Internet je ogromno more prepuno informacija. Problem pronalaska odgovarajućeg skupa informacija je skoro nerešiv bez upotrebe pomoćnih alata, odnosno, mehanizama pretrage.

***Istorijat***

Posmatrajući istoriju mehanizama pretrage, primećuje se jedna vrlo zanimljiva činjenica, a to je da su osnivači jednih od najpoznatijih mehanizama bili upravo studenti.

Alan Emtage, administrator sistema za departman informacionih tehnologija i student kanadskog Univerziteta McGill, u septembru 1990. godine objavljuje prvi Internet mehanizam pretrage pod nazivom *Archie*.

Nekoliko godina i mehanizama pretrage kasnije, apsolutnu dominaciju u Internet svetu ima *Yahoo*. Njegovi kreatori su studenti Stanford Univerziteta, Jerry Yang i David Filo.

1998. godine dvojica studenata, Larry Page i Sergey Brin, koji su takođe sa Stanford Univerziteta odlaze u *Yahoo* sa ponudom da im prodaju svoj biznis za milion dolara, međutim bili su odbijeni. Ono što tada *Yahoo* nije znao jeste da je odbio budućeg giganta koji i dan danas vlada Internetom

Uzmiajući u obzir prednosti korišćenja mehanizama pretrage i mnogobrojne projekte i pitanja koje studenti postavljaju prilikom njihove izrade, dolazi se do DebugIt-a, platforme koja obuhvata sva implementaciona i diskusiona pitanja studenata, nezavisno od predmeta i tehnologija. Na ovakav način se postiže:

* Svaki student može da pogleda bilo koje pitanje i njegove odgovore, bez obzira na to kada je postavljeno.
* Studenti, asistenti i profesori mogu da odgovore na postavljena pitanja
* Najčešći problemi koji se javljaju tokom izrade projekata će vrlo brzo imati odgovore i neće biti potrebe za dupliranjem istih pitanja.
* Studenti se navikavaju da rešenje koje su pronašli prilagode svom kodu.

***Postojeća rešenja***

Nabrojati samo ono sa slajda.

***Elasticsearch mehanizam pretrage***

*Elasticsearch* je distribuirani, *open source* mehanizam za pretragu i analitiku, čiji temelj predstavlja *Apache Lucene* bibilioteka. Omogućava skladištenje, pretragu i analiziranje velike količine podataka u realnom vremenu.

*Elasticsearch* nastao je kada je Shay Banon želeo da kreira mehanizam pretrage koji bi njegova žena mogla da koristi za svoje recepte za kuvanje.

***Struktura Elasticsearch mehanizma pretrage***

Nabrojati samo kao na slici.

***Index***

*Elasticsearch* koristi svoju ugrađenu NoSQL bazu koja podatke čuva kao JSON dokumente. Dokumenti predstavljaju osnovnu jedinicu podatak koju je moguće grupisati u indeks (*eng. index*), koncept sličan tabeli kod relacionih baza podataka. Da bi se dokumenti grupisali u indeks, moraju imati slične karakteristike.

Postoje 2 tipa strukture podataka koje se koriste kod mehanizama pretrage u cilju čuvanja i organizovanja podataka:

* ***forward index*** - dokumenti se mapiraju na termine koje sadrže
* ***inverted index*** - termini se mapiraju na dokumente u kojima se nalaze. Elasticsearch koristi upravo ovu strukturu za smeštanje svojih podataka, jer je kod nje pretraga veoma brza, pošto se podaci unapred čuvaju kao upiti.

***Shard***

*Shard* predstavlja osnovnu gradivnu jedinicu *Elasticsearch* distribuirane arhitekture. Indeks može da sadrži veoma veliku količinu podataka, a bi upravljanje podacima bilo što efikasnije, ti podaci se obično podele u nekoliko *shard* komponenti.

Postoje 2 tipa *shard* komponenti:

* **primarni** (*eng. primary*) - sadrže originalne podatke i obavljaju indeksiranje i pretragu.
* **kopije** (*eng. replica*) - predstavljaju kopiju primarnih *shard* komponenti.

***Segment***

Svaka *shard* komponenta se sastoji od više *segment* komponenti, koje imjau *inverted index* strukturu podataka.

Prilikom indeksiranja dokumenta otprilike svake sekunde se kreira novi *segment*.

Sadržaj *segment* komponente se ne može menjati.

Što je više *segment* komponenti, time je pretraga sporija. Zbog toga se vrši spajanje *segment* komponenti istih veličina, počev od manjih.

***Cluster i node***

*Node* predstavlja server na kome se izvršava jedna *Elasticsearch* instanca. Jedna ili više *node* komponenti povezanih zajedno čine *cluster*. Svaki *node* u okviru jedne *cluster* komponente zna za ostale *node* komponente, što omogućava preusmeravanje zahteva ka odgovarajućoj *node* komponenti.

*Node* komponente imaju svoje uloge (*eng. roles*) i na osnovu toga se razlikuje više node tipova: master-eligible node, data node, ingest node, remote-eligible node, machine-learning node, transform node, coordinating node.

***Način rada Elasticsearch mehanizma***

Kako korisnci znaju od svih ponuđenih dokumenata koji je najbolji, odnosno, koji najviše odgovara njihovom upitu? *Elasticsearch* je rešenje našao u tome da svakom dokumentu dodeli *score*. U zbirnom rezultatu se dokumenti ređaju u opadajućem redosledu na osnovu *score* vrednosti.

Moguće je koristiti različite tipove upita, odnosno prertrage, među kojima se najčešće koriste:

* **struktuirana pretraga** (*eng. structured search*)
* **nestruktuirana pretraga** (*eng. unstructured search*)
* **kombinovni upiti** (*eng. combining queries*)

***Text analysis***

Prilikom čuvanja i pretraživanja podataka, sva tekstualna polja prolaze kroz *text analysis* proces, koji ima dve funkcije:

* **tokenizacija** (*eng. tokenization*) - proces u kome *tokenizer* komponenta vrši deljenja teksta na tokene, najčešće reči, na osnovu nekog definisanog pravila.
* **normalizacija** (*eng. normalization*) - proces u kome se vrše transformacije tokena. Transformacije mogu biti različite, od *stemming* procesa do posmatranju sinonima, *stop words* i tako dalje.

*Analyzer* modul izvršava *text analysis* proces i sastavljen je od tri tipa komponenti kroz koje tekst prolazi u toku *text analysis* procesa:

* ***character filters***
* ***tokenizer***
* ***token filters***

*Stemming* je proces u kome se reč redukuje na svoj koren. Koren na koji je reč redukovana ne mora biti prava reč, odnosno, ne mora ni zaista biti koren te reči.

*Stemming token filters* su zaduženi za *stemming* i postoje dve vrste:

* ***algorithmic stemmers***
* ***dictionary stemmers***

*Stop words* predstavljaju skup reči koje ni na koji način ne doprinose pretrazi kako bi njen rezultat bio bolji, pa se iz tih razloga ignorišu. To su reči koje se često koriste u svakodnevno govoru, a za srpski jezik bi to bili: veznici, zamenice, predlozi, pomoćni glagoli i slično.

*Synonyms* predstavljaju jednu od ključnih stvari za uspešnu pretragu. Njihovim korišćenjem se poboljšava rezultat pretrage, jer je moguće naći i one dokumente koji sadrže sinonime umesto otkucanih reči u upitu.

*Scoring* mehanizam se zasniva na *BM25* algoritmu. Ovaj algoritam posmatra tri parametra prilikom određivanja *score* vrednosti dokumenta:

* ***term frequency (TF)***
* ***inverse document frequency (IDF)***
* ***field length normalization***

Prilikom pretrage, često se desi da korisnici pogrešno otkucaju reč ili nekoliko reči u celom upitu. Ipak, idalje žele da dobiju iste rezultate kao i u slučaju tačno otkucanih reči. *Elasticsearch* je to rešio korišćenjem *fuzzy query* koji koristi Levenštajnov algoritam udaljenosti (*eng. Levenshtein Distance Algorithm*) da bi izračunao udaljenost reči iz upita od onih koje se nalaze u *index* komponenti.

Levenštajnov algoritam udaljenosti računa distancu između reči tako što poredi karaktere reči redom. Za svaki karakter koji se razlikuje dodaje se vrednost 1 na ukupnu vrednost distance.

Jedna od čestih grešaka pri kucanju je transponovanje slova. Da se distanca ne bi računala kao +2 na trenutnu vrednost, koristi se transpositions parametar kojim se omogućuje da se prilikom transponovanja slova doda +1 na trenutnu vrednost.

***Zaključak***

Većina Internet sajtova ima neki oblik pretrage, za koju se najčešće koristi neki od mehanizama kako bi se njeni rezultati poboljšali. Osim dobijanja odgovarajućeg skupa informacija, jedan od glavnih zahteva je i brzina izvršavanja pretrage, čime se *Elasticsearch* ponosi.

*DebugIt* platforma integriše *Elasticsearch* mehanizam pretrage i uz pomoć nje studenti uče da istražuju i postavljaju pitanja kada im je potrebna pomoć, ali i da pomažu kolegama. Osim čitanja stvari relevantnih za problem koji imaju, studenti će zasigurno u toku svog istraživanja pročitati i neke dodatne informacije koje će im u nekom momentu biti korisne. Na takav način se dosta poboljšava proces učenja.

Svi studenti, asistenti i profesori jednog fakulteta čine jednu ogromnu zajednicu i kao takvoj joj je potrebno mesto u okviru koje će moći da se vode diskusije, što se upravo postiže korišćenjem *DebugIt* platforme.