TEHNIČKI FAKULTET-RIJEKA
Preddiplomski studij računarstva

Izvješće o izvedenoj stručnoj praksi s dnevnikom prakse

LogIn d.o.o.

Mateo Acinger
0069089579

Rijeka, Srpanj 2022.

SADRŽAJ:

1.UVOD 3

1.1 O LOGIN D.O.O. 3

1.2 MOJE ZADUŽENJE 3

2.ZADATAK: WEB STRANICA ZA JADRANKA TRGOVINE 4

2.1 FUNKCIONALNOSTI I IZGLED APLIKACIJE 4

2.1.1 NAČIN RADA STRANICE 4

2.1.2 POKRETANJE STRANICE I SERVERA 6

2.2 HTML KOD STRANICE 7

2.2.1 OPIS INDEX.HTML KODA 8

2.2.2 OPIS O_NAMA.HTML KODA 10

2.3 CSS KOD STRANICE 11

2.4 OPIS I IZGLED JAVASCRIPT KODA 13

3. ZAKLJUČAK 17

LITERATURA 18

DODATAK:DNEVNIK PRAKSE 19

1.UVOD

1.1 O Login d.o.o.

Login je tvrtka koja je bavi izradom poslovnih aplikacija tipa ERP(Enterprise resource plannning).Uz to rade na razvijanju mobilnih i web aplikacija. Također, bave se i cloud tehnologijom. Danas tvrtka Login d.o.o. zajedno sa svojom sestrinskom tvrtkom Login sustavi d.o.o. zapošljava više od 30 IT stručnjaka stvarajući stabilne temelje za rast i zapošljavanje novih kvalitetnih kadrova. Jedan od najvećih proizvoda tvrtke Login je Virga. Virga se koristi u nekoliko trgovačkih lanaca za sustav kasa i izdavanja računa. Uz sve navedeno tu su još i konzultantske usluge ,projektno upravljanje, vizualizacija podataka i slično. Rezultat ERP rješenja je pozitivna poslovna transformacija tvrtki kroz rasterećenje radnih mjesta od suvišnog operativnog rada u cilju povećanja profitabilnosti i rasta poslovnih prihoda.

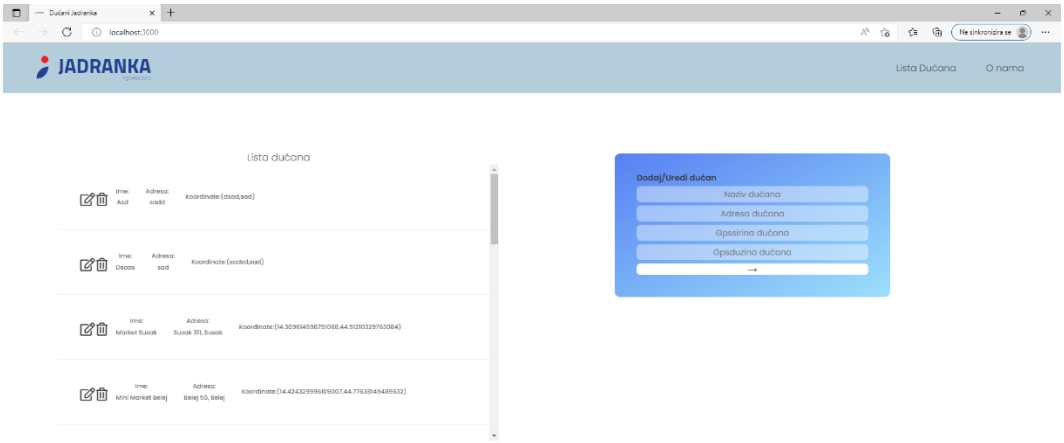
1.2 Moje zaduženje

Kada sam prvi put došao u tvrtku dodijeljen mi je mentor te mi je postavljeno pitanje kojim se područjem želim baviti. Odlučio sam se za područje kreiranja web stranica i web aplikacija te sam krenuo sam slušanjem JavaScript tečaja. JavaScript je programski jezik koji služi za postavljanje ponašanja web stranice i za kreiranje web stranica. Osim toga JavaScript se danas koristi u mnogim drugim stvarima i jedan je on najpopularnijih programskih jezika zbog svoje relativno jednostavne sintakse. U tečaju sam naučio sintaksu JavaScripta te sam naučio logički razmišljati i stvarati rješenja za određene probleme na području web aplikacija. U svemu tome mi je pomogao mentor koji mi je razjasnio stvari i neke probleme koji su se javljali tijekom prolaženja tečaja. Kada sam završio sam tečajem dobio sam zadatak da smislim i kreiram web aplikaciju za trgovine Jadranka. Pošto sam prošao samo JavaScript tečaj prije nego što sam počeo stvarati stranicu morao sam naučiti i osnove HTML-a i css-a.HTML je jezik u kojim se stvaraju objekti i sve što se vidi na web stranici. Ti objekti sami po sebi ništa ne znače jer nemaju određeno ponašanje. Ponašanje definiramo JavaScriptom, a stil i izgled tih objekata definira se u CSS-u. Na početku mi je dodijeljen papir sa specifikacija i uputama što stranica mora sadržavat i što mora raditi. Sve ostalo uključujući kreiranje ,raspored elemenata, dizajn i kako će stranica raditi to što mora raditi bilo je na meni da osmislim i implementiram.

2.ZADATAK: WEB STRANICA ZA JADRANKA TRGOVINE

2.1 Funkcionalnosti i izgled aplikacije

Ideja cijele aplikacije je da daje osnovne informacije o Jadranka trgovinama. Na način da postoji O nama sekcija. Uz to postoji i sekcija Lista dućana u kojoj su sve glavne funkcionalnosti. Kada se uđe na web stranicu pomoću URL-a pozicionirani smo u sekciju Lista dućana. Na lijevo strani stranice nalazi se lista koja prikazuje sve dućane iz baze ,te nam omogućuje mogućnost brisanja i uređivanja dućana pomoću za to predviđenih ikonica. Uz sve to daje nam vidni prikaz svih bitnih informacija o nekom dućanu. Na desnoj strani stranice nalaze se polja koja koristimo kada u bazu dodajemo novi dućan te isto tako u tim poljima možemo urediti već postojeći dućan na način da stisnemo ikonicu za uređivanje. Sve opisano možemo vidjeti na slici ispod.

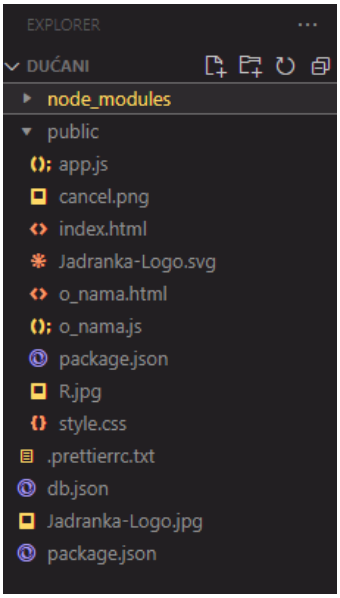


Slika 1: Izgled web stranice

2.1.1 Način rada stranice

Za programiranje kompletne stranice koristili smo Visual Studio Code u kojem smo kreirali mapu Dućani i u njoj sve potrebne datoteke.Za početak smo kreirali 3 datoteke. JavaScript datoteku koja ima ekstenziju .js, HTML datoteku sa .html i CSS datoteku sa .css. Zatim smo otvorili terminal u JavaScript datoteci i instalirali smo npm package manager koji nam je potreban za korištenje API-a. Velike aplikacije i inače projekti imaju front-end dio , to je dio koji se bavi čisto izgledom i funkcionalnosti i back-end koji se bavi podacima. U nekim manjim aplikacijama kao što je ova jako često se koristi REST API umjesto back-end dijela. API je software koji dopušta komunikaciju između više aplikacija. U ovom projektu ja sam koristio REST API koji simulira klijent – server način rada te omogućuje zahtjev prema serveru ,te

odgovor nazad sa određenim podatkom. Nakon uspješno instaliranog npm package managera u terminal pišemo komandu(npm install -g json-server) i pomoću nje instaliramo REST API. U mapi dućani sada su se kreirale svakakve datoteke koje su potrebne za rad REST API-a. U datoteci db.json su nam podaci i informacije o svakom dućanu i sada ih možemo koristiti. U sljedećem podpoglavlju ćemo objasniti kako. Na slici dva možemo vidjeti kako izgleda mapa sa datotekama. U node-modules su datoteke od npm-a koje ne diramo ,a u mapi public su sve prije spomenute datoteke. Na slici 3 vidimo izgled db.json datoteke u kojoj je primjer podataka za dućan.

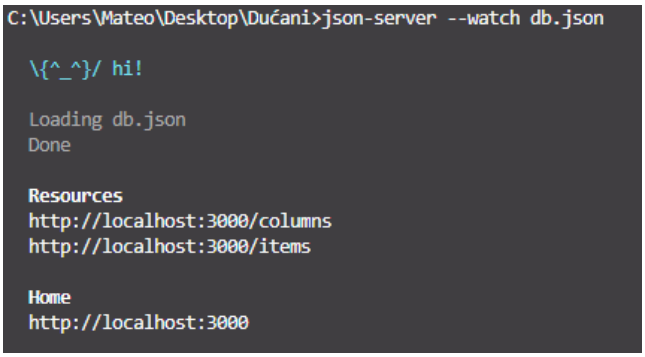


Slika 2: Izgled Dućani mape

Slika 3:db.json datoteka sa podacima

2.1.2 Pokretanje stranice i servera

Za pokretanje stranice i servera u terminal u JavaScript datoteci pišemo naredbu(json-server --watch db.json).Zatim dobijemo nekoliko URL-ova. Na home je naša stranica, a na resources gdje piše items su podaci u JSON formatu koji se otvore u posebnom tabu u browseru i izgledaju isto kao na slici 3.



Slika 4: Pokretanje stranice

Za komunikaciju između klijenta i servera koriste se HTTP metode. Metode koje mi koristimo u našoj aplikaciji su GET,PUT,POST i DELETE. U JavaScriptu imamo funkciju kao nam služi za slanje zahtjeva.

```
//funckija za request
async function customHttp(url = "", method = "GET", data = null) {
  const response = await fetch(url, {
    method: method,
    mode: "cors",
    cache: "no-cache",
    credentials: "same-origin",
    headers: {
      "Content-Type": "application/json",
    },
    redirect: "follow",
    referrerPolicy: "no-referrer",
    body: data ? JSON.stringify(data) : undefined,
  });

  return response.json();
}
```

Slika 5:HTTP funkcija

Ova funkcija nam je asinkrona što znači da kada pozovemo tu funkciju kao odgovor dobijemo promise. Pošto sa funkcijom fetch unutar funkcije customHTTP stvaramo request prema serveru trebat će neko vrijeme dok ne dobijemo odgovor zato je funkcija asinkrona i vraća promise. Promise je nekakav rezervirani prostor u koji dođe odgovor. On može bit ispunjen ili ne ispunjen. Zbog toga kada koristimo asinkronu funkciju obaveznu uz nju koristimo i await koji zatim dostavi podatke(ovisno dali prođe request ili ne) u promise. I zatim nam nakon poziva funkcija customHTTP vraća taj promise s kojim zatim dalje baratamo. Primjećujemo i to da kod return-a koristimo i metodu .json() koja služi za prevođenje response-a u čitljiv format.

2.2 HTML KOD STRANICE

Prije smo spomenuli kako nam HTML kod služi za kreiranje svega što vidimo na stranici uključujući objekte, naslovnu traku, polja za upis podataka i ostalo. U razvoju ove aplikacije koristili smo dvije HTML datoteke. To su indeks.html i o_nama.html. Index datoteka je datoteka u kojoj su svi elementi sekcije Lista Dućana uključujući i naslovnu traku, a u datoteci o_nama je naslovna traka i elementi stranice O nama. Na slici 6 ispod možemo vidjeti izgled sekcije O nama.

Slika 6:O nama sekcija

Bitno je primijetiti da kada smo na sekciji O nama u URL-u stranice dodan je kao argument o_nama.html isto tako kada pritisnemo na sekciju Lista Dućana u URL-u je dodan indeks.html. Kada uđemo u stranicu prvi put toga nema jer se po default-u zna da je indeks.html. Ti argumenti se mijenjaju jer smo povezali u kodu da kada pritisnemo u naslovnoj traci određenu sekciju da se automatski promjeni HTML kod i to vidimo na slici 7. Također vrijedi napomenuti da smo dodali i logo trgovine u lijevom kutu naslovne trake i u tab prostoru stranice dodali smo naziv Dućani Jadranka i malenu ikonicu.

```
    <ul class="nav__links">
      <li class="nav__item">
        <a class="nav__link" href="./index.html">Lista Dućana</a>
      </li>
      <li class="nav__item">
        <a class="nav__link" href="./o_nama.html">O nama</a>
      </li>
    </ul>
  </nav>
```

Slika 7: Povezivanje više HTML datoteka kod naslovne trake

2.2.1 Opis index.html koda

Index.html osnovan je HTML kod ove aplikacije u njemu su elementi naslovne trake, liste za ispis podataka o trgovinama i polja za upis podataka. Naslovna stranica se sastoji od liste koja sadrži dva gumba(Lista Dućana i O nama) i slike. Kod možemo vidjeti iznad na slici 7. Na lijevoj strani stranice nalazi se maleni naslov i lista koja sadrži sekciju za svaki dućan. Te sekcije se dinamično dodaju u listu i to ćemo pokazat kasnije kod pregleda JavaScript koda. Na desnoj strani stranice nalazi se naslov Dodaj/Uredi dućan te polja za upis imena, adrese i koordinata trgovine. Uz sve to na kraju je i potvrdni gumb koji služi za potvrdu dodavanja ili uređivanja. Važno je napomenuti i to da svaki put kada se pritisne ikonica za brisanje ili uređivanje iskoči nam privremeni modul koji nas pita jesmo li sigurni da želimo izbrisati ili urediti dućan. Potvrdni modul se sastoji od dva gumba i ikone x u desnom vrhu za prekid operacije. Na slici 8 možemo vidjeti kod za listu sa lijeve strane stranice, na slici 9 možemo vidjeti kod za polja i potvrdnog gumba ,a na slici 10 i 11 možemo vidjeti kod za modul te njegov izgled. U kasnijem dijelu ćemo vidjeti kako se lista zapravo puni podacima.

```
<div class="buttons">
  <p class="list_headline">Lista dućana</p>

  <div class="movements">
  </div>
</div>
```

Slika 8: Lista podataka(na početku prazna)

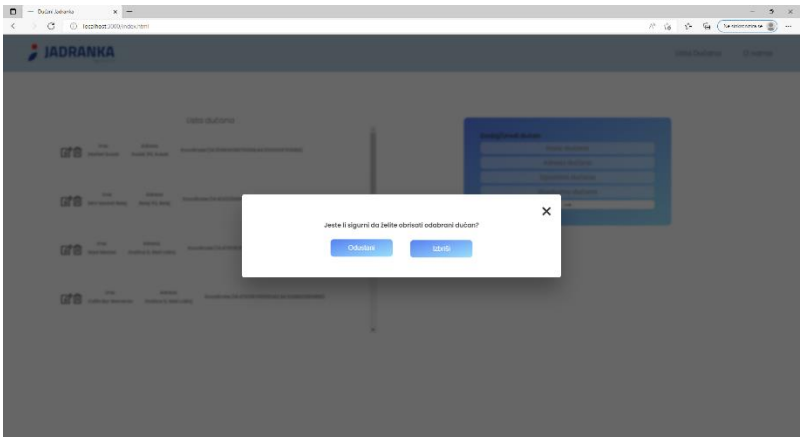
```html
<div class="buttons">
  <div class="operation operation--add">
    <h2>Dodaj/Uredi dućan</h2>
    <form class="form form--add">
      <input
        type="text"
        class="form__input name__add"
        placeholder="Naziv dućana"
      />
      <input
        type="text"
        class="form__input adress__add"
        placeholder="Adresa dućana"
      />
      <input
        type="text"
        class="form__input gpssirina__add"
        placeholder="Gpssirina dućana"
      />
      <input
        type="text"
        class="form__input gpsduzina__add"
        placeholder="Gpsduzina dućana"
      />
      <button type="button" class="form__btn form__btn--add">&rarr;</button>
    </form>
  </div>
```

Slika 9: Kod sekcije za dodavanje i uređivanje dućana

```html
<div class="modal hidden">
  <button class="exit-modal-btn exit-modal-btn-2">&times;</button>
  <h2>Jeste li sigurni da želite obrisati odabrani dućan?</h2>
  <button class="cancel-modal-btn">Odustani</button>
  <button class="confirm-modal-btn">Izbriši</button>
</div>
<div class="overlay hidden"></div>

<div class="modal modal-edit hidden">
  <button class="exit-modal-btn exit-modal-edit">&times;</button>
  <h2>Jeste li sigurni da želite potvrditi promjene?</h2>
  <button class="cancel-modal-btn cancel-modal-edit">Odustani</button>
  <button class="confirm-modal-btn confirm-modal-edit">Potvrdi</button>
</div>
```

Slika 10: Kod za skočni modul uređivanja i brisanja



Slika 11: Izgled skočnog modula

2.2.2 Opis o_nama.html koda

Prvi dio koda isti je kao i kod index.html datoteke. To znači da isto sadrži dio za naslovnu traku i dio di se pritiskom na gumb povezuju html datoteke za Listu dućana i za O nama sekciju. Ono što je drugačije je to da osim toga o_nama.html sadrži još samo paragraf u kojem je tek koji opisuje Jadranka trgovine kao tvrtku. U obje datoteke postoji i kod koji stvara warning , succes i error poruke na dnu stranice ,no to ćemo vidjeti kasnije kada opišemo detaljno način rada uz JavaScript primjere. Na slici 12 vidimo kod za paragraf i time smo opisali kompletan HTML kod korišten u aplikaciji.

```
<div class="o_nama">
  <p class="jadranka_text">
    Za vaš dobar dan na Lošinju i Cresu Jadranka trgovina d.o.o. jedan je od
    vodećih trgovačkih lanaca na Kvarneru, a bavi se veleprodajom i
    maloprodajom te proizvodnjom pekarskih i slastičarskih proizvoda.
    Vlastita maloprodajna mreža i veleprodaja opslužuju se iz
    servisnoskladišne zone sa 5000m2 zatvorenog prostora, u odvojenim i
    specijaliziranim skladištima za piće, industrijsku robu, prehrambene
    artikle, te hladnjača za suhomesnate proizvode, mlijeko i mliječne
    proizvode, svježe i smrznuto meso, smrznutu ribu i povrće, svježe voće i
    povrće. Od 2005. godine članica je udruženja hrvatskih trgovačkih kuća
    Ultra gros. Jadranka trgovina d.o.o. zapošljava 160 djelatnika tijekom
    cijele godine, a za vrijeme turističke sezone taj broj naraste i do 230,
    što je čini jednim od važnijih privrednih subjekata na otoku. Poslovanje
    se odvija na otocima Lošinju i Cresu gdje za vrijeme sezone funkcionira
    25 maloprodajnih prodavaonica raznih formata, a objekti koji su otvoreni
    tijekom cijele godine, omogućavaju stanovništvu Maloga Lošinja, ali i
    stanovništvu manjih mjesta na otoku, stalnu opskrbu živežnim i drugim
    namirnicama. Ključ uspješnog poslovanja tvrtke leži u vještom
    balansiranju između poslovanja u turističkoj sezoni i izvan nje, kao i
    zadovoljenja potreba gostiju i potreba lokalnog stanovništva. Brigu o
    lokalnom stanovništvu ponajviše pokazuju prodavaonice otvorene tijekom
    cijele godine u manjim mjestima na otocima (Punta Križa, Osor, Ćunski,
    Susak...). Poseban je trud i marketinški rad usmjeren na zadovoljstvo
    kupaca i stanovnika otoka,i to kroz aktivnosti koje se provode
    samostalno ili uz podršku dobavljača. Redovito se provode razne akcije
    (loyalty programi, popusti za umirovljenike itd.), a sponzoriraju se i
    razni događaji na otoku. Surađuje se i s mnogim lokalnim obiteljskim
    poljoprivrednim gospodarstvima čiji su proizvodi posebno označeni ili
    smješteni na posebne pozicije u trgovini. Jadranka trgovina je ušla u
    finale nagrade Zlatna košarica za 2017. godinu u kategoriji Društveno
    odgovorna kompanija.
  </p>
</div>
```

Slika 12: Paragraf opisa trgovine

2.3 CSS KOD STRANICE

U CSS datoteci smo kreirali cjelokupni izgled stranice kao i položaj svih elemenata. CSS nije programski jezik nego jezik za pisanje stila. Stil pišemo na način da uzmemo ime HTML klase za određeni objekt te unutar toga definiramo određena polja. Ima bezbroj mnogo mogućnosti za kreiranje stilova i položaja ali najbitnije je za znat da elementi mogu biti raspoređeni ovisno jedan o drugom da nema preklapanja i da između njih bude nekakav razmak(margina) ili mi sami definiramo poziciju na kojoj će se nalaziti element. Pošto je cijeli kod više-manje isti pokazat ćemo na slikama kako to izgleda na određenim elementima. CSS kod ima oko 400 redova i u njemu su za sve klase iz HTML-a kreirani nekakvi stilovi. Na slici 13 vidimo kako izgleda stil za potvrdi gumb modula i cancel gumb modula. Na slici 14 vidimo stil klase movements koja nam definira izgleda kontejnera u koji se sprema lista dućana. Također na slici 14 u polju display vidimo da je postavljeno na flex što znači da će elementi biti prirodno jedan ispod drugog ili jedan pored drugog.

```css
.cancel-modal-btn {
  background-color: #4caf50; /* Green */
  border: 10px;
  border-radius: 8px;
  border-color: #333;
  color: white;
  padding: 12px 40px;
  text-align: center;
  text-decoration: none;
  font-size: 16px;
  margin: 20px;
  width: 150px;
  background-image: linear-gradient(to top left, #9ce1fc, #5781f3);
}
.confirm-modal-btn {
  background-color: #4caf50; /* Green */
  border: 10px;
  border-radius: 8px;
  border-color: #333;
  color: white;
  padding: 12px 40px;
  text-align: center;
  vertical-align: middle;
  text-decoration: none;
  font-size: 16px;
  margin: 20px;
  width: 150px;
  background-image: linear-gradient(to top left, #9ce1fc, #5781f3);
}
```

Slika 13: Stil elemenata modula

```css
/* MOVEMENTS */
.movements {
  grid-row: 2 / span 3;
  background-color: #fff;
  border-radius: 1rem;
  width: 800px;
  height: 500px;
  overflow: auto;
}

.movements__row {
  padding: 2.25rem 4rem;
  display: flex;
  align-items: center;
  justify-items: auto;
  text-align: center;
  border-bottom: 1px solid #eee;
}
```

Slika 14: Stil movements elemenata

Zanimljivo je naglasiti da ne moramo mi raditi stil po klasi elementa. Ako negdje koristimo iste klase onda će svugdje bit raspoređeni isti stilovi jedino je moguće da su pozicije drugačije da se ne bi preklapalo sve. Možemo isto tako ako imam da je jedna klasa parent i ima mnoštvo poklasa u sebi možemo napisati da sva djeca te parent klase dobiju isti stil. Na slici 15 možemo vidjeti da smo definirali stil HTML oznaci za naslove i onda di god mi u kodu koristimo h1 font za naslov svi će imati isti izgled.

```css
h1 {
  font-size: 5.5rem;
  line-height: 1.35;
}

h5 {
  font-size: 2.4rem;
  height: 60px;
}
```

Slika 15: Stil font oznake

2.4 OPIS I IZGLED JAVASCRIPT KODA

Kao što smo i prije spomenuli JavaScript kod je zaslužan za sve što mi možemo raditi na stranici. Gore smo vidjeli funkciju koja šalje request i sada ćemo uz nju opisati i druge stvari u kodu. Na početku svakog koda nalaze se definirane varijable. U našem kodu 27 varijabli koje ćemo koristiti. Većina tih varijabli je zapravo napravljeno na način da smo se ajmo reć povezali na neki element preko imena HTML klase. Pod time mislimo na to da sljedeći put kada budemo nešto radili s tom varijablom ta varijabla nam je neki konkretan objekt u kodu. To može biti naprimjer gumb, naslovna traka, polje za upis podataka ili nešto slično na slici 16 vidimo definirane varijable.

```
const container = window.document.querySelector(".movements");
const button_add = window.document.querySelector(".form__btn--add");
const button_delete = window.document.querySelector(".form__btn--delete");
const button_add_name = window.document.querySelector(".name__add");
const button_add_adress = window.document.querySelector(".adress__add");
const button_add_gpssirina = window.document.querySelector(".gpssirina__add");
const button_add_gpsduzina = window.document.querySelector(".gpsduzina__add");
const cancel_button = window.document.querySelector(".cancel-modal-btn");
const confirm_button = window.document.querySelector(".confirm-modal-btn");
const exit_button = window.document.querySelector(".exit-modal-btn-2");
const modal = window.document.querySelector(".modal");
const overlay = window.document.querySelector(".overlay");
const pop_red = document.querySelector(".alert-red");
const pop_yellow = document.querySelector(".alert-yellow");
const pop_green = document.querySelector(".alert-green");
const exit_red = document.querySelector(".exit-red");
const exit_yellow = document.querySelector(".exit-yellow");
const exit_green = document.querySelector(".exit-green");
const modal_edit = document.querySelector(".modal-edit");
const overlay_edit = document.querySelector(".overlay-edit");
const exit_modal_edit = document.querySelector(".exit-modal-edit");
const confirm_modal_edit = document.querySelector(".confirm-modal-edit");
const cancel_modal_edit = document.querySelector(".cancel-modal-edit");
let button_add_position = 0;
let load_position = 0;
let element = document.createElement("warning");
let novi = {
  id: "",
  ducan_id: 192,
  naziv: "",
  adresa: "",
  gpssirina: "",
  gpsduzina: "",
  timestamp: 3893227,
  thumb_id: 192001,
};
```

Slika 16: Varijable korištene u kodu

Općenito JavaScript kod je najčešće napisan pomoću funkcija koje se zatim zovu u nekom dijelu programa. Osim toga može biti i objektno orijentiran kod ali mi ga tu nismo koristili. U kodu ima mnogo funkcija koje su sporedne i nekoliko glavnih. Glavne ćemo objasniti. Pod sporedne funkcije mislim na funkcije koje služe za neke sporedne poslove kao naprimjer

generiranje nekog random broja , ispravljanje malog slova u veliko di treba i tako nešto slično.

Jedna od glavnih funkcija u programu je funkcija loadEvent();. Ona služi za učitavanje podataka sa servera na sučelje stranice u listu dućana kada se stranica loada. Znači kada mi uđemo na stranicu automatski se na load efekt pokreće ta funkcija u kojoj se šalje GET zahtjev i uzima se to što je vratio iz promise-a i onda se to pomoću HTML koda stavlja u listu i prikazuje se na stranici. Ako se nešto ne izvrši kako treba na ekranu radimo jedan crveni element koji javlja error. Stavio bi sliku kako to izgleda ali nisam uspio napraviti error. Uz error imamo jos žuti prozor za warning i zeleni ako je sve uspješno napravljeno(vidit ćemo poslije na slici).Na slici 17. vidimo izgled loadEvent funkcije koja koristi try i catch.

```
const loadEvent = function () {
  const data = customHttp("http://localhost:3000/items", "GET");
  try {
    data.then(
      (value) => {
        container.innerHTML = "";
        value.forEach((element) => {
          let html = `<div class="movements__row" id="${element.id}">
      <div class="buttons__edit__delete movements__icons">
            <i class="far fa-trash-alt" style="font-size: 24px"></i>
            <i class="far fa-edit" style="font-size: 24px"></i>
          </div>
          <!--
      <div class="movements__type movements__type--deposit">ID:${element.id}</div>
      -->
      <div class="movements__date">Ime:<br>${element.naziv}</div>
      <div class="movements__date">Adresa:<br> ${element.adresa}</div>
      <div class="movements__date">
       Koordinate:(${element.gpsduzina},${element.gpssirina})
      </div>
      <div class="movements__value"></div>
    </div>`;
          container.insertAdjacentHTML("afterbegin", html);
        });
      },
      (e) => {
        element.innerHTML = "Dogodio se problem prilikom loada";
        check_hidden(element);
        pop_red.classList.remove("hidden");
        pop_red.appendChild(element);
      }
    );
  } catch (e) {
    element.innerHTML = "Dogodio se problem prilikom loada";
    check_hidden(element);
    pop_red.classList.remove("hidden");
    pop_red.appendChild(element);
  }
};
```

Slika 17: LoadEvent() funkcija

Druga bitna funkcija je funkcija za dodavanje dućana koja radi na način da kada koristim upiše podatke u desnu kućicu i pritisne gumb stvara se novi dućan u listi dućana na stranici i isti se dodaje pomoću POST poziva u db.json file u kojem su svi dućani. Na slici 18 vidimo samo mali dio funkcije za dodavanje dućana jer je funkcija jako dugo a kod je sličan kao i u LoadEvent funkcije jer isto imamo neko rješavanje errora ispod.

```
novi.id = getRndInteger(90000000, 100000000);
name = capitalizeOnlyFirstLetter(name);
novi.naziv = name;
novi.adresa = adress;
novi.gpsduzina = gpsduzina;
novi.gpssirina = gpssirina;
customHttp("http://localhost:3000/items", "POST", novi);
```

Slika 18: Slanje post zahtjeva kod dodavanja dućana

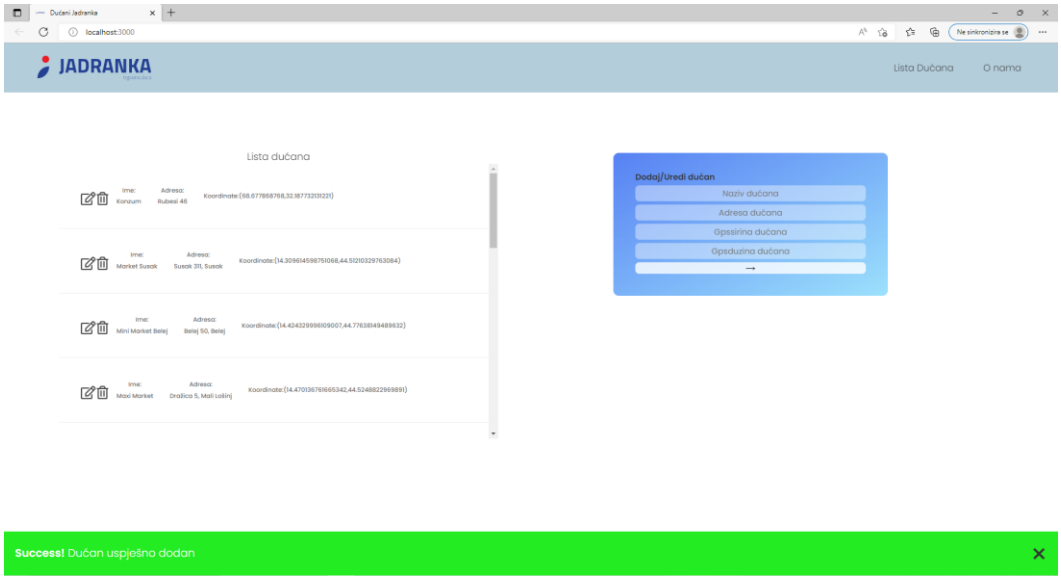
Zadnja bitna funkcija je funkcija koja nam radi i najviše posla. Ona znači gleda kada je stisnut gumb za editiranje a kada je stisnut gumb za brisanje i ona odlučuje šta se radi ovisno koji je stisnut. Ako je za brisat ona onda šalje modul i potvrđuje brisanje te šalje zahtjev za brisanje iz db.json file-a. A ako je za editiranje pokupi podatke iz polja i onda promjeni podatke na način da pošalje zahtjev sa promijenjenim podacima i isto je odgovorna za modul. Uz sve to pazi na manipulaciju podataka tj. Na mala i velika slova na izgled podataka i šalje odgovarajuće erorre , warninge i uspjehe. Dio koda koji šalju zahtjev za brisanjem i za editiranje vidimo na slikama 19 i 20 a na slici 21 možemo vidjeti kako to izgleda kada na stranici dobijemo element za uspjeh kada je dućan dodan.

```
//dio kada stisnemo delete ikonicu
if (e.target.classList.value === "far fa-trash-alt") {
  const parent = e.target.closest(".movements__row");
  modal.classList.remove("hidden");
  overlay.classList.remove("hidden");
  // * DELETE FUNKCIONALNOST - Ako je kliknuta ikonica za brisanje postavim logiku za callback i čekam da se potvrdi u modalu
  confirmButtonCallback = () => {
    customHttp(
      `http://localhost:3000/items/${Number(parent.id)}`,
      "DELETE"
    );
    loadEvent();
    modal.classList.add("hidden");
    overlay.classList.add("hidden");
    element.innerHTML = "Dućan uspješno izbrisan";
    check_hidden(element);
    pop_green.classList.remove("hidden");
    pop_green.appendChild(element);
    console.log("IZBRISANO");
    // * DELETE FUNKCIONALNOST - Nakon potvrde resetiram callback funkciju tako da se može ponovno vezati kada se klikne ikonica za
    brisanje
    confirmButtonCallback = null;
  };
}
```

Slika 19: Isječak koda za delete ikonicu

```
confirmButtonCallbackEdit = () => {
  customHttp(`http://localhost:3000/items/${novi.id}`, "PUT", novi);
  loadEvent();
  modal_edit.classList.add("hidden");
  overlay_edit.classList.add("hidden");
  confirmButtonCallbackEdit = null;
  button_add_adress.value = "";
  button_add_name.value = "";
  button_add_gpssirina.value = "";
  button_add_gpsduzina.value = "";
  button_add_position = 0;
};
//kada stisnemo submit button
```

Slika 20: Isječak koda za ikonicu editiranja



Slika 21: Prikaz uspješnog dodavanja dućana

Na slici 21 vidimo uspješno dodavanje dućana i kada to napravimo(dodamo dućan) automatski se napravi i ovaj dolje zeleni modul koji nam govori kako je akcija uspješno napravljena isto tako ako imamo neki warning kao naprimjer da nismo unijeli sve podatke za editiranje onda nam dođe žuti modul, a ako nešto pukne kada se radi zahtjev ili nešto tako dobit ćemo crveni error modul. Sve te module možemo zatvoriti u gornje desnom kutu modula na način da stisnemo x ali ni ne moramo jer ćemo sa svakom sljedećom akcijom jednostavno prebrisat ovaj modul i ispisat nov koji je tada potreban.

3. ZAKLJUČAK

U sklopu ove prakse naučio sam zaista mnogo o HTML-u, CSS-u i JavaScriptu. Na početku nisam znao skoro pa ništa ali sam počeo raditi projekt i pomalo učiti. Upoznao sam se sa osnovnim konceptima jezika i počeo sam to znanje primjenjivati na projektu .Na početku kod samog kreiranja izgleda stranice trebao sam biti kreativan jer sam sam trebao osmisliti izgled i položaj elemenata i tu sam izgubio možda i najviše vremena. Kada sam napokon završio taj dio počeo sam sa programiranjem stranice. Taj dio je bio jako zabavan i samo rješavanje problema je bilo dosta zanimljivo . U jednom dijelu projekta naišao sam na jedan problem koji nisam mogao sam riješiti pa mi je mentor pomogao da to uspijem. Svakih nekoliko dana kada bi napravio neke promjene na stranici mentor bi to pregledao i preporučio određene promjene u kodu ili dodavanje nekih novih stavki. Najteži dio bio je da si posložio cijelu logiku stranice u glavi i to implementiram bez greške taj glavni dio morao sam mijenjati nekoliko puta jer bi se uvijek našla neka greška koja nije toliko česta ali bi se znala dogodi. Do kraja prakse uspio sam završiti cijelu stranicu sa implementacijom bez error-a. Posto error jedino u try catch-u ali to su exceptioni na koje ne možemo utjecati . U budućnosti ću još probati implementirati GPS mapu pomoću kooridnata svakog dućana te napraviti valjani prikaz. Praksa je sve u svemu bila odlična, zabavna i ljudi s kojima sam radio bili su druželjubivi, pristojni i susretljivi . Ponekad je jedino odgovor na neko postavljeno pitanje bio dostavljen nakon nekoliko dana ali uzevši u obzir ozbiljnost firme i posla to je sasvim razumljivo. Nadam se ću se u budućnosti nastaviti razvijati u ovom području te da ću napredovati i učiti svakim danom.

LITERATURA

1. Jonas Schmedtmann JavaScript course
2. NPM package manager:
   https://nodejs.dev/learn/an-introduction-to-the-npm-package-manager
3. Fetch API:

   https://developer.mozilla.org/en-US/docs/Web/API/Fetch_API
4. CSS Grid:

   https://css-tricks.com/snippets/css/complete-guide-grid/
5. HTML buttons:

   https://developer.mozilla.org/en-US/docs/Web/HTML/Element/button

DODATAK:DNEVNIK PRAKSE

DATUM	MJESTO	OPIS AKTIVNOSTI
3.4.2022. - 20.4.2022.	LOGIN D.O.O	JAVASCRIPT COURSE
21.4.2022.	LOGIN D.O.O	PROBLEMATIKA ZADATKA
22.4.2022.-28.4.2022	LOGIN D.O.O	HTML I CSS KOD PARALELNO
29.4.2022.-3.5.2022	LOGIN D.O.O	FETCH API I NPM
4.5.2022.-20.5.2022.	LOGIN D.O.O	JAVASCRIPT KOD I DEBUGGING
20.5.2022.-26.5.2022.	LOGIN D.O.O	ERROR ,WARING I SUCCES
27.5.2022.-2.6.2022	LOGIN D.O.O	TESTIRANJE,POPRAVLJANJE,BUDUĆNOST