
https://www.elastic.co/
[7] Lucene
https://lucene.apache.org/

Kratka biografija:

Veljko Mošorinski rođen je 29.05.1996. godine u Zrenjaninu. Godine 2015. upisao je Fakultet Tehničkih Nauka u Novom Sadu, odsek Računarstvo i automatika. 2017. upisuje usmerenje Primenjene računarske nauke i informatika. U septembru 2019. godine je diplomirao i upisao Master akademske studije.

Kontakt: vmosorinski@gmail.com