Razvoj korisničkog sučelja web aplikacija uz primjenu web komponenti

Besednik, Mateo

Undergraduate thesis / Završni rad

2022

Degree Grantor / Ustanova koja je dodijelila akademski / stručni stupanj: **University of Zagreb, Faculty of Organization and Informatics / Sveučilište u Zagrebu, Fakultet organizacije i informatike**

Permanent link / Trajna poveznica: https://urn.nsk.hr/urn:nbn:hr:211:658152

SVEUČILIŠTE U ZAGREBU
FAKULTET ORGANIZACIJE I INFORMATIKE
V A R A Ž D I N

Mateo Besednik

Razvoj korisničkog sučelja web aplikacije uz primjenu web komponenti

ZAVRŠNI RAD

Varaždin, 2022.

SVEUČILIŠTE U ZAGREBU

FAKULTET ORGANIZACIJE I INFORMATIKE

V A R A Ž D I N

Mateo Besednik

Matični broj: 35918/07–R

Studij: Primjena informacijske tehnologije u poslovanju

Razvoj korisničkog sučelja web aplikacije uz primjenu web komponenti

ZAVRŠNI RAD

Mentor:

Prof. dr. sc. Dragutin Kermek

Varaždin, rujan 2022.

Mateo Besednik

Izjava o izvornosti

Izjavljujem da je moj završni/diplomski rad izvorni rezultat mojeg rada te da se u izradi istoga nisam koristio drugim izvorima osim onima koji su u njemu navedeni. Za izradu rada su korištene etički prikladne i prihvatljive metode i tehnike rada.

Autor/Autorica potvrdio/potvrdila prihvaćanjem odredbi u sustavu FOI-radovi

---

Sažetak

U ovom radu biti će prikazane najpopularnije tehnologije za izradu web aplikacije, s time da će poseban naglasak biti na poglavlje „web komponenti“. Cilj rada je objasniti što su web komponente, kako se koriste i čemu služe. Osim web komponenti u radu će biti prikazane osnove HTML-a, CSS-a te JavaScript jezika, pošto te tehnologije imaju veliku ulogu pri izradi web stranice odnosno aplikacije. Pri opisu osnovnih web tehnologija neće se ulaziti u detalje kao nabrajanje i objašnjavanje svih HTML oznaka, CSS stilova te JavaScript funkcija jer je naglasak ovog rada na web komponente.

Ključne riječi: HTML5, CSS, JavaScript, web, web komponente, DOM, DOM u sjeni, prilagođeni elementi, višekratna uporaba, web tehnologije

Sadržaj

1. Uvod

Web aplikacija je softverska aplikacija koju korisnik pokreće u pregledniku. Web aplikacije postale su vrlo popularne 1990-ih i 2000-ih. Programeri ne moraju pripremati različite verzije iste aplikacije za Microsoft Windows, Mac OS, Linux itd. Aplikacija se izrađuje samo jednom za bilo koju platformu i može raditi na bilo kojem operacijskom sustavu. Međutim, različita realizacija i druga sučelja u preglednicima može uzrokovati probleme tijekom razvoja web aplikacija i njihove daljnje podrške.

U zadnjih 30 godina web je jako napredovao, web je u početku bio sredstvo za objavljivanje, te dijeljenje informacija i sadržaja, ali s vremenom se razvio u snažnu i fleksibilnu platformu koja podržava veliki i raznolik niz slučaja upotrebe. Web platforma je stekla puno novih mogućnost i uzastopnih verzija biblioteka (eng. library), okvira (eng. framework) te alata, koji pomažu programerima da ispune sve zahtjeve korisnika te uostalom da sami sebi olakšaju proces izrade web aplikacije. Velika transformacija izrade web aplikacija nastala je kada su se pojavile web komponente (eng. web components). Rastavljanje koda na komponente, odnosno kreiranjem komponenata pojedinačnih za strukturu, stil te funkcionalnost web aplikacije, pomoglo je programerima da lakše organiziraju strukturu projekta tj. Koda i da se s vremenom bolje snalaze u kodu. Kada se jednom napravi komponenta, ona se može koristiti na više mjesta u projektu (eng. reusable component), te se na taj način smanjuju linije koda i općenito je kod uredniji i organiziraniji. Komponente se također mogu koristiti i u više različitih projekata, te si na taj način programeri olakšavaju posao.

Popularni web okviri poput React-a, Angular-a, Vue-a pomogli su voditi komponentu revolucija i doista većina komponenti danas je specifična za a zadani okvir ili biblioteka.

Web komponentne predstavljaju izraz za novu cjelinu unutar web značajki koje nude izravnu podršku za razvoj komponenti za višestruku upotrebu. Prilagođeni elementi (eng. custom elements) omogućuju HTML-u da proširi svoje mogućnosti, definiranje vlastitih oznaka koje besprijekorno funkcioniraju. Sjena DOM (eng. Shadow DOM) omogućuje odabir enkapsulacije izvornog stila, osiguravajući da se CSS pravila komponente se ne krše s CSS pravilima koja nisu napisana u sjeni DOM-a.

2. Povijest web aplikacija

Povijest razvoja web aplikacija nije uobičajena. Programeri su morali pronaći rješenje za postojeće probleme, nije bilo lako natjerati web aplikacije da rade na različitim operacijskim sustavima. Svaka je aplikacija imala svoj unaprijed sastavljeni klijentski program i morala se zasebno instalirati na računalo svakog korisnika. Nadalje, komponente klijenta i poslužitelja bile su čvrsto vezane za određeni operacijski sustav i arhitekturu računala. Zbog toga je prijenos aplikacija na druge sustave bio skup. Korisnik je u ranijim danima Web-a, kao web stranicu dobio statični dokument. Bilo je teško imati interaktivno iskustvo dok se radilo s takvom stranicom. Kada ste unijeli bilo kakve izmjene na web stranicu, trebalo vam je vremena da osvježite ovu stranicu u onoj mjeri u kojoj ste se vratili na njezin poslužitelj. [1]

1995. godina ključna je u eri web-a. Netscape Communications predstavio je JavaScript, skriptni jezik na strani klijenta (eng. client-side) koji programerima omogućuje poboljšanje korisničkog sučelja s dinamičkim elementima. JavaScript je učinio web bržim i produktivnijim jer se podaci više nisu slali poslužitelju za generiranje cijele web stranice. JavaScript je jedna od tri najznačajnije tehnologije (s HTML-om i CSS-om) proizvodnje sadržaja za WWW. Ima programsko sučelje za aplikacije koje stručnjacima omogućuje rad s tekstovima, datumima i raznim regularnim izrazima. [1]

1996. predstavljen je „Macromedia Flash“. To je također bila revolucionarna inovacija koja je web učinila "svjetlijim" i interaktivnim. On je omogućio programerima da obogate web stranice animacijom. Ova multimedijska softverska platforma radi s animacijama, različitim vrstama igara preglednika, vektorskom grafikom te internetskim i mobilnim aplikacijama. Bio je to solidan napredak kada je Adobe Flash u svoju animaciju uključio strujanje zvuka i videa. Ova platforma omogućuje korisniku interakciju s uređajem uz pomoć miša, mikrofona i tipkovnice. Štoviše, bilo koje interakcije programa na strani klijenta više ne zahtijevaju komunikaciju s poslužiteljem. Od 1996. godine primjećuje se sve veća popularnost različitih interaktivnih online video igara zahvaljujući revolucionarnoj tehnologiji koju nudi Macromedia Flash. Web stranice su dobile svoj redoviti izgled. Korisnikov rad više nisu ometali čudni i neočekivani oglasi i streaming videozapisi utoliko što su usporili rad web stranice i potrošili dodatni promet. Ipak, još uvijek postoje jeftine web stranice koje koriste Flash na svojim web stranicama. U današnje vrijeme Adobe Flash se uglavnom koristi za izradu raznih video igara i interaktivnih aplikacija za pametne telefone i tablete. [1]

Godine 1999. pojam web aplikacije pojavio se na „Java“ jeziku. Kasnije, 2005., Ajax je predstavio Jesse James Garrett u svom članku "Ajax: Novi pristup web aplikaciji". Ovaj kompleks tehnika web razvoja omogućio je programeru sastavljanje asinkronih web aplikacija. Web aplikacije mogu slati podatke poslužitelju i preuzimati ih s njega bez ometanja rada na određenoj stranici. Ne mora preuzimati cijelu stranicu. Ajax je prvi put stvoren za Internet Explorer, no vrlo brzo su ga prihvatili i preglednici poput Opera, Mozilla i Safari. Google ovu tehniku intenzivno koristi u Gmailu i Google kartama od 2005. godine. [1]

Najnovija verzija HTML-a, HTML5, u svijet je ušla 2014. HTML5 služi za predstavljanje sadržaja u WWW-u i njegovo slaganje u logičke strukture. HTML5 je nova i poboljšana verzija jezika. Njegova je uloga podržati potpuno novu vrstu multimedije koja se sada stalno razvija. Ako govorimo o animaciji, trebali bismo reći da HTML5 nije autonomna tehnologija. Ne opskrbljuje web stranice animacijama ili raznim streaming videozapisima. Ovaj jezik mogu čitati ljudi i računala. Omogućuje programerima stvaranje web aplikacija koje su doista neovisne o web preglednicima i platformama. Što je s popularnošću HTML5? Prema istraživanju, najmanje 34% najpopularnijih web stranica koristilo je HTML5. Stoga se važnost ove tehnologije ne može precijeniti. Povijest razvoja web aplikacija prilično je komplicirana. Postoje mnoge tehnologije (Flash, Java, Silverlight itd.) koje čine rad na Internetu što je moguće lakšim. Interaktivnost Web-a postala je ogromna te će u budućnosti biti još učinkovitija i raznolika. Ajax je jedan od najboljih primjera skupa tehnologija koje poboljšavaju razinu interaktivnosti između korisnika i stroja. [1]

3. Osnovne web tehnologije

Budući da računala još uvijek ne mogu međusobno komunicirati na način na koji to ljudi rade, potrebno im je dodijeliti naredbe kroz kodove. Web tehnologije su jezici za pisanje, označavanje i programiranje web stranica. To su jezici kao HTML, CSS i JavaScript te razne biblioteke i web okviri koji pomažu u izradi stranica.

Preglednici traže informacije, te a zatim prikazuju korisnicima sadržaj na način na koji ih korisnici (ljudi) mogu razumjeti. Najpopularniji preglednici su:

- **Google Chrome** - trenutno najpopularniji preglednik koji vam je ponudio Google
- **Safari** - Appleov web preglednik
- **Firefox** - preglednik otvorenog koda koji podržava Mozilla Foundation
- **Internet Explorer** - Microsoftov preglednik
  [2]

Najpopularnije tehnologije za izradu web stranica su HTML i CSS. Zahvaljujući HTML-u, web preglednici znaju što trebaju prikazati kada dobiju zahtjev. CSS je opisuje kako se HTML elementi prikazuju na ekranu. Iako su HTML i CSS važne web tehnologije, bez primjene nekog programskog jezika poput JavaScripta ne može se napraviti puno, zapravo mogu se napraviti samo statičke web stranice. JavaScript se koristi baš iz razloga da se stvori dinamička web stranica, stranica koja ima interakciju sa korisnikom. Korištenjem te tri tehnologije, stvaraju se brze, lijepe i dinamičke stranice.

Ovo su samo osnovne tehnologije, danas postoji sve više knjižnica (eng. library) i okvira (eng. framework) koje ubrzavaju i ujedno olakšavaju pisanje koda. Naime kod je već unaprijed napisan i spremljen, te se poziva preko imena funkcija.

HTML, CSS, JavaScript te dinamičke i statičke web stranice biti će detaljnije objašnjene kasnije u radu.

3.1 HTML

HTML je skraćenica od „Hypertext Markup Language“. HTML ima jednostavnu sintaksu te se koristi za kreiranje hipertekstualnih dokumenata, neovisnih o platformi na kojoj se koriste. HTML-om se određuje struktura i sadržaj web stranice odnosno aplikacije. Struktura sadržaja napisana u HTML-u kasnije se povezuje sa CSS-om te se uređuje. [3]

HTML je jezik oznaka, oznaka može predstavljati email, multimediju, hipertekstualnu vijest, izbornik, rezultate upita na bazu i još mnogo toga. [4]

HTML element definiran je početnom oznakom, nekim sadržajem i završnom oznakom. Oznaka koja ide na kraju se u pravilu uvijek piše, ali neki HTML elementi će se prikazati ispravno čak i u slučaju kada se oznaka za zatvaranje elementa zaboravi napisati, nikada se ne treba oslanjati na to. [5]

```
<oznaka>Ovdje ide sadržaj...</oznaka>
```

3.2 CSS

CSS omogućuje stvaranje pravila koja određuju kako će se sadržaj napisan u jeziku HTML pojaviti u pregledniku. CSS je kratica za kaskadni stil formatiranja (eng. Cascading Style Sheets). [6]

Dok se nije pojavio CSS, dizajn sadržaja se mogao i može se urediti u samom HTML dokumentu, ali do određene granice. Na primjer nije se moglo postavljati animacije, pozicionirati elemenate korištenjem „flexbox-a“ ili „grid-a“. Uređivanje web stranice unutar HTML dokumenta stvorilo je problem miješanja koda čija je svrha prezentacije sa strukturom samog sadržaja. HTML kod koji se tada koristio za stiliziranje oznaka, morao se ponovno pisati na svakoj oznaci. Pojavom CSS-a se riješio taj problem. Trenutačna verzija CSS-a je CSS 3, a prva verzija je nastala 1996. godine. [7]

CSS kod se sastoji od CSS pravila odnosno uputa.

```
h1 {
color: blue;
}
```

Upute navedene u primjeru iznad oblikuju tekst naslova prve razine, u ovom slučaju mijenja se boja teksta unutar oznake `<h1>` u plavu boju. Dio CSS uputa sastoji se od selektora, u ovom slučaju selektira se h1. Dio koji se nalazi unutar zagrada zove se svojstvo, color je svojstvo u ovom primjeru. Blue predstavlja vrijednost svojstva color. Svojstvo i vrijednost zajedno čine deklaraciju. [7]

3.3. JavaScript

JavaScript je uveden 1995. kao način dodavanja programa na web stranice u pregledniku Netscape Navigator. Jezik su od tada prihvatili svi drugi glavni grafički web preglednici. Omogućio je izradu moderne web aplikacije s kojima je moguće izravno komunicirati bez ponovnog učitavanja stranice za svaku radnju. [8]

Važno je napomenuti da JavaScript nema gotovo nikakve veze s programskim jezikom pod nazivom Java. Sličan naziv inspiriran je marketinškim razlozima. Kad je JavaScript bio uveden, jezik Java se jako prodavao i dobivao je na popularnosti.

Nakon usvajanja izvan Netscapea, napisan je standardni dokument koji opisuje način rada JavaScripta tako da različiti dijelovi softvera koji tvrde da podržavaju JavaScript zapravo govore o istom jeziku. To se naziva standard ECMAScript, prema organizaciji Ecma International koja je izvršila standardizaciju. U praksi se izrazi ECMAScript i JavaScript mogu koristiti naizmjenično - dva su nazivi za isti jezik. [8]

3.4 Sadržaj stranice

Sadržaj web stranice može biti:

- Statički
- Dinamički

3.4.1 Statične web stranice

Statične web stranice obično dolaze s fiksnim brojem stranica s određenim izgledom. Kada se stranica pokreće u pregledniku, sadržaj je doslovno statičan i ne mijenja se kao odgovor na radnje korisnika. Statička web stranica obično se izrađuje s HTML-om i CSS-om u jednostavnim uređivačima teksta poput Notepada. Za izradu statične nije potrebno toliko vremena ni truda kao u slučaju dinamičkih web stranica. Ako stranice statične web stranice moraju izgledati drugačije, HTML kod se može lako duplicirati na svakoj od ovih stranica, uključujući potrebne promjene. [9]

Iako će web stranica prikazivati istu stvar bez zamršenih navigacijskih detalja, statične web stranice ne moraju sadržavati samo običan tekst. Zapravo se mogu koristiti različiti multimedijski elementi i videozapisi. HTML web stranica može izgledati lijepo, ali izvorni kod stranice se neće promijeniti, bez obzira na radnje koje korisnik poduzme. [9]

3.4.2 Dinamične web stranice

U usporedbi sa statičnim web stranicama koje su isključivo informativne prirode, dinamična web stranica je funkcionalnija. Omogućuje korisnicima interakciju s podacima navedenim na stranici. Naravno, to zahtijeva korištenje više od HTML koda. [9]

Statične web stranice koriste samo klijentski HTML i CSS kod, dok se dinamičke web stranice oslanjaju i na klijentske i na poslužiteljske skriptne jezike kao što su JavaScript, PHP ili ASP. Kada korisnik pristupi dinamičnoj web stranici, web mjesto se može promijeniti pomoću koda koji se pokreće u pregledniku i/ili na poslužitelju. Za generiranje dinamičkog sadržaja takve web stranice koriste kombinaciju skriptiranja na strani poslužitelja i klijenta. Skriptiranje na strani klijenta odnosi se na kôd koji izvršava preglednik, obično s JavaScriptom. U međuvremenu, skriptiranje na strani poslužitelja odnosi se na kôd koji poslužitelj izvršava (prije nego što se sadržaj pošalje u preglednik korisnika). [9]

3.5 Dodavanje JavaScript-a u html datoteku

JavaScript kod se dodaje u HTML dokument upotrebom HTML oznake `<script>` koja se omotava oko JavaScript koda. Oznaka `<script>` se može postaviti unutar oznake `<head>` HTML-a ili u oznaku `<body>`, ovisno o tome kada je potrebno učitati JavaScript. [10]

```
<script>
document.getElementById("primjer").innerHTML = `Umetanje JavaScript-a`;
</script>
```

JavaScript se osim u oznaci `<script>` može spremiti u posebnu datoteku te pozvati unutra HTML dokumenta unutar `src="“` atributa koji se nalazi u početnoj oznaci `<script>` oznake. Da bi to bilo moguće, datoteka sa JavaScriptom mora imati nastavak „.js“. [10]

```
<script src="primjer.js"></script>
```

Kao i u slučaju u prethodnom poglavlju, sa uključivanjem CSS-a u HTML, puno je bolje odvojiti kod u zasebnu datoteku, na taj način je kod organiziraniji i programeru se neće biti teško snaći.

Vanjska skripta može se referencirati na 3 različita načina:

- S punim URL -om (puna web adresa)

```
<script src="https://www.mbesednik/primjer.js"></script>
```

- S putanjom datoteke (poput /js /)

```
<script src="/mbesednik/primjer.js"></script>
```

- Bez ikakve putanje [10]

```
<script src="primjer.js"></script>
```

4. Web komponente

Web komponente su skup API-ja web platformi koji omogućuju stvaranje novih prilagođenih, enkapsuliranih HTML oznaka za višekratnu uporabu za upotrebu na web stranicama i web aplikacijama. Prilagođene komponente temelje se na standardima web komponenti, radit će u modernim preglednicima i mogu se koristiti sa bilo kojom JavaScript knjižnicom ili okvirom koji radi s HTML-om. [11]

Web komponente temelje se na postojećim web standardima. Značajke za podršku web komponenti trenutno se dodaju specifikacijama HTML-a i DOM-a, što web programerima omogućuje jednostavno proširenje HTML-a novim elementima s enkapsuliranim stilom i prilagođenim ponašanjem. [11]

Web komponentama se nastoji riješiti problem višestrukog pisanja istog ili sličnog koda. One se sastoje od tri glavne tehnologije koje se mogu koristiti zajedno za stvaranje svestranih prilagođenih elemenata s enkapsuliranom funkcionalnošću koji se mogu ponovno koristiti gdje god želite bez straha od miješanja koda. [12] Upotrebom web komponenti nastoji se omogućiti višestruka upotreba koda (eng. reusable code) čime se znatno smanjuju linije koda.

4.1 Prilagođeni elementi

Sada postoji standardni način generiranja prilagođenih elemenata unutar raznih okvira i platforma. Jezgra HTML-a, jedinstveni element je otvoren za sve. Ovo je prvi korak u izgradnji HTML-a u ono što su web aplikacije oduvijek trebale biti. Ovdje se radi o mnogo više od stvaranja novih tekstualnih oznaka. [13]

Prilagođeni elementi su zapravo skup JavaScript API-ja koji omogućuju definiranje prilagođenih elemenata i njihovog ponašanja, koji se zatim mogu koristiti po želji na korisničkom sučelju. [12]

```
<ul class="product-listing">
<li class="product">
<img src="img/log.jpg" alt="log" />
<h3>log</h3>
```

```
<p>
Lorem ipsum dolor sit amet<br />
consectetur adipisicing elit. <br />
Minus harum tenetur rem quos <br />
laboriosam eius veniam laborum. <br />
Cupiditate nulla eveniet esse <br />
eligendi rerum natus. <br />
</p>
<ul class="reviews">
<li class="review">
<div class="reviewer">
<img src="img/rcrumb.jpg" alt="R. Crumb" />
<div class="reviewr-name">R. Crumb</div>
</div>
<p>
Lorem ipsum, dolor sit amet consectetur adipisicing elit.
Voluptates repellat itaque ullam blanditiis voluptatibus
consequatur.
</p>
</li>
</ul>
</li>
</ul>
```

Oba HTML dokumenta prikazuju istu stvar, no može se vidjeti kolika je razlika u pisanju i jasnoći čitanja.

```
<product-listing>
<product-desc name="log" img="img/log.jpg">
Lorem ipsum dolor sit amet<br />
consectetur adipisicing elit. <br />
Minus harum tenetur rem quos <br />
laboriosam eius veniam laborum. <br />
Cupiditate nulla eveniet esse <br />
eligendi rerum natus. <br />
<product-reviews>
<product-review>
<product-reviewer name="R. Crumb" img="img/rcrumb.jpg" />
Lorem ipsum, dolor sit amet consectetur adipisicing elit.
Voluptates repellat itaque ullam blanditiis voluptatibus
consequatur.
</product-review>
</product-reviews>
<product-desc/>
</product-listing>
```

Upotrebom prilagođenih elemenata uklonjeni su vanjske oznake i svojstva. Osnovna namjena prilagođenih elemenata je pružiti enkapsulacijsku funkcionalnost koja sadrži mehanizam za ponovno korištenje istog načina. Ovu značajku ne podržavaju svi preglednici pa je prije samog korištenja preporuka pogledati da li se preglednik nalazi na listi podražavajućih preglednika. Da bi preglednik prepoznao i podržao prilagođeni element, mora se prvo registrirati u pregledniku. To se radi putem funkcije document.registerElement, koji uzima dva argumenta. Prvi argument je naziv prilagođenog elementa koji se registrira. Drugi argument je objekt opcija koji omogućuje definiranje prototipa iz kojeg elementi nasljeđuju. Nakon što je prilagođeni element registriran, može se referencirati i koristiti kao izvorni element. Kreiranje prilagođenih elemenata se vrši pomoću „define“ na način gdje je „word-count“ element čija instanca pripada klasi WordCount i proširuje element odlomka.

```
customElements.define('word-count', WordCount, {extends: 'p'});
```

Postoje dva tipa prilagođenih elemenata koji se zovu autonomni prilagođeni element i prilagođeno ugrađeni elementi. Autonomni prilagođeni element je samostalan element koji se ne proširuje iz HTML elementa. Oni se pozivaju kao HTML element upisujući njihov naziv unutar HTML oznaka. Prilagođeno ugrađeni element je prošireni element iz HTML elementa. Kod kreiranja ovakvog tipa elementa potrebno je odrediti iz kojeg HTML elementa su prošireni i to na način da se upotrijebi „is“ atribut `<p is="word-count">`.

Prilagođene elemente je moguće proširivati dodavanjem svojstva „extends“ u objekt `document.registerElement`. Prošireni elementi i njihove baze je moguće pozivati na 3 načina:

Deklarirane unutar HTML dokumenta:

```
<opis-slike></opis-slike>
<tekstualni-opis></tekstualni-opis>
```

Putem document.createElement

```
let opisSlike = document.createElement('opis-slike');
let tekstualniOpis = document.createElement('tekstualni-opis');
```

Novi operator

```
let opisSlike = new Slika();
let tekstualniOpis = new Tekst();
```

Nativne elemente je isto moguće proširit. Oni se proširuju na isti način kao i prilagođeni elementi uz pomoć „extends“.

Prilagođeni elementi mogu sadržavati svoja svojstva i metode što omogućuje developerima da kreiraju javni API. Proces dodavanja svojstva se ne razlikuje previše u odnosu na dodavanje svojstva objektu. Za dodavanje svojstva i metoda koriste se konstruktori unutar kojih se mogu definirati razne metode i svojstva.

```
class WordCount extends HTMLParagraphElement {
constructor() {
super();

// funkcionalnost elemenmta se piše ovdje

...

}
}
```

4.1.1 Korištenje stilova prilagođenim elementima

Stiliziranje prilagođenih elemenata ne razlikuje se od uobičajenog stiliziranja ostalih elemenata. Jedina razlika je što postoji novi pseudo selektor koji sprječava FOUCs (eng. flashes of unstyled content) za one elemente koje DOM nije stigao registrirati. Preglednik nije stigao definirati prilagođeni element, pa dolazi do toga. Postoje slučajevi u kojima će se prvo svi elementi definirati u pregledniku a zatim izvršiti (eng. rendering) dok kod nekih to bude obrnuto te se na prilagođenim elementima neće izvršiti postavljanje stila. To se može spriječiti upotrebom pseudo selektora :unresolved koji će izvršiti postavljanje stilova za sve one elemente koji se nisu stigli izvršiti. Jedan od načina sprječavanja je postavljanje tom elementu vidljivost na 0% , te ga učiniti „nevidljivim“ oku.

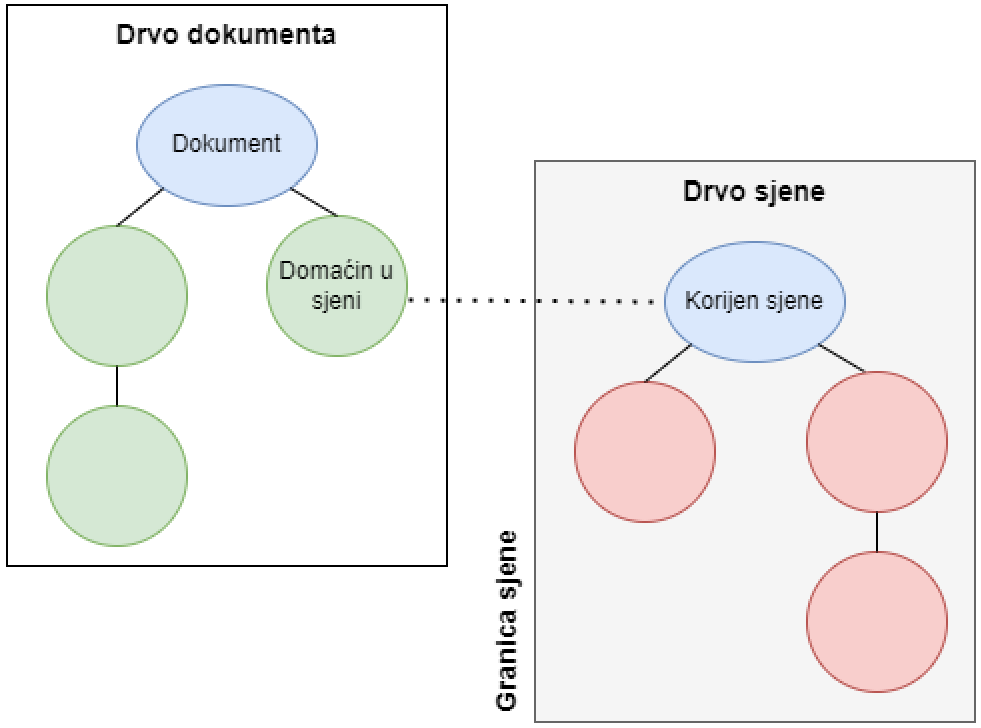
4.2 Dom u sjeni

Jedan od važnijih aspekata web komponenti je enkapsulacija. Enkapsulacija je grupiranje podataka i metoda pomoću kojih se manipulira tim podatcima ili se ograniči izravan pristup nekim komponentama tog objekta. [14]

Skup JavaScript API-ja za pričvršćivanje enkapsuliranog „sjenčanog" DOM stabla na elementu, koji se iscrtava odvojeno od DOM-a glavnog dokumenta. Na taj način se značajke elementa mogu zadržati privatnima, tako da se mogu skriptirati i stilizirati bez straha od sudara s drugim dijelovima dokumenta. [12]

DOM u sjeni (eng. Shadow DOM) ima dvije ključne razliku u odnosu na običan DOM:

- Kako se kreira i koristi
- Kako se ponaša u odnosu na ostatak stranice [13]



Slika 1 Prikaz interakcije DOM-a i DOM-a u sjeni (vlastiti izvor)

Kreiraju se DOM čvorovi i dodaju se kao podređeni ostalim elementima. To se naziva svijetlo stablo, odnosno obično pod-stablo DOM-a. Složeno je pomoću HTML elemenata.

Stablo u sjeni je skriveno pod-stablo DOM-a. Stablom u sjeni kreiramo obično DOM pod-stablo koje je spojeno na određeni element dokumenta, ali je odvojeno i skriveno od svojih podređenih. Element na koji se spaja je njegov domaćin u sjeni. Sve što se doda u sjeni postaje lokalno za element na koji je spojen, te se tako postiže enkapsuliranje css-a.

4.2.1 Korištenje DOM-a u sjeni

```
const h1 = document.createElement(„header‟);
const root_u_sjeni = h1.attachShadow({ mode: „open‟ });
root_u_sjeni.innerHTML = „<h1> hello world! „;
```

Ovaj kod prikazuje kako se kreira DOM u sjeni. Korijen u sjeni je dio koji se spaja na element domaćin. Dodavanje korijena u sjeni je način na koji element dobiva svoj DOM u sjeni. Da bi se programski kod dodao DOM u sjeni koristi se funkcija `element.attachShadow()`. Na trećoj linija koda se samo popuni korijen. Moguće je koristiti i ostale funkcije, ne nužno `element.innerHTML`.

Može se kreirati i DOM u sjeni za vlastiti element. Kreira se novi vlastiti HTML tag i definiramo njegovo ponašanje pomoću JavaScripta.

```
vlastiti_element.define(„vlastiti_tag", class extends
HTMLElement {
constructor() {
super();
const korijen_u_sjeni = this.attachShadow({mode: „open"});
korijen_u_sjeni.innerHTML = „
...
<style> ... </style>
...
<div> ... </div>
...
„;
}
});
```

Programski kod napisan iznad prikazuje HTML element koji ima svoj vlastiti DOM u sjeni. Funkcijom `element.define()` se kreira klasa za novi HTML tag. Na sličan način kao i u prvom primjeru kreira se DOM u sjeni pomoću funkcije `element.attachShadow()`. U ovom primjeru prikazano je da se može dodati i stil unaprijed definiran za takav HTML tag. Na takav način osigurava se uvijek isti stil kada se koristiti taj HTML tag i štiti ga od nekakvih kasnijih promjena. Isto tako se mogu dodati zaštićene funkcionalnosti pomoću `<script>` elementa koji će se stalno pokretati pri korištenju istog. Zatvaranje korijena u sjeni postižemo postavljajući „mode:" u „close" unutar funkcije `element.attachShadow()`.

4.3 Predlošci

Osnovni dio skoro svih okruženja za razvoj programske podrške za web je koncept predložaka. Predlošci služe za razvoj dinamičnih dijelova i pomaže nam reducirati duplicirani kod. [14]

4.3.1 HTML predlošci

HTML5 s ugrađenim elementom `<template>` omogućava kreiranje vlastitih predložaka koji je moguće kasnije klonirati bez korištenja kopiraj i zalijepi metode. Osim što se unutar `<template>` elementa može pisati HTML sadržaj, podržava i stilove i skripte. U početku se neće primijeniti napisani stil i skripte, sve dok se template ne postavi negdje u dokument, jer `<template>` sadržaj se smatra da je „izvan dokumenta". [13]

Elementi `<template>` i `<slot>` omogućuju pisanje predložaka oznaka koji se ne prikazuju na prikazanoj stranici. Oni se tada mogu ponovno koristiti više puta kao osnova za strukturu prilagođenog elementa. [12]

4.3.2 Pokretanje predloška

Predlošci se nalaze u posebnom čvoru DOM-a. Konkretan predložak se ne ubacuje direktno, već kreira „djecu“ tog predloška i njima rukujemo, a to se postiže pomoću atributa `.content`. [13]

```
<>
<template id="predlozak">
<div>Hello, world!</div>
</template>
<script>
let test = document.createElement(„div“);
test.append(predlozak.content.cloneNode(true));
document.body.append(test);
</script>
</>
```

Programski kod napisan iznad prikazuje kako se predložak programski ubaci u dokument. U ovom slučaju kreiran je element `<div>`. Na njega je funkcijom dodan `element.append()` dijete predloška. Kloniranjem čvora predloška kreirano je dijete, te je početni element dodan na dokument. Tek nakon zadnje linije, odnosno nakon dodavanja početnog elementa na dokument se pokreću sve skripte i primjenjuju se stilovi napisani unutar predloška.

4.3.3 Predlošci i DOM u sjeni

Glavna svrha predložaka je rukovanje DOM-ovima u sjeni. Takav način rada je u redu sve dok se radi o manjim dijelovima koda ili ako se takav kod ne koristi na više različitih mjesta. U suprotnom je bolje umjesto takvih znakovnih nizova koristiti HTML5 predloške. [13]

4.3.4 Ugniježđeni predlošci

Ugniježđeni predlošci su predlošci unutar predložaka.

```
<template id="vanjski_predlozak">
<h1>Hello world!</h1>
<template id="unutarnji_predlozak">
<p>Ovaj hello world je teže aktivirati</p>
</template>
</template>
```

Ovdje je prikazano kako intuitivno napisati ugniježđene predloške. Problem kod aktivacije takvih predložaka je laka aktivacija vanjskog predloška, dok onaj unutarnji ostaje neaktivan. Oba predloška se moraju aktivirati posebno. Način na koji se aktivira ugniježđeni predložak je malo kompliciraniji. Ideja je ukloniti unutarnji predložak, odnosno djecu vanjskog predloška. Zatim se klonira čvor djeteta i dodaje vanjskom predlošku(sada aktiviranom) klonirani čvor (također sada aktiviran).

4.4 Komponente i stilovi

Jedno od obilježja DOM-a u sjeni je mogućnost izolacije. Prednost korištenja izoliranih stilova je mogućnost korištenja bez poznavanja i ometanja drugih globalnih stilova koji su dodijeljeni nekom elementu dok kao nedostatak može biti upravo to ograničenje upotrjebljivosti globalnih stilova.

4.4.1 Ponašanje stilova

U ovom dijelu rada se govori o ponašanju stilova, pristupanju domaćinu (eng. host), nasljeđivanju stilova, odabir elementa unutar hosta te primjenjivanje određenog stila.

Pomoću host selektora moguće je selektirati sjenu domaćina te pristupiti domaćinu ili elementu koji sadrži stablo u sjeni te koristiti željeni stil. Sve dok je komponenta u sjeni primjenjivat će se stil.

Ako unutar dokumenta postoje stilovi za neki određeni element, upotrebom selektora host je moguće nadjačati lokalne stilove. No postoji iznimka pomoću koje lokalni stil može nadjačat ostale uz upotrebu sintakse „important“ te time lokalni stilova nadjačavaju ostale i prema CSS pravilima ih nije moguće nadjačati. Osim što je moguće primjeniti stil za cijelu komponentnu moguće je i primijeniti samo za određeni dio komponente na način da se unutar selektora host dodaje atribut. Na primjer ako se nalaze neke komponentne koje su centralizirane moguće je samo na njih upotrijebiti drugačiji stil.

```
<template id ="temp1">
<style>
:host([centered]) {
position: fixed;
left: 50%;
top: 50%;
transform: translate(-50%, -50%);
border-color:blue;
}

:hoost {
display: inline-block;
border: 1px solid red,
padding: 10px;
}
</style>
<slot></slot>
```

Element :host sa atributom „centered“ koji se može pozivati za one elemente koji su centrirani te vršiti određene css promjene. Postoji još jedan način selektiranja koji donosi veću kontrolu od ostalih. Radi se o selektoru čijem se domaćinu može pristupiti u ovisnosti u kojem okruženju se nalazi. Da bi se koristilo selektiranje na ovaj način :host mora biti definiran unutar nekog grupirajućeg elementa koji sadrži atribut class. Selektiranje ovog tipa će biti istinito jedino ako se :host nalazi unutar elementa koji sadrži ime klase koje odgovara imenu definirano unutar style.

4.5 Događaji

HTML događaji su nešto što preglednik radi ili što korisnik napravi. Najbolji način za opis evenata u JavaScriptu je kroz primjere, a neki od primjera HTML evenata su:

- HTML web stranica je završila s učitavanjem
- Korisnik je pritisnuo HTML gumb
- Korisnik je upisao nove podatku u HTML formu [13]

Pri pojavi događaja, aplikacija pokreće set instrukcija. Blok koda koji se izvršava na taj način se naziva „event handler“ odnosno rukovatelj događajima. [13]

Rukovatelje događajima možemo dodati na više načina, a njih možemo vidjeti na sljedećoj slici.

Prvi način

```
<button onclick="nekakva_funkcija()">Pokreni funkciju</button>
```

Drugi način

```
const gumb_1 = this.querySelector(„button“);
gumb_1.addEventListener(„click“, e =>
nekakva_funkcija(parametar));
```

Treći način

```
const gumb_2 = this.querySelector(„button“);
gumb_2.onclick = nekakva_funkcija;
```

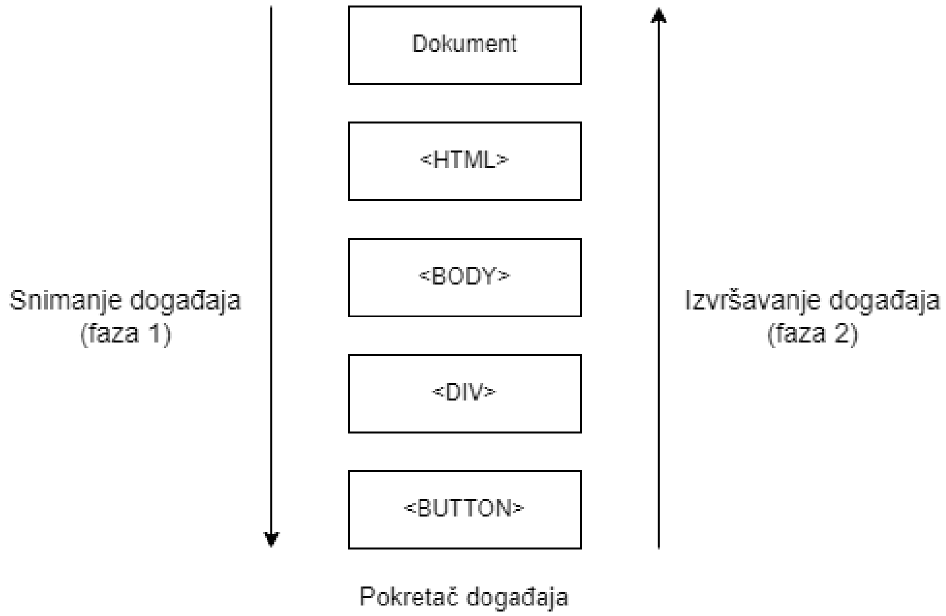
Osim navedenih događaja, dostupni su i ostali događaji. Dio najpopularnijih događaja mogu se vidjeti u tablici 1.

Funkcija	**Opis funkcije**
click	Pokreće se klikom miša
dblclick	Pokreće se dvostrukim klikom miša
drag	Pokreće se povlačenjem elementa
ended	Pokreće se kada je medijski zapis završio
load	Pokreće se učitavanjem dokumenta
pause	Pokreće se pauziranjem medijskog zapisa
scroll	Pokreće se listanjem dokumenta
submit	Pokreće se predajom HTML formi
undo	Pokreće se kada dokument poništi nekakvu radnju

Tablica 1 Vrste događaja

4.5.1 Propagiranje događaja

Važno je znati kako se događaji propagiraju. Na slici 2 vidi se kako bi to zapravo izgledalo. Klikom miša započinje propagacija i ona se izvršava dok ne dođe do svog cilja, u slučaju na slici do dokumenta.



Slika 2 Propagiranje događaja

Događaji se propagiraju unutar spljoštenog DOM-a. Ako je postavljena zastavica mode: „closed“ za stablo u sjeni, neće se vidjeti kako se točno propagira događaj, već će se vidjeti od domaćina u sjeni pa na gore. U slučaju da je zastavica „open“, tada se može vidjeti putanja od najnižeg sloja stabla u sjeni, pa skroz do kraja. [13]

4.5.2 Događaji u web komponentama

Događaji se mogu napisati unutar stabla u sjeni. Ako se pokrene događaj koji se nalazi unutar stabla u sjeni, taj interni rukovatelj događaja će prepoznati da je cilj bio unutar DOM-a u sjeni. Ukoliko se pokrene događaj izvan stabla u sjeni, dokumentni rukovatelj događaja će misliti da je cilj bio domaćin u sjeni. Što je dobra stvar jer vanjski dokument ne treba znati kako izgleda zapravo ta komponenta niti kako funkcionira. [13]

4.6 Utori

Rezervirana mjesta mogu se ispuniti vlastitim oznakama. Ta rezervirana mjesta se nazivaju utori. Utori (eng. slots) su automatski popunjeni sadržajem iz regularnog DOM-a. [13]

4.6.1 Uporaba utora

```
<slotovi>
<span slot="prvi_slot">Hello</span>
<span slot="drugi_slot">World</span>
<span slot="treci_slot">!</span>
</slotovi>
<script>
customElements.define(„slotovi‟, class extends HTMLElement
{connectedCallback()} {this.attachShadow({mode: „open‟})};
this.ShadowRoot.innerHTML =
`
<div>
<slot name="prvi_slot"></slot>
</div>
<div>
<slot name="drugi_slot"></slot>
</div>
<div>
<slot name="treći_slot"></slot>
</div>
`;
);
</script>
```

Ovo je jednostavan primjer korištenja utora. Preglednik uzima elemente iz DOM-a na svijetlu i pomoću njih puni utore DOM-a u sjeni. Za svaki `<slot name=…>` unutar DOM-a u sjeni preglednik traži istoimene utore unutar DOM-a na svijetlu i tako iscrtava podatke unutar utora. Spljošteni DOM postoji samo za iscrtavanje i zbog rukovanja događajima.

4.7 Korištenje povratnih poziva životnog ciklusa

Unutar samog prilagođenog elemenat moguće je definirati razne povratne pozive koji će se izvršavati u nekom trenutku. U ovom dijelu će se opisati povratni pozivi za:, `disconnectedCallback`, `attributeChangedCallback`, `observedAttributes`, `connectedCallBack` i `adoptedCallback`.

Funkcija `connectedCallBack` se poziva svaki put kada se prilagođeni element dodaje u element koji je povezan s dokumentom. Pozivat će se svaki put prilikom premještanja čvora. Može se dogoditi i prije nego što se cijeli sadržaj elementa u potpunosti prenese. Mana je u tome što se može pozvati čak i kad element više nije povezan sa dokumentom te se kod ovog poziva koristi funkcija `Node.isConnected` za provjeru. `DisconnectedCallback` se poziva svaku put kada prilagođeni element odvoji od DOM dokumenta. `AdoptedCallback` se poziva svaku put prilikom premještanja elementa u novi dokument. Kod bilo kakve promjene od atributa unutar prilagođenog elementa poziva se `attributeChangedCallback`. Tu promjenu mu dojavi `observedAttributes` koji prati svaki promjenu i javlja ukoliko dođe do nje.

Primjer korištenja attributeChangeCallback

```
atributeChangeCallback(name, oldValue, newValue) {
updateStyle(this);
}
```

Primjer korištenja observedAttrinutes funkcije

```
static get observedAttributes() { return ['c','l']; }
```

Moguće je promijeniti bilo koje svojstvo individualno, koristeći ime tog svojstva. No u ovom slučaju promjena će se vršiti preko funkcije `updateStyle` koja će promijeniti stil za taj element. Kao što je već spomenuto poziv će se izvršiti jedino ukoliko se ta promjena dojavi preko `observedAttributes`() koja vraća niz koji sadrži imena atributa nad kojim se vrši promjena.

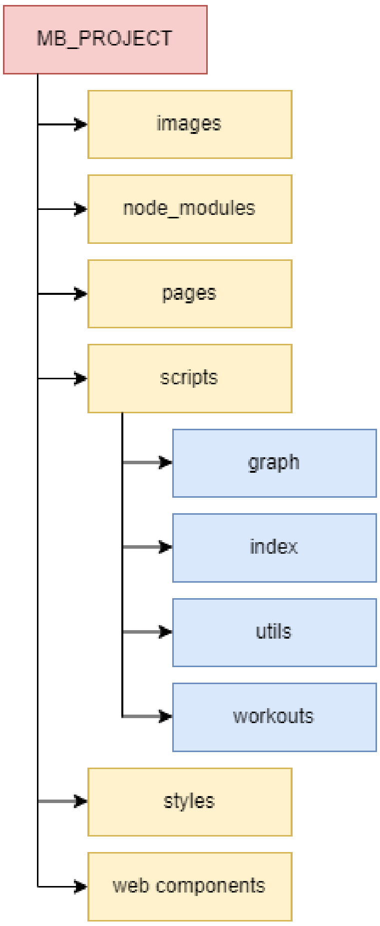
5. Praktičan dio

5.1 Ideja web aplikacije

Cilj web aplikacije je olakšati korisniku praćenje svojih dnevnih aktivnosti, poput hodanja i vježbanja. Korisnik za svaki dan može unijeti odrađene vježbe, serije i ponavljanja, te može dodati opis vježbe. Na taj način korisnik može pratiti svoj napredak kroz vrijeme. Svi podaci koji su uneseni unutar aplikacije spremaju se u bazu podataka izrađene pomoću mongoDB-a.

5.2 Struktura direktorija web aplikacije

Uvijek je bitna dobra organizacija direktorija, radi lakšeg snalaženja te radi budućih promjena na projektu. Glavni direktorij, odnosno polazišna točka projekta naziva se „MB_PROJECT“.



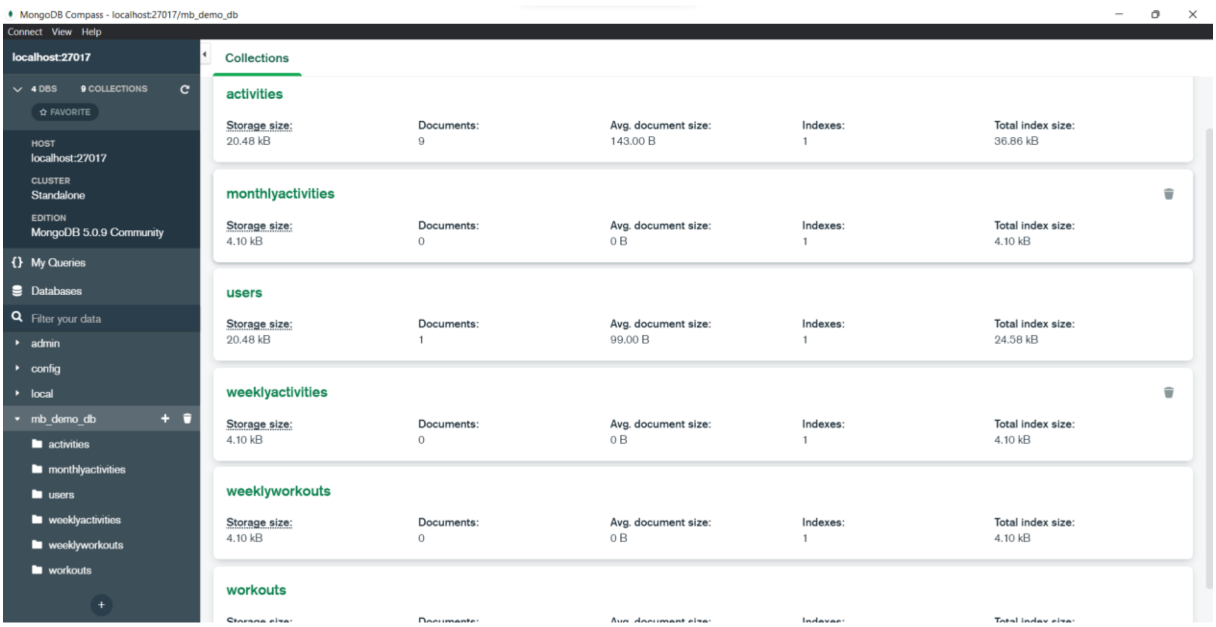
Slika 3 Struktura direktorija

Na slici 3 prikazana je struktura direktorija ove web aplikacije. Datoteke su razvrstane prema svojim ekstenzijama, sav HTML kod smješten je unutar „pages“ direktorija, JavaScript je smješten unutar „scripts“ i „web components“ zavisno o tome radili se o funkcionalnostima ili komponentama, te je CSS smješten unutar „style“ direktorija. Unutar „images“ nalaze se slike.

5.3 Baza podataka MongoDB

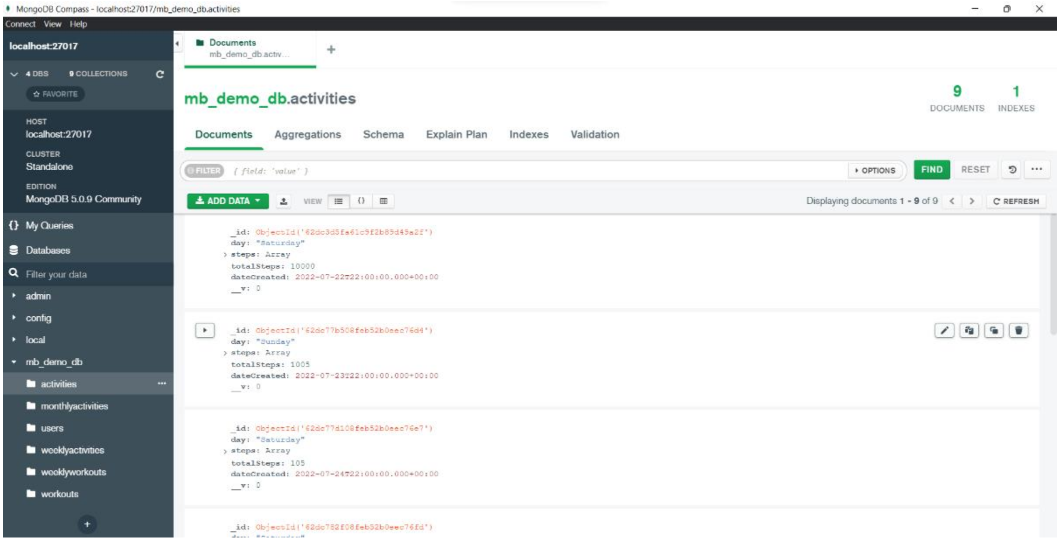
MongoDB je NoSQL program za upravljanje bazama podataka. MongoDB je baza podataka otvorenog koda (eng. Open source), te je moguće prilagođavati ga potrebama projekta. NoSQL se koristi kao alternativa tradicionalnim relacijskim bazama podataka. NoSQL baze podataka vrlo su korisne za rad s velikim skupovima distribuiranih podataka. Laički rečeno MongoDB je alat koji može upravljati informacijama o dokumentima, pohranjivati ili dohvaćati informacije.

Za manipulaciju s bazom podataka i njezinim lokalnim podacima korištena je aplikacija MongoDB Compass. Aplikacija ima mogućnost lokalnog spajanja baze podataka.



Slika 4 MongoDB Compass

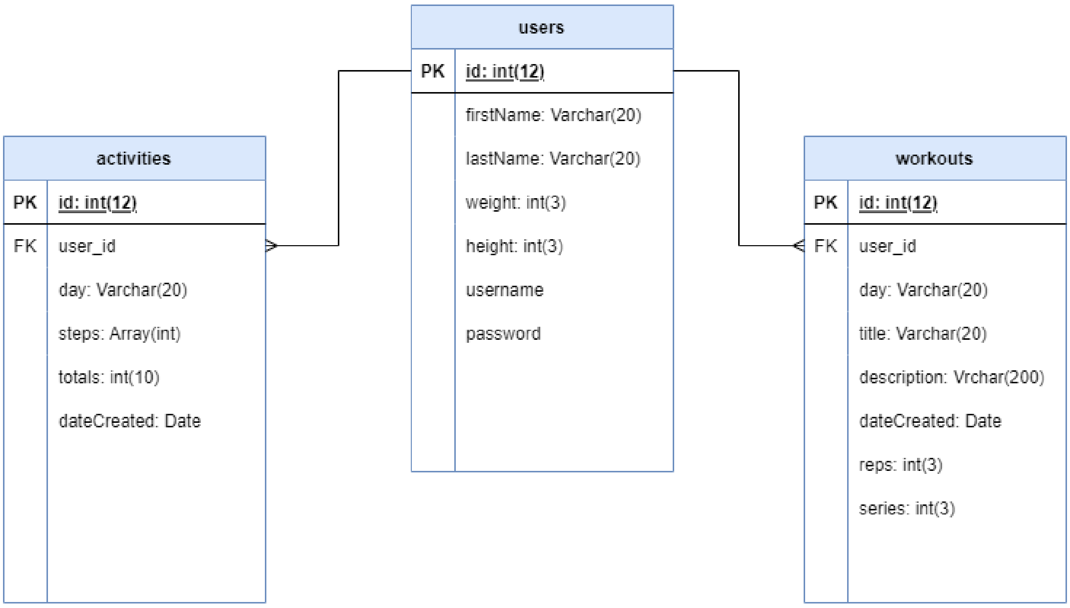
Na slici 4 se nalaze tablice u koje se spremaju lokalni podaci te su one poslužile za prikazivanje podataka na web stranici prilikom izrade praktičnog dijela.



Slika 5 Prikaz podataka preko MongoDB Compass aplikacije

Za podatke koji nisu pohranjeni lokalno na računalu, koristi se oblak (eng. cloud) MongoDB koji preko AWS-a (Amazon Web Server) pruža uslugu pohranjivanja servera.

5.4 Struktura baze podataka



Slika 6 Struktura baze podataka

Slika 6 prikazuje ERA dijagram baze podataka ovog projekta. Na dijagramu se nalaze 3 tablice. Tablica „activities“ i „workouts“ sadrže vanjski ključ koji je ujedno i primarni ključ u tablici „users“. Na taj način vježbe i aktivnost su povezane sa korisnikom. Svaka vježba ili aktivnost u sebi sadrži jedinstveni ključ korisnika, te se po tome dohvaćaju podatci za svakog korisnika. Baza je pojednostavljena i napravljena isključivo za svrhu projekta te je prema tome rađen i ovaj ERA dijagram. Svaka tablica ima svoj primarni ključ (eng. primary key) koji je id. Odnos tablice „users“ i ostalih je 1:M što znači da jedan korisnik može imati više vježbi ili aktivnost i svaka vježba ili aktivnost može imati samo jednog korisnika.

Podaci koji se nalaze u tablici su sljedeći:

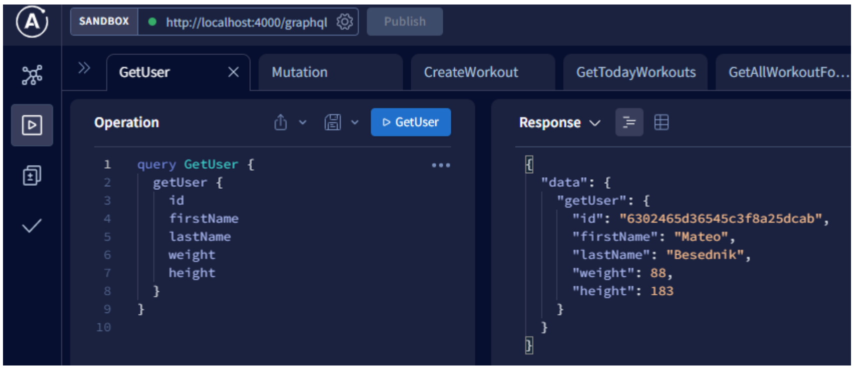
- Tablica „workouts“ sadrži podatke o korisnik id, danu, naslovu vježbe, opisu, datumu, odrađenim ponavljanima i serijama.
- Tablica „acrivities“ sadrži podatke o korisnik id, danu u kojem se hodalo, broju prijeđenih koraka, zbroju svih koraka, te datumu.
- Tablica „users“ sadrži podatke o korisniku, odnosno ime, prezime, njegova visina i težina.

5.5 GraphQL

Za upravljanje lokalnim i udaljenim podacima koristi se GraphQL. To je biblioteka koja omogućuje izradu zahtjeva koji povlače podatke iz više izvora podataka u jednom pozivu aplikacijskog programskog sučelja (eng. Application Programming Interface). GraphQL je poznata alternativa REST-u, te se u današnje vrijeme koristi u većini IT poduzeća. To je ujedno i razlog korištenja GraphQL-a u izradi ove aplikacije. [15]

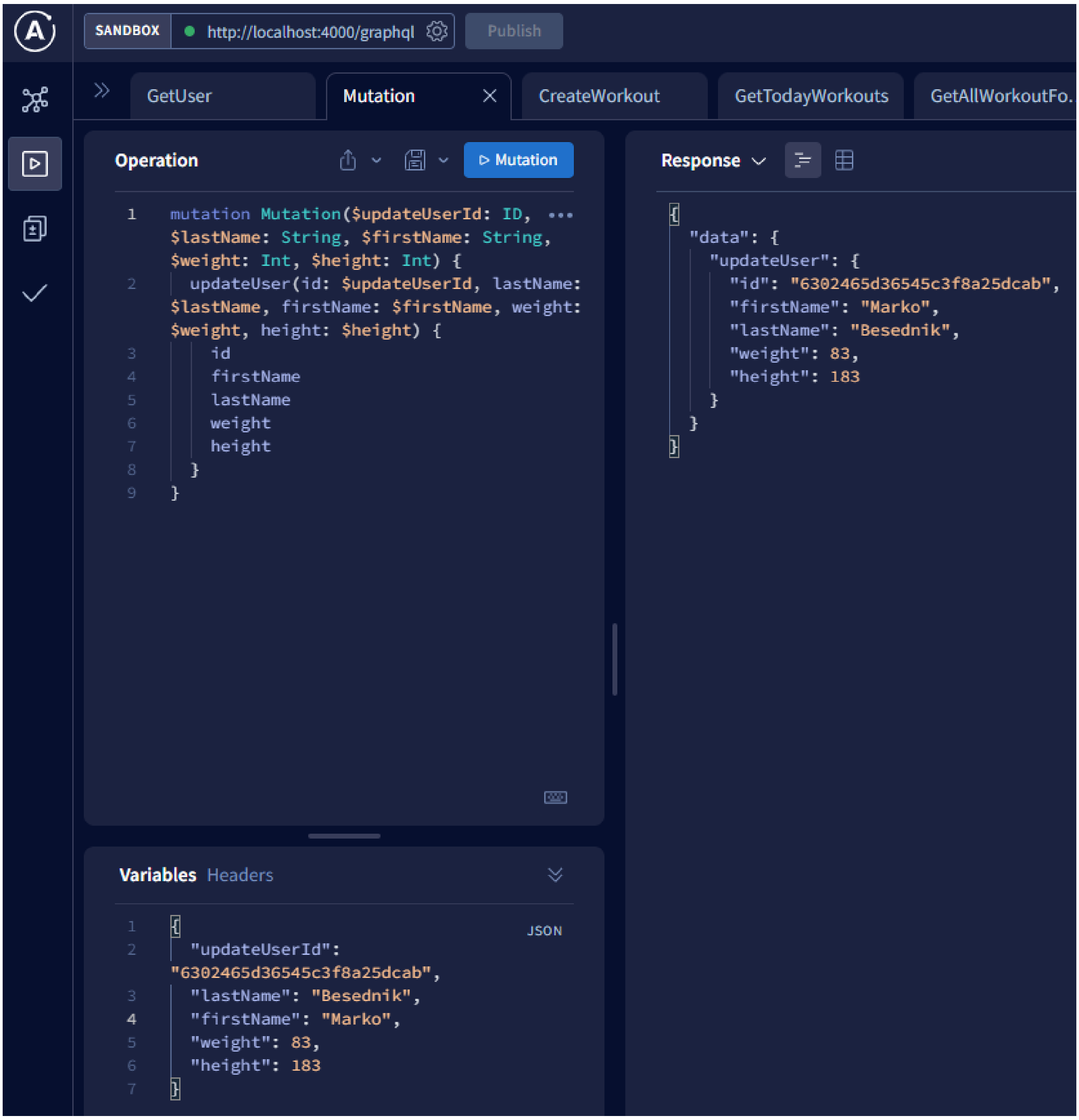
GraphQL kao i REST koristi metode poput dohvaćanja (eng. GET) i slanja (eng. POST) podataka, samo što se za dohvaćanje koriste upiti (eng. query), a za slanje se koristi mutacija (eng. mutation). Za provjeru ispravnosti API-a koristio se Apollo server.

Apollo Server je GraphQL poslužitelj otvorenog koda, koji je kompatibilan s bilo kojim GraphQL klijentom, uključujući Apollo klijent.



Slika 7 Apollo server primjer upita

Na slici 7 nalazi se primjer upita za dohvaćanje korisnika, koji se trenutno nalazi prijavljen u web aplikaciji. Na lijevoj strani prozora upisuje se upit, dok se na desnoj nalazi odgovor (eng. response) toga upita.



Slika 8 Apollo server primjer mutacije

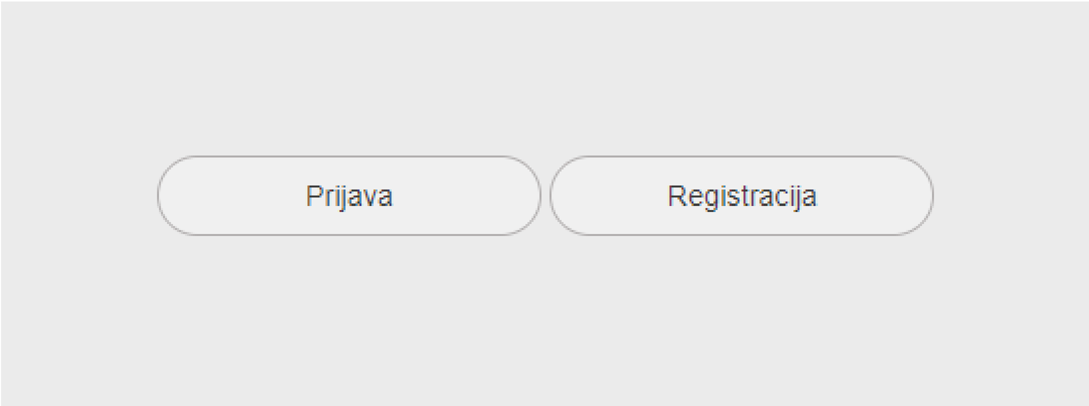
Slika 8 prikazuje primjer mutacije za promjenu podataka korisnika, koji se trenutno nalazi prijavljen u web aplikaciji. Na lijevoj strani prozora upisuje se mutacija, ispod toga nalazi se prozor za varijable s kojima se mijenjaju podaci, dok se na desnoj nalazi odgovor te mutacije nakon njezinog izvršenja.

Slika 9 prikazuje kako promjena podataka na Apollo serveru utječe na samu web aplikaciju, ovdje inače piše „Mateo“ a sada se promijenilo u „Marko“. Dakle ukoliko se za varijablu „firstName“ postavi iz „Mateo“ u „Marko“ unutar Apollo servera, pozvat će se mutacija koja ažurira korisnika i tu promjenu upisuje u bazu podataka te Apollo vraća novo ažurirano stanje. Ta promijena se vizualno može uočiti na web aplikaciji ili na Apollo serveru kao povratna informacija.

Slika 9 Mutacija na stranici

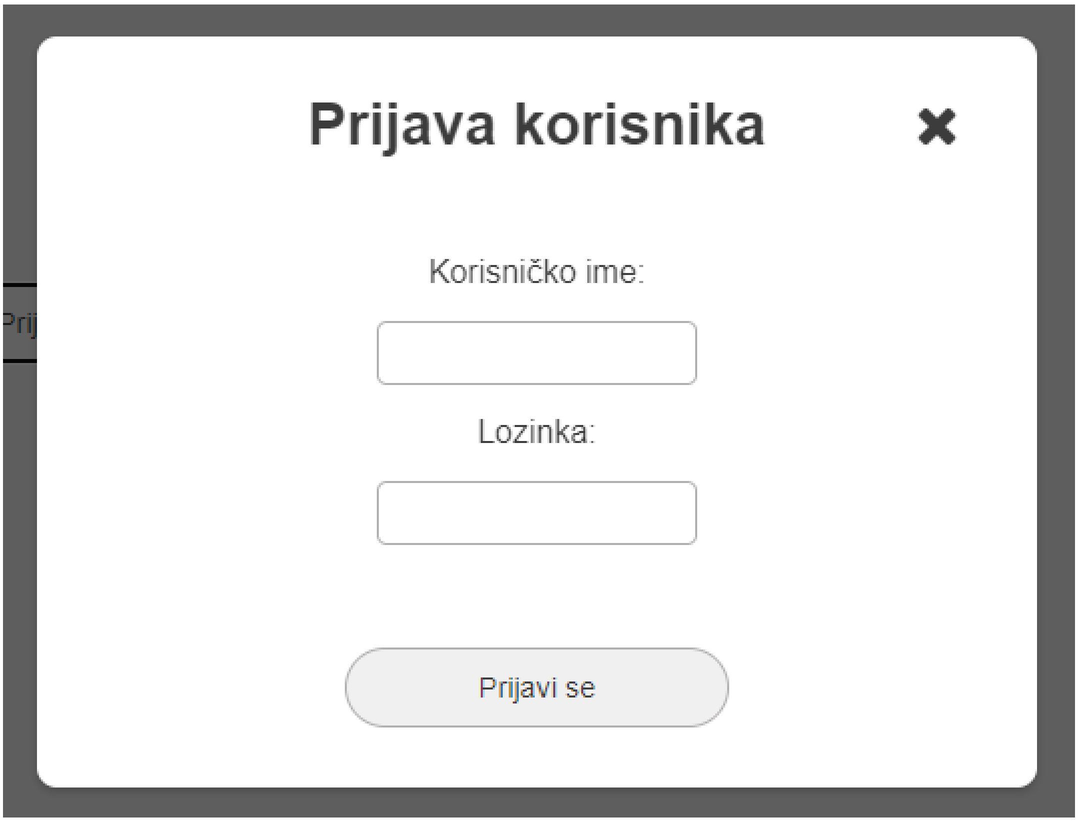
5.6 Web aplikacija

Prilikom pokretanja web aplikacije, korisnik se treba prijaviti ili registrirati ukoliko se već nije registrirao. Na slici ispod, prikazuju se dva gumba, koja ovisno o odabiru otvaraju modale za prijavu/registraciju.



Slika 10 Gumbi za prijavu i registraciju

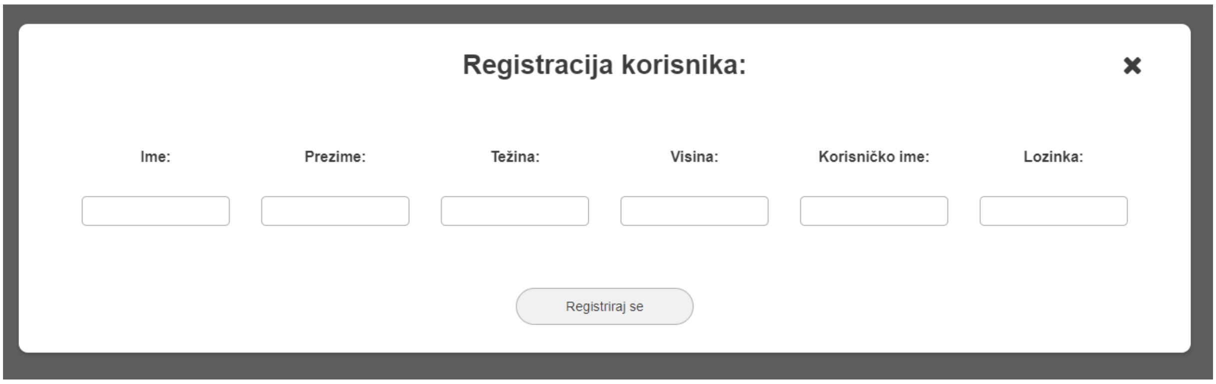
Ukoliko je korisnik stisnuo gumb prijava otvara mu se prozorčić unutar kojeg može upisati korisničko ime i lozinku s kojom se prethodno registrirao. To je prikazano na slici ispod, te pritiskom na gumb „Prijavi se“, poziva se funkcija „loginUser()“ koja provjerava da li se uneseno korisničko ime i lozinka nalaze u bazi podataka. Ako se nalazi onda se u kolačić sprema korisnikov jedinstveni identifikator, te korisnika preusmjerava na početnu stranicu web aplikacije (index.html), inače se prikazuje poruka za krivi unos korisničkog imena ili lozinke. Programski kod funkcije se nalazi ispod slike.



Slika 11 Obrazac za prijavu

```
async function loginUser(username, password) {
JsLoadingOverlay.show();
const result = await loginUserQuery(username, password);
if (result != null) {
Cookies.set('uuid', result.id);
location.assign('index.html');
} else {
alert('Krivo korisničko ime ili lozinku');
}
JsLoadingOverlay.hide();
```

Ukoliko je korisnik stisnuo gumb „Registracija“ otvara mu se prozorčić unutar kojeg može upisati svoje podatke potrebne za registraciju. To je prikazano na slici ispod, te pritiskom na gumb „Registriraj se“, poziva se funkcija „registerUser()“ koja provjerava da li su uneseni svi potrebni podaci. Ako su uneseni onda se poziva funkcija za unos novog korisnika u bazu podataka te vraća podatke o korisniku zajedno sa jedinstvenim identifikatorom korisnika koji se sprema u kolačić te kasnije koristi za dohvaćanje i manipulaciju podataka za tog korisnika. Nakon toga se korisnika preusmjerava na početnu stranicu web aplikacije (index.html). Ako nisu uneseni svi potrebni podaci onda se prikaže poruka za unos svih potrebnih podataka. Programski kod funkcije se nalazi ispod slike.



Slika 12 Obrazac za registraciju korisnika

```
async function registerUser() {
JsLoadingOverlay.show();
const result = await createNewUser(user);
if (result != null) {
Cookies.set('uuid', result.id);
location.assign('index.html');
} else {
alert('Greška ');
}
JsLoadingOverlay.hide();
}
```

Nakon prijave/registracije otvara se početna stranica koja sadrži tri bloka i navigaciju koja omogućuje kretanje po stranici.

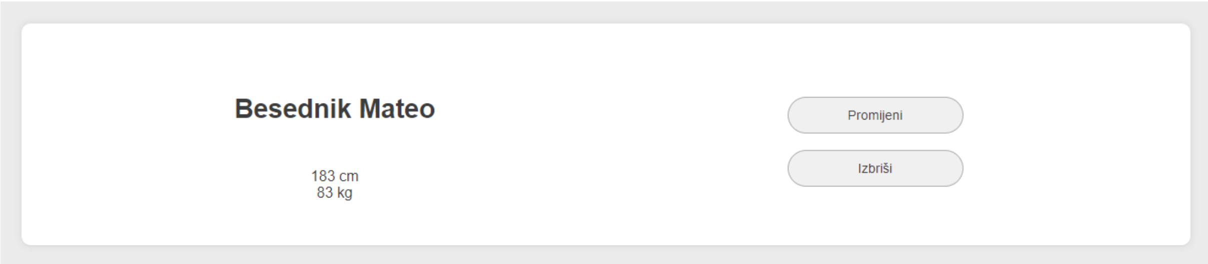
Slika 13 Početna stranica

Na početnoj stranici nalazi se više web komponenti, jedna od tih komponenti je navigacijska traka, ona se prikazuje na sve tri stranice ovog projekta. Navigacija se kreira jednom i zatim se poziva na svim stranicama. Na taj način smanjuje se broj linija koda, te je poboljšana preglednost koda.

```
class NavMenu extends HTMLElement {
constructor() {
super();
this.attachShadow({ mode: „open“ });
}
connectedCallback() {
this.render();
}
render() {
const { shadowRoot } = this;
const templateNode = document.getElementById(„nav-template“);
shadowRoot.innerHTML = „“;
if (templateNav) {
const instance = document.importNode(templateNav.content,
true);
instance.querySelector(„.nav-column“).innerHTML = `
<div>
<a href="index.html"><i class="fa fa-home" aria-
hidden="true"></i></a>
</div>
<div>
<a href="vježbe.html"><i class="fa fa-child" aria-
hidden="true"></i></a>
</div>
<div>
<a href="graf.html"
><i class="fa fa-bar-chart" aria-hidden="true"></i
></a>
</div>
`;
shadowRoot.appendChild(instance);
} else {
console.log(„Došlo je do greške“);
}
}
}
customElements.define(„nav-menu“, NavMenu);
```

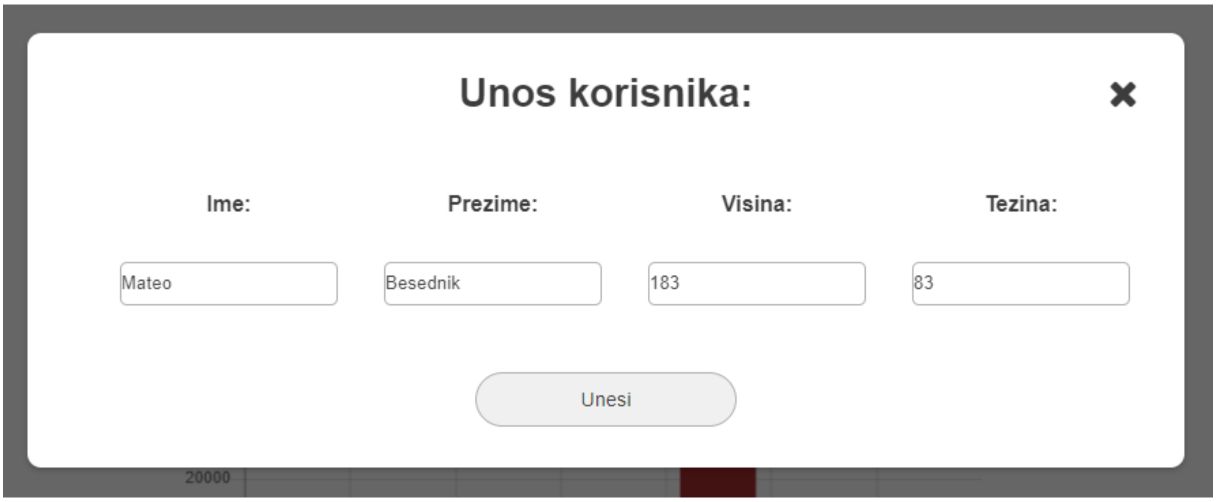
Svaka web komponenta kreira se unutar klase. Navigaciji se dodaje vlastiti DOM odnosno DOM u sjeni, pomoću „`attachShadow`“, na taj način komponenta ima vlastiti DOM i ne utječe na druge komponente na stranici. Sve što komponenta sadrži spremljeno je u varijablu „instance“. Pomoću funkcije `appendChild()` u shadowRoot ubacujemo varijablu instance u kojoj se nalazi korisničko sučelje te se na taj način prikazuje komponenta i korisnik može vidjeti sadržaj na stranici, bez funkcije `appendChild()` na stranici se ne bi ništa prikazalo. Na kraju dokumenta potrebno je deklarirati ime komponente koja se zatim poziva u HTML-u, u ovom slučaju „nav-menu“. Prilagođeni element kreira se pomoću `customElements.define`, tu je potrebno deklarirati ime komponente i klasu u kojoj je zapisana ta komponenta.

Unutar prvog bloka nalaze se podaci o korisniku, odnosno prezime i ime, visina te kilaža korisnika. Ime i prezime korisnika napisano je unutar komponente „card-title“, razlog iz kojeg je i to zapisano kao komponenta je ponavljanje naslova, odnosno sva tri bloka imaju naslov tako da se na taj način opet smanjuju linije koda. U ovom bloku se nalaze dvije komponente, „card-title“ i „user-info“, u drugu komponentu stavljeni su podaci o visini i težini korisnika.



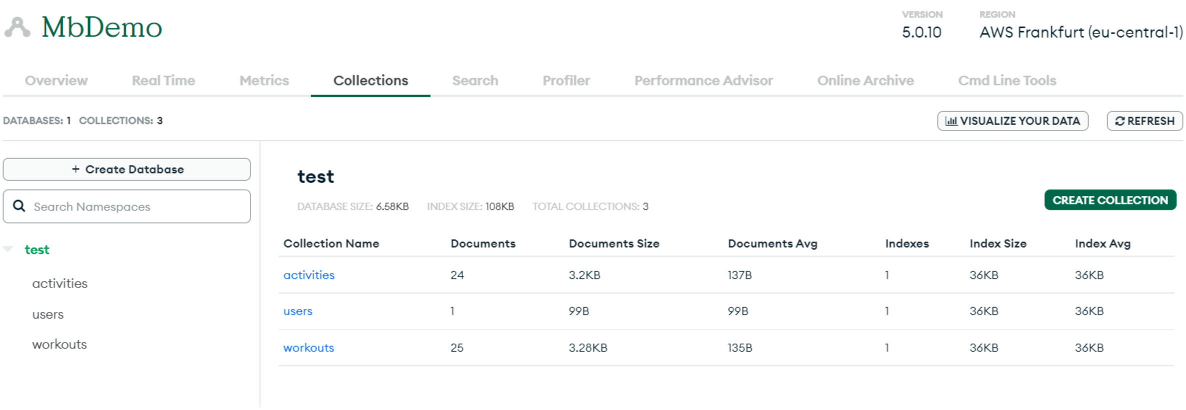
Slika 14 Blok s podacima o korisniku

Klikom na gumb „Promijeni“ otvara se prozorčić unutar kojeg se mogu promijeniti ti podaci, sve što se tu unese sprema se direktno u bazu podataka i prikazuje se na početnoj stranici u bloku sa korisnikovim podacima.



Slika 15 Prozor za promijenu podataka o korisniku

Uneseni podaci se u bazi podataka ažuriraju za korisnika koji sadrži jedinstveni identifikator koji je pročitan iz kolačića u entitet pod imenom „users“, što se može vidjeti lijevo na slici, dok se desno nalaze atributi tog entiteta.



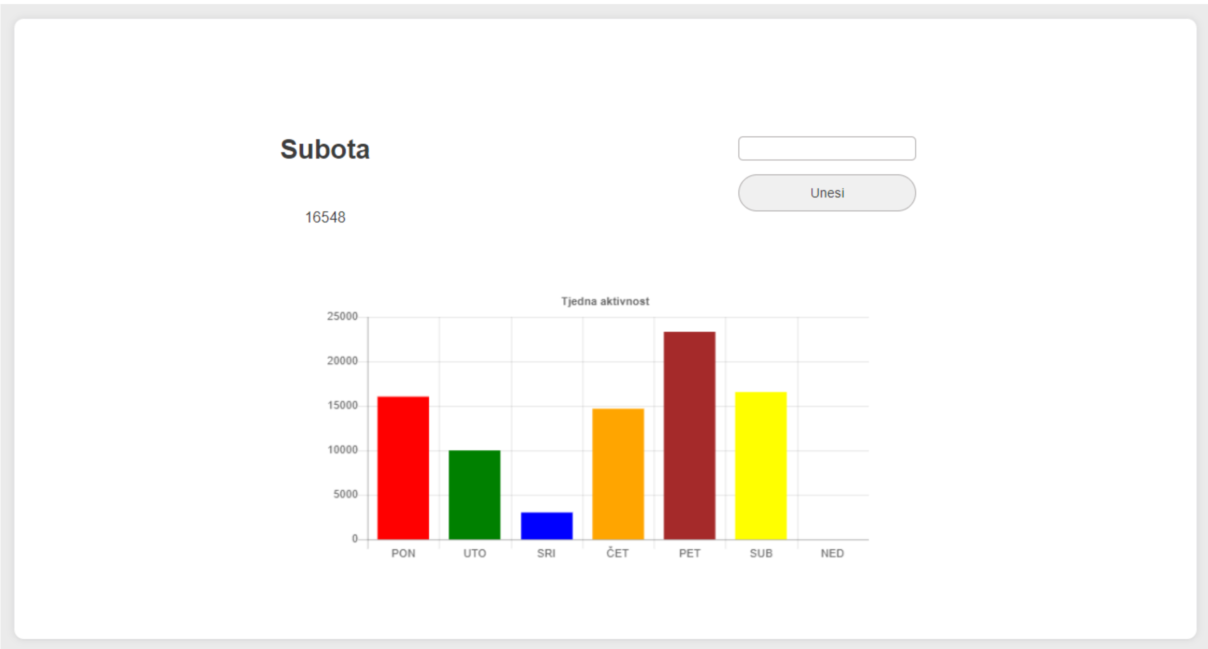
Slika 16 Cloud MongoDB

Drugi blok sadrži graf sa koracima za prijavljenog korisnika. Unutar bloka nalazi se dijagram sa koracima unesenim po danima, tu je potrebno za svaki dan upisati broj prijeđenih koraka i klikom na gumb „Unesi“ se unose koraci u bazu podataka. Nakon što je korisnik pritisnuo gumb, graf se automatski ažurira, te se broj unesenih koraka zbroji na već uneseni broj koraka.

Komponente koje sadrži ovaj blok su naslov koji je isto zapisan u komponenti „card-title“, te komponenta „graph-info-weekly“, koja se također prikazuje na stranici s grafovima u svakom bloku, čija se implementacija nalazi u narednom programskom kodu.

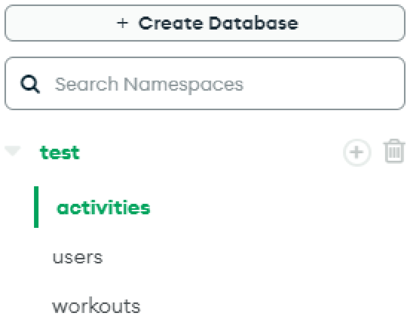
```
class GraphInfoWeekly extends HTMLElement {
get yValues() {
return this._yValues;
}
set yValues(values) {
this._yValues = values;
}
constructor() {
super();
this._yValues = „„;
this.attachShadow({ mode: „open“ });
}
connectedCallback() {
this.render();
}
render() {
const { shadowRoot } = this;
const instance =
document.importNode(templateGraphWeekly.content, true);
setClickListeners(instance);
const ctx = instance.querySelector(„#myChart“);
shadowRoot.appendChild(instance);
createChart(ctx, this.yValues.values);
}

}

function setClickListeners(instance) {}
customElements.define(„graph-info-weekly“, GraphInfoWeekly);
```

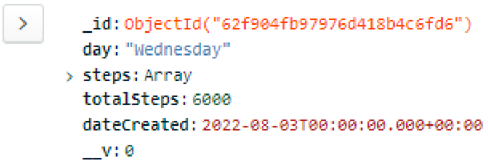
Komponeta je zapisana u klasi „GraphInfoWeekly“, te se u HTML-u poziva preko imena „graph-info-weekly“.



Slika 17 Blok s koracima na početnoj

Svi uneseni podaci, odnosno koraci spremaju se u bazu podataka u entitet „activities“.

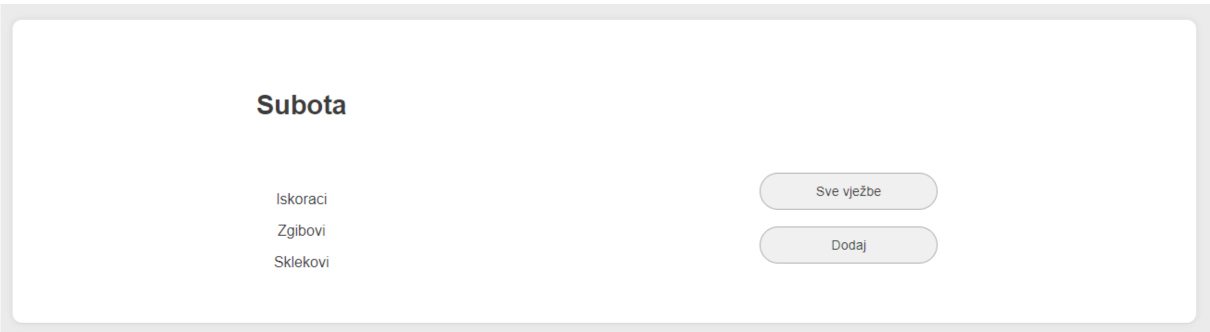




Slika 19 Atributi entiteta „activities“

Slika 18 Entitet "activities" u bazi podataka

Treći i ujedno posljednji blok na početnoj stranici je blok sa vježbama koje su odrađene u trenutnom danu. Na lijevoj strani bloka ispisuje se trenutni dan i nazivi odrađenih vježbi, dok se na desnoj strani nalaze gumb za unos odrađenih vježbi te gumb koji vodi na stranicu gdje se nalaze svi dani u kojima se vježbalo, sa serijama, ponavljanima te opisom za svaku vježbu.

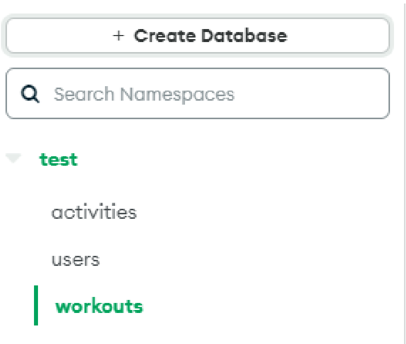


Slika 20 Blok s vježbama na početnoj

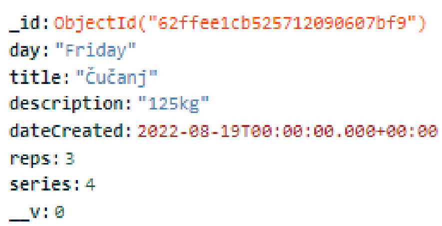
Klikom na gumb „Dodaj“ pojavljuje se prozorčić unutar kojeg je moguće unijeti naziv vježbe, odrađene serije i ponavljanja, te opis vježbe. Klikom na gumb unesi se poziva funkcija za unos nove vježbe pomoću koje se nova vježba, zajedno sa korisničkim identifikatorom zapisuje u bazu podataka.

Slika 21 Prozor za unos vježbi

Podaci koji su uneseni unutar trećeg bloka pohranjuju se u entitet pod nazivom „workouts“.



Slika 22 Entitet "workouts"



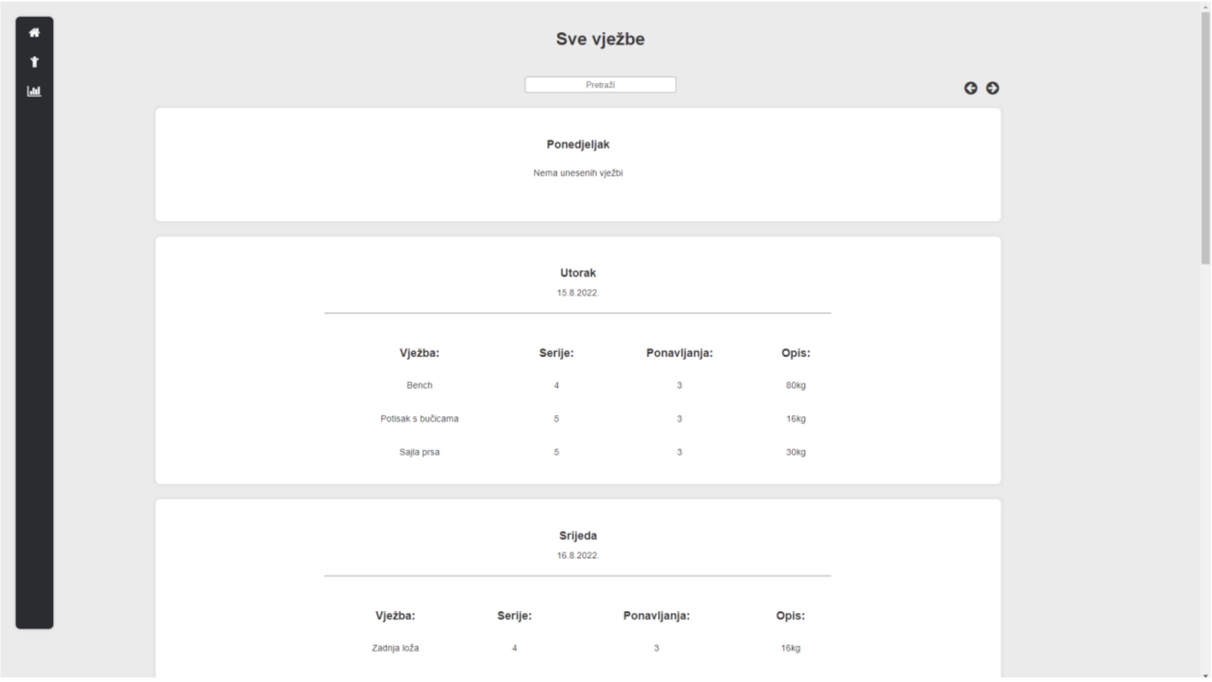
Slika 23 Atributi entiteta "workouts"

5.6.1 Stranica sa vježbama

Druga stranica sadrži sadrži vježbe, serije, ponavljanja te opis vježbe po tjednu koji su uneseni na početnoj stranici. Na stranicu sa vježbama korisnik može doći putem navigacije ili klikom na gumb „sve vježbe“ koji se nalazi u trećem bloku početne stranice. Prva komponenta na stranici je „big-title“ koja prikazuje naslov stranice.

```
class BigTitle extends HTMLElement {
constructor() {
super();
this.attachShadow({ mode: "open" });
}
connectedCallback() {
this.render();
}
render() {
const { shadowRoot } = this;
this.shadowRoot.innerHTML = `
<style>
.blok-title {
display: inline-block;
font-size: 2.3rem;
padding: 2.1rem 0 2.1rem 0;
width: 30rem;
}
</style>
<h1 class="blok-title">${this.getAttribute("title")}</h1>
`;


}
}
customElements.define("big-title", BigTitle);
```

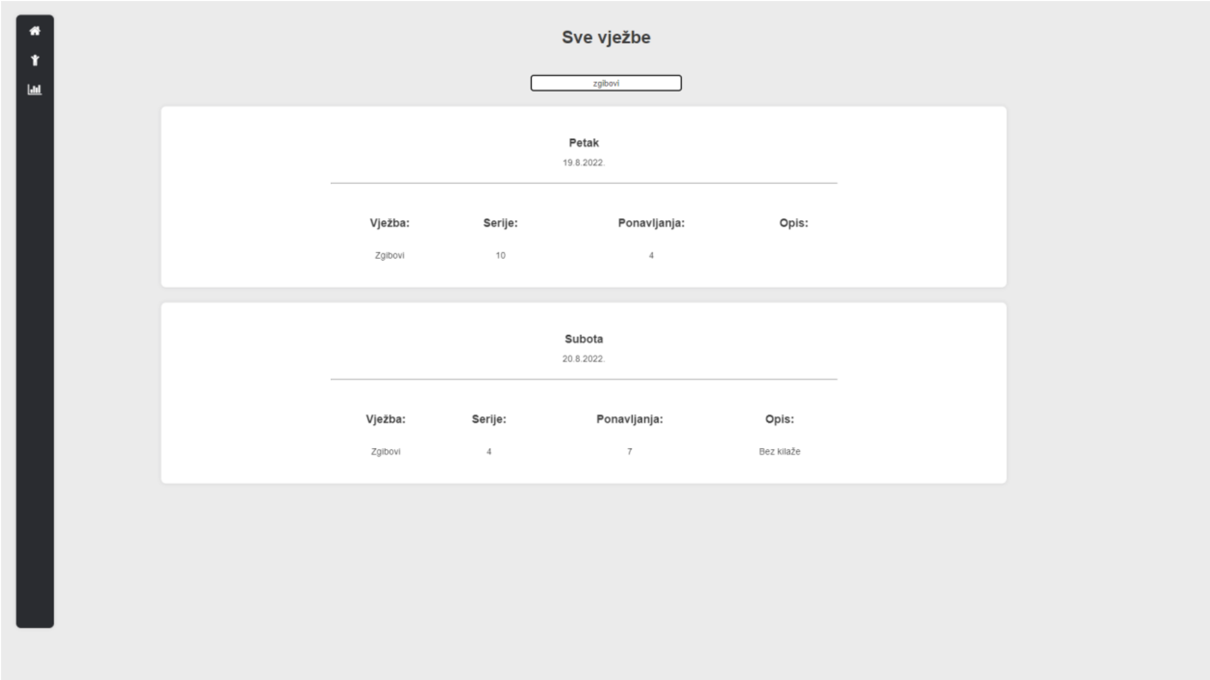
Programski kod komponente „big-title“ prikazana je iznad. Druga komponenta je navigacija, dok je treća komponenta je blok sa vježbama pod nazivom „workout-weekly-info“. Stranica sa vježbama sadrži sedam blokova, svaki blok pretstavlja jedan dan u tjednu.



Slika 24 Stranica sa svim vježbama

Unutar bloka korisnik može vidjeti datum i dan u tjednu u kojem je vježba, te naravno vježbe, serije, ponavljanja te opis odrađenih vježbi. Ukoliko korisnik želi vidjeti vježbe odrađene u prošlim tjednima, može kliknuti na gumb sa strelicom čime se prikažu vježbe odrađene u prošlom tjednu. U slučaju da korisnik nije trenirao u nekom danu u tjednu, u ovom slučaju to bi bio ponedjeljak, unutar bloka se ispisuje ime dana te poruka „Nema unesenih vježbi“.

Korisnik na ovoj stranici može pretraživati odrađene vježbe, kako bi si uštedio vrijeme i olakšao traženje odrađenih vježbi, dakle unutar inputa se unese naziv vježbe i pritisne „enter“, te se nakon toga poziva funkcija koja u bazi podataka traži ime vježbi ili opis vježbe za traženi unos, te ako postoji onda se prikažu odrađene vježbe.



Slika 25 Funkcionalnost pretraživanja

```
searchEnter.addEventListener("submit", async (e) => {
e.preventDefault();
JsLoadingOverlay.show();
const getWorkoutBySearchInput = await
getWorkoutBySearch(searchInput.value);
if (getWorkoutBySearchInput != null)
renderSearch(getWorkoutBySearchInput);
JsLoadingOverlay.hide();
});
```

Poziva se funkcija „getWorkoutBySearch“ koja za parametar poprima vrijednosti koji je korisnik unio u input.

```
function getWorkoutBySearch(searchInput) {
if (searchInput != "") {
blokIcon.classList.add("hide-blok");
blokIcon.classList.remove("show-blok");
return queryFetch(GET_WORKOUT_BY_SEARCH_QUERY, {
searchInput: searchInput,
}).then((res) => {
return res.data;
});
} else {
blokIcon.classList.remove("hide-blok");
blokIcon.classList.add("show-blok");
getWeeklyWorkouts(number);
}
}
```

Ova funkcija provjerava da li u unosu nešto postoji, te ako postoji onda iz baze traži rezultat na temelju unesene riječi u input i vraća rezultat ukoliko postoji. Ako je input prazan, onda se prikaže tjedan koji je korisnik gledao prije pretraživanja.

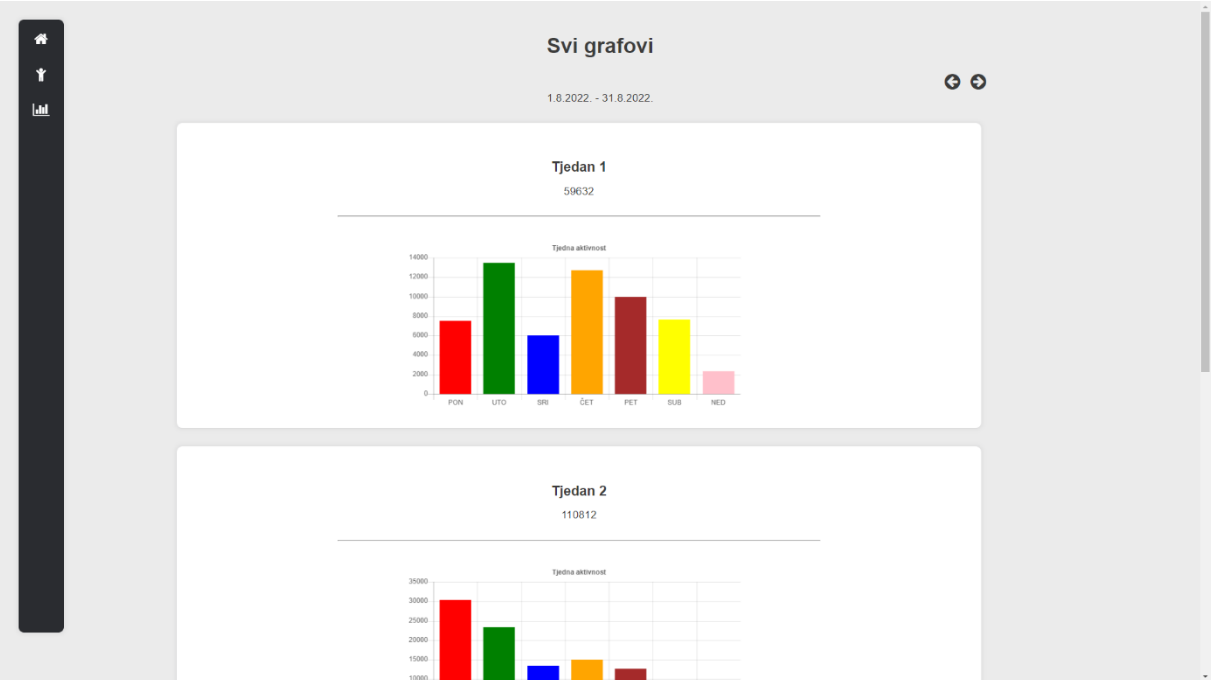
U slučaju da korisnik unese naziv vježbe koja nije odrađena ili nešto drugo ne vezano za stranicu, na stranici se ispisuje poruka „Nema rezultata!“. Programski kod koji to omogućuje napisan je ispod slike.

Slika 26 Slučaj da nema rezultata pretraživanja

```
document.getElementById("blok-proba").innerHTML =
noWorkoutView(NOT_FOUND);
```

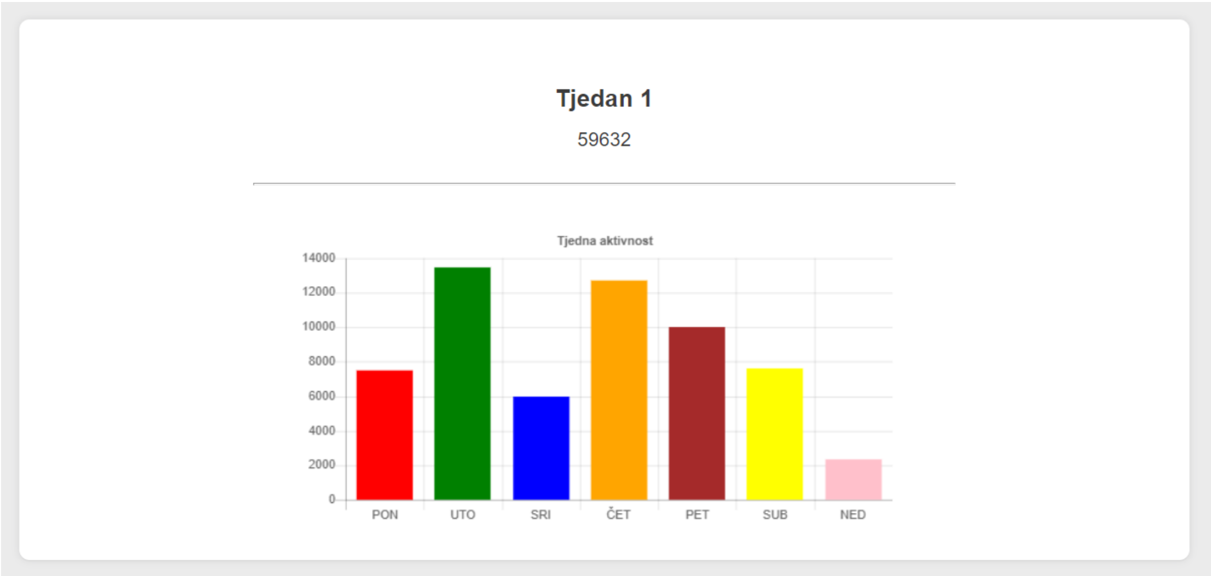
5.6.2 Stranica s koracima i grafovima

Na treću i ujedno posljednju stranicu dolazi se klikom na ikonu grafa koja se nalazi u navigaciji. Ova stranica sadrži blokove koja predstavlja tjedne. Unutar svakog bloka može se vidjeti ukupan broj koraka odhodanih u tjednu.



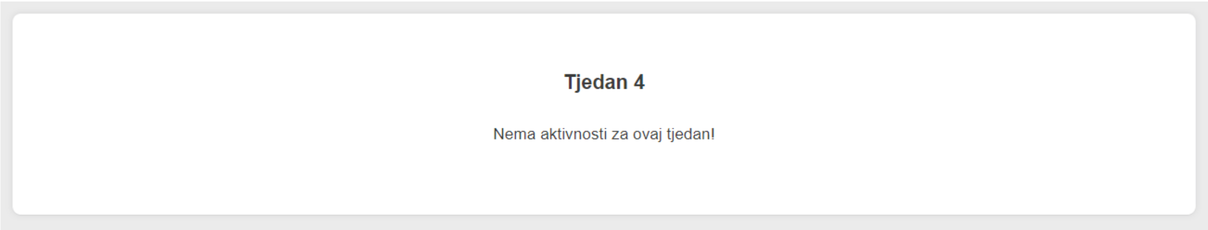
Slika 26 Stranica s grafovima i koracima

„Tjedan 1“ predstavlja prvi tjedan u mjesecu, na vrhu stranice ispod naslova može se vidjeti o kojem se mjesecu radi, u ovom slučaju to je kolovoz koji ima četiri tjedna te se zato trenutno na stranici prikazuju četiri bloka.



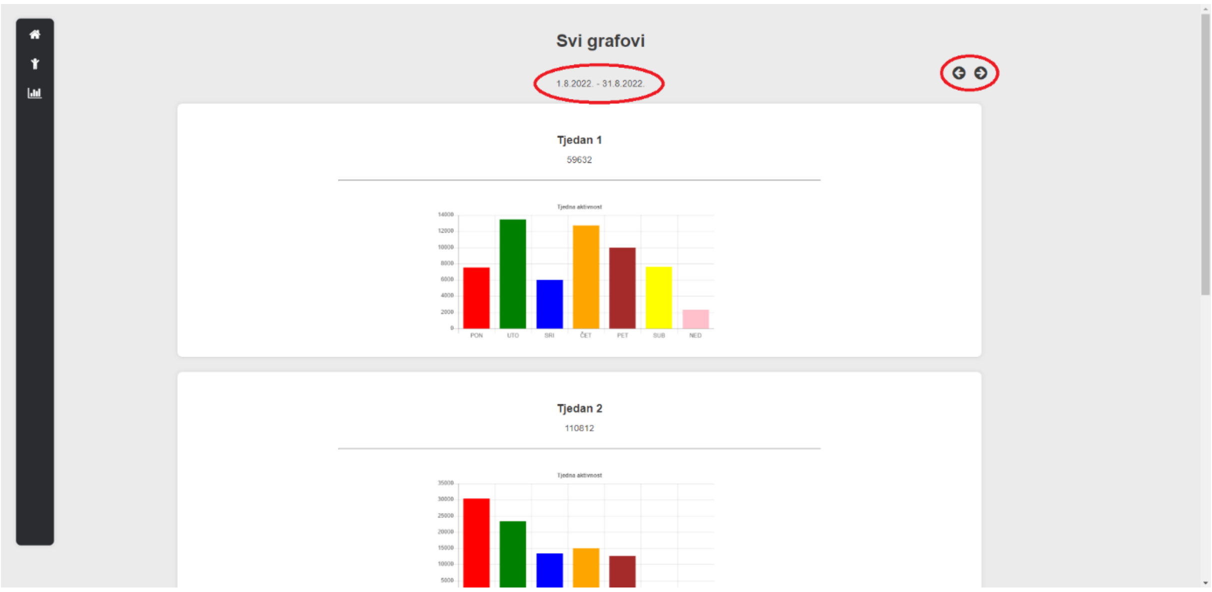
Slika 27 Blok sa svim koracima u jednom tjednu

Ukoliko tjedan još nije započeo ili korisnik iz nekog razloga nije unio korake ni za jedan dan u tjednu, unutar bloka se ispiše poruka „Nema aktivnosti za ovaj tjedan“.



Slika 28 Tjedan u kojem nisu uneseni koraci

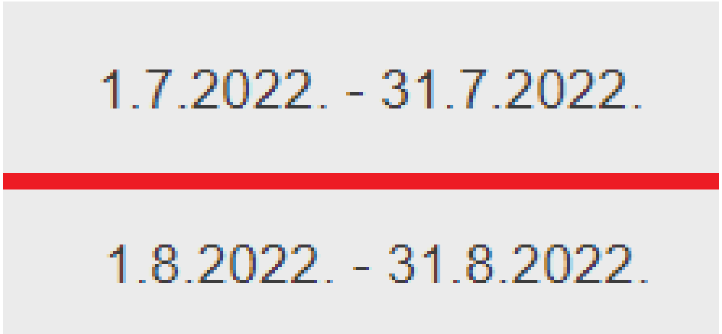
Klikom na ikonu strelice lijevo ili desno, korisnik može vidjeti podatke vezane uz odhodane korake u ostalim mjesecima. Podaci koji su učitani su spremljeni unutar niza, te zapravo klikom na lijevo/desno, korisnik mijenja indeks-e unutar niza.



Slika 29 Prikaz koraka za prethodni mjesec

```
async function getWeeklyActivities(date = new Date()) {
JsLoadingOverlay.show();
const a = await queryFetch(GET_MONTHLY_ACTIVITIES_QUERY, {
date: date.toString(),
});
const result = a.data.getMonthlyActivities;
console.log(result);
if (result.length == 0) {
return (blokGraf.innerHTML = noActivitiesView(EROR));
}
let weekNumber = 0;
id = [];
result.forEach((element) => {
const graphsContainer = document.createElement("graph-info-
monthly");
graphsContainer.element = { element: element, weekNumber };
weekNumber++;
blokGraf.appendChild(graphsContainer);
});
JsLoadingOverlay.hide();
}
```

Funkcija „getWeeklyActivities()" predstavlja funkcionalnost koja služi za prikaz koraka i grafova.



Slika 30 Trenutni i prethodni mjesec sa grafovima

Ispod naslova stranice automatski se mijenja datum mjeseca te na taj način korisnik vidi za koji mjesec gleda podatke, slika 30 to prikazuje.

```
let d = new Date(date);
const firstDayOfMonth = new Date(d.getFullYear(), d.getMonth(),
1);
const lastDayOfMonth = new Date(d.getFullYear(), d.getMonth() +
1, 0);
dateRange.innerHTML = `${createDateString(
firstDayOfMonth
)} - ${createDateString(lastDayOfMonth)}`;
blokGraf.innerHTML = ``;
```

Varijable „firstDayOfMonth“ i „lastDayOfMonth“ predstavljaju prvi i zadnji datum u mjesecu, dok „dateRange“ predstavlja znakovni niz koji korisniku prikazuje za koji mjesec trenutno gleda podatke o koracima.

6. Zaključak

Cilj ovog rada bio je definiranje web komponenti te prikaz načina korištenja istih. Veliki projekti zahtijevaju više linija koda, te korištenje komponenti smanjuju te linije koda, jer kad se jednom kreira komponenta, ona se može pozvati na više različitih stranica bez dodavanja nepotrebnih linija koda, što je prikazano u praktičnom dijelu završnog rada. Osim smanjenja linija koda, komponente omogućuju programerima da naprave uredan i pregledan projekt, te da se s vremenom mogu lakše snalaziti u njemu, kada će trebati napraviti neke preinake na njemu. Razvoj web komponenti bitan je za daljnji razvoj web tehnologija poput React-a, Angular-a te Vue-a.

Najveća prednost korištenja web komponenti je što omogućuje višekratnu upotrebu istih web komponenti na različitim stranicama. Na taj način se promjena dizajna vrši na samo jednom mjestu, a ne na svakoj stranici posebno, te se isto tako kod zapisuje samo na jednom mjestu unutar projekta te se poziva na različitim dijelovima projekta preko imena komponente. Još jedna prednost web komponenti je DOM u sjeni, on omogućuje izolaciju komponente od globalnih stilova web stranice. Dakle komponenta je neovisna o drugim stilovima, te gdje god se stavila unutar projekta neće se kršiti odnosno miješati s drugim stilovima projekta.

Web komponente imaju neka ograničenja, prilagođeni CSS pseudo selektori ne mogu se koristiti s web komponentama, osim toga ne rade besprijekorno s izvornim elementima i njima povezanim API-ima. Trenutačno web komponente imaju relativno slabu podršku za više preglednika.

7. Literatura

[1] *From History of Web Application Development | Devsaran*. Accessed: Sep. 14, 2021. [Online]. Available: https://www.devsaran.com/blog/history-web-application-development

[2] "9 Web Technologies Every Web Developer Must Know in 2021 | TMS." https://tms-outsource.com/blog/posts/web-technologies/ (accessed Sep. 14, 2021).

[3] Gorana Kurtović, "HTML." https://www.srce.unizg.hr/files/srce/docs/edu/osnovni-tecajevi/c201_polaznik.pdf (accessed Sep. 13, 2021).

[4] D. Kermek, "Izgradnja Web aplikacija," 2016.

[5] "HTML Elements." https://www.w3schools.com/html/html_elements.asp (accessed Sep. 13, 2021).

[6] C. Amery, *Design and build*, no. 5189. 2002. doi: 10.4324/9780203994979-11.

[7] Edin Mujadžević, "Uvod u CSS." https://www.srce.unizg.hr/files/srce/docs/edu/osnovni-tecajevi/c220_polaznik.pdf (accessed Sep. 12, 2021).

[8] M. Haverbeke, *Eloquent JavaScript*, vol. 24, no. 7. 2007. doi: 10.1190/1.9781560801597.index.

[9] "Static vs Dynamic Website: What Is the Difference?", Accessed: Sep. 14, 2021. [Online]. Available: https://wpamelia.com/static-vs-dynamic-website/

[10] "JavaScript Where To." https://www.w3schools.com/js/js_whereto.asp (accessed Sep. 14, 2021).

[11] "Introduction - webcomponents.org." https://www.webcomponents.org/introduction (accessed Sep. 14, 2021).

[12] "Web Components | MDN." https://developer.mozilla.org/en-US/docs/Web/Web_Components (accessed Sep. 14, 2021).

[13] J. Overson, *Developing Web Components*. 2015.

[14] J. Krause, "Web Components The future of modern Web Apps".

[15] Vieth David, "There's an Alternative to Rest API: GraphQL - Shockoe," Aug. 02, 2019. https://shockoe.com/ideas/graphql-an-alternative-to-rest/ (accessed Sep. 12, 2022).

8. Popis slika

9. Popis tablica