Primena softvera otvorenog koda kod vizualizacije podataka

Vladimir Otašević, Biljana Kosanović
Računarski centar Univerziteta u Beogradu, Beograd, Srbija
vladimir.otasevic@rcub.bg.ac.rs, biljana@rcub.bg.ac.rs

Rezime: Rad je napisan sa ciljem da se pokažu mogućnosti različitih alata otvorenog koda koji se mogu koristi za potrebe vizualizacije i grafičke interpretacije podataka. Takođe, navedeni su i objašnjeni osnovi aspekti koji utiču na grafičku interpretaciju podataka. Ovaj rad sadrži kratak pregled funkcionalnosti dve *javascript* biblioteke za vizualizaciju i to: *d3.js* i *chart.js*.

Ključne reči: javascript, otvoren kod, vizualizacija podataka, grafikon, interakcija

I. Uvod

Oblast vizualizacije podataka se vekovima razvijala [1]. Nije oduvek bila posebna oblast, već se dug period posmatrala kako sastavni element drugih oblasti kao što su fizika, matematika, kartografija i dr. Sa nastankom računara dolazi i do ubrzanog razvoja tehnika vizualizacije podataka. Danas, kada je većina poslova nezamisliva bez računara, vizualizacije predstavlja važan segment prilikom prenosa informacija.

Vizualizacija podataka bi se mogla definisati kao tehnika kojom se informacije i podaci grafički predstavljaju. Grafički objekti mogu biti različiti grafikoni, skale, stabla, histogrami i dr. Postoji još jedna važna odlika vizualizacije podataka, a to je da ona ne treba samo da grafički predstavlja informacije i podatke već i da svojim grafičkim objektima i interaktivnošću privuče korisnikovu pažnju kako bi jasnije razumeo predstavljenu informaciju [2].

Postoji veći broj alata za vizualizaciju podataka baziranih na *javascript* jeziku [3]. U ovom radu su opisane *d3.js* (dostupna na https://d3js.org/) i *chart.js* (dostupna na https://www.chartjs.org/) biblioteke zbog iskustva u radu sa njima. Navedene biblioteke iako se koriste za vizualizaciju podataka, one se međusobno razlikuju. Biblioteka *d3.js* je pogodna zbog svoje fleksibilnosti prilikom implementacije ali to uvodi veću kompleksnost u kodu. Sa druge strane *chart.js* je jednostavnija za primenu ali to dovodi do manje fleksibilnosti prilikom implementacije. Zbog načina implementacije i velikog izbora grafikona koji se mogu proizvesti pomoću ovih biblioteka, one predstavljaju jedno od mogućih rešenja za vizualizaciju.

II. Grafička interpretacija podataka

A. *Istorija*

Oblast razvoja tehnika grafičke interpretacije podataka je već dugo u fokusu pažnje istraživača [1]. Danas vizualizacija podataka predstavlja posebnu granu, a u prilog tome ide i to što postoje eksperti i stručnjaci koji se isključivo bave razvojem rešenja grafičke interpretacije podataka, a da pri tome ne moraju razumeti oblast kojoj pripadaju podaci koje obrađuju. Ovakav pristup ovoj oblasti nije postojao oduvek. Ranije je oblast vizualizacije podataka bila samo segment neke veće oblasti za koju postoji potreba da se određeni skup podataka prikaže na adekvatan način.

Počeci razvoja oblasti vizualizacije vezuju se za početak 16. veka. Period od 16. veka pa sve do danas moguće je podeliti po epohama. Svaka epoha obeležena je različitim otkrićima koja su doprinela razvoju vizualizacije [1].

Prva epoha obuhvata period u kojem je od značaja bilo da se podaci zapisuju na struktuiran način. To je podrazumevalo da se podaci npr. pišu u tabelama. Takođe, istu epohu su obeležile i nove kartografske tehnike. Za drugu epohu se vezuju dešavanja koja se odnose na merenja u fizici. Epoha je obeležena otkrićem Dekartovog koordinatnog sistema. Tada se javlja potreba za ozbiljnim analitičkim pristupom prema prikupljenim podacima koji su sakupljeni putem različitih popisa. Javljanje ideje i prihvatanje važnosti grafičke interpretacije podataka se dešava u 18. veku. Ta nova epoha obeležena je događajima koji se vezuju za otkrića iz oblasti statistike usled čega se javila potreba za razvojem i ispitivanjem novih oblika i formi koje mogu poslužiti za predstavljanje podataka. Od ovog trenutka pa sve do početka 20. intenzivno se razvijaju nove forme grafičke interpretacije podataka koje su posledica ubrzanog razvoja nauke, privrede i društva kao i dolazak do velikog broja novih otkrića. Usled dešavanja u Evropi i svetu, prestaje razvoj oblasti vizualizacije podataka sve do 1950 kada ponovo počinje intenzivan razvoj. Pojava računara omogućila je razvoj novih tehničkih rešenja koja treba da pomognu da se podaci koje koriste računari interpretiraju na adekvatan način.

Danas kada je svakodnevni život nezamisliv bez upotrebe uređaja poput računara ili mobilnog telefona vizualizacija podataka dobija sasvim novi značaj. Vizualizovani podaci nisu više samo podaci koji se tiču naučnih krugova, već se ozbiljno pristupa razmišljanu kako je najbolje preneti informacije ljudima. Važno je razmotriti kako se informacije koje treba preneti grafički interpretiraju kada čulo vida predstavlja značajno čulo kojim čovek prikuplja i brzo obrađuje informacije oko sebe. Posmatrano sa aspekta računarstva, vizualizacija predstavlja značajan segment u oblasti mašinskog učenja i virtualne stvarnosti.

B. *Elementi koji utiču na prenos informacija*

Grafička interpretacija podataka i informacija zavisi od nekoliko elemenata. Prilikom implementacije i odabira grafikona neophodno je razmotriti svaki element. To su elementi kao što su boja, dimenzije, oblici, dodatne funkcionalnosti i dr. Navedeni elementi su samo neki od atributa koji utiču na to kako će grafikon biti protumačen. Jedan pogrešno definisan element može dovesti do toga da se pogrešno protumače predstavljeni podaci i informacije.

Boja predstavlja značajan element prilikom implementacije grafikona [4]. Često je ovaj element taj koji se koristi kada je potrebno pronaći način kako jasno kategorizovati podatke. Pogrešno odabrane boje mogu dovesti do toga da se podaci izmešaju i na taj način dovesti do toga da korisnik dođe do pogrešnih zaključaka. Postoje dve značajne osobine koje se često koriste prilikom odabira boja, a to su kontrast i nijanse. Kontrast je tehnika kod koje se boje biraju na takav način da su jasno uočljive i da se međusobno odabrane boje jasno razlikuju. Za potrebe prikazivanja kategoričkih podataka ova tehnika može biti od značaja. Sa druge strane nijansa je tehnika kod koje se podaci predstavljaju istom bojom različitog intenziteta. Kada je potrebno predstaviti podatke koji imaju hijerarhijsku strukturu onda se podaci koji pripadaju različitim grupama strukture mogu obojiti različitim nijansama iste boje i to bi ukazivalo da ti podaci pripadaju istoj nadgrupi. Treba voditi računa i da odabrane boje budu u skladu sa ostalim elementima aplikacije.

Dimenzije predstavljaju element koji se može posmatrati i tumačiti na više načina. Važno je da dimenzije grafikona budu usklađene sa ostatkom aplikacije na takav način da ne postoji prazan prostor niti da postoje grafički elementi koji se usled svojih dimenzija preklapaju. Sa druge stane dimenzije se mogu odnositi i na sastavne elemente grafikona. Ako govorimo npr. o koordinatnom sistemu, dimenzija tačaka u njemu može predstavljati dodatnu informaciju koja govori o broju elemenata sa istim koordinatama.

Dimenzije se često kombinuju sa oblicima sastavnih elemenata grafikona. Kao i boja, oblici mogu nositi informacije o grupnoj/klasnoj pripadnosti podataka. Oblici bi trebali da budu dovoljno jasni kako bi se međusobno razlikovali i da budu što jednostavniji kako ne bi nepotrebno skretali korisnikovu pažnju.

Element koji se razlikuje u potpunosti od svih navedenih elemenata jeste implementacija dodatnih funkcionalnosti. Savremene tehnologije i moderne tekovine su načinile da korisnik očekuje interaktivne mogućnosti od strane grafičko-korisničkog interfejsa. Vizualizacija podataka se može posmatrati kao deo grafičko-korisničkog interfejsa koji je pogodan za implementaciju dodatnih funkcionalnosti. Postoji veći broj različitih alata i biblioteka koje pružaju proširenje grafikona na način da im omogućavaju dinamičko ponašanje. Treba biti umeren kao i kod ostalih elemenata. Ne treba implementirati sve mogućnosti svim korišćenim grafikonima, jer bi to dovelo samo do zabune i nerazumevanja informacija. Biblioteke pružaju jedan deo mogućnosti za implementaciju dinamičkih karakteristika grafikona. Prilikom implementacije grafikona neophodno je da se u zavisnosti od razvojnog okruženja programiraju i delovi koji služe da se grafikon poveže sa ostatkom sistema.

III. Softverski alati otvorenog koda za vizualizaciju podataka

Postoji čitav spektar alata koji vizualizuju podatke. Alati se mogu svrstati po različitim kriterijumima. Neki od tih kriterijuma su: licenca softvera, tip podataka koji se vizualizuje, vrsta platforme za koju su grafičke predstave namenjene i dr. U zavisnosti od licenciranja alati mogu biti redovno održavani, transparenti i bezbedni za korišćenje. Što se tiče vrste podataka, ona utiče na ograničenje skupa dostupnih grafikona koji se mogu primeniti. Ako govorimo o vrsti platforme onda treba praviti razliku da li će grafikoni biti deo eksternog alata (veb rešenja) ili deo internog alata (desktop rešenja).

A. *Biblioteka d3.js*

Biblioteka kao što je *d3.js* (pun naziv na eng. *Data-Driven Documents*) je rešenje koje se može primeniti kada je potrebno kreirati složeniji oblik grafičke interpretacije podataka. Biblioteka je namenjena vizualizaciji podataka i obuhvata veliki spektar mogućnosti koje nisu direktno povezane sa samom vizualizacijom kao što je npr. čitanje i obrada formata fajlova kao što su *JSON*, *csv* ili *psv* formata. Biblioteka objedinjuje i koristi tehnologije kao što su *HTML*, *CSS*, *JavaScript* i *SVG*. Biblioteka *d3.js* je opisana u ovom radom pod pretpostavkom da se biblioteka upotrebljava sa klijentske strane, iako se ista može primeniti i sa serverske strane. Alat kao što je *d3.js* karakteriše 3 osobine i to: kompatibilnost, debagovanje i performanse. Kompatibilnost podrazumeva da se alat može kombinovati sa drugim *JavaScript* bibliotekama ili uopšte ostalim bibliotekama sa ciljem da se ostvari što bolji osećaj prilikom korišćenja veb aplikacija [5]. Debagovanje kao važan mehanizam otklanjanja grešaka i nedostataka. Kao poslednje, performanse sa ciljem da se rezultati primene *d3.js* biblioteke iskoriste kako bi se implementacija samog veb servisa poboljšala na način da korisnik nikada ne oseti kako nešto radi sporo ili ne radi uopšte.

Alat poseduje veliki broj funkcionalnosti. Neke od funkcija koje su od značaja za vizualizaciju podataka kod veb aplikacija su: kreiranje grafičkog objekta, dohvatanje objekata, operacije nad podacima, interaktivni dodaci [6]. Zbog hijerarhiske strukture, alat pruža mogućnost da se prilikom implementacije pristupi pojedinačno svakom od elemenata strukture i na taj način omogućava da se rade fina podešavanja ili obrade izuzeci. Dohvatanje objekata predstavlja važan segment alata. Nije uvek jednostavno sa složene veb stranice izvući tačno određen objekat pogotovu ako su stranice veb servisa obogaćene različitim bibliotekama ili su objekti definisani složenim identifikatorima. *D3* biblioteka ima implementirane kvalitetne funkcije za dohvatanje objekata koje mogu u zavisnosti od prosleđenih parametara objekat identifikovati po različitim kriterijumima. Takođe, dohvatanje objekata ne mora biti samo slučaj kada se traži eksplicitno jedan objekat koji ispunjava uslov, već se mogu dohvatati nizovi objekata koji ispunjavaju kriterijume prosleđene kao argument funkcija za

dohvatanje objekata. Podaci koji se vizualizuju se mogu obrađivati uz pomoć *D3* biblioteke. To omogućava da se isti podaci mogu obrađivati na različite načine što dovodi do toga se na kraju od istih početnih podataka dobiju različite interpretacije podataka i to ne samo po grafičkoj strukturi već i po nameni. Kako bi se korisnik zainteresovao za elemente grafikona koji su od značaja ili da bi se veliki skup podataka predstavio višeslojnom strukturom, alat *d3.js* pruža mogućnost implementacije interaktivnih elemenata. Interaktivni elementi služe da prilikom korisnikovog rada sa grafikonom, koji može biti prelazak „miša“ preko grafikona, pritisak tastera i dr., dođe do transformacije grafikona i na taj način se prikažu novi elementi. Treba biti umeren sa interaktivnim funkcionalnostima iz razloga što preterana primena može dovesti do toga da se korisniku u potpunosti skrene pažnja sa suštine i time dođe do ne razumevanja informacija.

U okviru *NI4OS* projekta, realizovan je veći broj različitih grafikona primenom *d3.js* biblioteke. Razlog za primenu *d3.js* biblioteke jeste taj što je bilo neophodno razviti složenu logiku i kompleksnu grafičku interpretaciju. Realizovana rešenja dostupna su na https://ni4os.eu/survey-results/

Alat sadrži veliki skup primera koji se mogu primeniti. Kao početan izvor grafikona veoma je korisna lista gotovih primera dostupnih na https://observablehq.com/@d3/gallery. Navedeni servis nudi mogućnost da pre nego što se donese odluka o primeni određenog grafikona izvrše modifikacije i testiranja sa konkretnim podacima i na taj način uštedi na vremenu ako se ispostavi da željeni grafikon nije najadekvatniji za primenu. Odabrani grafikon se može naknadno modifikovati i prilagoditi. Dodatno, kod većine primera postoji i hijerahiska struktura koja dozvoljava izmene i promene kako bi se detalji prilagodili određenoj vrsti podataka.

Jedan primer izvorne verzije grafikona dostupan je na https://gist.github.com/kerryrodden/766f8f6d31f645c39f488a0befa1e3c8. Isti navedeni primer je korišćen prilikom realizacije grafičkog rešenja za potrebe NI4OS projekta. Usklađivanje i modifikacije se ogledaju pre svega u potrebi da se grafikon podesi na način da adekvatno predstavlja podatke sa složenom strukturom i da tako predstavljeni podaci budu jasni korisniku. Na slici 1 je prikazana prilagođena verzija grafikona. Takođe, bilo je neophodno pronaći adekvatan način obeležavanja delova grafikona kako bi oni bili usklađeni sa ostalim elementima veb stranice.

Slika 1: Prilagođena verzija realizovana za potrebe NI4OS projekta

B. Biblioteka chart.js

Još jedna biblioteka koja se može koristiti za vizualizaciju podataka jeste *chart.js* biblioteka. Ta biblioteka je takođe pogodna da se kombinuje sa programima na *JavaScript* jeziku. Poseduje vrlo jednostavan interfejs za korišćenje. Pogodna je za implementaciju iz razloga što se veoma brzo i lako mogu napraviti grafikoni koji ispunjavaju većinu korisničkih zahteva. Poseduje veliki broj gotovih grafikona dostupnih na https://www.chartjs.org/samples/latest/. Osobina koju treba istaći jeste mogućnost transformisanja grafikona. Grafikoni su jednostavnije strukture i vrlo lako se može jedan grafikon transformisati u neki drugi oblik grafikona. Kada je reč o podešavanju postojećih grafikona, postoji veći broj funkcija koje imaju jednostavan oblik pozivanja i koje omogućavaju implementaciju dodatnih funkcionalnosti. Nedostatak kod *chart.js* je taj što ako korisnik želi određenu modifikaciju koja nije predviđena od strane samog alata za vizualizaciju mora se dodatno razviti deo programa koji će implementirani željenu modifikaciju.

Biblioteka *chart.js* sadrži veliki broj gotovih primera koji se uz promenu skupa podataka koji se vizualizuju i uz često male promene u programu kreira željeni grafikon. Postoji mnoštvo grafikona koji se po tipu mogu klasifikovati kao ista vrsta grafikona ali zbog sitnih detalja međusobno se razlikuju. Sitni detalji po kojima se grafikoni razlikuju predstavlja jednu vrstu modifikacije grafikona. Primer grafikona sa ovom osobinom može se videti na https://www.chartjs.org/samples/latest/charts/area/line-boundaries.html.

Značajna prednost *chart.js* biblioteke je jednostavna integracija sa radnim okruženjima kao što su *Angular* i *React*. Ako govorimo o *Angular* radnom okruženju može se u projekat dodati paket pod nazivom *ng2-charts* [7]. Navedeni paket omogućava da se u *Angular* razvojno okruženje dodaju grafikoni kreirani *chart.js* bibliotekom. Biblioteka *chart.js* zbog svoje osobine da se veličina grafikona dinamički prilagođava veličini ekrana uređaja, predstavlja čest izbor i dobru kombinaciju kada se veb servisi razvijaju tehnologijom kao što je *Angular*. Ako se koristi *React* radno okruženje, *chart.js* se sa lakoćom može integrisati pre svega zbog svoje jednostavne i jasno definisane strukture objekata. Primer paketa koji bi se mogao primeniti kako bi se koristila *chart.js* biblioteke jeste *react-charts-2* [8][9]. Navedeni paket predstavlja omotač za *chart.js*.

IV. Zaključak

Postoji veći broj elemenata koji utiču na izgled grafikona i na način na koji će grafikoni preneti informacije. Navedene biblioteke su samo jedna od mogućih rešenja. Prilikom implementacije bilo kakvih grafičkih elemenata treba biti obazriv. Prenos informacija vizualizacijom podataka je svakako jedan način kako se ljudi mogu informisati.

Navedene biblioteke imaju svoje prednosti i mane. Biblioteka *d3.js* može biti veoma zahtevna po pitanju neophodnog znanja za njenu primenu i mnoge funkcionalnosti su nepotrebne za konkretnu implementaciju ali biblioteka pruža veliki stepen

prilagođavanja i pogodna je za vizualizaciju velikih količina podataka. Sa druge strane *chart.js* biblioteka poseduje manji skup grafikona i teže je prilagoditi grafikon konkretnoj implementaciji dok sa druge strane je biblioteka jednostavna za primenu i lako se integrira sa postojećim *JavaScript* programom [10].

Zahvalnica

Autori su zahvalni Milici Ševkušić, bibliotekarki u Tehničkom institutu SANU i prof. Nadici Miljković na datoj prilici i pruženoj podršci.

Rad je napisan u okviru projekta NI4OS-Europe - National Initiatives for Open Science in Europe (H2020 no. 857645).

Literatura

[1] Friendly, Michael & Chen, Chun-houh & Härdle, Wolfgang Karl & Unwin, Antony. (2008). A Brief History of Data Visualization. 10.1007/978-3-540-33037-0_2

[2] Marvin A. Ruder. The Visual Display of Quantitative Information. https://hci.iwr.uni-heidelberg.de/system/files/private/downloads/385875839/marvin-ruder_report.pdf

[3] Jakub Majorek. 14 JavaScript Data Visualization Libraries in 2020. https://www.monterail.com/blog/javascript-libraries-data-visualization

[4] Maureen Stone. Choosing Colors for Data Visualization. January 17, 2006. (dostupno na https://www.perceptualedge.com/articles/b-eye/choosing_colors.pdf)

[5] Deqing Li, Honghui Mei, Yi Shen, Shuang Su, Wenli Zhang, Junting Wang, Ming Zu, Wei Chen. ECharts: A declarative framework for rapid construction of web-based visualization. Visual Informatics. Volume 2, Issue 2. 2018. Pages 136-146. ISSN 2468-502X. https://doi.org/10.1016/j.visinf.2018.04.011 (http://www.sciencedirect.com/science/article/pii/S2468502X18300068)

[6] Guldamlasioglu, Selin. (2015). Web-based information visualization using JavaScript. (Dostupno na https://trepo.tuni.fi/handle/10024/97846)

[7] Sebastian Eschweiler. Angular & Chart.js (with ng2-charts). (dostupno na https://medium.com/codingthesmartway-com-blog/angular-chart-js-with-ng2-charts-e21c8262777f)

[8] Peter Cook. Creating a dashboard with React and Chart.js. Jan 28, 2019. (dostupno na https://www.createwithdata.com/react-chartjs-dashboard/)

[9] Dmitry Rogozhny. Quick Introduction to Displaying Charts in React with Chart.js and react-chartjs-2. March 10th, 2020. (dostupno na https://www.newline.co/@dmitryrogozhny/quick-introduction-to-displaying-charts-in-react-with-chartjs-and-react-chartjs-2--a85b4e2e)

[10] Omar Almootassem and Syed Hamza Husain and Denesh Parthipan and Qusay H. Mahmoud. A Cloud-based Service for Real-Time Performance Evaluation of NoSQL Databases. (2017). 1705.08317. arXiv. (dostupno na https://arxiv.org/ftp/arxiv/papers/1705/1705.08317.pdf)