Web aplikacija za prikazivanje višedimenzionalnih podataka

Blažević, Josip

Master's thesis / Diplomski rad

2021

Degree Grantor / Ustanova koja je dodijelila akademski / stručni stupanj: **Josip Juraj Strossmayer University of Osijek, Faculty of Electrical Engineering, Computer Science and Information Technology Osijek / Sveučilište Josipa Jurja Strossmayera u Osijeku, Fakultet elektrotehnike, računarstva i informacijskih tehnologija Osijek**

Permanent link / Trajna poveznica: https://urn.nsk.hr/urn:nbn:hr:200:158639

Rights / Prava: In copyright / Zaštićeno autorskim pravom.

Download date / Datum preuzimanja: **2024-04-20**

Repository / Repozitorij:

Faculty of Electrical Engineering, Computer Science and Information Technology Osijek

SVEUČILIŠTE JOSIPA JURJA STROSSMAYERA U OSIJEKU
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA

Sveučilišni studij računarstva

Web aplikacija za prikazivanje višedimenzionalnih podataka

Diplomski rad

Josip Blažević

Osijek, 2021.godina

Sadržaj:

1. UVOD ... 1

1.1. Zadatak diplomskog rada ... 1

2. PRIKAZIVANJE VIŠEDIMENZIONALNIH PODATAKA ... 2

2.1. Alati za prikazivanje višedimenzionalnih podataka ... 2

2.1.1. Alati opće namjene ... 3

2.1.2. Alati niske razine ... 3

2.1.3. Prikazivanje tablica i podatkovnih rešetki ... 4

2.1.4. Prikazivanje tokova podataka i gantograma ... 5

2.1.5. Prikazivanje geoprostornih podataka ... 5

2.1.6. Prikazivanje oblaka riječi ... 6

2.1.7. Alati za izradu 3D prikaza ... 6

3. WEB APLIKACIJA ZA PRIKAZIVANJE VIŠEDIMENZIONALNIH PODATAKA ... 8

3.1. Primijenjeni alati i tehnologije ... 8

3.1.1. React ... 8

3.1.2. TypeScript ... 9

3.1.3. Material-UI ... 9

3.1.4. Recharts ... 10

3.1.5. React Hook Form ... 10

3.2. Arhitektura i dizajn aplikacije za prikazivanje višedimenzionalnih podatka ... 11

3.3. Implementacija rješenja ... 16

3.3.1. Unos podataka ... 16

3.3.2. Dohvaćanje podataka ... 18

3.3.3. Prikazivanje grafova ... 19

3.3.4. Pomoćne funkcije ... 21

4. ISPITIVANJE FUNCKIONALNOSTI I DEMONSTIRANJE MOGUĆNOSTI APLIKACIJE 26

4.1. Upravljanje pogreškama ... 26

4.2. Dohvaćanje podataka koristeći API ... 27

4.3. Dohvaćanje podataka iz statičke baze podataka ... 29

5. ZAKLJUČAK ... 31

Sažetak: ... 33

Abstract: ... 34

Životopis ... 35

1. UVOD

Informacije i podaci se nalaze posvuda u okolini. Važno je imati mogućnost prilagodbe podataka u ljudski razumljiv oblik. Ljudima su najpogodniji grafički i matematički modeli na koje se nastoje svesti podaci.

Rad je podijeljen na 5 dijelova: uvod, prikazivanje višedimenzionalnih podataka, web aplikacija za prikazivanje višedimenzionalnih podataka, ispitivanje funkcionalnosti i demonstriranje mogućnosti aplikacije i zaključak.

U poglavlju prikazivanje višedimenzionalnih podataka, biti će opisane postojeće aplikacije i vrste alata za prikaz višedimenzionalnih podataka. U poglavlju web aplikacija za prikazivanje višedimenzionalnih podataka, biti će navedeni i opisani primijenjeni alati i tehnologije koji su potrebni za izradu web aplikacije za prikazivanje višedimenzionalnih podataka, biti će prikazan dizajn aplikacije te će biti detaljan opis programskoga rješenja za prikaz podataka. Unutar poglavlja ispitivanje funkcionalnosti i demonstriranje mogućnosti aplikacije će biti demonstriran neispravan unos i prikaz njegovih poruka te iscrtavanje linijskih i stupčastih grafova uz korištenje podataka dobivenih pomoću statičke baze podataka ili API-ja.

1.1. Zadatak diplomskog rada

Zadatak diplomskog rada je opisati i usporediti alate za prikazivanje višedimenzionalnih podataka. Uz to je potrebno izraditi web aplikaciju za prikazivanje višedimenzionalnih podataka koja podatke učitava pomoću statičke baze podataka ili koristeći API. Dizajn sustava nastoji univerzalno rješavati problem vizualizacije podataka. Korisnik odabire tip prikaza i podatke koji se prikazuju.

2. PRIKAZIVANJE VIŠEDIMENZIONALNIH PODATAKA

Potreba za vizualizacijom podataka je dobro poznati problem koji se često može riješiti na više načina. Podatke je često potrebno prikazati na svojevrsnome grafu kako bi se veliki skupovi podataka mogli prikazati u ljudski razumljivom obliku te nakon toga analizirati. Na temelju toga se mogu donositi bolje odluke vezane uz podatke. Samu vizualizaciju možemo definirati kao prilagodbu velikih skupova podataka u grafove i druge vizualne prikaze. Područje vizualizacije podataka je široko rasprostranjeno i korišteno u raznim sektorima sa svrhom prikazivanja statističkih, financijskih i drugih podataka u lakše shvatljivom obliku [1]. Ono predstavlja efikasan način prezentiranja podataka u komunikaciji.

Na tržištu postoje brojni alati koji prikazuju podatke na grafovima unutar meteoroloških, burzovnih i drugih aplikacija specifične namjene te postoje aplikacije koje pokušavaju biti univerzalni alati za prikazivanje podataka na velikom broju različitih tipova grafova. Takve aplikacije često koriste linijske, stupčaste i mnoge druge često korištene načine prikaza podataka.

2.1. Alati za prikazivanje višedimenzionalnih podataka

AmCharts (https://live.amcharts.com/) je kreiran kao web alat koja pokušava biti univerzalni alat za prikazivanje grafova. Ima široki raspon od preko 10 vrsta grafova i preko 60 unaprijed definiranih podvrsta grafova za vizualizaciju. Podaci se unose tablično ili kao JSON objekt. Stilovi se mijenjaju pomoću polja za unos te se takvi grafovi mogu izvesti kao HTML kod.

TradingView (https://www.tradingview.com/) je web alat za prikazivanje grafova kreiran sa specifičnom namjenom. Primjena mu je vizualizacija kretanja financijskih sredstava poput dionica, fondova, obveznica, kripto valuta, parova valuta i sirovina. Prikazuje vrijednosti na linijskome grafu te omogućava dodavanje proizvoljnih pravaca, elipsa, kružnica i drugih oblika, indikatora kretanja, usporedbi s drugim sredstvima te omogućuje još brojne druge mogućnosti za lakšu vizualizaciju podataka i predviđanje kretanja financijskih sredstava.

Canva Graph Maker (https://www.canva.com/) je univerzalni web alat za prikazivanje grafova. Nudi široki raspon grafova za prikaz od linijskih i stupčastih kao češćih vrsta do Venn-ovih dijagrama i konceptualnih mapa. Podaci se unose u tablicu kako bi se prikazali na grafu.

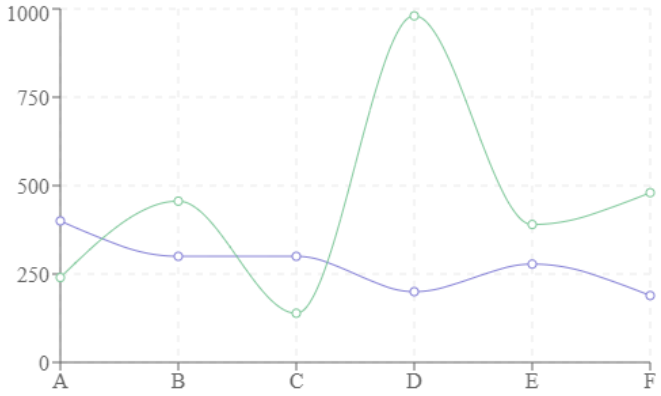
LucidCharts (https://www.lucidchart.com/) je web alat sa specifičnom primjenom. Specijaliziran je za izradu UML dijagrama, tokova podataka, organizacijskih dijagrama i ostalih grafova posebne

primjene. Elementi dijagrama se dodaju povlačenjem elementa na ekran i upisivanje vrijednosti unutar njih.

Adobe Spark (https://spark.adobe.com/) je web alat koji omogućuje izradu vrlo jednostavnih grafova. Nudi izradu linijskih, stupčastih i ispunjenih i u sredini praznih kružnih grafova. Podaci se unose kao niz parova ključ-vrijednost.

2.1.1. Alati opće namjene

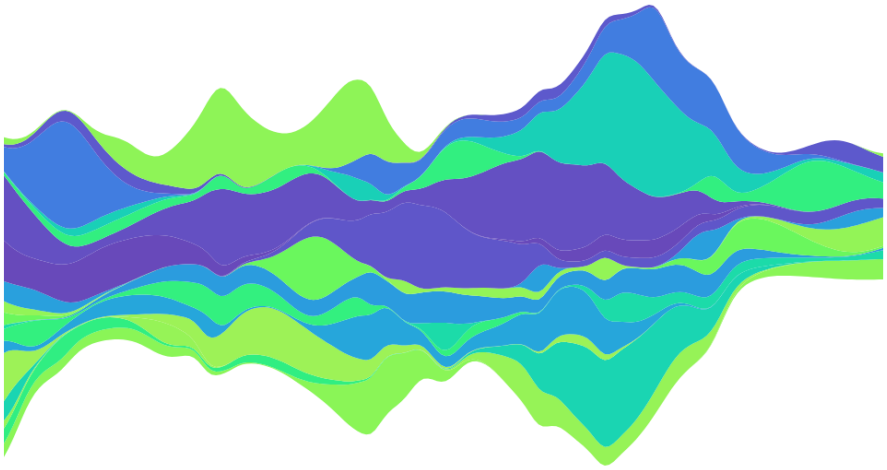
Alati opće primjene su najčešće korišteni alati za vizualizaciju podataka. Većina podataka se može prikazati pomoću grafova koji u sebi mogu, ali ne moraju sadržavati vrijeme [2]. Ovakvi alati su namijenjeni kako bi omogućili laganu implementaciju dobro poznatih grafova. Grafovi koje oni često sadrže su stupčasti grafovi, linijski grafovi (Slika 2.1.), kružni grafovi te mnogi drugi. Neke od biblioteka za vizualizaciju podataka opće namjene su Chart.js, Recharts, Highcharts, Chartist.js, Victory, React-vis, amCharts i AnyCharts.



Slika 2.1. Linijski graf izrađen pomoću Recharts biblioteke [3]

2.1.2. Alati niske razine

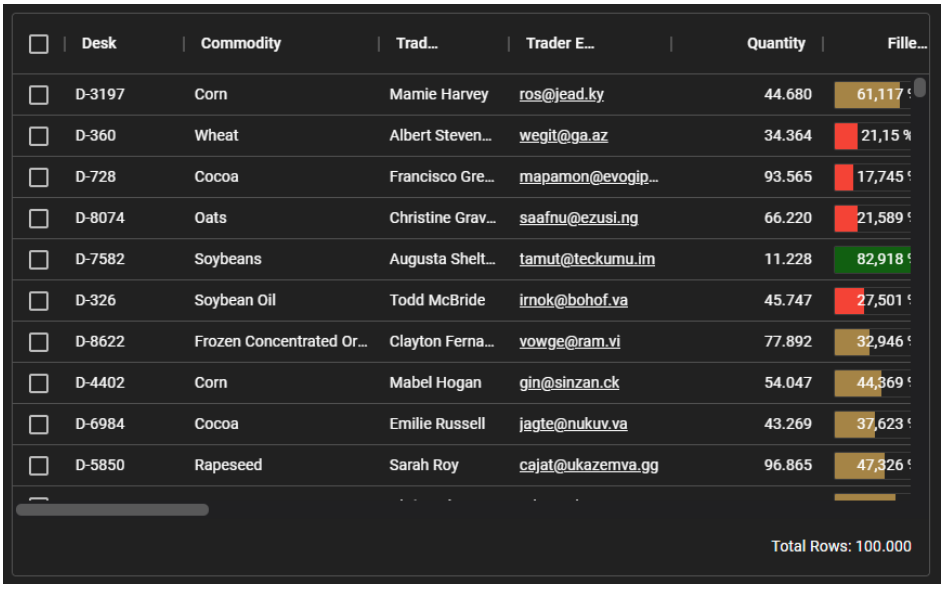
Alati za vizualizaciju niske razine su primjenjivi za naprednije i kompliciranije vizualizacije od alata opće primjene. Pomoću ovih alata se mogu ručno kreirati grafovi koji se inače koriste i kod alata opće namjene, no ovakvi alati imaju najveću primjenu ako je potrebna prilagođena, složena vizualizacija (Slika 2.2.) ili vrlo visok stupanj interaktivnost. Ovakvi alati zahtijevaju znatno veću razinu znanja iz područja vizualizacije podataka i izrade SVG datoteka [2]. Neke od biblioteka niske razine za vizualizaciju podataka su D3.js, Plotly.js, C3.js, Semiotic, Taucharts, Apache ECharts i Raphaël.



Slika 2.2. Graf protoka izrađen pomoću D3.js biblioteke [4]

2.1.3. Prikazivanje tablica i podatkovnih rešetki

Neki od podataka se lakše prikazuju unutar tablica te se onda i vizualiziraju pomoću tablice ili podatkovne rešetke (Slika 2.3.) [2]. Neke od biblioteka za prikaz tabličnih podataka i podatkovnih rešetki su ag-Grid i Material-UI.



Slika 2.3. Interaktivna tablica kreirana pomoću Material-UI biblioteke [5]

2.1.4. Prikazivanje tokova podataka i gantograma

Neki od alata opće namjene već imaju unutar sebe ovu vrstu vizualizacije, no s obzirom da nije uvijek takav slučaj se izdvaja ova grupa kao posebna. Ovi alati, kao što i naziv govori, prikladni su za prikaz tokova podataka (Slika 2.4.) i gantograma [2]. Neke od biblioteka za prikaz tokova podataka i gantograma su Highcharts, amCharts i vis-timeline.

Slika 2.4. Prikaz toka podataka izrađen pomoću vis-timeline biblioteke [6]

2.1.5. Prikazivanje geoprostornih podataka

Alati za prikaz geoprostora su namijenjeni za prikaz geografskih sadržaja. Podaci koje mogu prikazivati su karte, područja, ulice, lokacije i bilo koji drugi podaci koji se mogu geografski prikazati (Slika 2.5.) [2]. Neke od biblioteka za prikaz geoprostora su Leaflet, Mapbox, Google Maps i amCharts.

Slika 2.5. Prikaz lokacije na karti izrađen pomoću Leaflet biblioteke [7]

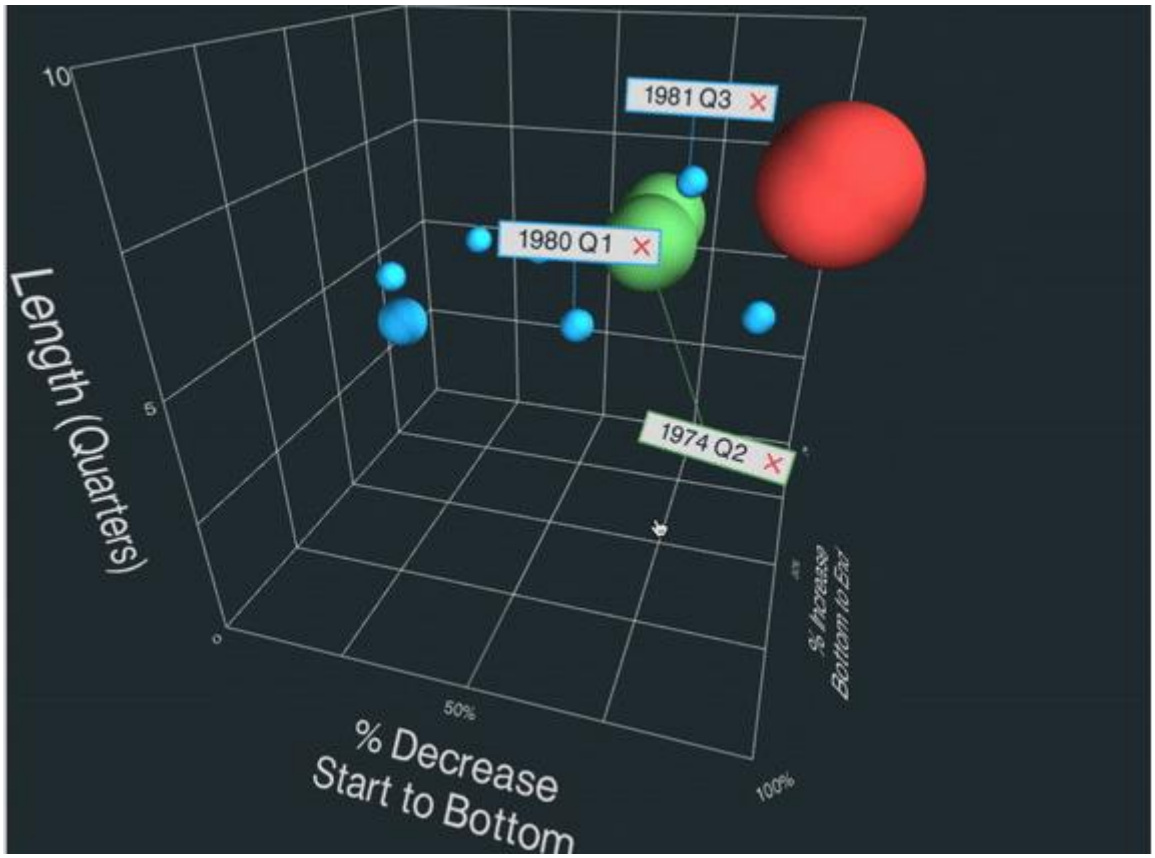
2.1.6. Prikazivanje oblaka riječi

Alati za prikaz oblaka riječi služe za prikazivanje podataka koji se ne mogu kvantificirati unutar oblaka riječi (Slika 2.6.) [2]. Neke od biblioteka za prikaz oblaka riječi su d3-cloud, amCharts i HighCharts.

Slika 2.6. Prikaz oblaka riječi izrađenoga pomoću d3-cloud biblioteke [8]

2.1.7. Alati za izradu 3D prikaza

Za razliku od ranije spomenutih alata koji su specijalizirani za 2D planarne prikaze podataka, ovi alati su specijalizirani za prikaz 3D objekata (Slika 2.7.) [2]. Neke od biblioteka za prikaz 3D objekata su Three.js, Babylon.js i vis-graph3d.



Slika 2.7. Prikaz podataka u 3D koordinatnom sustavu izrađen pomoću Three.js biblioteke [9]

3. WEB APLIKACIJA ZA PRIKAZIVANJE VIŠEDIMENZIONALNIH PODATAKA

3.1. Primijenjeni alati i tehnologije

Prilikom izrade web aplikacije za prikazivanje višedimenzionalnih podataka, korištene su brojne JavaScript biblioteke koje olakšavaju izradu stranica, stilova, i vizualizacija podataka (Tablica 3.1.). Web aplikacija je izrađena uz pomoć React biblioteke. TypeScript je korišten kako bi se podaci mogli imati navedene tipove i tako se olakšala implementacija i otklanjanje pogrešaka. Stilovi su dodani u aplikaciju uz pomoć Material-UI biblioteke koja olakšava njihovo dodavanje. Grafovi su kreirani uz pomoć Recharts biblioteke opće namjene za vizualizaciju podataka. Formama se upravlja pomoću React Hook Form biblioteke te se podaci primaju i šalju s Axios bibliotekom.

Tablica 3.1. Tablični prikaz primijenjenih tehnologija

Programsko okruženje	React
Tipovi podataka	TypeScript
Implementacija dizajna	Material-UI
Izrada grafova	Recharts
Forme	React Hook Form
HTTP zahtjevi	Axios
Kontrola koda	Git
Uređivač koda	Visual Studio Code

Kod implementiran pomoću React biblioteke se prije izvršavanja, automatiziranim postupcima, pretvara u HTML, CSS i JavaScript kod kako bi se mogao koristiti unutar internetskog preglednika.

3.1.1. React

React je JavaScript biblioteka za izradu korisničkih sučelja web aplikacija. React je deklarativan, što olakšava pisanje koda [10]. Unutar njega se mogu jednostavno kreirati prikazi za svako stanje u aplikaciji. React učinkovito ažurira i prikazuje komponente prilikom promjena podataka.

Deklarativni pogledi čine kod preglednim, jednostavnim za razumijevanje te lakšim za otklanjanje pogrešaka unutar koda.

React je temeljen na izradi komponenata. Omogućuje izradu cjelovitih komponenata koje upravljaju vlastitim stanjima, čijim povezivanje se mogu izraditi kompleksna korisnička sučelja.

Kao alternativa React-u su se mogli koristiti Angular ili Vue, no React se pokazao najbolji zbog najrasprostranjenije upotrebe, detaljne dokumentacije, brzine, lakoće korištenja te fleksibilnosti.

3.1.2. TypeScript

TypeScript je jezik otvorenog koda koji se nadovezuje na JavaScript. Glavna funkcija mu je dodavanje definicija statičkih tipova. Statički tipovi pružaju način za opis tipova podataka objekata, čime pružaju bolju dokumentaciju te dopuštaju TypeScriptu da provjeri ispravnost koda.

Sav valjani JavaScript kod također je TypeScript kod. Postoji mogućnost dobivanja pogreške u provjeri tipa, no i dalje se može pokrenuti rezultirajući JavaScript kod. Najčešće se definira strogo ponašanje unutar aplikacije, no TypeScript nudi fleksibilnost prilikom definiranja [11].

TypeScript nema relevantnih alternativa. Najjednostavniji i najopširniji način za postavljanje tipa unutar željenog JavaScript koda je korištenjem TypeScript biblioteke.

3.1.3. Material-UI

Material-UI je jednostavna i prilagodljiva biblioteka komponenata za izgradnju bržih, ljepših i pristupačnijih React aplikacija. Biblioteka je izrađena s uzorom na Material Design tvrtke Google [12].

Material-UI nudi veliki broj unaprijed stiliziranih komponenti kojima se stilovi mogu lako mijenjati prema korisničkim željama. Nudi stilizirane elemente poput gumbova, unosa, unosa datuma i vremena, radio gumbova, tekstualnih polja, menija, ladica, navigacija, dijaloga, ikona, tablica, list, itd.

Material-UI nudi „makeStyles“ funkciju pomoću koje se mogu mijenjati stilovi željenih komponenti unutar komponenata koje se koriste. Napisani stilovi se unutar aplikacije prikazuju kao objekt koji unutar sebe sadrži klase kao svojstva.

Uz njega biblioteka pruža stilizirane komponente i komponente višeg stupnja kako bi se mogla definirati komponenta sa korisnički željenim stilovima te kako bi se mogla na više mjesta koristiti uz minimalne ili nikakve promjene unutar komponenti.

Kao alternativa Material-UI biblioteci, mogu se koristiti biblioteke Ant Design, React-Bootstrap, Reactstrap te brojne druge. Material-UI je pogodan zbog mogućnosti povezivanja s funkcijskim komponentama, mogućnosti kreiranja stiliziranih komponenti, detaljne dokumentacije i velikog broja pred definiranih elemenata.

3.1.4. Recharts

Recharts je biblioteka za izradu vizualizacija podataka opće namjene, kreirana pomoću React i D3 biblioteke [13]. Glavna primjena ove biblioteke je olakšanje pri izradi grafova u React aplikacijama. Omogućava definiranje velikog broja često korištenih grafova uz predavanje podataka, bez potrebe za pisanjem kompleksnog koda za prikaz.

Alternative Recharts biblioteci su Chart.js, Highcharts, Chartist.js, Victory, React-vis, amCharts i AnyCharts. Recharts je pogodan zbog lakoće korištenja s React bibliotekom, mogućnostima koje nudi te sa svojom jednostavnosti.

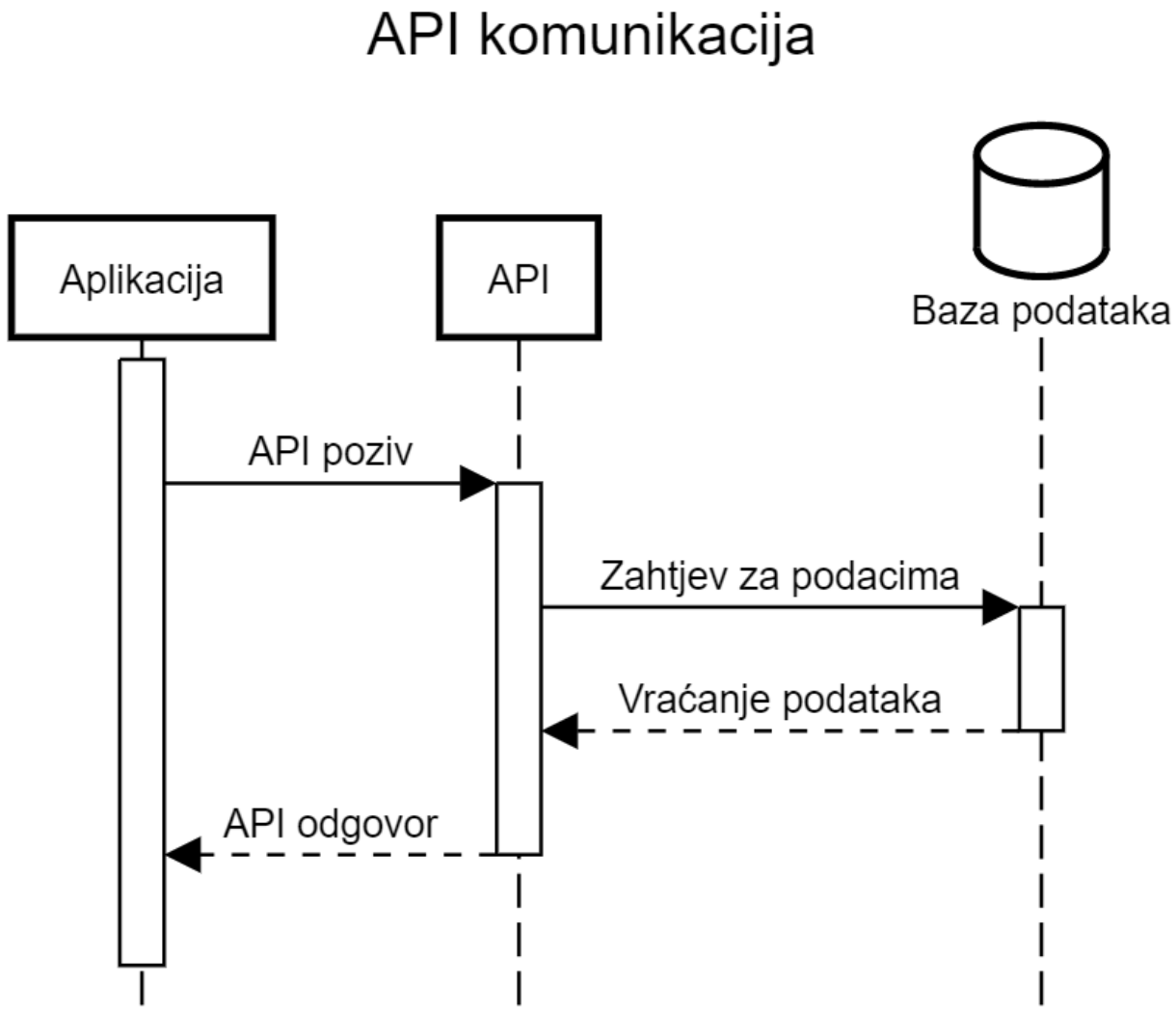
3.1.5. React Hook Form

React Hook Form biblioteka je biblioteka koja kao glavnu primjenu ima lakše upravljanje formama za unos podatka. Dodavanjem ove biblioteke smanjuje se potrebna količina koda za upravljanje formama te može smanjiti velik broj ponovnih prikazivanja komponenti unutar rada. Na taj način se povećavaju performanse stranice [14].

Kao alternativa, mogli su se koristiti Formik, KendoReact Form te React Final Form. React Hook Form je korišten zbog svoje lakoće integracije sa funkcijskim komponentama unutar React-a, omogućava lako korištenje ne kontroliranih komponenti forme te jednostavan način za upravljanje pogreškama.

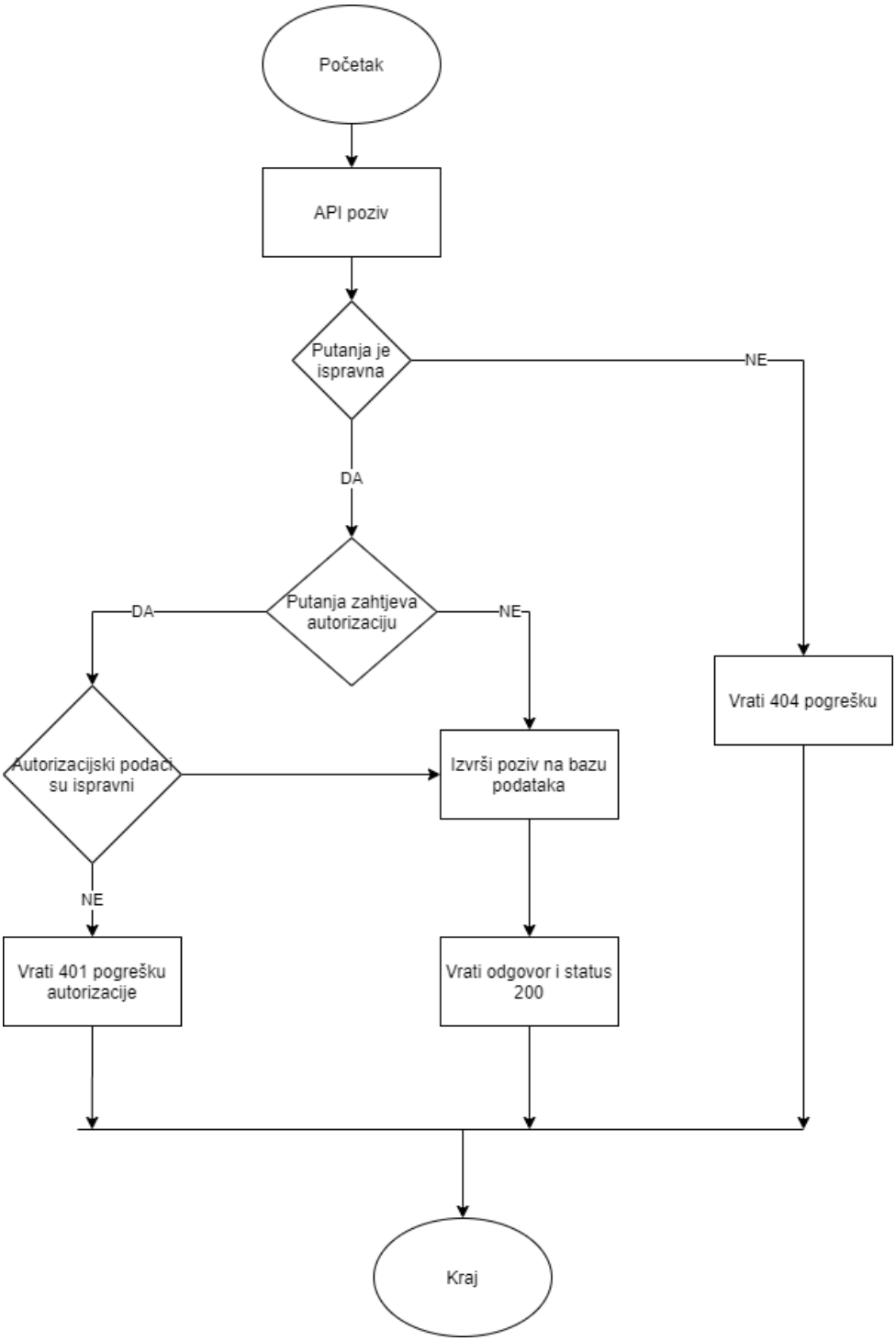
3.2. Arhitektura i dizajn aplikacije za prikazivanje višedimenzionalnih podatka

Komunikacija koristeći API, prikazana na slici 3.1., ostvaruje se na način da se izvršava asinkroni poziv za dobivanje podataka iz aplikacije. API u pozadini pristupa bazi podataka iz koje dohvaća podatke. Nakon toga se vraća odgovor za korišteni API. API potom šalje željene vrijednosti u aplikaciju.



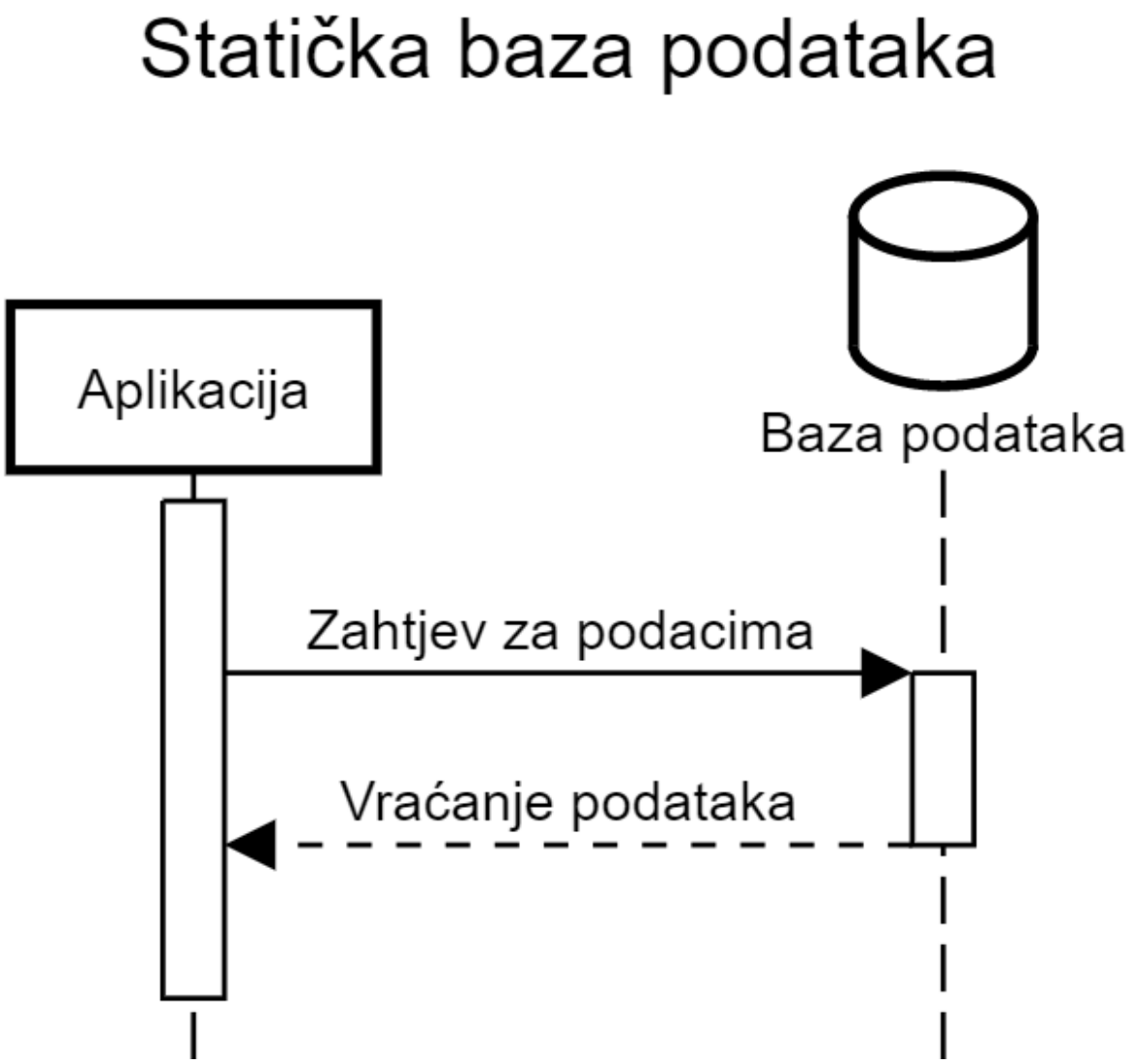
Slika 3.1.: Sekvencijalni dijagram API komunikacije

Detaljan dijagram komunikacije je vidljiv na dijagramu toka API komunikacije (Slika 3.2.), gdje je prikazano upravljanje pogreškama uz samo kretanje podataka. Ukoliko putanja nije ispravna, potrebno je vratiti HTTP 404 pogrešku ne ispravne putanje. Ukoliko je potrebna autorizacija, vraća se HTTP 401 pogreška autorizacije, ukoliko autorizacijski podaci nisu ispravni. Ako je sve uspješno obavljeno, korisniku se vraćaju podaci uz HTTP 200 uspješan poziv.



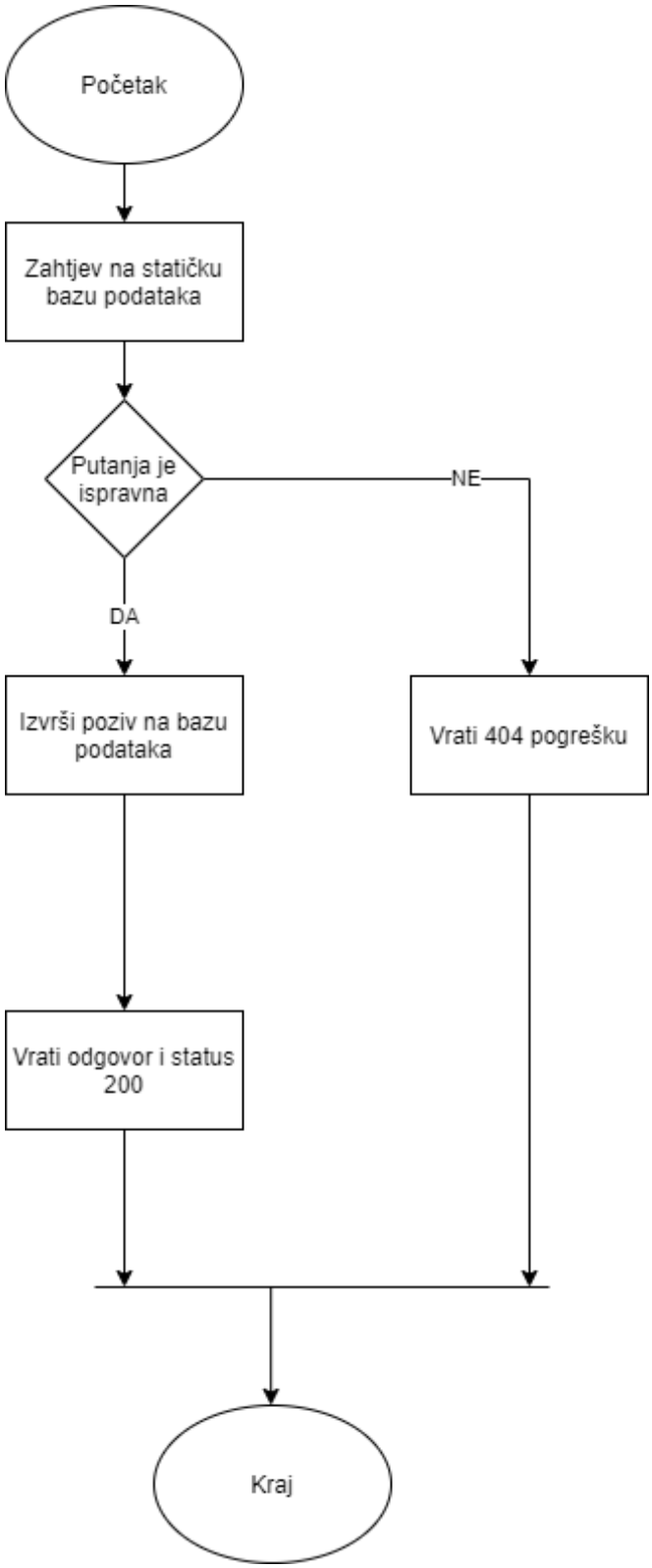
Slika 3.2. Dijagram toka API komunikacije

Prilikom korištenja statičke baze podataka, asinkroni zahtjev se direktno šalje iz aplikacije na bazu podataka koja vraća odgovor (Slika 3.3.). S obzirom da u ovome slučaju se podacima pristupa direktno, ovaj slučaj je jednostavniji od onoga uz korištenje API-ja.



Slika 3.3.: Sekvencijalni dijagram statičke baze podataka

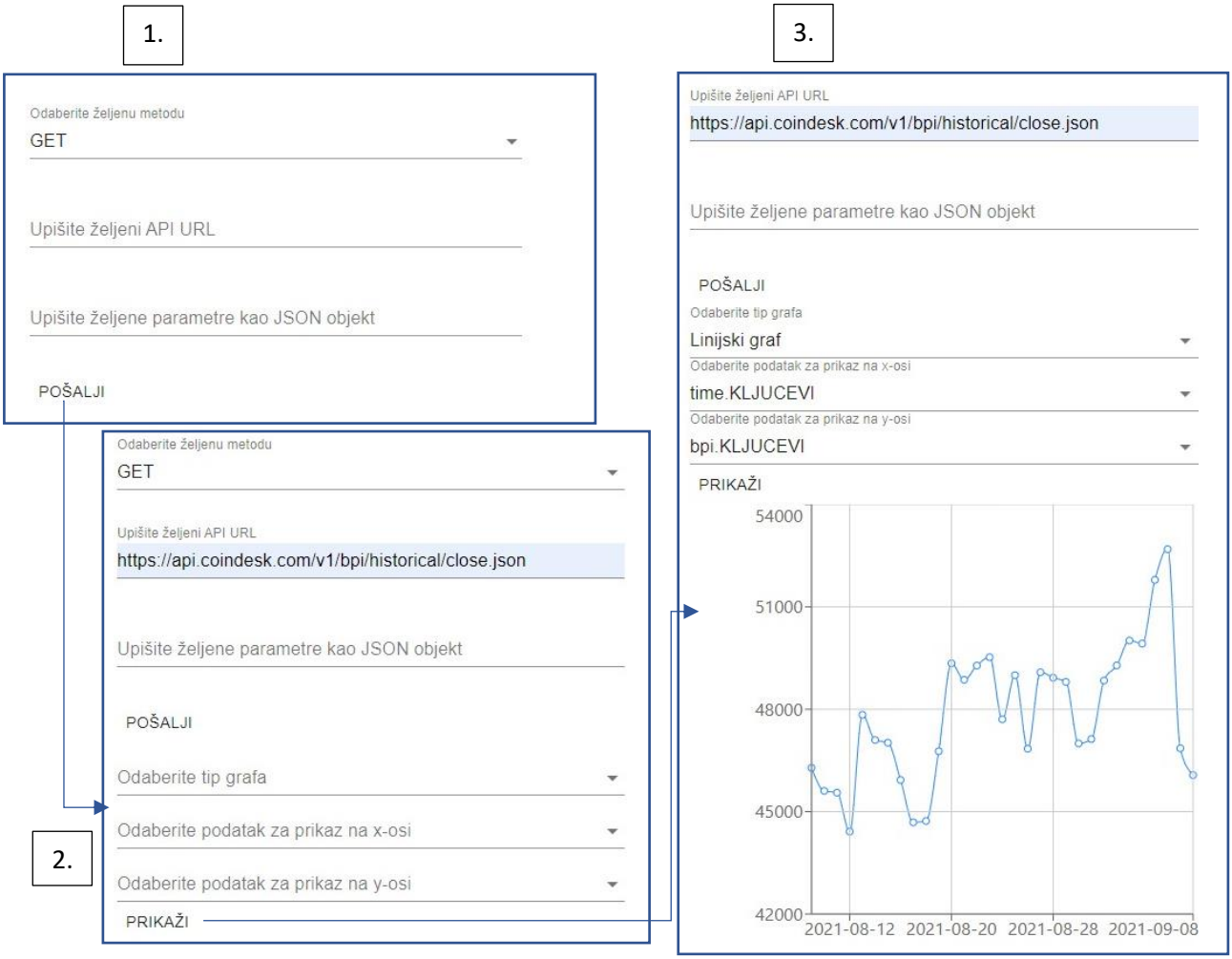
Detaljan dijagram komunikacije je vidljiv na dijagramu toka statičke baze podataka (Slika 3.4.), gdje je prikazano upravljanje pogreškama uz kretanje podataka. Ponašanje je slično kao i kod API poziva, uz manji broj grananja. Ukoliko putanja nije ispravna, potrebno je vratiti HTTP 404 pogrešku ne ispravne putanje. Ako je sve uspješno obavljeno, korisniku se vraćaju podaci uz HTTP 200 uspješan poziv.



Slika 3.4. Dijagram toka statičke baze podataka

Aplikacija je dizajnirana na način da se polja za unos zajedno s gumbovima nalaze na vrhu stranice te se nakon toga nalazi graf s prikazanim podacima nakon što se unesu svi podaci. Prema slici 3.5. se može vidjeti prikazivanje aplikacije kroz korake te je aplikacija u cijelosti prikazana na slici 3.6.

Polja za odabir metode, upis željenog URL-a, upis željenih parametara i gumb za slanje tih podataka se nalaze na stranici prilikom pokretanja stranice. Nakon primanja odgovora sa podacima, prikazuju se polja za odabir vrste grafa, odabir podataka na x i y osima te gumb za prikazivanje podataka. Nakon pritiska gumba za prikazivanje podataka, podaci se prikazuju na željenome grafu ukoliko ih je moguće prikazati.



Slika 3.5. Prikazivanje stranice u koracima

Slika 3.6. Dizajn aplikacije

3.3. Implementacija rješenja

Glavni dio web aplikacije je definiran unutar „App.tsx“ datoteke, kao unaprijed zadane glavne datoteke unutar React programskoga okruženja. U njoj su definirane forme za unos podataka unutar aplikacije te funkcije koje osluškuju slanje tih formi. Grafovi su definirani kao zasebne gotove komponente koje se koriste unutar generičke graf komponente koja sadrži podatke za iscrtavanje. Upravljanje podacima i API pozivi su definirani unutar pomoćnih funkcija. Konstante su definirane u odvojenim datotekama kako bi se njihovim vrijednostima moglo pristupati koristeći bolje pamtljiva imena pozivanjem konstanti.

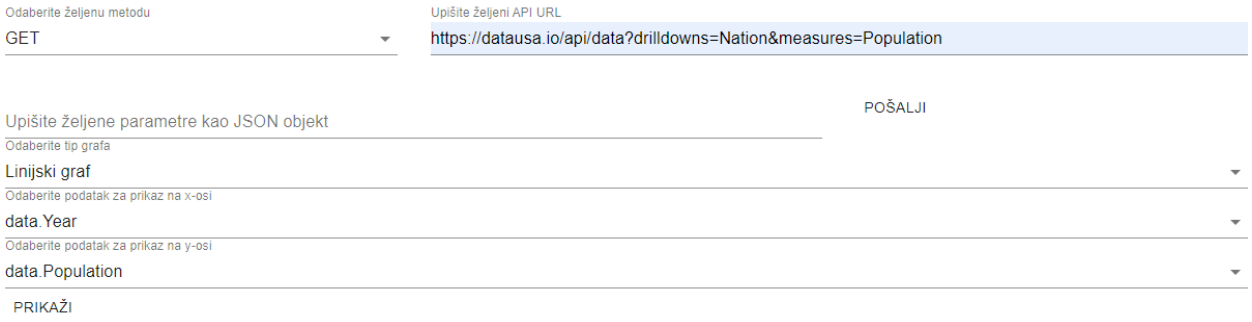
3.3.1. Unos podataka

Polja za unos podataka (Slika 3.7.) se definiraju unutar kontrolera React Hook Form biblioteke koja upravlja tim poljima. Polja za unos se definiraju korištenjem „TextField“ oznake kao složene komponente koja na sebi ima ugrađeno upravljanje pogreškama, prikazivanje naziva te ta komponenta može prikazivati razne izvorne načine unosa podataka. Svako polje ima svoj naziv

prema kojemu se prepoznaju unosi nakon slanja forme. Od polja za spajanje se koriste odabir vrste zahtjeva (GET ili POST), putanja za spajanje te dodatni parametri ukoliko su potrebni u JSON formatu.

```
<Controller
    render={({ field: { name, value, onChange }, fieldState: { error }}) => (
        <TextField

            fullWidth
            name={name}
            value={value}
            onChange={onChange}
            label={polja.putanja.rezerviraniTekst}
            error={Boolean(error)}
            helperText={error?.message || ''}

         />
     )}
     control={control}
     name={polja.putanja.naziv}
     defaultValue=""
     rules={{ required: { value: true, message: porukePogreske.putanja } }}
/>
```

Programski kod 3.1. Polje za unos putanje za spajanje na API ili lokalnu bazu podataka



Slika 3.7. Polja za unos podataka

Slanjem forme s podacima za spajanje na API ili lokalnu bazu podataka se poziva funkcija koja kao parametar prima objekt sa svim poljima za unos. Koristeći dobivene vrijednosti, poziva se pomoćna funkcija koja izvršava zahtjev za dohvaćanje podataka. U slučaju da se podaci uspješno dohvate, podaci se spremaju kao stanje unutar glavne komponente.

```
const posaljiZahtjev = React.useCallback((vrijednosti) => {
    dohvatiPodatke(vrijednosti[polja.putanja.naziv], vrijednosti[polja.tipPoziva.
naziv], vrijednosti[polja.parametri.naziv]).then(
        (odgovor) => {
            postaviPodatke(odgovor);
```

```
        }
    );
}, []);
```

Programski kod 3.2. Funkcija na slanje forme s podacima za spajanje na API ili lokanu bazu

Nakon dohvaćanja podataka, prikazuju se polja za odabir tipa grafa te za odabir podataka na osima. Nakon unošenja željenih podataka i njihovog slanja, poziva se funkcija za iscrtavanje grafa koja se nalazi u komponenti osnovnoga grafa. Kao parametar se šalje objekt koji unutar sebe sadrži upisane vrijednosti unutar forme.

```
const odabirPodataka = React.useCallback((vrijednosti) => {
    grafRef.current?.nacrtajGraf(vrijednosti);
}, []);
```

Programski kod 3.3. Funkcija na slanje forme s podacima za prikazivanje grafa

3.3.2. Dohvaćanje podataka

Pomoću Axios biblioteke, izvršavaju se AJAX HTTP zahtjevi. Uz korištenje pomoćne funkcije za dohvaćanje podataka, uz primanje parametara tipa zahtjeva, putanje za spajanje i dodatnih parametara iz polja za unos, Axios biblioteka šalje zahtjev na API ili lokalnu bazu podataka i vraća rezultate ili pogrešku u obliku obećanja zbog svoje asinkrone naravi.

```
export const dohvatiPodatke = async (url: string, tipPoziva: TipoviPoziva, paramt
eri: string): Promise<any> => {
    try {
        let podaci: AxiosResponse<any>;
        if (paramteri.length > 0) {
            podaci = await axios({
                method: tipPoziva,
                url,
                ...JSON.parse(paramteri),
            });
        } else {
            podaci = await axios({
                method: tipPoziva,
                url,
            });
        }
        return podaci.data;
    } catch (pogreska) {
        throw pogreska;
    }
};
```

Programski kod 3.4. Pomoćna funkcija za dohvaćanje podataka

3.3.3. Prikazivanje grafova

Podaci dobiveni slanjem forme s podacima za prikaz podataka, šalju se u osnovnu komponentu grafa koja u ovisnosti o odabranome tipu grafa, vraća specifičnu komponentu grafa. Primjer rezultata dobivenog korištenjem komponente linijskog grafa je prikazan na slici 3.8, te primjer rezultata dobivenog korištenjem komponente stupčastog grafa je prikazan na slici 3.9.

```
if (!prikaz || tipGrafa.length === 0 || xOs.length === 0 || yOs.length === 0) ret
urn null;
else if (tipGrafa === tipoviGrafova.linijskiGraf) {
    return <LinijskiGraf podaci={prikaz} xOs={xOs} yOs={yOs} />;
} else if (tipGrafa === tipoviGrafova.stupcastiGraf) {
    return <StupcastiGraf podaci={prikaz} xOs={xOs} yOs={yOs} />;
} else return null;
```

Programski kod 5.5. Prikaz specifične komponente grafa u ovisnosti o ulaznim podacima

Specifične komponente grafa unutar sebe ne sadrže dodatnu logiku, već samo pozivaju komponente iz Recharts biblioteke sa njima predanim parametrima.

```
const LinijskiGraf = (props: Props) => {
    const { podaci, xOs, yOs } = props;

    return (
        <ResponsiveContainer minHeight={400}>
            <LineChart data={podaci} margin={{ left: 50, right: 5 }}>
                <Line type="monotone" dataKey={yOs} stroke={boje.plava} />
                <CartesianGrid stroke={boje.siva} />
                <XAxis dataKey={xOs} />
                <YAxis domain={['auto', 'auto']} />
                <Tooltip />
            </LineChart>
        </ResponsiveContainer>
    );
};

export default LinijskiGraf;
```

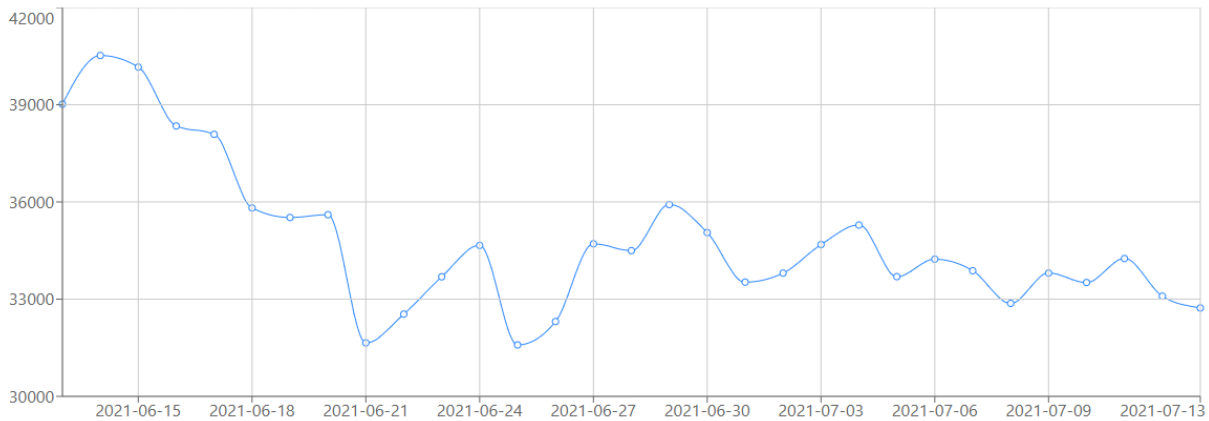
Programski kod 3.6. Komponenta za izradu linijskoga grafa

Podaci nakon što se dobiju u osnovnu komponentu za graf, unutar funkcije za iscrtavanje grafa, prilagođavaju se u oblik pogodan za iscrtavanje pomoću Recharts biblioteke te se spremaju unutar React stanja. Prilagodba se događa unutar pomoćnih funkcija koje vraćaju potrebne rezultate.

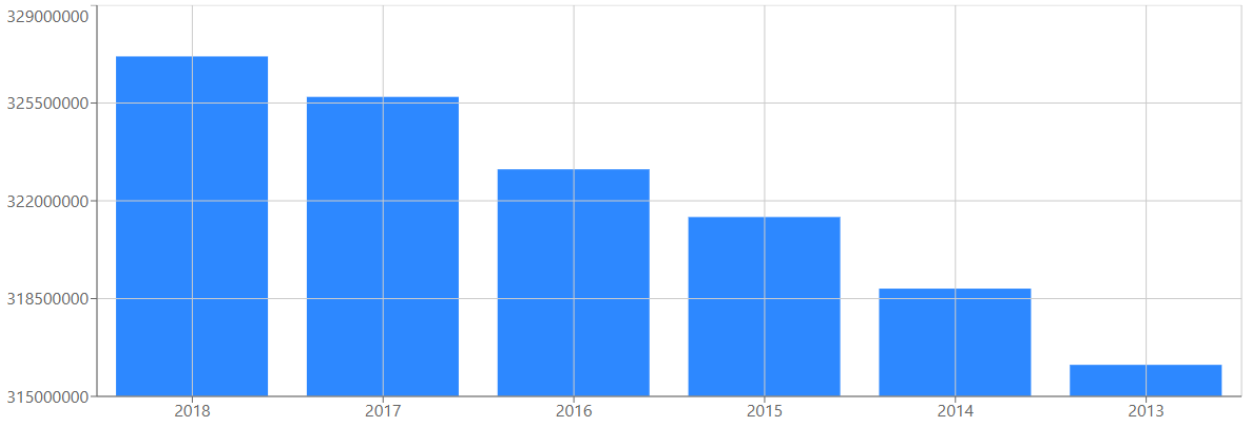
```
const [prikaz, postaviPrikaz] = React.useState(podaci);
const [tipGrafa, postaviTipGrafa] = React.useState('');
const [xOs, postaviXOs] = React.useState('');
const [yOs, postaviYOs] = React.useState('');
```

```
const nacrtajGraf = React.useCallback(
    (vrijednosti: any) => {
        if (vrijednosti[polja.yOs.naziv].includes('KLJUCEVI')) {
            const podatkovniObjekt = dohvatiPodatkovniObjekt(podaci, vrijednosti[
polja.yOs.naziv]);

            const keyObject = kljuceviUObjekt(podatkovniObjekt);

            postaviPrikaz(keyObject);
            postaviTipGrafa(vrijednosti[polja.tipGrafa.naziv]);
            postaviXOs('kljuc');
            postaviYOs('vrijednost');
        } else {
            const pomaknutiPodaci = pomakDoPodatkovnogPolja(podaci, vrijednosti[p
olja.xOs.naziv], vrijednosti[polja.yOs.naziv]);

            postaviPrikaz(pomaknutiPodaci.podaci);
            postaviTipGrafa(vrijednosti[polja.tipGrafa.naziv]);
            postaviXOs(pomaknutiPodaci.xOs);
            postaviYOs(pomaknutiPodaci.yOs);
        }
    },
    [podaci]
);
```

Programski kod 5.7. Stanja potrebna za graf i funkcija za iscrtavanje grafa koja popunjava stanja



Slika 3.8. Linijski graf koji prikazuje kretanje cijene Bitcoina u USD unutar jednoga mjeseca



Slika 3.9. Stupčasti graf koji prikazuje Američku populaciju kroz godine

3.3.4. Pomoćne funkcije

Prije slanja zahtjeva za podatke, ukoliko je upisano nešto u polje parametara, potrebno je provjeriti jesu li parametri zadani u JSON formatu. Potrebna je provjera kako bi se spriječile daljnje pogreške unutar aplikacije.

```
export const tipJSON = (kod: string) => {
    try {
        JSON.parse(kod);
    } catch (pogreska) {
        return false;
    }
    return true;
};
```

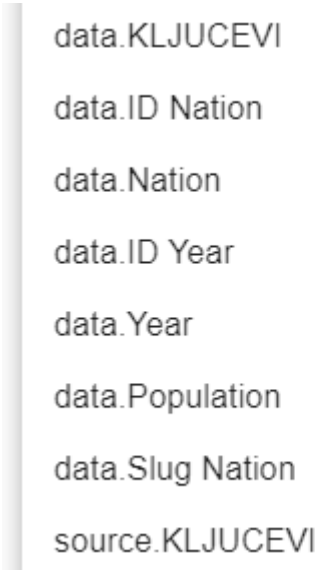
Programski kod 3.8. Funkcija za provjeru valjanost JSON zapisa

Nakon dobivanja podataka, potrebno je raščlaniti podatke tako da se dobiju putanje do svih svojstava podatkovnih objekata, kako bi se podaci mogli odabrati za prikaz na grafu (Slika 3.10.). Funkcija je rekurzivna kako bi se mogli raščlaniti podaci do bilo koje dubine. Potrebno je vođenje podatka o prethodnim dubinama trenutnog svojstva objekta kako bi se mogla dobiti cjelovita putanja. Putanja koja na kraju svoje putanje ima oznaku za ključeve će imati dodatnu pretvorbu ključeva svojstava objekata u vrijednosti.

```
export const mapirajPodatkovneKljuceve = (odgovor: any, roditelj?: string) => {
    const podaci: string | any[] = [];

    if (roditelj) {
        podaci.push(roditelj + '.KLJUCEVI');
    }
```

```
    if (Array.isArray(odgovor)) {
        podaci.push(mapirajPodatkovneKljuceve(odgovor[0], roditelj));
    } else if (odgovor) {
        Object.keys(odgovor).forEach((kljuc) => {
            if (typeof odgovor[kljuc] === 'object') {
                if (roditelj) {
                    podaci.push(mapirajPodatkovneKljuceve(odgovor[kljuc], roditel
j + '.' + kljuc));
                } else {
                    podaci.push(mapirajPodatkovneKljuceve(odgovor[kljuc], kljuc))
;
                }
            } else {
                if (roditelj) {
                    podaci.push(roditelj + '.' + kljuc);
                } else {
                    podaci.push(kljuc);
                }
            }
        });
    }

    return podaci.flat(Infinity);
};
```

Programski kod 3.9. Funkcija za raščlanjivanje svojstava podatkovnog objekta

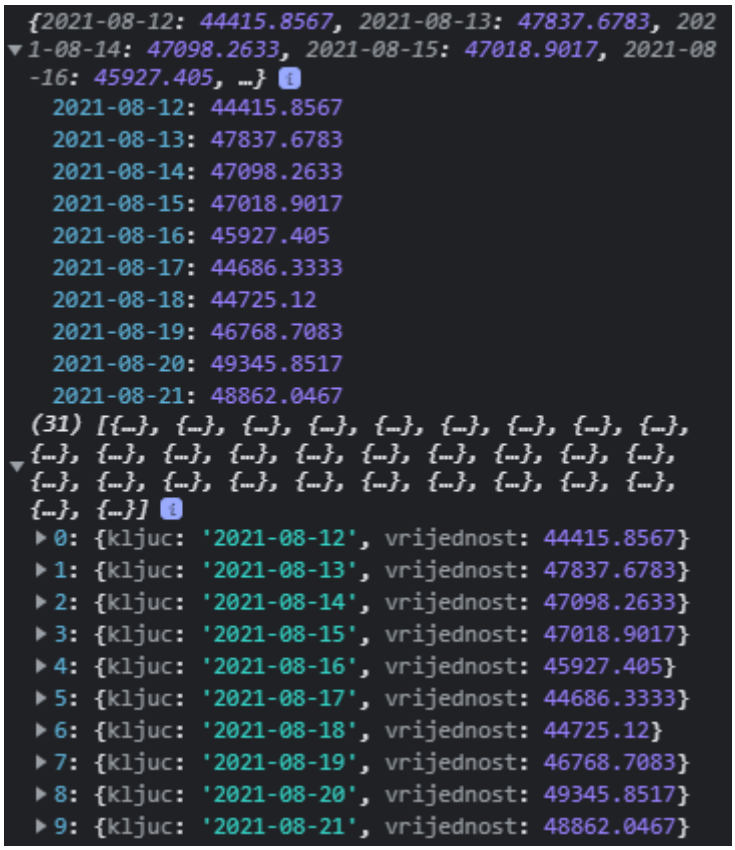


Slika 3.10. Dio podataka za odabir nakon pozivanja funkcije za raščlanjivanje svojstava podatkovnog objekta

U slučaju da se dio potrebnih podataka dobiva kao ključ, a ne vrijednost objekta, potrebno je izraditi novi objekt kojemu je ključ jedno, a vrijednost drugo svojstvo objekta. Takvi objekti se nakon toga stavljaju u polje kako bi se dobilo polje objekata s tih dviju vrijednosti (Slika 3.11.). Česti primjer ovakvoga zapisa je vrijednost nečega na određeni datum.

```
export const kljuceviUObjekt = (objekt: any) => {
    return Object.keys(objekt).map((artikl) => {
        return { kljuc: artikl, vrijednost: objekt[artikl] };
    });
};
```

Programski kod 3.10. Funkcija koja pretvara ključeve objekta u vrijednosti



Slika 3.11. Podaci prije i nakon izvršavanja funkcije koja pretvara ključeve objekta u vrijednosti

Raščlanjene podatke o putanji unutar objekta koji se dobiju kao parametri forme, potrebno je podijeliti po točki i pomicati se kroz objekt. Kada se dođe do željene lokacije potrebno je vratiti objekt.

```
export const dohvatiPodatkovniObjekt = (podaci: any, putanja: string) => {
    const dijeloviPutanje = putanja.split('.');

    let objekt = podaci;

    dijeloviPutanje.forEach((dioPutanje) => {
        if (!dioPutanje.includes('KLJUCEVI')) {
            objekt = podaci[dioPutanje];
        }
    });

    return objekt;
};
```

Programski kod 3.11: Dohvaćanje objekta na danoj lokaciji

Recharts biblioteka zahtjeva da podaci budu predani kao polje objekata s podacima. Podaci za osi se navode kao relativne putanje unutar tih objekata. U slučaju da dobiveni podatkovni objekt ne sadrži direktno u sebi polje objekata, potrebno je pomicanje glavnoga dijela podataka kako bi se

došlo do polja. Ujedno je potrebno pomaknuti putanje do svojstava objekata. Funkcija je rekurzivna te se ponavlja skroz dok se ne dođe do polja podatkovnih objekata za prikaz.

```
export const pomakDoPodatkovnogPolja = (podaci: any, xOs: string, yOs: string) =>
 {
    if (!Array.isArray(podaci)) {
        const xSplit = xOs.split('.');
        const xShift = xSplit.shift();
        const ySplit = yOs.split('.');
        const yShift = ySplit.shift();

        if (xShift !== yShift) {
            throw new Error('Nekompatibilni podaci');
        } else {
            podaci = podaci[xShift!];
            const noviPodaci = pomakDoPodatkovnogPolja(podaci, xSplit.join('.'),
ySplit.join(''));

            podaci = noviPodaci.podaci;
            xOs = noviPodaci.xOs;
            yOs = noviPodaci.yOs;
        }
    }
    return { podaci, xOs, yOs };
};
```

Programski kod 3.12. Pomicanje kroz objekt do polja objekata

4. ISPITIVANJE FUNCKIONALNOSTI I DEMONSTIRANJE MOGUĆNOSTI APLIKACIJE

Aplikacija omogućava uvoz podataka iz statičke baze podataka ili pomoću API-ja. Odabirom željenih podataka se prikazuje graf. U slučaju da neki od potrebnih podataka nisu uneseni, prikazati će se pogreška na ekranu. Osim toga ako se podaci ne mogu iscrtati na grafu, prikazati će se prazan graf.

4.1. Upravljanje pogreškama

Putanja do sjedišta datoteke je obavezna prilikom unosa podataka te u slučaju da nije unesena, prilikom pokušaja slanja forme, odmah će se prikazati pogreška na ekranu bez da se pošalju podaci (Slika 4.1.).

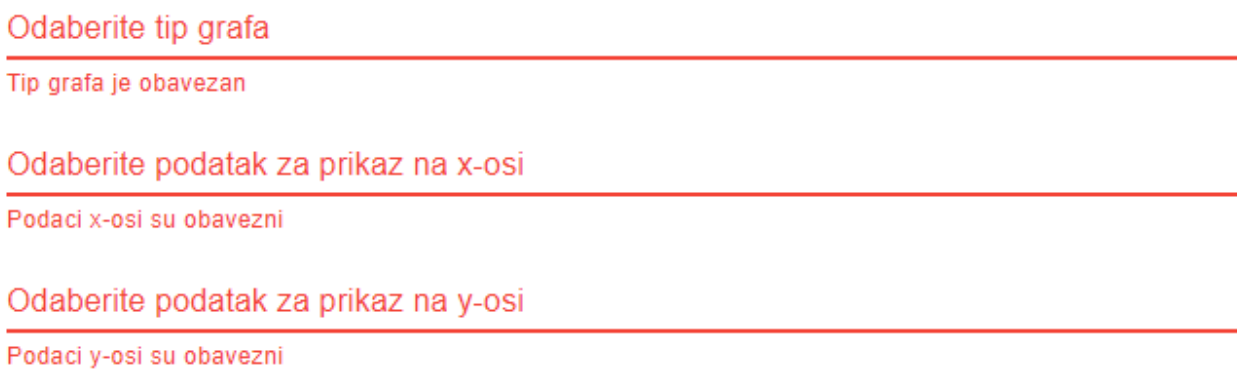
Slika 4.1. Prazno polje za unos putanje do sjedišta

Parametri, ukoliko se unose, moraju biti uneseni u valjanom JSON formatu te u slučaju da nisu, prilikom pokušaja slanja forme, odmah će se prikazati pogreška na ekranu bez da se pošalju podaci (Slika 4.2.).

Slika 4.2. Neispravan unos parametara

U slučaju da postoji pogreška prilikom dohvaćanja podataka sa navedenoga web sjedišta, aplikacija neće omogućiti korak dalje dok se ne unese valjana putanja.

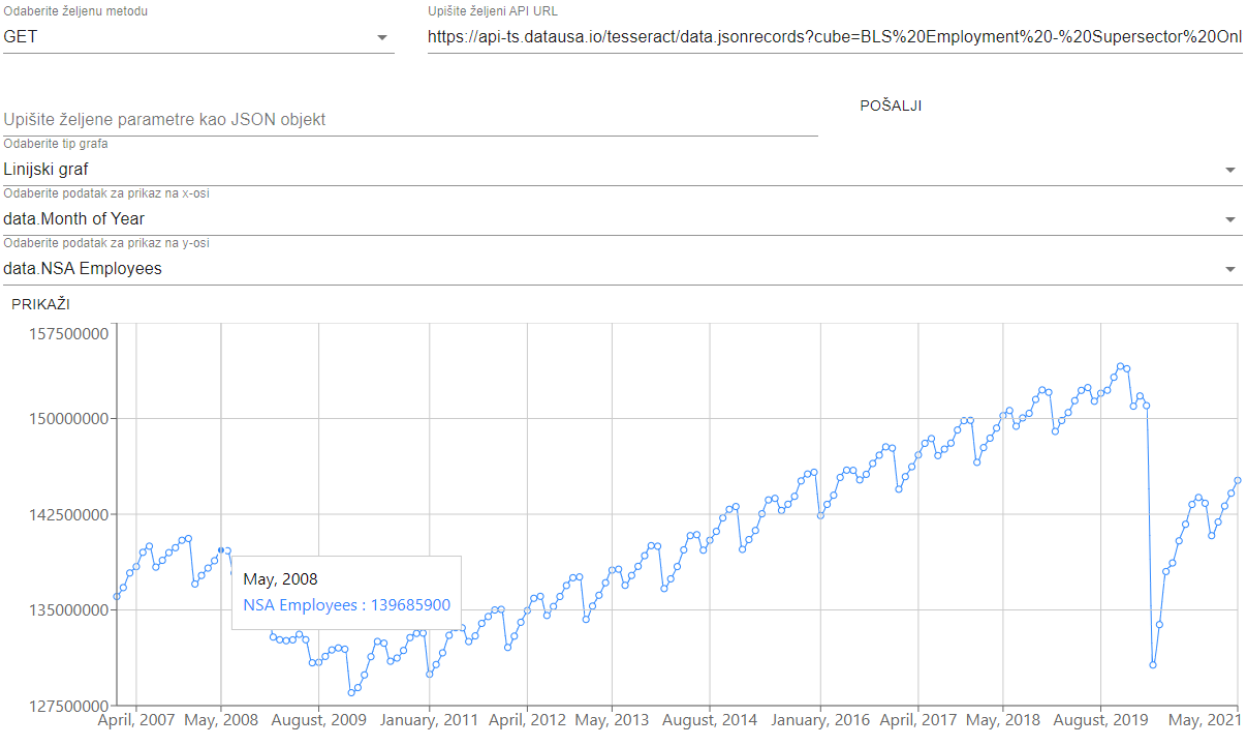
U sljedećem koraku prilikom odabira podataka za unos, sva polja su obavezna. Ukoliko se ne odaberu podaci, prilikom pokušaja slanja forme, odmah će se prikazati pogreška bez prikazivanja grafa (Slika 4.3.).



Slika 4.3. Prazna polja za odabir tipa grafa, x-osi i y-osi

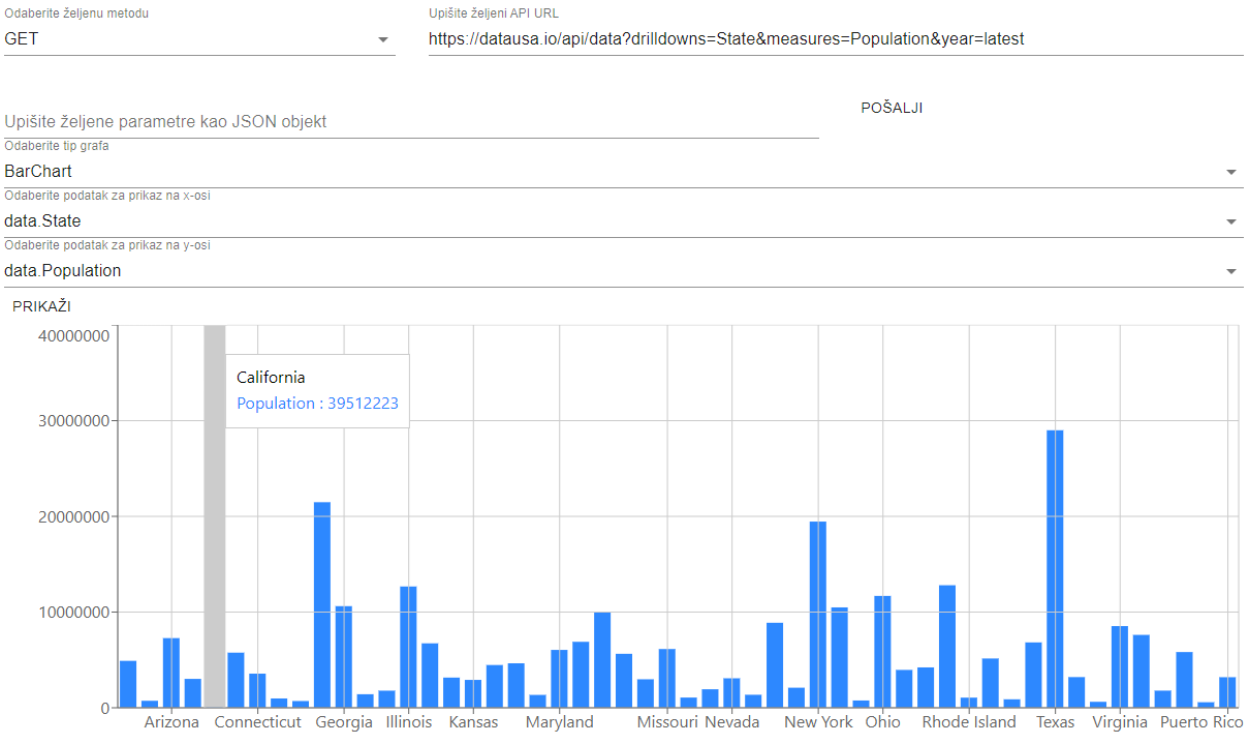
4.2. Dohvaćanje podataka koristeći API

Dohvaćanje podataka s API-ja te njegovo prikazivanje na linijskome grafu je opisano na primjeru prikazivanja broja zaposlenika nacionalne sigurnosti Amerike kroz mjesece (Slika 4.4.). Unutar aplikacije se ostavlja unaprijed odabrani HTTP GET tip poziva, upisuje se putanja do željenog API-ja te se nakon toga šalje poziv. Nakon odgovora servera, odabire se linijski tip grafa, za x-os se odabiru mjeseci unutar godina, za y-os se odabire broj zaposlenika nacionalne sigurnosti Amerike te se nakon toga podaci prikazuju. Prikazani su kružići za podatkovne točke te su te točke povezane krivuljama. Prelaskom miša preko jedne od točaka, prikazuje se oblačić koji prikazuje podatke vezane uz obje osi, odnosno mjesec i broj zaposlenih.



Slika 4.4. Broj zaposlenika nacionalne sigurnosti Amerike kroz mjesece

Uz prošlo navedeni primjer dohvaćanja podataka s API-ja, naveden je još jedan primjer uz njegovo prikazivanje na stupčastome grafu (Slika 4.5.). Unutar aplikacije se ostavlja unaprijed odabrani HTTP GET tip poziva, upisuje se putanja do željenog API-ja te se nakon toga šalje poziv. Nakon odgovora servera, odabire se stupčasti tip grafa, za x-os se odabiru savezne države, za y-os se odabire stanovništvo te se nakon toga podaci prikazuju. Prikazani stupci prikazuju stanovništvo unutar svake savezne države. Prelaskom miša preko jednog od stupaca, prikazuje se oblačić koji prikazuje podatke vezane uz obje osi, odnosno ime savezne države i broj stanovnika.

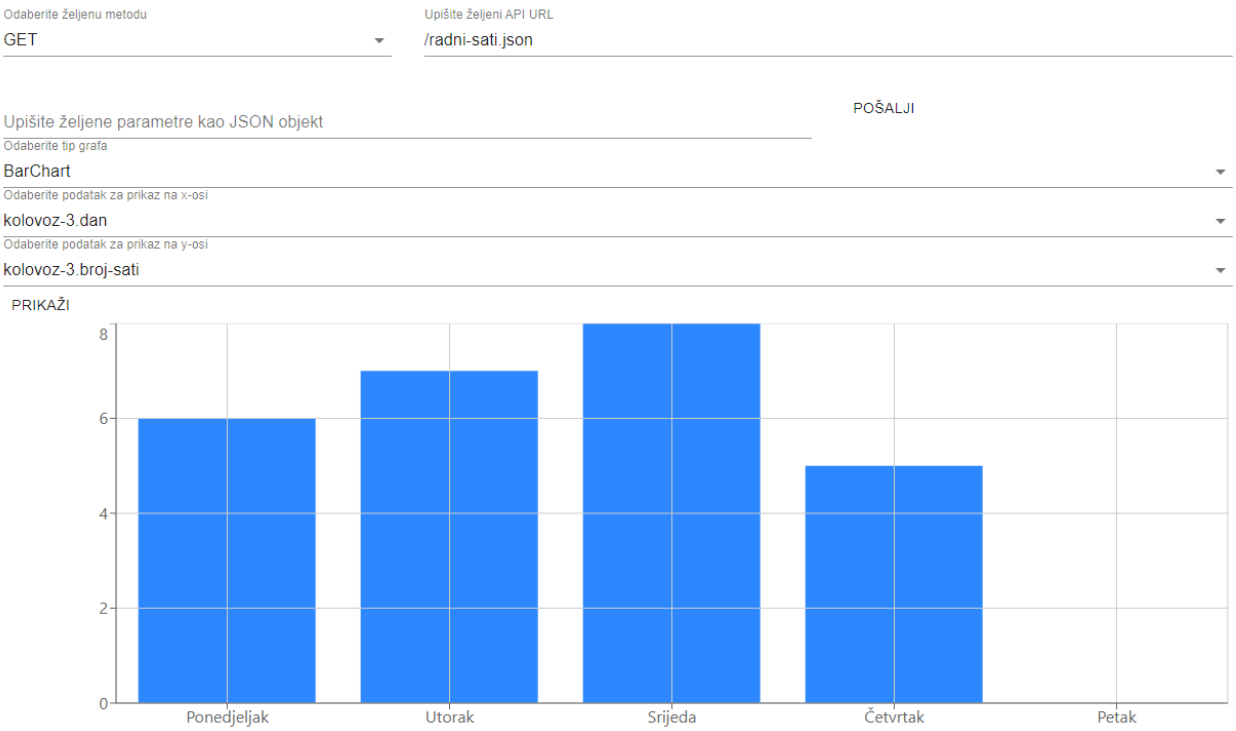


Slika 4.5. Američka populacija prema saveznim državama

4.3. Dohvaćanje podataka iz statičke baze podataka

Za dohvaćanje podataka iz statičke baze podataka, aplikacija se ponaša isto kao i kod korištenja API-ja. Jedina razlika je što se mogu uz apsolutne putanje do podataka koristiti i relativne putanje ako se podaci poslužuju sa istoga servera kao i sama stranica.

Dohvaćanje podataka iz statičke baze podataka te njihovo prikazivanje na stupčastome grafu je opisano na primjeru prikaza radnih sati za 3. tjedan kolovoza (Slika 4.6.). Unutar aplikacije se ostavlja unaprijed odabrani HTTP GET tip poziva, upisuje se putanja do željene datoteke unutar statičke baze podataka te se nakon toga šalje poziv. Nakon odgovora servera, odabire se stupčasti tip grafa, za x-os se odabiru dani u tjednu, za y-os se odabire broj odrađenih sati te se nakon toga podaci prikazuju. Prikazani stupci prikazuju odrađene radne sate unutar tjedna.



Slika 4.6. Radni sati za 3.tjedan kolovoza

5. ZAKLJUČAK

Tema ovoga rada je Web aplikacija za prikazivanje višedimenzionalnih podataka, prema kojoj je izrađena aplikacija. Kao preduvjet tome, bilo je potrebno proučiti vrste alata i specifične biblioteke za prikaz višedimenzionalnih podataka te prepoznati potreban tip alata koji će biti primijenjen unutar sustava. Nakon odabira tipa alata, bilo je potrebno proučiti i odabrati alate i tehnologije izrade. Web aplikacija je kreirana pomoću React biblioteke za izradu web sučelja. Recharts biblioteka je odabrana za prikaz podataka zbog jednostavne primjene prilikom prikaza grafova opće namjene te zbog jednostavne integracije s React bibliotekom. Podaci se uvoze iz statičke baze podataka ili pomoću API-ja u aplikaciju te se prikazuju na željenim grafovima. Korisnik odabire podatke za prikaz i tip grafa. U slučaju da nisu ispravne vrijednosti ispisuju se pogreške. Ovakav izrađeni sustav za prikazivanje višedimenzionalnih podataka, može poslužiti svim pojedincima koji trebaju na jednostavan način biti u mogućnosti imati prikazan jedan od često korištenih grafova, uz proizvoljne podatke. Sustav bi se mogao dodatno poboljšati dodavanjem mogućnosti prikaza kružnih, površinskih, plošnih, polarnih i mnogih drugih grafova. Uz to, poboljšanje bi se moglo ostvariti dodatnom prilagodbom podataka, kako ne bi morali biti prezentirani kao objekti unutar polja te da podaci ne moraju biti uvezeni u stranicu unutar JSON formata.

Literatura:

[1] K.Brush, E. Burns, „What is data visualization and why is it important?“, https://searchbusinessanalytics.techtarget.com/definition/data-visualization, TechTarget zadnji pristup 16.9.2021.

[2] N. Kakuev, „Understanding front-end data visualization tools ecosystem in 2021“, https://cube.dev/blog/dataviz-ecosystem-2021/, Cube Dev, Inc, 2021., zadnji pristup 8.9.2021.

[3] Recharts Group, „Recharts“, https://recharts.org/en-US/, zadnji pristup 8.9.2021.

[4] M. Bostock, „Streamgraph Transitions“, Observable, Inc., 2018., https://observablehq.com/@d3/streamgraph-transitions, zadnji pristup 8.9.2021.

[5] Material-UI, „Data Grid“, https://material-ui.com/components/data-grid/, zadnji pristup 8.9.2021.

[6] vis.js, „vis-timeline“, https://github.com/visjs/vis-timeline, zadnji pristup 8.9.2021.

[7] V. Agafonkin, „Leaflet“, Leaflet, https://leafletjs.com/, zadnji pristup 8.9.2021.

[8] J. Davies, „Word Cloud“, https://www.jasondavies.com/wordcloud/, zadnji pristup 8.9.2021.

[9] C. Pacak, „Visualizing Recessions with Pandas and Three.js“, Towards Data Science, 2018., https://towardsdatascience.com/visualizing-recessions-with-pandas-and-three-js-2df3f514dc44, zadnji pristup 8.9.2021.

[10] Facebook, „React“, GitHub, inc., https://github.com/facebook/react, zadnji pristup 8.9.2021.

[11] Typescript, „TypeScript is JavaScript with syntax for types.“, https://www.typescriptlang.org/, zadnji pristup 8.9.2021.

[12] Material-UI, „Material-UI“, https://material-ui.com/, zadnji pristup 8.9.2021.

[13] Recharts Group, „Recharts“, GitHub, Inc., https://github.com/recharts/recharts, zadnji pristup 8.9.2021.

[14] B. Luo, „React Hook Form“, React Hook Form, https://react-hook-form.com/, zadnji pristup 8.9.2021.

Sažetak:

Naslov: Web aplikacija za prikazivanje višedimenzionalnih podataka

Navedene su i opisane postojeće aplikacije za prikaz višedimenzionalnih podataka. Detaljno su objašnjeni alati, biblioteke i vrste prikaza višedimenzionalnih podataka. Za izradu grafova unutar aplikacije, korišten je Recharts alat opće namjene zbog svoje lake integracije sa React bibliotekom za izradu internetskih stranica. Uz React su od biblioteka odabrane TypeScript za dodavanje tipova podataka, Material-UI za izradu dizajna aplikacije i React Hook Form za upravljanje formama, kao alati koji olakšavaju implementaciju internetske stranice. Izrađen je sustav pomoću kojega se mogu prikazati linijski i stupčasti grafovi, pomoću podataka dobivenih uz dohvaćanje podataka iz statičke baze podataka ili API-ja. Korisnik odabire tip grafa na kojemu želi imati prikazane podatke te odabire podatke koje želi da se prikažu. Poruke pogreške se prikazuju unutar aplikacije u slučaju da unos podataka nije valjan.

Ključne riječi: aplikacija za prikaz podataka, Recharts, vizualizacija

Abstract:

Title: Multidimensional data visualization web application

Existing applications for multidimensional data visualization are listed and described. Tools, libraries, and types of multidimensional data display are explained in detail. The general-purpose of Recharts tool was used to create the application due to its easy integration with the React website creation library. In addition to React, TypeScript for adding data types, Material-UI for creating application design, and React Hook Form, for form management have been selected as libraries that work as tools for website implementation. A system that can display line and bar graphs using data obtained by retrieving data from a static database or API has been developed. The user selects the suitable graph type and selects the data he wants to display. Error messages are displayed within the application in case the entered data is invalid.

Keywords: data visualization application, Recharts, visualization

Životopis

Josip Blažević, rođen je u Osijeku 16.10.1997. godine. Prebivalište mu je u Josipovcu, prigradskom naselju grada Osijeka. Pohađao je osnovnu školu Josipovac i Elektrotehničku i prometnu školu Osijek, smjer tehničar za računalstvo. Trenutno pohađa Fakultet elektrotehnike, računarstva i informacijskih tehnologija, diplomski sveučilišni studij računarstva. Osvojio je 2. mjesto na županijskom natjecanju Osječko-baranjske županije iz osnova informatike na Infokupu. Pohađao je tečaje Ljetne škole programiranja i Software startup academy. Obišao je #INIT2019 IT konferenciju u Banja Luci. Uz brojne poslove koje je do sada obavljao, radio je kao demonstrator na fakultetu na kolegijima Programiranje 1, Programiranje 2, Osnove elektrotehnike 1 i Komunikacijskim mrežama, radio kao programer u Igloo obrtu za programiranje te trenutno radi kao programer u Binaryx dev ops d.o.o. Trenutno se usavršava na području izrade korisničkih sučelja internetskih stranica.