Zbornik radova Fakulteta tehničkih nauka, Novi Sad

UDK: 007:005
DOI: https://doi.org/10.24867/07BE35Zekovic

IMPLEMENTACIJA REACT-TABLEQL BIBLIOTEKE OTVORENOG KODA ZA TABELARNI PRIKAZ PODATAKA

IMPLEMENTATION OF REACT-TABLEQL OPEN SOURCE LIBRARY FOR TABULAR VIEW OF DATA

Danilo Zeković, *Fakultet tehničkih nauka, Novi Sad*

Oblast – ELEKTROTEHNIKA I RAČUNARSTVO

Kratak sadržaj – *U ovom radu predstavljena je biblioteka otovrenog koda React-TableQL, koja služi da se podaci pribave i prikažu u formi tabele. Biblioteka sadrži dve komponente koje uz određenu konfiguraciju dodaju ili modifikuju različite funkcionalnosti tabele. Podaci se nabavljaju pomoću GraphQL upita koji se prosledi Apollo Client-u. Za prikaz tabele i obradu podataka korišten je programski jezik JavaScript i biblioteka React.*

Ključne reči: *Biblioteka, React, tabela, GraphQL*

Abstract – *This paper presents the open source library React-TableQL, which is used for fetching and displaying data in a form of a table. Library contains two components, which with certain configuration add or modify different functionalities of the table. Data is fetched using GraphQL query that is passed to Apollo Client. JavaScript and react are used for display and parsing of data.*

Keyords: *Library, React, table, GraphQL*

1. UVOD

U dvadeset i prvom veku, usled intenzivnih društvenih promena, ubrzanog života i transformacije radnih uslova, izuzetno bitan aspekt poslovanja predstavlja brzina i efikasnost u radu zaposlenih, i to ne samo u proizvodnji potrošne robe ili prehrambenih proizvoda, već i u razvoju softvera. Frederick P. Brooks piše u svojoj knjizi "The Mythical Man Month" kako predviđanje trajanja razvoja jednog softverskog sistema predstavlja jedan od težih i često pogrešno izvedenih koraka u samom razvoju. On ističe da su programeri često previše optimistični i da u tom optimizmu prave greške u predviđanjima. Takođe, naglašava da povećanje broja ljudi koji su angažovani na istom projektu najčešće ima negativan efekat na krajnji rezultat i trajanje razvoja samog softvera. Prema Brooksovim tvrdnjama, kompleksnost nekog softvera programeru deluje mnogo jednostvanija nego što ona u stvarnosti zaista jeste, i u tom neslaganju dolazi do velikog gubitka u vremenu i novcu [1].

Jedan od najčešćih načina prikazivanja informacija je u vidu tabele. Kao takav postoje mnoge varijacije i funkcionalnosti koje bivaju dodate samoj komponenti proširujući njenu kompleksnost.

NAPOMENA:
Ovaj rad proistekao je iz master rada čiji mentor je bila dr Gordana Milosavljević.

Sa porastom kompleksnosti raste i vreme utrošeno na implementaciju, a potom i na ažuriranje, odnosno održavanje same komponente. Da bi se neki podaci prikazali u tabeli potrebno je prvo doći do njih i to se na vebu najčešće vrši putem HTTP/HTTPS poziva. Zahvaljujući inžinjerima iz Facebook-a[1] danas, pored REST[2] poziva, imamo mogućnost da koristimo GraphQL za dobavljanje i manipulaciju podacima. GraphQL[3] sve više raste u popularnosti i sama njegova primena je u usponu.

2. TEHNOLOGIJE

TableQL je konstruisan od dve ključne tehnologije i dve biblioteke koje su izvedene iz njih. Dve ključne tehnologije su JavaScript i GraphQL. Prva biblioteka koja je korištena, a napisana je JavaScript programskim jezikom, je React. Druga je Apollo[4] biblioteka koja je napisana JavaScript jezikom da bi se GraphQL lakše uvezao i koristio u veb aplikacijama.

2.1 JavaScript

JavaScript je programski jezik kreiran od strane Brendana Eich-a [2]. Prvi put je predstavljena 1995. godine, kao programski jezik za pisanje programa za Netscape Navigator pretraživač (browser). Međutim, posle sporijih početaka, svi vodeći pretraživači su usvojili JavaScript kao programski jezik koji se koristi da veb stranicama i aplikacijama omogući interaktivnost i funkcionalnost bez potrebe da se stranica ažurira posle svake izvršene akcije korisnika [3].

Posle usvajanja JavaScript kao programskog jezika od strane pretraživača, kreiran je standard poznat kao ECMAScript[5] standard. Organizacija pod nazivom Ecma International je sastavila standard i do današnjeg dana objavljeno je više verzija pod njihovim imenom [3]. JavaScript je postao popularniji u periodu od 2000. do 2010. godine sa verzijom ECMAScript 3, poznatiji pod skraćenim imenom ES3. 2008. godine planirane su radikalne izmene za verziju ES4, međutim verzija 4 nikada nije bila puštena zbog svojih drastičnih izmena u samom jeziku. Umesto verzije ES4, iste godine puštena je verzija ES5 sa manjim izmenama u odnosu na ES3, i na taj način je još više podigla popularnost JavaScript-a. Posle verzije ES5 usledila je verzija ES6 u 2015. godini, i

[1] https://www.facebook.com
[2] https://restfulapi.net/
[3] https://graphql.org/
[4] https://www.apollographql.com/
[5] https://www.ecma-international.org/

od tad je svake godine izlazila nova verzija sa manjim ili većim izmenama [2].

2.2 React

React je JavaScript biblioteka otvorenog koda za kreiranje korisničkog interfejsa. Inicijalno je pušten u javnost za upotrebu 2013. godine od strane tehnološkog giganta Facebook-a i do danas je održavan i razvijan od strane Facbook-a i open source zajednice. Od tada React tim je radio i stvorio mogućnosti da se React koristi i na mobilnim uređajima pomoću React Nativ-a[6], i da može pomoću Node.js-a[7] da se vrši generisanje na serverskoj strani. Pored raznovrsnosti primene, najveća prednost React-a je da se brzo i lako uči. React se zasniva na komponentama koje su osnovni deo React aplikacije [4].

2.3 GraphQL

GraphQL je jezik za upite (query language) za API. Drugim rečima, GraphQL je set pravila za opis podataka koje prednja strana potražuje od zadnje strane, servera. GraphQL omogućava programerima da opišu podatke koji su im potrebni, i da ne brinu o tome kako će upit biti poslat i podaci da se dobave. Uglavnom se koristi preko HTTP poziva, iako nije ograničen na HTTP [5].

Prednosti GraphQL-a u odnosu na REST API je da sprečava dovlačenje više ili manje podataka u zavisnosti od situacije. Ponekad kad pošaljemo zahtev za korisnika, sve što nam je potrebno su ime i datum rođenja. REST će vratiti celog korisnika, ne samo ime i datum rođenja, već i boju očiju, mesto rođenja, ime majke i tako dalje. Sa druge strane, GraphQL nam omogućava da tražimo tačno šta nam treba vratiće nam samo to što je traženo. U drugom slučaju, imamo situaciju gde nije dovoljno podataka dobijeno prilikom prvog zahteva. Pretpostavimo da je korisnik sportista i da je igrao u više timova. Potreba aplikacije je da se ispišu nazivi svih timova za koje je igrao. Timovi za koje je igrao su vezani za korisnika preko njihovih indentifikacionih brojeva, i da bi došli do njihovih imena morali bismo da pozovemo API onoliko puta koliko ih ima, što stvara nepotreban broj poziva u slučaju da se koristi REST. GraphQL taj problem rešava tako što dozvoljava da se podaci opišu ugneždenim upitima.

Još neke prednosti GraphQL-a su da GraphQL ima samo jednu putanju koju gađa za dobavljanje podataka. Ta putanja komunicira preko HTTP POST metode, kojoj prosleđuje objekat sa svim potrebnim opisima podataka koji su klijentu potrebni. Na taj način rešen je problem nekontrolisanog broja putanja.

2.4 Apollo Client

Apollo Client je JavaScript biblioteka otvorenog koda za upravljanje stanjem. Omogućava da programer napiše GraphQL zahteve, a Apollo Client se pobrine za pravljenje upita i keširanje podataka, kao i za ažuriranje korisničkog interfejsa. Apollo Client ima odvojene biblioteke za različite JavaScript radne okvire, među kojima je React biblioteka najpopularnija. Postoje tri osnovne komponente i funkcionalnosti Apollo Clienta: *query*, *mutation*, i upravljanje lokalnim stanjem [6].

[6] https://facebook.github.io/react-native/

[7] https://nodejs.org/en/

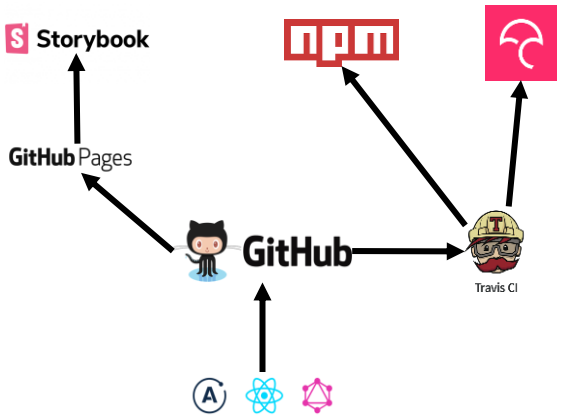
3. REACT-TABLEQL

React-tableql koja je predmet ovog master rada je JavaScript React biblioteka. Napisana je i objavljena kao open source (otvoren kod) projekat sa ciljem da dosegne što šire krugove među React programerima. Svrha biblioteke je da omogući brži i efikasniji razvoj React aplikacija omogućavajući da se sa nekoliko redova koda doda tabela sa svim kompleksnim funkcionalnostima koje može da poseduje. Takođe, za cilj ima i da se omogući da se što manje rizika i grešaka pravi tokom razvoja iste.

TableQL je komponenta koja predstavlja samu srž react-tableql biblioteke. Ona omogućava tabelarni prikaz podataka, u skladu sa definisanom konfiguracijom. Međutim, to ne čini ovu biblioteku drugačijom od drugih već postojećih generatora tabelarnog prikaza podataka. Ono što izdvaja react-tableql od drugih sličnih biblioteka i komponenti je da sadrži omotač oko same komponente za prikaz podataka koji koristi GraphQL da napravi zahtev za podacima serveru koji ima definisan GraphQL API, sa dodatinim slojem, koji pruža dodatni nivo apstrakcije i samim tim smanjuje prostor za grešku prilikom razvoja. Komponenta koja sadrži spoljašnji omotač koji je zadužen za GraphQL API komunikaciju je ApolloTableQL.

3.1 Organizacija i struktura

React-tableql kao open source projekat ima svoj *source code* javno dostupan. Kod se čuva i može se videti kao i doprineti njegovom razvoju na GitHub-u. GitHub, kao centralna komponenta u arhitekturi projekta, igra vezivnu ulogu između napisanog koda i njene isporuke široj javnosti, kao i pristupačnost same dokumentacije.



Slika 1. *Struktura projekta*

Dokumentacija je veoma bitan sastavni deo svakog softvera. Kako bi programeri znali da koriste react-tableql biblioteku, velika pažnja je posvećena pisanju dokumentacije i njenoj dostupnosti. Pored dokumentacije koja je napisana u sklopu samog projekta u radnom dokumentu, kreirana je Storybook[8] prezentacija funkcionalnosti biblioteke. Sadržaj readme fajla može se pročitati na GitHub-u kao i na sajtu www.npmjs.com/package/react-tableql. Sa druge strane imamo GitHub Pages koje nam omogućavaju da direktno iz GitHub repozitorijuma imamo sajt za projekat. GitHub Pages su integrisane sa GitHub-om, i u konfiguraciji projekata postoji mogućnost da se servira veb stranica koja služi kao veb prezentacija projekta. URL stranice je automatski sastavljen od naziva projekta i korisničkog imena vlasnika projekta, što može

[8] https://storybook.js.org/

biti osoba ili organizacija. U slučaju react-tableql biblioteke URL glasi https://danilo-zekovic.github.io/react-tableql.
Prilikom svakog doprinosa biblioteci korišćenjem git-a i komitovanja koda na GitHub-u okida se kontinuirana integracija. Za kontinuiranu integraciju korišćen je Travis CI[9]. Travis CI se lako integriše sa GitHub-om i automatski se pokreće preko GitHub-a. Prilikom svakog pokretanja validiraju se različite stvari koje je moguće definisati kroz konfiguracioni dokument. Taj dokument je deo projekta i Travis CI ga automatski prepoznaje jer konvencija nalaže da se zove .travis.yml [7].
U sklopu react-tableql biblioteke proverava se ispravnost sledećih uslova:

1. da li svi testovi prolaze bez javljanja grešaka,
2. kolika je pokrivenost koda testovima,
3. da li je napisani kod pravilno formatiran,
4. da li se biblioteka i Storybook stranica grade (build) bez grešaka

Isporuka se vrši na npm. Npm je paket menadžer za JavaScript. Svaki paket ima jedinstveno ime preko kojeg je indentifikovan. ReactTableQL biblioteka takođe ima svoje ime u npm registru, react-tableql. Jednostavno se može dodati u projekat korišćenjem komande *npm install react-tableql* u konzoli. Međutim, da bi mogla da se instalira poslednja stabilna verzija biblioteke prvo mora da se sadržaj paketa doda u npm registar. Proces dodavanja u registar je automatizovan korišćenjem Travis-a. Nakon sto Travis završi svoje provere i izgradnju modula oni se šalju na npm uz pomoć API ključa koji identifikuje datu pošiljku kao validnu i kod postaje dostupan preko npm konzole. Slanje koda u registar se ne dešava svaki put kad je kod komitovan na GitHub i kad prođe provere na Travis-u. To se dešava samo kad se tag verzije promeni. Tag verzije se čuva u *package.json* dokumentu koji je izuzetno bitan za svaki projekat koji koristi npm. U njemu se čuvaju svi detalji vezani za projekat i pakete koji se koriste u njemu. Na primer, u react-tableql biblioteci package.json dokument sadrži podatke o verziji biblioteke, podatke o autoru, nazive i verzije paketa koji su potrebni za funkcionisanje biblioteke, kao i komande za pokretanje projekta.

3.2 Funkcionalnosti i primena

React-tableql biblioteka sadži dve komponente, TableQL i ApolloTableQL. Obe komponente imaju svoje parametre koji omogućavaju da se tabela konfiguriše na različite načine. Većina parametara su indentični za obe komponente, međutim, ima ih nekoliko po kojima se razlikuju.

Raznovrsnost parametara ima za cilj da omogući veću prilagodljivost i fleksibilnost korisnikovim potrebama. Komponente imaju svoju osnovnu konfiguraciju koja zadovoljava potrebe jedne aplikacije, ali kako su aplikacije različite, tako su i potrebe za različitim komponentama potrebne. Iz tog razloga težilo se ka omogućavanju korisniku biblioteke da promeni i podesi svaki aspekat komponenti prema svojim potrebama.

Još jedan aspekt koji je bio motivacija za razvoj biblioteke je da se kod brže i efikasnije razvija. To je ostvareno tako što je kompleksnost funkcionalnosti sakrivena. Programer piše manje koda a dobija kompleksne rezultate. Na slici 2 moguće je videti minimalno potreban broj linija koda da bi se dobila tabela pomoću ApolloGraphQL komponente:

```
import React from 'react'
import { ApolloTableQL } from 'react-tableql'

const ExampleComponent = () => {
  const GET_ALL_FILMS = `
    {
      allFilms {
        films {
          title
          episodeID
        }
      }
    }
  `
  return <ApolloTableQL query={GET_ALL_FILMS} />
}

export default ExampleComponent

```

Slika 2 Primer minimalnog koda za generisanje tabele

Pre nego što se tabela konfiguriše, potrebno je dodati react-tableql u projekat. Dodavanje biblioteke je jednostavno i može se ostvariti jednom komandom u *root* folderu projekta, *npm install --save react-tableql*. Da bi se ta komanda pokrenula, neophodno je prethodno instalirati npm u konzoli. Navedena komanda sadrži dodatnu oznaku *--save*, koji dodaje react-tableql u *packege.json* projekat u kojem je pokrenuta. Komanda će iz npm registra svući poslednju stabilnu verziju paketa i dodati je projektu. U tom trenutku su komponente spremne da se učitaju u kod i koriste. Neke od najbitnijih parametara konfiguracije su *query*, *columns* i *pagination*.

Query je jedini obavezan parametar ApolloTableQL komponente. Tip vrednosti *query*-ja je string, koji predstavlja GraphQL upit. Ovaj parametar je ključan jer od njega zavisi koji će sepodaci dovući sa servera. Nakon što se string koji predstavlja upit prosledi ApolloTableQL komponenti, ona prosledi tu vrednost funkciji *gql* koja se učita iz biblioteke graphql-tag. Graphql-tag biblioteka omogućava da se string raščlani i pretvori u validan GraphQL upit [8]. Nakon što se formira ispravan format upita, on se prosleđuje Apollo Client komponenti *Query*, koja pored parametra koji sadrži upit prima i mnoge druge o kojima će biti više reči kasnije u radu. Nakon što je prosleđen ovaj parametar, vrši se upit ka GraphQL API-ju, a u isto vreme se varijabla *loading*, što na engleskom znači učitavanje, prosleđuje TableQL komponenti da se naznači da se podaci čekaju na učitavanje. API vraća podatke u obliku niza koji se zatim prosleđuju TableQL komponenti. U trenutku kad se prime podaci i prosledi *loading* parametar, menja se vrednost u *false* i naznaka učitavanja prestaje. Podaci se obrađuju i bivaju prikazani u tabeli.

Columns predstavlja parametar koji omogućava detaljnije izmene i prilagođavanje komponente specifičnim zahtevima. *Columns* nije obavezan parametar. *Columns* može biti niz stringova i objkata, mogući parametri objekta su *id*, *label*, *component*, *customolumn*, *headerStyle*, *nodeStyle* i *sort*. Od svih parametara jedino je *id* obavezan kako bi bilo moguće indentifikovati i povezati kolonu sa podacima koji su vraćeni sa servera.

React-tableql ima opciju da se uključi paginacija, kao i da se konfiguriše prema potrebama koje aplikacija iziskuje. Opcija da se prosledi jedinstvena paginacija još uvek ne postoji, iako je u planu da u nekim budućim verzijama

[9] https://travis-ci.org/

bude dodata. *Pagination* je opciono svojstvo, koje prima dva tipa vrednosti *boolean* i objekat sa svojstvima *pageLimit, pageNeighbors, currentPage, onPageChange* i *style.*

4. POSTOJEĆA REŠENJA

Tabela je jedan od najpopularnijih vidova prikaza podataka, i kao takav ima i mnoga rešenja za generisanje i prikaz za različite platforme. React je veoma rasprostranjen među veb aplikacijama i samim tim ima široku ponudu komponenti koje omogućavaju prikaz kompleksnih tabela na jednostavan način. Mnoga rešenja su deo biblioteka koje pružaju i druge komponente za izgradnju korisničkog interfejsa, kao na primer, Material Design [10] ili Semantic UI [11]. Samim tim što sadrže mnogo više od jedne komponente, kompleksnost te komponente je svedena na minimum i funkcionalnosti koje pruža su ograničene. Sa druge strane postoji biblioteka react-table [12], čiji fokus je isključivo na tabelarnom prikazu podataka.

React-table je React komponenta za prikaz kompleksnih tabela. React-table je brz i prilagodljiv sa mnogo testiranih funkcionalnosti. Neke od funkcionalnosti koje poseduje su: sortiranje, filtriranje, paginacija i još mnoge druge. Orginalno je kreirana biblioteka od strane Tanner Linsley-a.

5. ZAKLJUČAK

React-tableql je JavaScript biblioteka koja pruža korisniku mogućnost da na lak i jednostavan način prikaže podatke u tabeli. Ovaj projekat napravljen je kako bi se našao način da se ubrza razvoj aplikacija, a u isto vreme smanji obim koda koji treba da se napiše. U radu je opisano rešenje koje ujedinjuje više različitih tehnologija otvorenog koda u jednu jedinstvenu biblioteku. Biblioteka pruža dve React komponente koje uz minimalno podešavanja omogućavaju da se napravi upit za podatke korišćenjem GraphQL-a, a potom i prikažu u stilizovanom i unapred podešenim funkcionalnostima tabele.

TableQL i ApolloTableQL su za sada jedine komponente koje čine react-tableql. One pružaju raznovrsne funkcionalnosti poput paginacije, sortiranja i automatizovanog parsiranja podataka bez prethodnog podešavanja od strane korisnika. Osnovna prednost korišćenja komponenti iz biblioteke opisane u ovom radu jeste da one omogućavaju bezbedniji i brži razvoj aplikacija. To je postignuto kroz apstrakciju funkcionalnosti koje jedna tabela može da ima. I pored visokog nivoa apstrakcije, ostavljen je prostor za prilagođavanje svakog aspekta komponenti.

[10] https://material-ui.com/
[11] https://react.semantic-ui.com/
[12] https://www.npmjs.com/package/react-table

7. LITERATURA

[1] Brooks, Frederick. *The Mythical Man-Month*. Addison-Wesley, 1995

[2] Frost, Aaron. *JS.next: a Manager's Guide*. O'Reilly Media, 2015.

[3] Haverbeke, Marijn. *Eloquent Javascript: a Modern Introduction to Programming*. No Starch Press, 2018.

[4] "React – A JavaScript Library for Building User Interfaces." – *A JavaScript Library for Building User Interfaces*, https://reactjs.org/.

[5] Porcello, Eve, i Alex Banks. *Learning GraphQL: Declarative Data Fetching for Modern Web Apps*. O'Reilly Media, 2018.

[6] "Apollo Docs." *Apollo*, https://www.apollographql.com/docs/react/.

[7] "Travis CI - Test and Deploy Your Code with Confidence." *Travis CI* , https://travis-ci.org/.

[8] "Graphql-Tag." *Npm*, https://www.npmjs.com/package/graphql-tag.

Kratka biografija:

Danilo Zeković rođen je 11.05.1993 u Novom Sadu. Osnovnu školu "Petefi Šandor" u Novom Sadu završava 2008. godine kao dobitnik Vukove diplome. Iste godine upisuje gimnaziju "Svetozar Marković" u Novom Sadu, opšti smer. Četvrtu godinu srednje škole upisuje u Munster High School u Sjedinjenim Američkim Državama u mestu Munster, Indijana. 2012. završava srednju školu uz počasti za odličan prosek. Potom upisuje Saint Joseph College u Rensselaer, Indijana. Diplomira u maju 2016. godine na smeru Computer Science. 2017. godine upisuje master studije na smeru Softversko inžinjerstvo i informacione tehnologije. Poslednji ispit položio januara 2019. godine. Prosek položenih ispita na master studijama je 9.83.