Zbornik radova Fakulteta tehničkih nauka, Novi Sad

UDK: 004.738.1:766
DOI: https://doi.org/10.24867/15BE33Nikolic

ANALIZA GRAPHQL I REST API-IJA ZA INFORMACIJE O AKADEMSKIM ENTITETIMA

ANALYSIS OF GRAPHQL AND REST API IN INFORMATION RETRIEVAL OF ACADEMIC ENTITIES

Dušan Nikolić, *Fakultet tehničkih nauka, Novi Sad*

Oblast – PRIMENJENE RAČUNARSKE NAUKE I INFORMATIKA

Kratak sadržaj – *U radu su analizirane performanse između GraphQL i REST API-ija prilikom pretrage informacionog sistema naučno-istraživačke delatnosti. Preuzeti su podaci o akademskim entitetima sa Elsevier Scopus platforme. Prikazana je arhitektura informacionog sistema sa dijagramom razmeštaja i dijagramom komponenti. Implementiran je GraphQL API i diskutovani su rezultati odziva prilikom pretrage datih entiteta u oba API-ija.*

Ključne reči: *informacioni sistemi naučno-istraživačke delatnosti, migracija, GraphQL, REST.*

Abstract – *This paper analyzes performance differences between GraphQL and REST API's information retreival of a current research information system. Data regarding academic entities has been taken from the Elsevier Scopus platform. Information system architecture is shown with deployment and component diagrams. In addition to recording the implementation of the GraphQL API, the present study also discusses the results of system response gathered during information retrieval of the given entities in both APIs.*

Keywords: *current research information systems, migration study, GraphQL, REST.*

1. UVOD

Predmet istraživanja ovog rada jeste analiza performansi između GraphQL i REST API-ija prilikom pretrage akademskih entiteta informacionog sistema naučno-istraživačke delatnosti. U radu je opisan proces migracije API-ija sa tradicionalnog REST stila na GraphQL.

Kako bi vršenje upita nad informacionim sistemom bilo moguće, preuzeto je preko 13 hiljada informacija o akademskim entitetima sa Elsevier *Scopus* platforme. Preuzete informacije su pretežno publikacije u časopisu, monografije radova i radovi objavljeni u zbornicima sa naučno-stručnih konferencija koje pripadaju autorima sa Univerziteta u Novom Sadu.

1.1. Informacioni sistemi naučno-istraživačke delatnosti

Kroz upotrebu informacionih sistema naučno-istraživačke delatnosti (eng. *Current Research Information Systems – CRIS*), institucije koje se bave naukom dobijaju sveobuhvatan pogled na aktivnosti i delatnosti datih istraživača u objedinjenoj bazi podataka. Ovakve aktivnosti obuhvataju proces prikupljanja, obrade i upravljanja informacijama o ljudima, publikacijama, patentima, tezama, opremi i projektima od istraživačkog značaja [1].

Postoje informacioni sistemi koji korisnicima olakšavaju proces pretrage citata i publikacija. Scopus, razvijen od strane *Elsevier* kompanije, predstavlja bazu podataka (eng. *abstract and citation database*) sa preko 75 miliona indeksiranih apstrakata i citata koji su dobijeni analizom publikacija, monografija radova i radova sa naučno-stručne konferencije objavljenih nakon 1966. godine [2] Funkcionalnosti koje Scopus nudi su: osnovna pretraga, pretraga po autoru, napredna pretraga, brza pretraga.

1.2. REST

REST (eng. *Representational State Transfer*) predstavlja stil softverske arhitekture namenjen distribuiranim sistemima. Stil arhitekture specificira ograničenja kao što su: način identifikacije resursa, uniforman interfejs sa određenom semantikom, samoopisive poruke, *stateless* interakcija.

API protokoli su se menjali i razvijali tokom vremena. Uzimajući u obzir sve veću dostupnost podataka, potražnja za fleksibilnijim i javno dostupnim API-ijima je takođe veća i REST je u industriji trenutno najzastupljeniji API protokol. Međutim, ograničenje ovakvog protokola je u tome što pozivi ka API-iju uglavnom zahtevaju više povezanih resursa, rezultujući u višestrukim zahtevima ka serveru. Ovakvo ograničenje otežava aplikacijama da se efikasno integrišu sa REST API-ijima, jer integracija podrazumeva pristup ka resursima iz više uzastopnih zahteva.

1.3. GraphQL

GraphQL je upitni jezik otvorenog koda razvijen od strane kompanije *Facebook*. Predstavlja upitni jezik nad API-ijem (eng. *Application Programming Interface*), namenjen za implementiranje veb servisa.

Osnovna karakteristika GraphQL-a je mogućnost kreiranja proizvoljnih upita ka serveru. Upiti su kreirani time što se na serverskoj strani implementira statička šema (eng. *schema*) nad bazom podataka. Moguće je implementirati i dinamičku (eng. *runtime*) šemu. Šema se kreira unapred i predstavlja pogled na bazu podataka,

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

odnosno API koji klijenti mogu da pozivaju. Ovakva implementacija upita je kontrast u odnosu na REST pristup. U REST sistemima se programski implementiraju operacije (eng. *endpoints*) koje klijenti mogu da pozovu, dok u GraphQL sistemima klijenti kreiraju upite nad „*export*-ovanom" bazom podataka.

1.4. Pregled relevantne literature

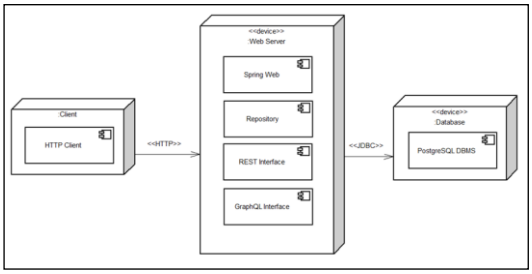
U radu [3] dokumentovan je process migriranja 7 GitHub API-ija sa REST-a na GraphQL. Predmet istraživanja u datom radu je pružanje odgovora na pet pitanja (eng. *research questions)*: 1) glavne karakteristike GraphQL-a i njegove prednosti; 2) nedostaci GraphQL tehnologije; 3) smanjenje broja poziva ka API-iju upotrebom GraphQL-a nasuprot REST pozivima; 4) smanjenje vraćenih polja od strane servera upotrebom GraphQL; 5) smanjenje veličine dokumenta vraćenog od strane servera upotrebom GraphQL. Odgovor na pitanje karakteristika GraphQL-a je dat analizom dokumentacije, ali i blogova koji izveštavaju o novim tehnologijama (eng. *grey literature*). Ističu se dve karakteristike GraphQL-a: hijerarhijski model podataka koji smanjuje potrebu za više *endpoint*-a i mogućnost da klijenti postavljaju proizvoljne upite ka serveru. Za potrebe faze analize performansi GraphQL-a, migrirano je pet GitHub REST projekata otvorenog koda i dva arXiv [4] projekta. Kao rezultat migracije dobijeno je 29 REST API poziva mapiranih na 24 GraphQL upita. Prema dobijenom broju *endpoint*-a, zaključak je da ne postoji značajno smanjenje broja poziva ka serveru upotrebom GraphQL-a Sa druge strane, demonstrirana je značajna razlika u broju polja i veličine dokumenta vraćenih od strane servera ove dve tehnologije. U poređenju sa pozivima REST stila, GraphQL implementacija smanjuje veličinu vraćenog JSON dokumenta za čak 94% u broju vraćenih polja i 99% u broju vraćenih bajtova [3].

Rad [5] prikazuje studiju migriranja dela sistema za upravljanje pametnom kućom na GraphQL API. U izveštaju evaluiraju se performanse na primeru brzine odziva dva *endpoint*-a koja su implementirana u REST i GraphQL. Prvi *endpoint* predstavlja primer poziva koji zahteva mali broj polja od strane servera. Ovakav upit autori nazivaju „atomičan" (eng. *atomic*) i nije uočena razlika u brzini odziva sistema nakon migracije na GraphQL. Drugi *endpoint* je složeniji i zahteva više različitih polja sa servera. Za upit nad drugim *endpoint*-om, GraphQL je 54% brži od REST ekvivalentnog poziva.

Rad [6] iz 2020. godine sprovodi kontrolisan eksperiment sa 22 studenta koji poseduju određen stepen znanja o veb programiranju. Cilj eksperimenta je da studenti implementiraju 8 operacija (*endpoints*) na određenom veb servisu koristeći REST i GraphQL. Rezultati eksperimenta su pokazali da vreme neophodno za implementaciju REST operacije raste sa porastom broja njegovih parametara. Medijana vremena potrebnog za implementaciju REST operacije je 9 minuta, dok za GraphQL iznosi 6 minuta. Kao rezultat, zaključeno je da su GraphQL operacije jednostavnije za studente koji nisu ranije imali susret sa datom tehnologijom. Prema medijanama pokazano je da je studentima potrebno manje truda za implementiranje GraphQL operacije.

2. SPECIFIKACIJA SISTEMA ZA INFORMACIJE O AKADEMSKIM ENTITETIMA

U ovom poglavlju prikazana je arhitektura sistema sa aspekta UML dijagrama razmeštaja i dijagrama komponenti. Dijagram razmeštaja (slika 1.) prikazuje softverske komponente i veze uz pomoć kojih komuniciraju date komponente. Dijagram komponenti na slici 2. prikazuje organizaciju i veze između komponenti aplikacije.

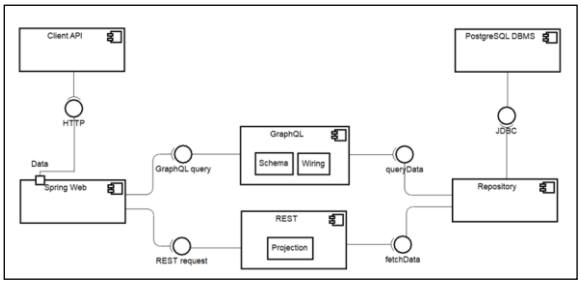


Slika 1. *Dijagram razmeštaja*

Client **–** sadrži komponentu *Http Client*. Komponenta *HTTP Client* može predstavljati proizvoljan veb čitač ili namenski softver za kreiranje HTTP poziva.

Web Server – sadrži komponente *Spring Web*, *Repository*, *REST i GraphQL Interface*.

Database **–** sadrži komponentu *PostgreSQL DBMS* kao izabrani sistem za upravljanje bazom podataka.



Slika 2. *Dijagram komponenti*

Prema dijagramu komponenti sa slike 2, klijentski HTTP zahtev stiže do *Spring Web* komponente, te se dalje usmerava ka GraphQL ili REST komponenti. Za potrebe GraphQL upita, data komponenta je sastavljena od *Schema* i *Wiring* dela (eng. *parts*). Zadatak *Schema* komponente je definisanje polja koja GraphQL API podržava.

Ta polja se kasnije mapiraju na bazu podataka informacionog sistema. Sa druge strane *Wiring* komponenta mapira pristigli klijentski upit na odgovarajući entitet iz baze podataka. REST komponenta se sastoji od *Projections* biblioteke, čiji zadatak je mapiranje korisničkog zahteva na entitet iz *Repository* sloja. Konačno, JPA *Repository* komponenta uz pomoć JDBC interfejsa komunicira sa PostgreSQL bazom podataka.

3. REZULTATI I TUMAČENJA

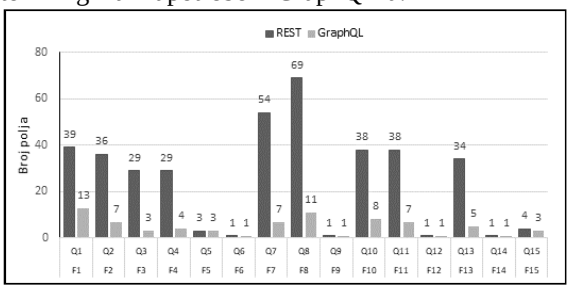
U ovom poglavlju se prikazuju i diskutuju dobijeni rezultati pri analizi performansi prilikom upita nad GraphQL i REST API-ijima.

Kako bi se evaluirale performanse, implementirano je 15 reprezentativnih upita $[Q1, \dots, Q15]$ ka akademskim entitetima u GraphQL tehnologiji i 15 ekvivalentnih REST poziva $[F1, \dots, F15]$.

Odeljak 3.1 poredi broj vraćenih polja u JSON dokumentu oba API-ija i prikazuje medijane u odnosu na 15 GraphQL upita i REST poziva. Odeljak 3.2 poredi veličinu dokumenta vraćenog od strane servera u bajtima.

3.1. Vraćena JSON polja

Kada je broj vraćenih polja u JSON odgovoru u pitanju, rezultati obe tehnologije prikazani su na grafikonu 1. Grafikon prikazuje broj vraćenih polja na pozive iz upita $[Q1, \dots, Q15]$ koji su implementirani uz pomoć REST-a, a potom migrirani upotrebom GraphQL-a.



Grafikon 1. *Broj vraćenih REST i GraphQL polja u JSON odgovoru*

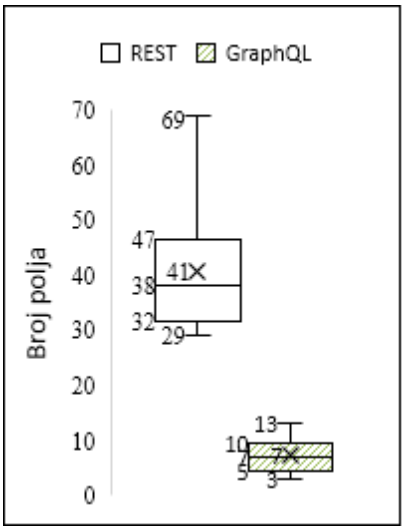
Uočava se znatno manji broj vraćenih polja upotrebom GraphQL-a. Ovaj stepen redukcije se menja u zavisnosti od upita i varira od razlike u 1 polju (F14, Q14) do razlike u 58 polja (F8, Q8). Razlog za znatan REST *overfetch*, na primeru (F8, Q8), je u samoj implementaciji Spring *Projections* tehnologije. Projekcija vraća rezultat entiteta sa svim svojim korenskim poljima. Međutim, uzimajući u obzir povezanost akademskog entiteta koji pripada pozivu (F8) sa većim brojem drugih entiteta, projekcija takođe vraća i povezane entitete. Kao posledica, rezultat vraća JSON sa 38 korenskih i 31 ugnježđenih polja.

Sa druge strane, GraphQL eliminiše *overfetching* time što se unapred znaju polja koja se zahtevaju u telu zahteva. U konkretnom upitu Q8, zahtevaju se 11 nepraznih polja *PaperMonograph* - monografije radova akademskog entiteta koja su prikazana u listingu 1.

```
query Q8 {
findPaperMonographByScopusID(scopusID: "85019996128" ) {
    id
    startPage
    endPage
    titleEng
    abstractEng
    language {
      code
      name
    }
    monographOfPapers {
      id
      editionTitle
    }
  }
}
```

Listing 1. *7 korenskih i 4 ugnježđena polja GraphQL upita*

Grafikon 2 prikazuje *box plot* sa distribucijom broja JSON polja vraćenih od strane REST i GraphQL API-ija. *Box plot* prikazuje uređenu seriju od najmanjeg ka najvećem uzorku i sadrži medijanu i kvartile. Medijana je srednja vrednost po položaju koja deli uzorak na dva jednaka dela. Jedna polovina vrednosti uzorka je manja od medijane, a druga polovina je veća. Kvartili su srednje vrednosti po položaju koje dele uređenu seriju na četiri jednaka dela. Iz grafikona su izuzeti podaci o upitima i i funkcijama koje vraćaju isti broj polja. Medijana broja polja vraćenih sa REST pozivima je 41, dok kod GraphQL poziva je 7. Mera prvog kvartila je 32 (REST) i 5 (GraphQL). Četvrti kvartil iznosi 47 (REST) i 10 (GraphQL).

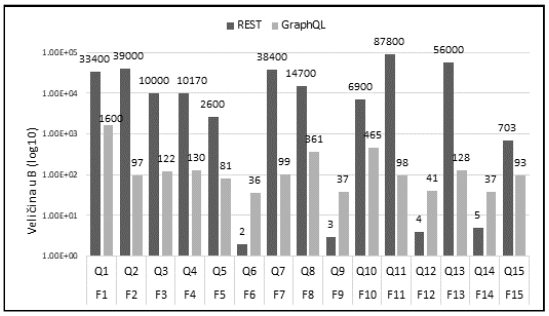


Grafikon 2. *Box plot broja vraćenih REST i GraphQL polja*

3.2. Veličina JSON dokumenata

Nakon što su implementirani GraphQL i REST upiti $[Q1, \dots, Q15]$, izmerena je veličina odgovora JSON odgovora i prikazana na grafikonima 3 i 4. Grafikon 3 prikazuje veličinu JSON dokumenata merenu u bajtima sa log10 poravnanjem. Na vrhu kolona prikazana je labela koja predstavlja izmeren stvaran broj bajtova.

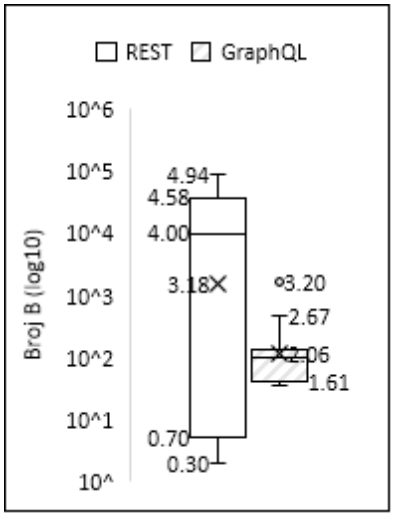
U skoro svim upitima uočava se znatna razlika veličine JSON dokumenta nakon migracije na GraphQL. Na primeru Q8, upotrebom REST poziva odziv je 14.7KB. Isti upit implementiran uz pomoć GraphQL-a iz listinga 11 je 361B. Raskorak je izraženiji ukoliko REST implementacija upita vraća više povezanih entiteta.



Grafikon 3. *Veličina JSON odgovora prilikom REST i GraphQL upita*

Grafikon 4 prikazuje *box plot* sa distribucijom veličine JSON dokumenata vraćenih od strane REST-a i GraphQL. REST odgovori vraćaju 1.51KB (medijana), nasuprot 114B medijani GraphQL odgovora. Samim tim, prema medijani je postignuta memorijska redukcija

GraphQL API-ija oko jednog reda veličine u odnosu na REST. Zbog uticaja atomičnih upita donji REST kvartil je 5B, dok kod GraphQL je 40B. Gornji kvartil je ~38KB (REST) i 128B (GraphQL).



Grafikon 4. *Box plot veličine JSON dokumenata pri REST i GraphQL pozivima*

4. ZAKLJUČAK

Implementiran je GraphQL API kao *wrapper* oko postojećeg REST interfejsa i prilikom postavljanja 15 različitih upita meren je odziv sistema prema dva kriterijuma.

Prvi kriterijum predstavljao je broj vraćenih polja od strane REST i GraphQL API-ija. Kao rezultat, prema medijanama je pokazano da GraphQL vraća do 6 puta manje polja od REST ekvivalentnog poziva.

Drugi kriterijum predstavljao je veličinu dokumenta vraćenog od strane servera. Pokazalo se da GraphQL vraća za red veličine manji JSON dokument (medijana – 114B) od REST API-ija (medijana – 1.51KB). Ovi rezultati pokazuju znatnu uštedu resursa pri svakom zahtevu.

4.1. Dalji razvoj sistema

Uzimajući u obzir da je u ovom radu podržana samo pretraga, jedan od mogućih proširenja u smeru dalje analize performansi GraphQL i REST API-ija bi bila implementacija podrške kreiranja, ažuriranja i brisanja akademskih entiteta na informacionom sistemu. Pomoću GraphQL API-ija takva podrška podrazumeva implementaciju mutacija, dok uz pomoć REST API-ija bi se implementirale dodatne operacije (*endpoints*). Takođe, prema uštedi memorije pri pozivima koji su prikazani u ovom radu, od značajnije koristi bi bila dalja istraživanja merenja performansi GraphQL-a u kontekstu mobilnih i distribuiranih aplikacija. Takvo istraživanje bi utvrdilo da li je moguće dodatno optimizovati resurse pri razmeni podataka na mobilnim i mikroservisnim aplikacijama.

5. LITERATURA

[1] Azeroual, O., & Schöpfel, J. (2019). Quality issues of CRIS data: An exploratory investigation with universities from twelve countries. Publications, 7(1), 14.

[2] https://blog.scopus.com/posts/scopus-roadmap-whats-coming-up-in-2020-2021 (pristupljeno u septembru 2021.).

[3] Brito, G., Mombach, T., & Valente, M. T. (2019). Migrating to GraphQL: A practical assessment. In 2019 IEEE 26th International Conference on Software Analysis, Evolution and Reengineering (SANER) (pp. 140-150). IEEE.

[4] https://arxiv.org/ (pregledano u septembru 2021.).

[5] Vogel, M., Weber, S., & Zirpins, C. (2017). Experiences on migrating RESTful web services to GraphQL. In International Conference on Service-Oriented Computing (pp. 283-295). Springer, Cham.

[6] Brito, G., & Valente, M. T. (2020). Rest vs graphql: A controlled experiment. In 2020 IEEE International Conference on Software Architecture (ICSA) (pp. 81-91). IEEE.

Kratka biografija:

Dušan Nikolić rođen je 16.10.1997. godine u Loznici. Smer računarstvo i automatika na Fakultetu tehničkih nauka u Novom Sadu upisao je 2016 godine. Osnovne studije završio je u septembru 2020. godine. Od oktobra 2020. godine upisuje master studije i angažovan je kao saradnik u nastavi.