CATCHBUS

Powered by CatchBus team

Sadržaj

Kreiranje projekta

Rješavanje problema

Traženjem svakodnevnih problema htjeli smo pronaći problem koji se može riješiti putem web-aplikacije.

Taj problem je bio snalaženje u prometu sa autobusnim linijama. Većina ljudi se žalilo kako ne znaju kojim autobusom moraju ići da bi došli do željene lokacije.

Tu je stupio CatchBus Team koji je napravio PWA (progressive web application) za lakše snalaženje autobusnim linijama. Aplikacija je namijenjena svim dobnim skupinama. Zato smo uz moderni ali jednostavni dizajn htjeli olakšati što lakše korištenje i snalaženje aplikacijom.

CatchBus je dvojezična aplikacija koja ima mogućnost biranja jezika – hrvatski / engleski. Htjeli smo napraviti aplikaciju i na engleskom jeziku zbog turista koji su trenutno najzaslužniji za rast hrvatske ekonomije. Htjeli bi da imaju najbolje iskustvo u Republici Hrvatskoj te da ostavljaju najbolje recenzije.

Također CatchBus aplikacija i napravljena za djecu koja polaze u škole, a moraju ići autobusom, te za starije ljude kada idu u doktora, pijacu...

Funkcionalnost aplikacije

CatchBus je multifunkcionalna web-aplikacija koja korisniku omogućava lakše snalaženje i prometovanje koristeći mrežu gradskih autobusnih linija. Naime, aplikacija je izvorno izrađena za grad Split što ne znači da je strogo vezana uz pojedini grad.

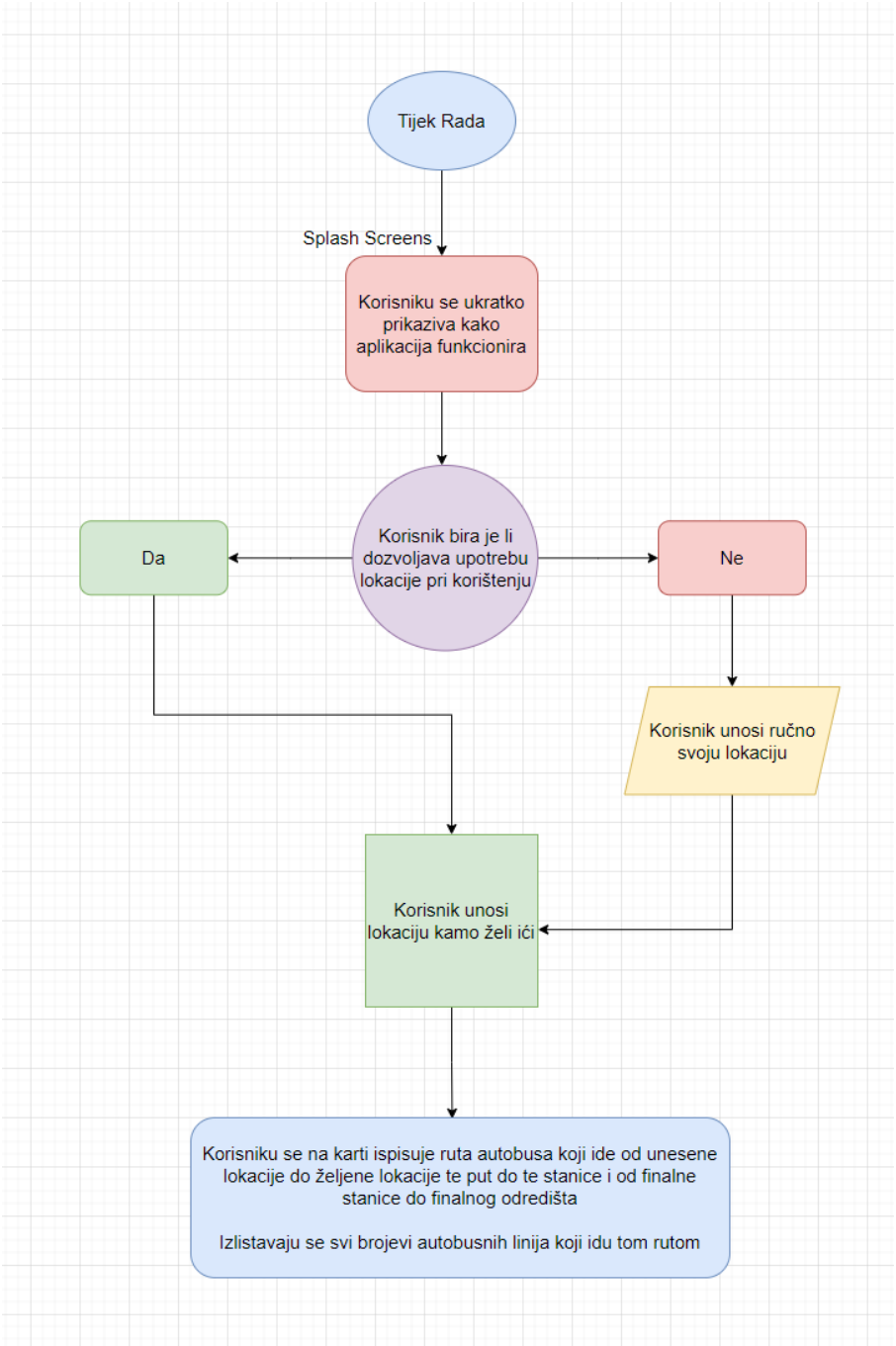
Glavni detalj našeg algoritma je u tome što je sustav moguće implementirati za bilo koji grad.

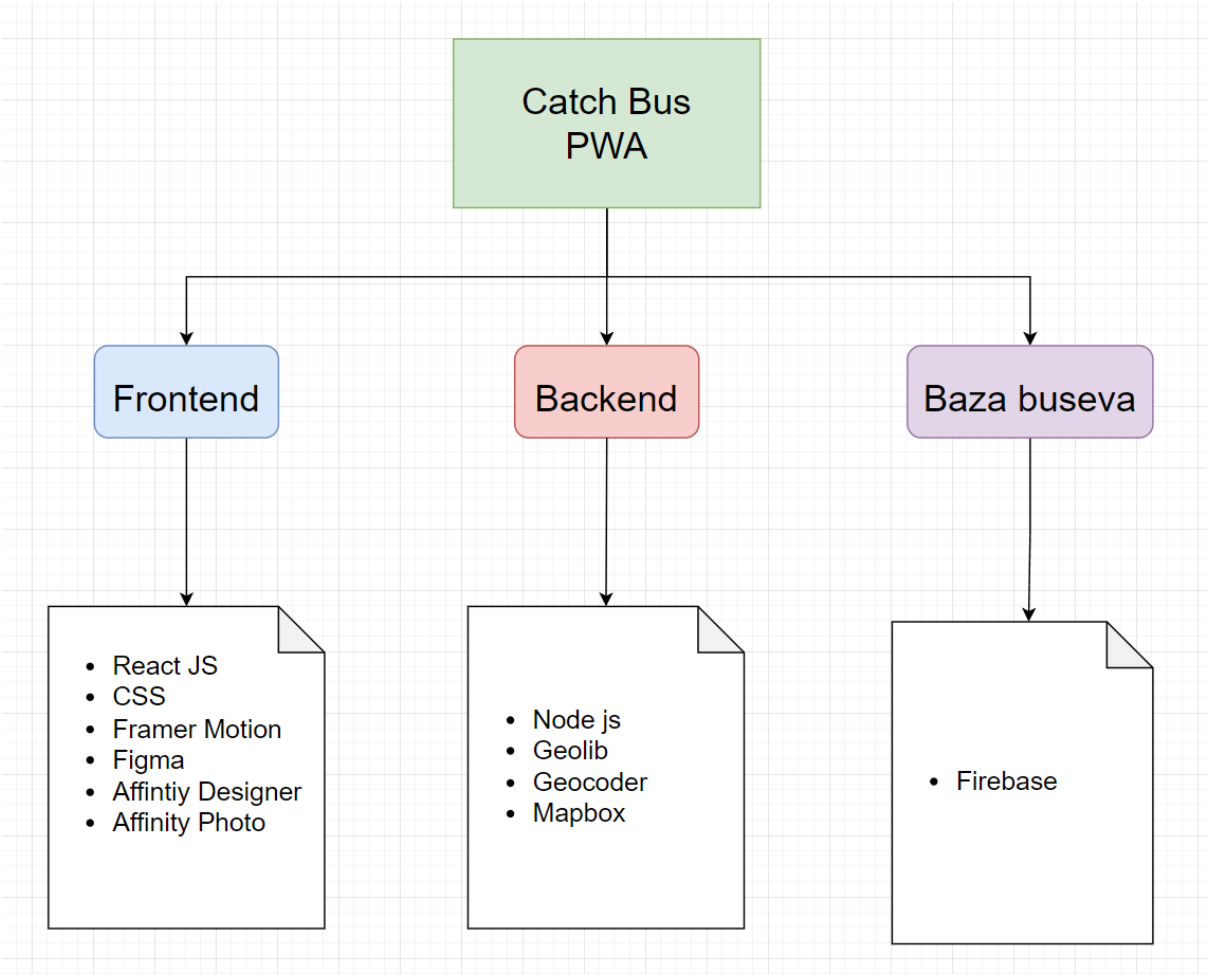
Sve što je potrebno za implementaciju naše aplikacije za određeni grad je nadograđivanje baze podataka dodavanjem pojedinih informacija kao što su linije prijevoza te podaci o svim stanicama koje definirane linije prijevoza koriste.

Također aplikacija je softverski nevezana uz autobuse tako da je algoritam moguće iskoristiti i pri organizaciji mreže linije tramvaja, autobusa, trolejbusa ili bilo kojeg drugog oblika javnog prijevoza.

Sitemap

Izradom kvalitetnog sitemap-a omogućili smo lakše vizualiziranje samog projekta od tijeka rada do dizajna.




Catch Bus
PWA
Frontend
Backend
Baza buseva
React JS
CSS
Framer Motion
Figma
Affintiy Designer
Affinity Photo
Node js
Geolib
Geocoder
Mapbox
Firebase

Tehnologije

Tehnologije također utječu na način rada te su svjesno izarbane na osnovu mogućnosti koje su nam bile potrebne za razradu funkcijonalnosti aplikacije. Bila nam je potrebna tehnologija koja nam može pružiti mogućnost brzog povezivanja s bazom podataka te praktično uzastopno rerenderanje pojednih komponenti, ali također da aplikacija zadrži visoke performanse na svim uređajima. React je bio odgovor na sva postavljena pitanja.

React

React, jedan od vodećih JavaScript knjižnica (engl. librarya) za izradu korisničkih sučelja te ujedno i tehnologija uz pomoć koje smo izradili CatchBus. Razvijena je od strane Facebook-a i objavljena 2013. godine te postaje najutjecajnija knjižnica korisničkog sučelja. Koristimo je za izgradnju komponenti koje predstavljaju logične dijelove korisničog sučelja (engl. User Interface) za višekratnu upotrebu. Prednost React-a je u tome što je jednostavnost izrade komponente svedena na svoj teoretski minimum, to je samo javascript funkcija. Povratna vrijednost iz ove funkcije je vaš html ili korisnički interfejs koji je napisan u posebnoj sintaksi zvanoj jsx.

Prednosti odabira React-a

Glavna prednost Reacta je što će cijelu stranicu učitati samo jednom dok će određeni utjecaj korisnika rerenderati samo potrebne komponente. Recimo da postoji varijabla *x* koju koristi komponenta *komponentaY*, komponenta će se učitati na prvom učitavanju stranice te će se ponovno učitati kada god se varijabla *x* promijeni. U našem slučaju, na glavnoj sekciji, kada god se promijeni jedna od dvije lokacije (početna lokacija ili lokacija odredišta) ponovno će se pokrenuti algoritam i rerenderati mapa iz korisničkog sučelja.

Frontend

Uvod

Frontend je jedna od bitnijih stavki svake veće aplikacije, stranice, software-a...

Kreiranjem CatchBus PWA htjeli smo da korisnici imaju najbolje korisničko iskustvo i korisničko sučelje. (PWA jest web „progresivna web aplikacija“ koja izgleda i ponaša se baš kao mobilna aplikacija.)

To smo ostvarili korištenjem modernih tehnologija kao što su: React JS (JavaScript biblioteka za izgradnju korisničkih sučelja), Framer Motion (knjižnica za animacije), CSS4...

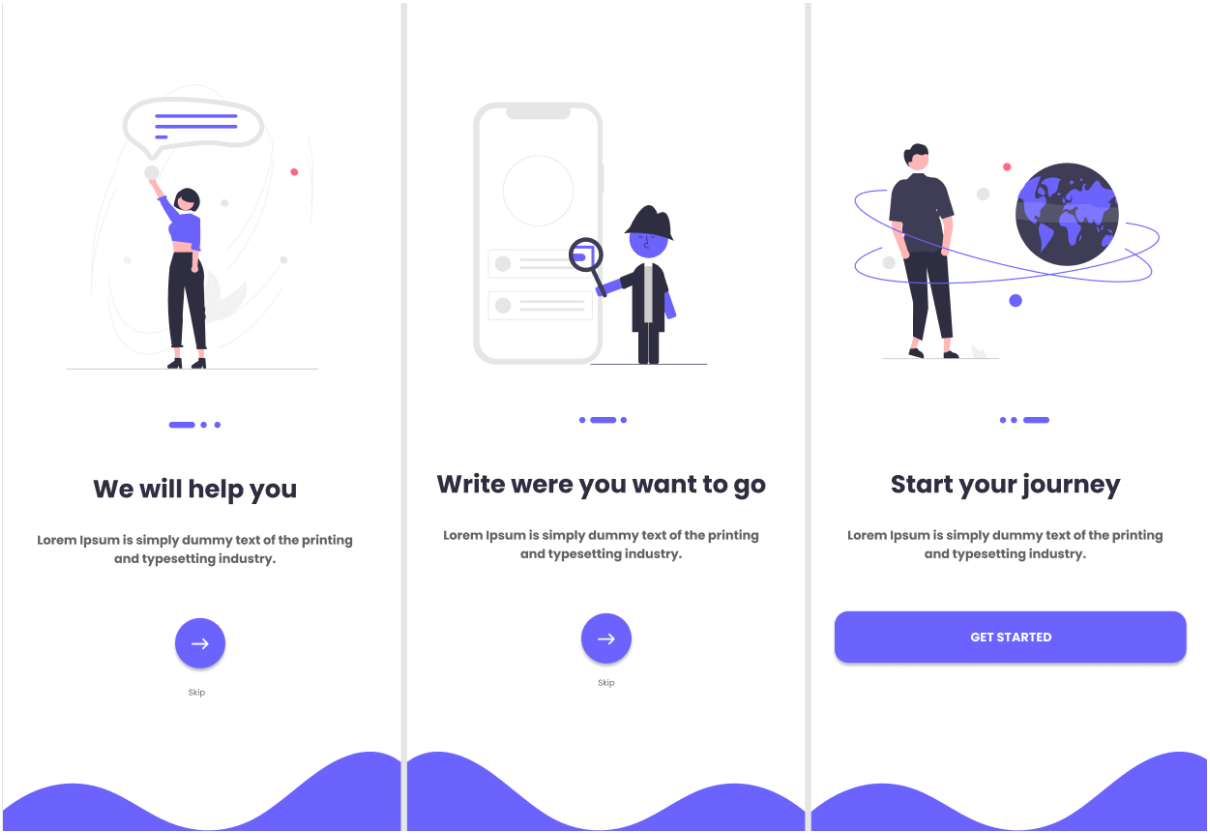
Dizajn aplikacije smo radili pomoću alata kao što su: Figma alat za dizajn sučelja za suradnju), Affinity Designer (grafika, dizajn), Affintiy Photo (uređivač slika).

Dizajn

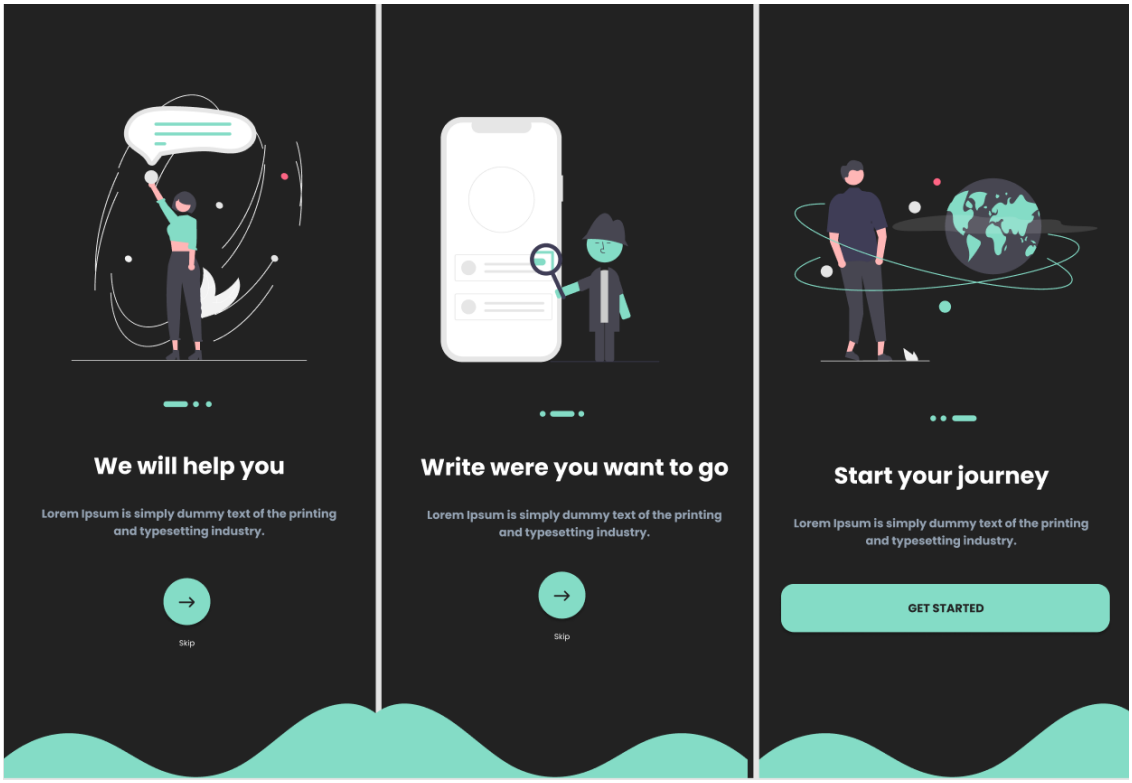
U samom dizajnu htjeli smo korisnicima što više olakšati korištenje CatchBus aplikacije. U dizajn je implementirana svijetla i tamna tema koja se može promijeniti u postavkama, a kao zadana tema jest tema uređaja.

Dizajn smo podijelili u nekoliko sekcija kako bi lakše vizualizirali ideju:

1. **Splash Screen** (ubacili smo splash screen-ove kako bismo korisnicima u nekoliko koraka objasnili kako koristiti aplikaciju – oni se pojavljuju prilikom prvog ulaska u aplikaciju)



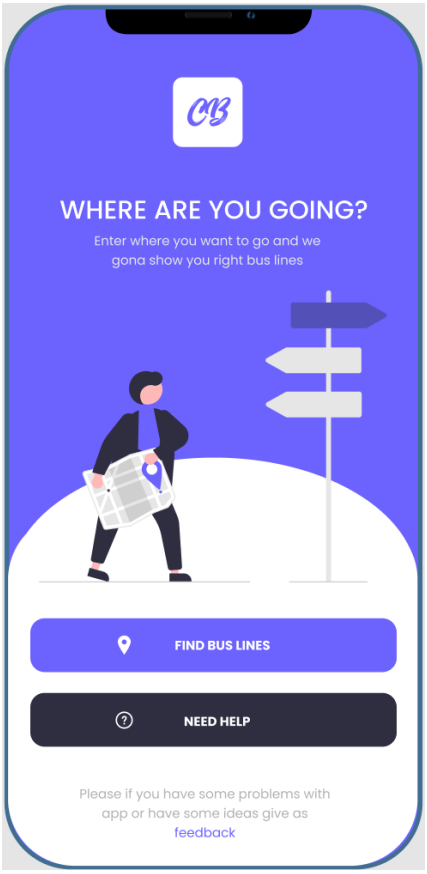
Splash Screen – Svijetla Tema

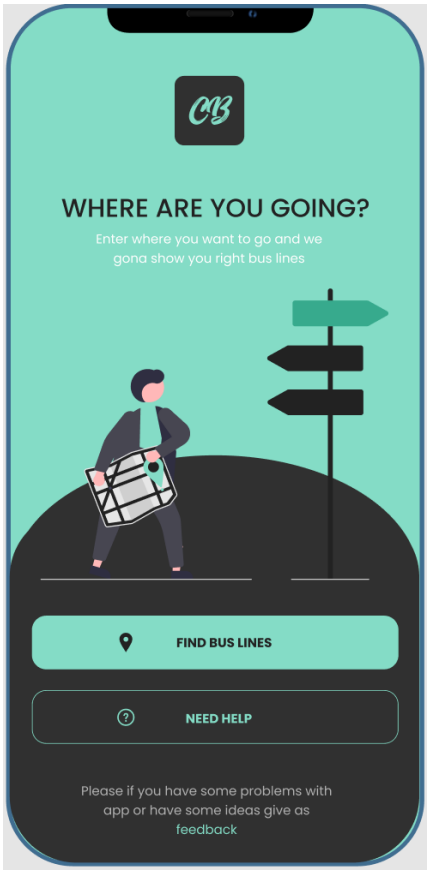


Splash Screen –Tamna Tema

2. **Naslovna**

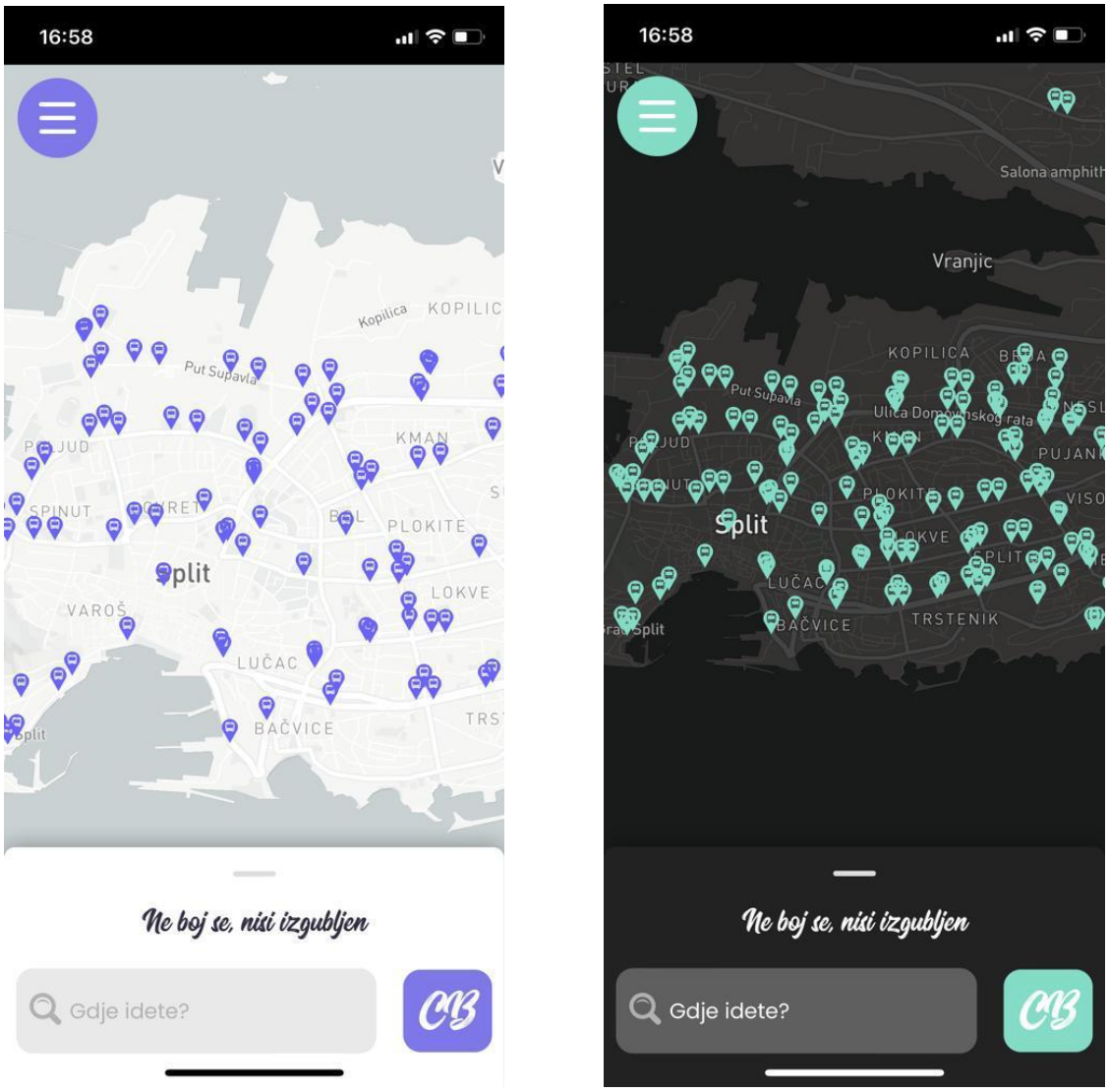
Prikazuje poruku dobrodošlice s kratkim smjernicama za pronalazak ispravnog autobusa.



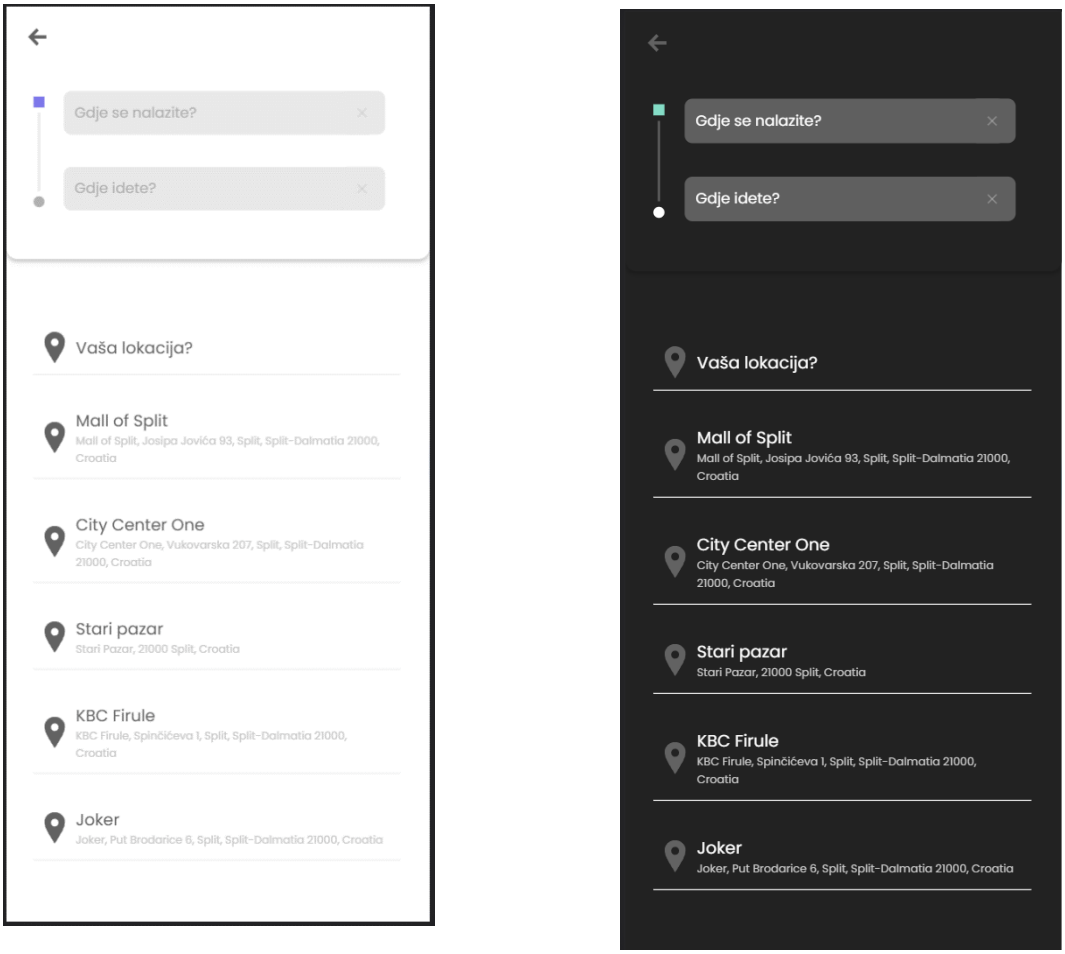


3. Mapa

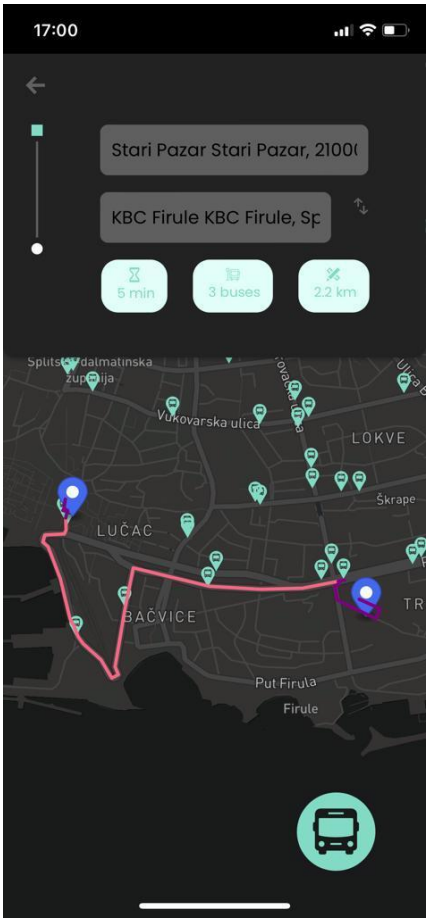
Interaktivna mapa pokazuje sve stanice kojim prometuju autobusi – nakon unosa lokacije gdje želimo ići prikazuje se najbrži put do stanice zajedno sa brojevima autobusnih linija koje prolaze tom stanicom do finalnog stanice. Također od finalne stanice prikazuje put do finalnog odredišta.

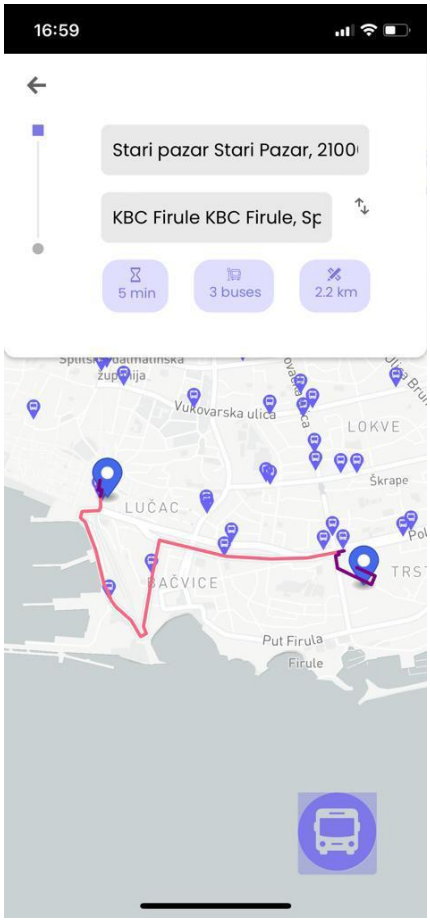


Mapa sa svim stanicama



Ponuđene lokacije

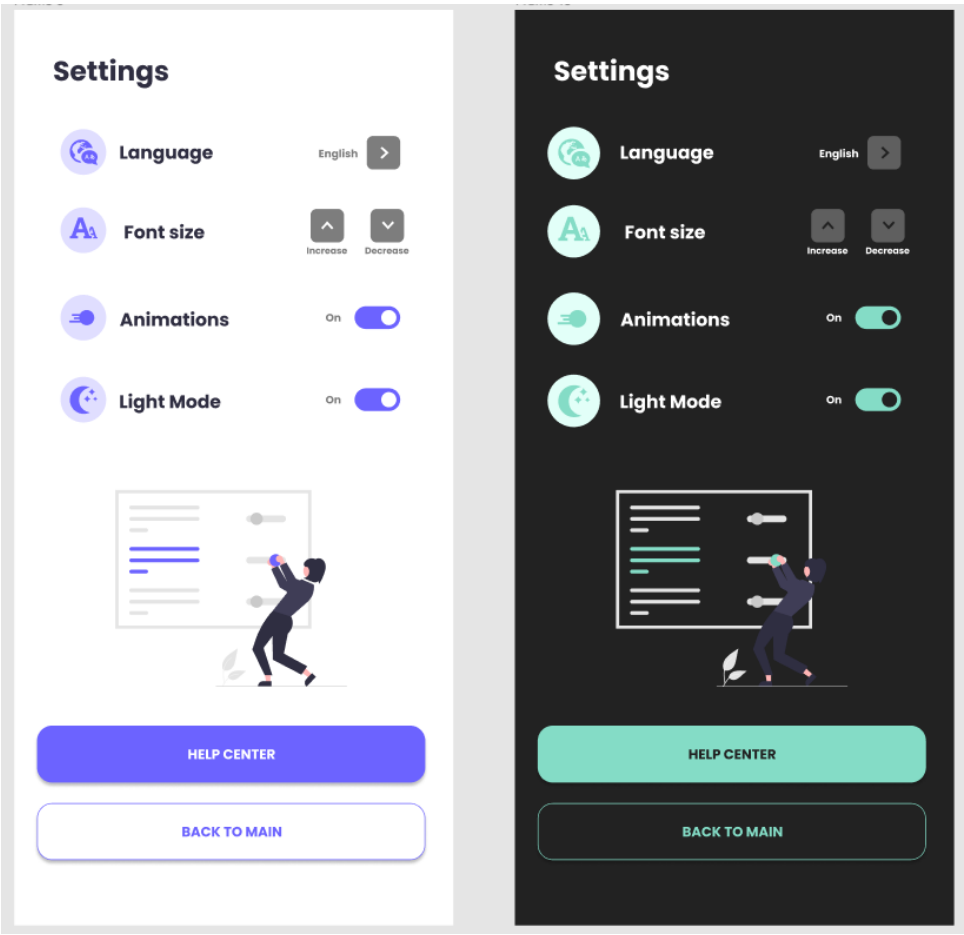




Izlistana autobusna ruta zajedno sa autobusima koji prolaze tom rutom

4. Postavke

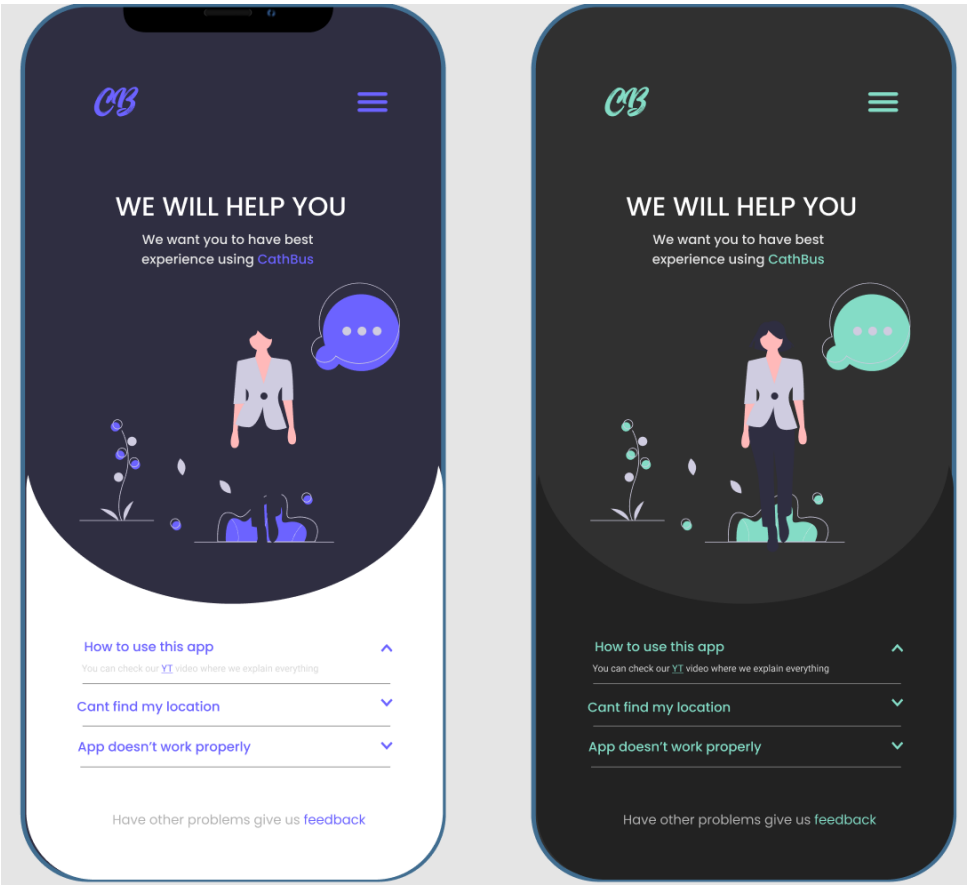
U postavkama se podešava veličina fonta, mijenjanje jezika, mijenjanje teme – tamna, svjetla.



Svijetla i tamna tema postavki

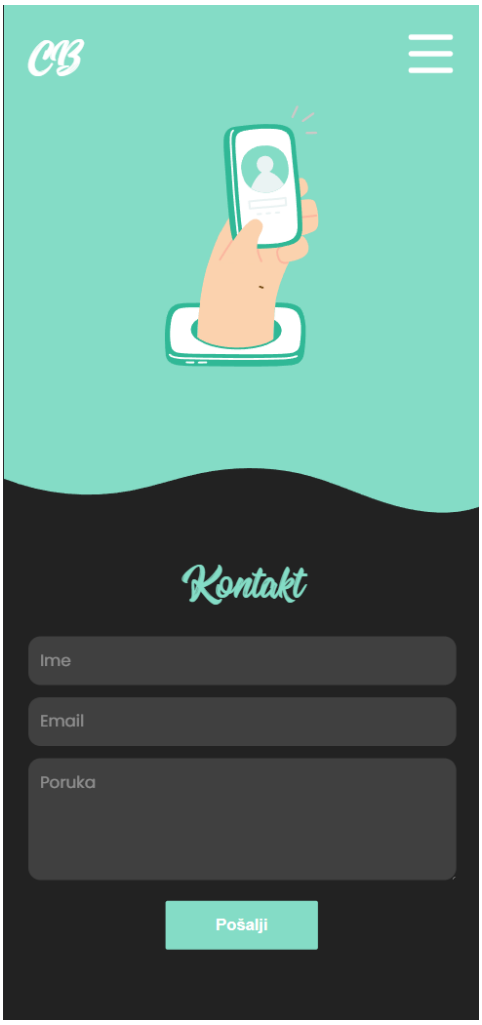
6. Pomoćni centar

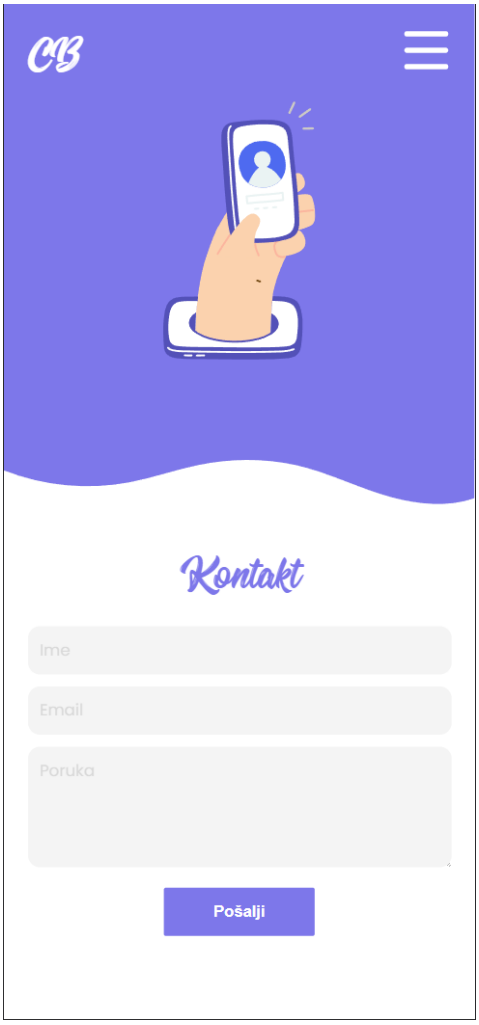
U pomoćnom centru se nalazi „accordion“ sa najčešćim pitanjima i odgovorima.



8. Kontakt

Ako korisnik ima primjedbi ili pohvala može nas kontaktirati putem email-a preko aplikacije.





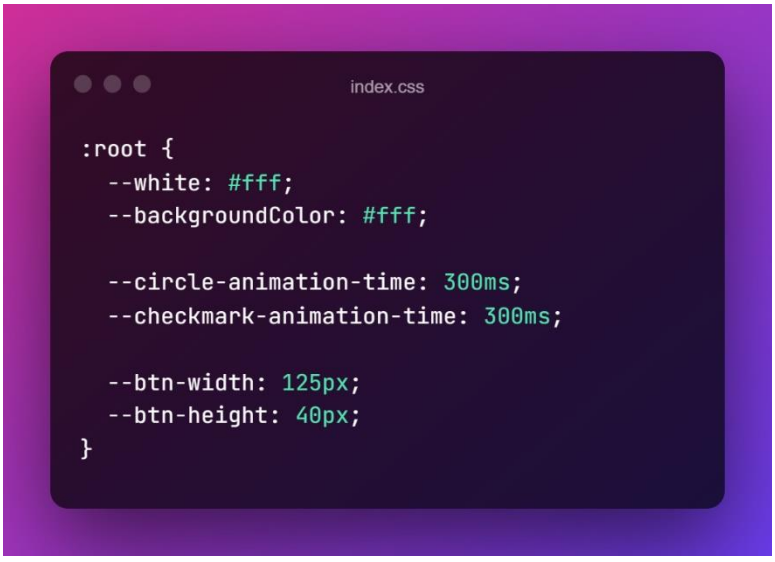
Dodaci i alati

Za izradu što ljepšeg korisničog sučelja korišteni su neki dodaci i paketi kako bi korisničko iskustvo bilo što bolje.

- Za kreiranja animacija koristili smo **Framer Motion**
- Za kreiranje dvojezičnosti (eng/hr) koristili smo **i18next** sa njegovim dodacima kao što su : browser-language detector, http-backend, react-18next.
- Za otvaranje više stranica korišten je „react-router-dom“
- Da bi se sve to lijepo upakiralo korišten je „react-app-rewired“
- Za kreiranje klizača stranica korištena je knjižnica „react-slick“ zajedno sa „slick-carousel css“

Izrada korisničkog sučelja

Za kreiranje grid-a frontend-a korišten je CSS Flexbox. Pomoću CSS varijabli smo deklarirali: boje, veličine, trajanje animacija...



Promjena jezika uz pomoć i18next i njegovim dodacima napravljena je na sljedeći način:

1. Napravljena je „locales“ mapa koja sadrži 2 JSON datoteke. Jedan JSON je ispunjen sa engleskim jezikom dok je drugi ispunjen hrvatskim jezikom
2. Uz pomoć i18next radimo „config file“ gdje upisujemo zadane postavke jezika za aplikaciju (zadani jezik, fallback jezik, detektor jezika)
3. U svakoj stranici aplikacije implementiramo JSON datoteke koje povlače riječi i rečenice te se prikazuju na aplikaciji ovisno o odabranom jeziku

Dodavanje animacije je napravljeno uz pomoć Framer Motion na sljedeći način:

animation.js

```
<motion.div initial="hidden" animate="visible" variants={{
            hidden: {
                opacity: 0,
                y: -100
            },
            visible: {
                opacity: 1,
                y: 0,
                transition: {
                    delay: 0.5,
                    duration: 0.5
                }
            }
        }}>

  <section>
    <h1>Content goes here...</h1>
  </section>
</motion.div>
```

Backend

Uvod

Glavna ideja i zamisao je bila predstaviti korisniku najbolji način za doći od prve lokacije, koja može predstavljati korisnikovu trenutnu lokaciju ali i ne mora, do odredišne lokacije (lokacija do koje korisnik želi doći). Naravno, aplikacija nije u startu imala finalnu verziju algoritma, već se algoritam povremeno unapređivao ovisno o pronađenim problemima (bug-ovi).

Način rada

Pri prvom učitavanju aplikacije se dohvaćaju stanice i autobusa iz baze podataka. Autobusi su u kodu formirane u obliku niza objekata gdje svaki objekt predstavlja jedan autobus te svaki objekt ima broj autobusa (npr. 9), ime autobusa (npr. "R. Njive – Traj. Luka") i niz u kojem se nalaze sva imena stanica kojima taj autobus prolazi (npr. ["StajalisteBrda", "Hercegovacka4A", "Hercegovacka5A",...]).

Autobusne stanice su u kodu formirane također u obliku niza objekata te svaki objekt ima ime stanice i lokaciju stanice (atribut lokacija je objekt sa svoja dva atributa *longitude* i *latitude*).

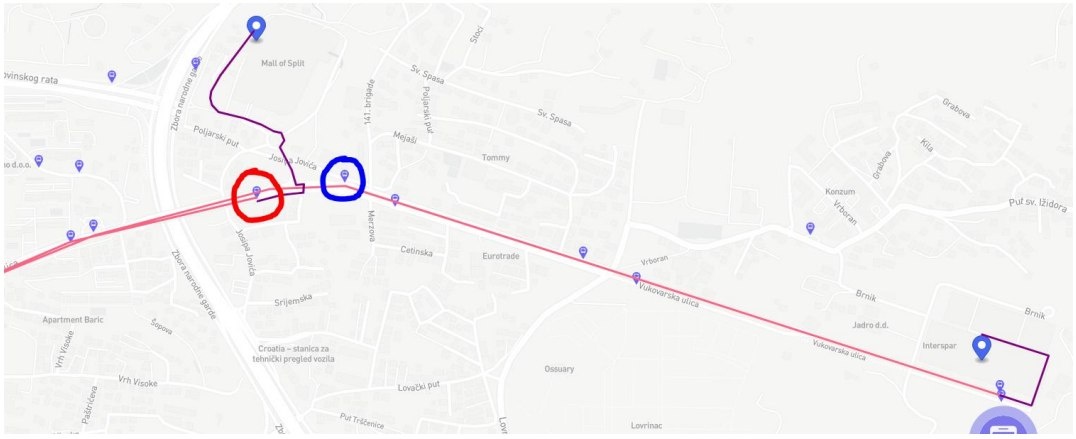
Nakon što korisnik unese obje lokacije u pozadini se poziva funkcija u kojoj se sortiraju sve stanice iz baze podataka po udaljenosti od korisnikove lokacija i odredišne lokacije na način da se kreiraju dva niza stanica gdje svaka stanica predstavlja objekt. Sortiranje se izvršava uz pomoć geolib knjižnice (engl. *library*) koja kao povratnu informaciju vraća sortirani niz te vrijedi isti postupak i za sortiranje stanica po udaljenosti od prve lokacije i za sortiranje po udaljenosti od druge lokacije.

```
const R = 6371e3; // radijus zemlje u metrima
const ϕ1 = lat1 * Math.PI/180; // latitude prve lokacije u radijanima
const ϕ2 = lat2 * Math.PI/180; // latitude stanice u radijanima
const Δϕ = (lat2-lat1) * Math.PI/180;
const Δλ = (lon2-lon1) * Math.PI/180;

const a = Math.sin(Δϕ/2) * Math.sin(Δϕ/2) +
          Math.cos(ϕ1) * Math.cos(ϕ2) *
          Math.sin(Δλ/2) * Math.sin(Δλ/2); //formula za izracun
const c = 2 * Math.atan2(Math.sqrt(a), Math.sqrt(1-a));

const d = R * c; // udaljenost u metrima
```

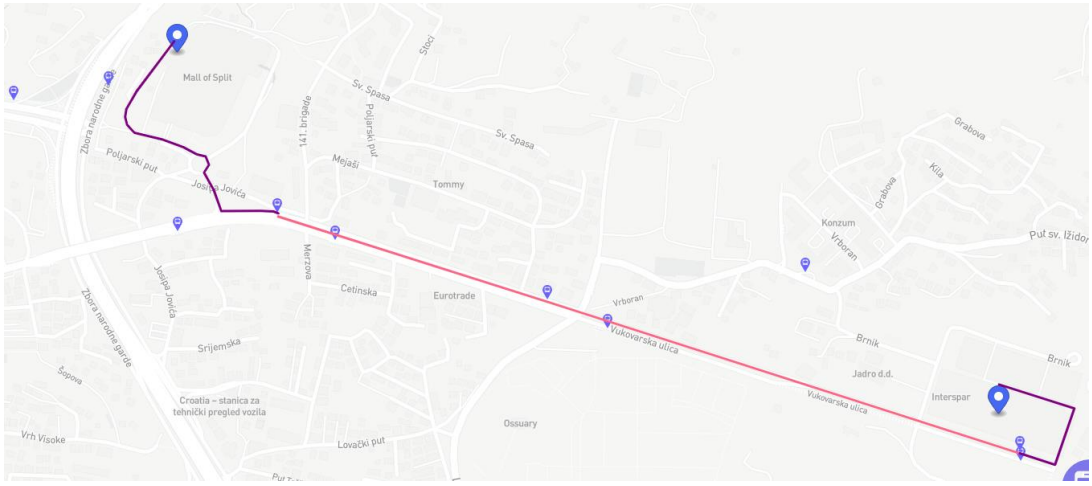
Zatim slijedi funckija *pronadiStanice()* koja se poziva kada god se jedna od lokacija promijeni. Navedena funkcija predstavlja ujedno i glavni dio koda s obzirom da se u njoj nalazi glavni dio algoritma za pronalazak matematički najbolje opcije dolaska do odredišta. Funkcionira na način da provjerava po redu najbolje moguće opcije. Na primjer, najbolja opcija bi bila kada bi korisnik u autobus ušao na stanici koja je njemu najbliža, a izašao na stanici koja je najbliža lokaciji do koje želi stići. U slučaju da ne postoji niti jedan autobus koji spaja te dvije stanice provjerava se sljedeća najbolja opcija.

```
useEffect(() => {
    if (sortiraneStaniceA && sortiraneStaniceB) {
      pronadiStanice()
    }
  }, [sortiraneStaniceA, sortiraneStaniceB]);
```

Nakon što se pronađu dvije stanice koje su povezane bilo kojom autobusnom linijom slijedi provjera potencijalne greške koja bi predstavljala da korisnik kruži cijelom rutom autobusa kako bi došao do odredišta umjesto da uđe na stanicu s druge strane ceste.



Slika prikazuje kako bi radio algoritam bez provjere potencijalne greške na relaciji između lokacija “City Center One” i “Mall of Split” s obzirom da je crveno zaokružena stanica prepoznata kao stanica koja je bliža od plavo zaokružene te su povezane istim autobusom.



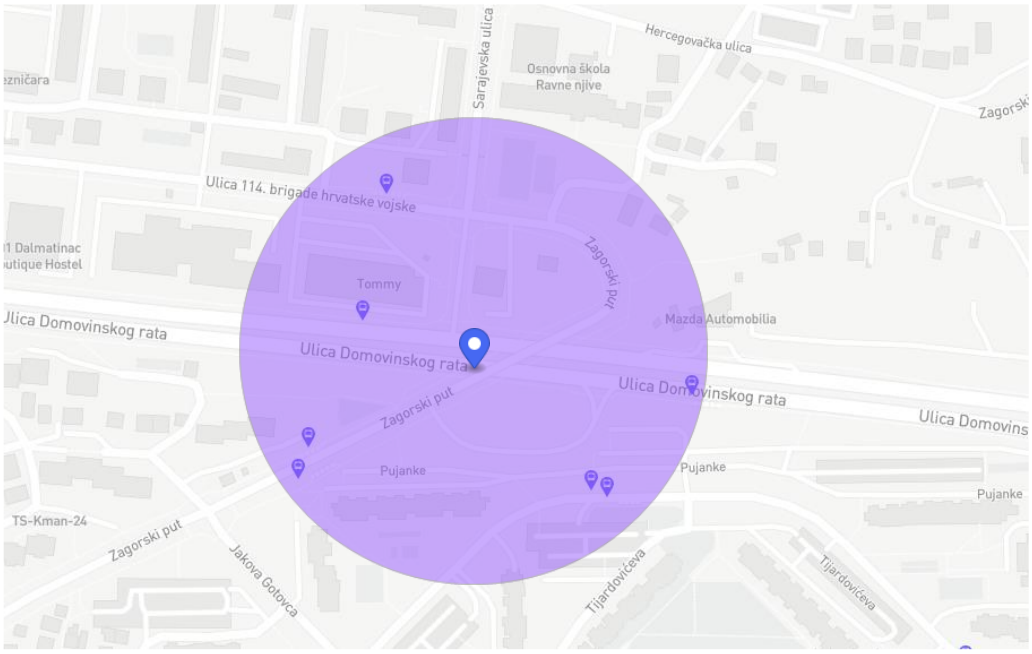
Slika prikazuje rad aplikacije kada se u algoritmu nalazi funkcija provjere potencijalne greške gdje algoritam prepoznaje da je riječ o kruženju te da korisnik može ući u stanicu koja se nalazi na drugoj strani ceste što bi predstavljalo puno bolju opciju.

Prva verzija algoritma

CatchBus je u početku predstavljalo algoritam koji će korisnika spojiti do prve najbliže stanice i odvesti ga do stanice koja je najbliža odredištu. Ta verzija algoritma je bila u funkciji u toku razrade ideje i izrade baze podataka. Algoritam je tada koristio funkciju *findNearest()* iz geolib knjižnice koja kao povratnu vrijednsot vraća samo jednu stanicu, stanicu koja je najbliža lokaciji.

Druga verzija algoritma

U drugoj verziji algoritma je dodana mogućnost provjere povezanosti prve dvije najbliže stanice od prve lokacije sa prve dvije najbliže stanice od druge lokacije. Na prvu je to izgledalo kao finalno rješenje no ipak problem nije bio ni blizu konačnog rješenja s obzirom da se u većini slučajeva u krugu radijusa 200m nalazi 4 ili više stanica.



Treća verzija algoritma

Treća verzija algoritma, ujedno i verzija koju trenutno aplikacija koristi za rad, prvenstveno nudi rješenje problema iz prethodne verzije. Algoritam ne uzima u obzir samo prvu najbližu stanicu za početnu lokaciju i prvu najbližu stanicu za odredišnu lokaciju, niti uzima određeni skup najbližih stanica već postoji n! mogućnosti gdje n predstavlja broj stanica u bazi podataka. Algoritam redom provjerava sve potencijalne opcije sve dok ne nađe dvije stanice koje su međusobno povezane u barem jednom od autobusnih linija gdje prestaje s pretraživanjem. Također, u ovoj verziji se pojavljuje prethodno navedeni dodatak algoritmu koji provjerava potencijalne greške i dodatak koji računa potrebno vrijeme čekanja i udaljenost između te dvije lokacije.

Algoritam u razvoju

Prethodna verzija nudi ispravne opcije do trenutka kada smo u bazu podataka unijeli i grad Solin gdje se postavlja pitanje vezano za presjedanje. U tom trenutku se pojavljuje potreba mogućnosti presjedanja. CatchBus tim trenutno radi na tome te je pronađeno rješenje za taj problem, no još uvijek su u toku razne provjere i testiranja. Algoritam bi trebao funkcionirati na način da nakon što algoritam pronalaženjem izađe iz kruga radijusa određene udaljenosti da se pretražuje opcija presjedanja. Algoritam bi presjedanje pretraživao na način da na osnovu autobusnih linija koje prolaze u blizini početne lokacija i autobusnih linija koje prolaze u blizini odredišne lokacije traži presjek dviju autobusnih linija, lokaciju gdje bi korisnik trebao izaći iz prvog autobusa i ući u drugi.

Navigacija

Navigacija čini veliki dio funkcije CatchBus aplikacije s obzirom da je jedan od ciljeva prikazati korisniku put do početne stanice i od odredišne stanice do odredišta. Prikazuje se i put autobusa kojim će se korisnik voziti. Izbor platforme za prikaz mape i ostalih pojedinosti usko povezanih s mapom je bio između Google Maps platforme i Mapbox GL.

Google Maps predstavlja vodeću platformu kad je u pitanju prikaz mape ili bilo koji API vezano za mapu (*Directions API, Geocoding API, Places API*) no s obzirom da smo se odlučIli za React kao glavnu tehnologiju, Mapbox GL je puno bolje rješenje. Naime, Mapbox GL nudi React verziju svoje platforme pod nazivom *react-map-gl* koja se puno bolje slaže sa React-om od bilo koje platforme za prikaz mape.

Mapbox GL također koristi i *Uber* što smo uzeli u obzir jer mapa u CatchBus-u ima relativno sličnu ulogu kao i kod *Uber* aplikacije.

Prikaz rute

Prikaz rute korisniku se generira uz pomoć Mapbox Matching API-a i Mapbox Directions API-a. Na mapi se generiraju 3 rute. Prva ruta predstavlja hodanje korisnika od početne lokacija do početne stanice te API kao ulazne paremetre prima samo korisnikovu početnu lokaciju i lokaciju odgovarajuće početne stanice (stanica na kojoj će korisnik ući u autobus).

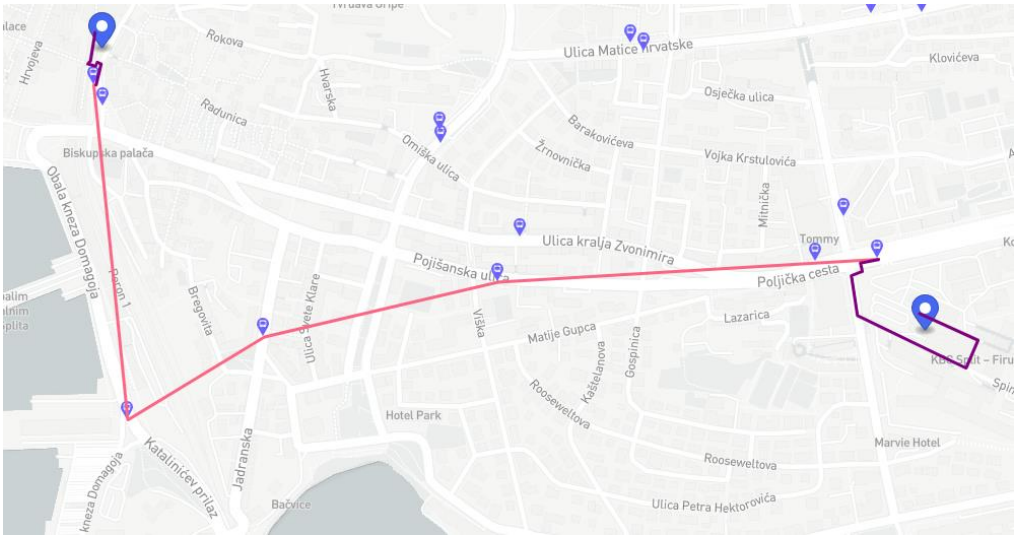
Druga ruta predstavlja rutu autobusa kojom će autobus doći od stanice na kojoj korisnik ulazi do stanice na kojoj korisnik izlazi iz autobusa. To je realizirano na način da nakon što se odrede početna stanica, stanica odredišta i odgovarajući autobusi se uzima jedan od autobusa za generiranje rute. Zatim se provjerava indeks početne stanice u nizu stanica odabranog autobusa i indeks odredišne stanice nakon čega se iz niza stanica odgovarajućeg autobusa generira niz stanica koje se nalaze između tih dviju stanica.

```
const potrebneStanice = odgovarajuciBus.stanice.slice(indexA, indexB + 1);
```

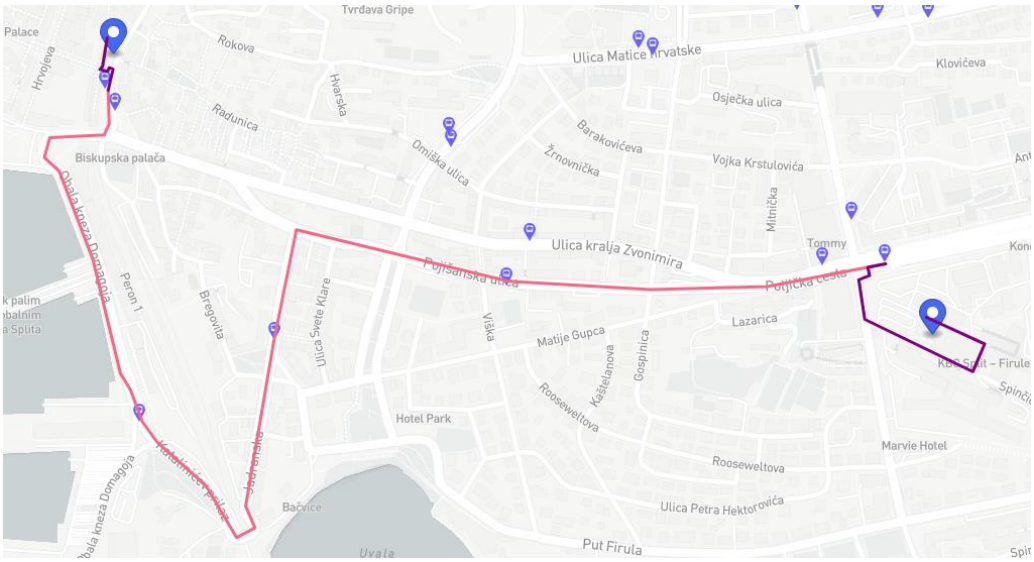
Niz formiran od koordinata stanica se prosljeđuje u API koji kao povratnu vrijednost vraća koordinate rute

Matching API

Mapbox Map Matching API na osnovu nejasnih GPS koordinata i osnovu mreže cesta i puteva uz pomoć Directions API-a generira čiste staze koje se mogu prikazati na karti ili koristiti za druge analize.



Prikaz rute bez Matching Api-a



Prikaz rute sa Map Matching API-em

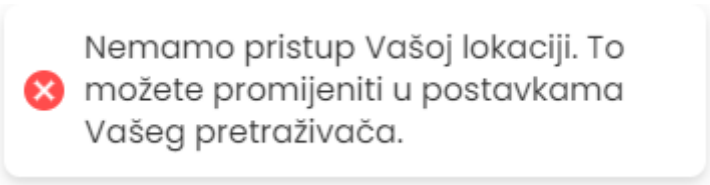
Map Matching API vraća niz koordinata koje se potom prosljeđuju u *PolylineOverlay.jsx* koji na temelju dobivenih koordinata iscrtava rutu.

Lociranje korisnika

Aplikacija pri prvom učitavanju traži pristup lokaciji korisnika nakon čega se korisnikova lokacija odma prikazuje na mapi. U slučaju da korisnik odbije pristup njegovoj lokaciji, aplikacija će normalno nastaviti sa radom.

```
const locirajKorisnika = () => {
      navigator.geolocation.getCurrentPosition(position => {
        setKorisnikLociran(true)
        setUserLocation({
            longitude: position.coords.longitude,
            latitude: position.coords.latitude
        })
      })
    }
locirajKorisnika();
```

Korisniku se pri odabiru lokacija automatski nudi opcija “Vaša lokacija”. Klikom na tu opciju se šalje request Geocoding API-u koji na temelju lokacije (longitude i latitude) vraća naziv lokacije/ulice u kojoj se korisnik nalazi. U slučaju da korisnik odbije pristup aplikacije njegovoj lokaciji te odabere opciju “Vaša lokacija” pojavljuje se *toast* sa obavijesti da je pristup lokaciji odbijen te upute za mogućnost dopuštenja pristupa lokaciji.



Pristup lokaciji odbijen - Toast

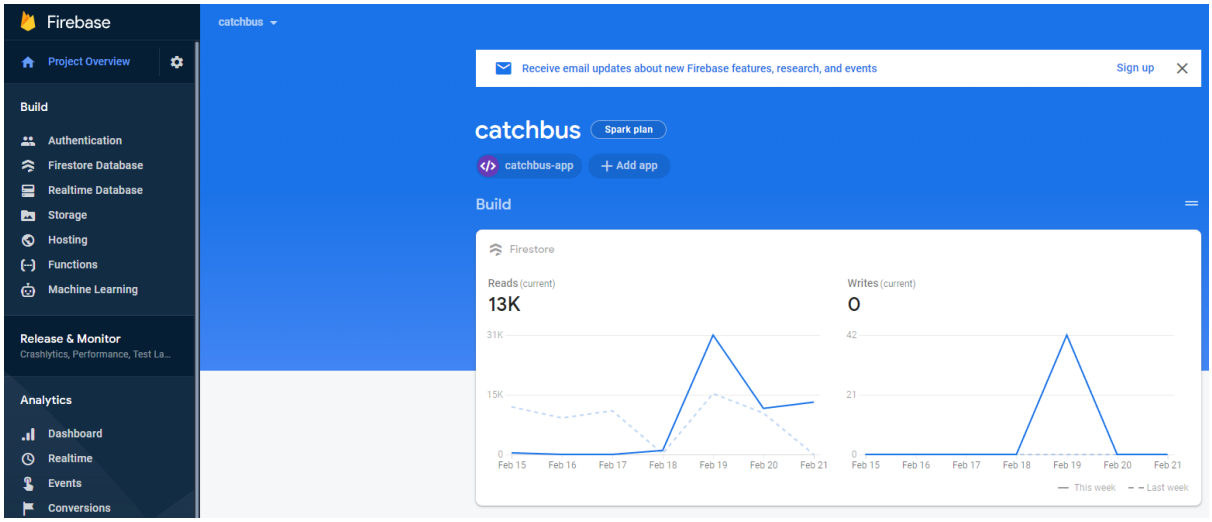
Baza Podataka

Uvod

Baza podataka je, uz dizajn i sam kod ove aplikacije, jednako važan dio projekta jer bez adekvatne, funkcionalne i baze sklonoj doradi i unaprjeđenju, ne bi bilo moguće odraditi ovaj projekt.

Baza podataka ove web-aplikacije služi za dohvaćanje i korištenje podataka o kojima ćemo nešto kasnije.

Za izradu baze podataka koristili smo Firebase, Google-ovu platformu za izradu online baze podataka koja se, ovisno o korisnikovom projektu, povezuje na ostale platforme za programiranje. Firebase je jednostavan za korištenje, no povećanjem projekta i traženjem većih mogućnosti, Firebase može i koštati.



Slika 3.1. Sučelje platforme Firebase

Na isti način, mi smo našu bazu podataka povezali na projekt u React.js-u (Visual Studio Code 2019.) te tako napravili našu bazu podataka funkcionalnom.

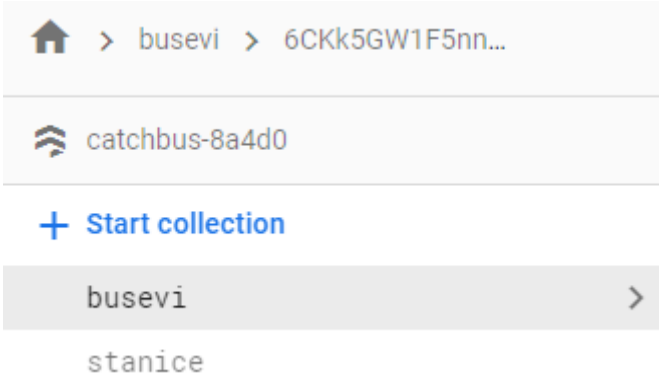
Pomoću funkcije „useEffect“ u samo nekoliko linija koda sakupljamo podatke zapisane u bazi podataka. Kod se nalazi na slici ispod:

Slika 3.2. Kod sakupljanja podataka iz baze podataka u aplikaciju

Također, valja napomenuti da za svaku stanicu, koja je u Firebase-u zabilježena kao dokument, aplikacija sakuplja svaki taj traženi dokument ovisno o potrebi korisnika.

Struktura

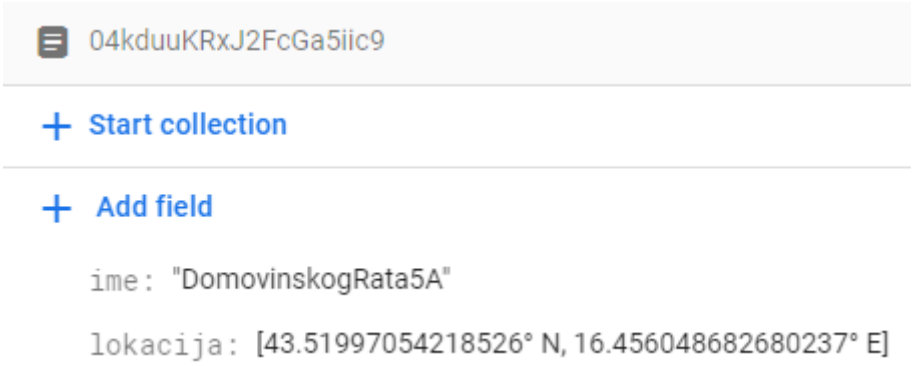
Baza podataka sastoji se zapravo od dvije baze, bazu autobusnih stanica te bazu samih autobusnih linija.



Slika 3.3. Grananje na bazu Autobusa i Stanica

Baza Autobusnih Stanica

Baza podataka autobusnih stanica sastoji se od nekoliko vrijednosti, a to su: Ime (naziv stanice) i lokacija stanice (geopoint).

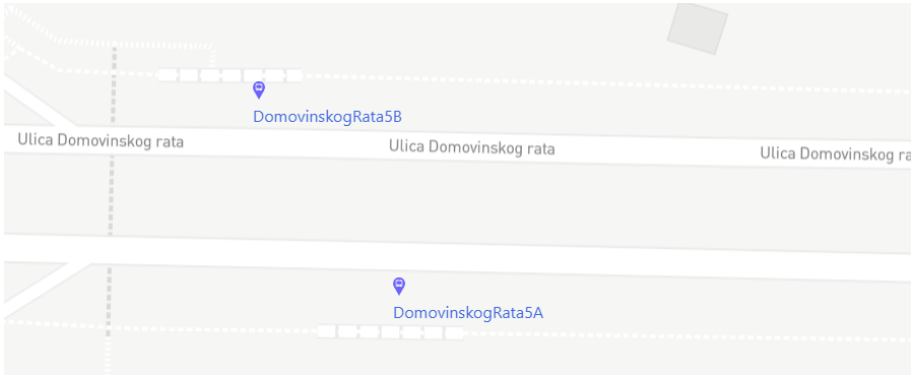


Slika 3.4. Primjer jedne stanice unutar baze pod.

Ime autobusne stanice

Imena stanica dodjeljivali smo jedinstvenim nazivom koji se dobije kombiniranjem naziva ulice u kojoj se nalazi, te smjera u kojem vozi (ovisno je li vozi prema centru grada ili obratno).

Tako na primjer, dobijemo stanicu „Poljicka7A“, što bi značilo da se ova stanica nalazi na Poljičkoj cesti, nalazi se u smjeru koji ide put centra grada, te je sedma u nizu tih stanica.



Slika 3.5. Primjer imenovanja stanice u jednom i u drugom smjeru

Jedinstveni naziv stanica omogućava funkcionalno korištenje istih kako bi se na mapi grada prikazivale upravo te stanice prilikom odabira jednog od autobusa.

Lokacija autobusne stanice

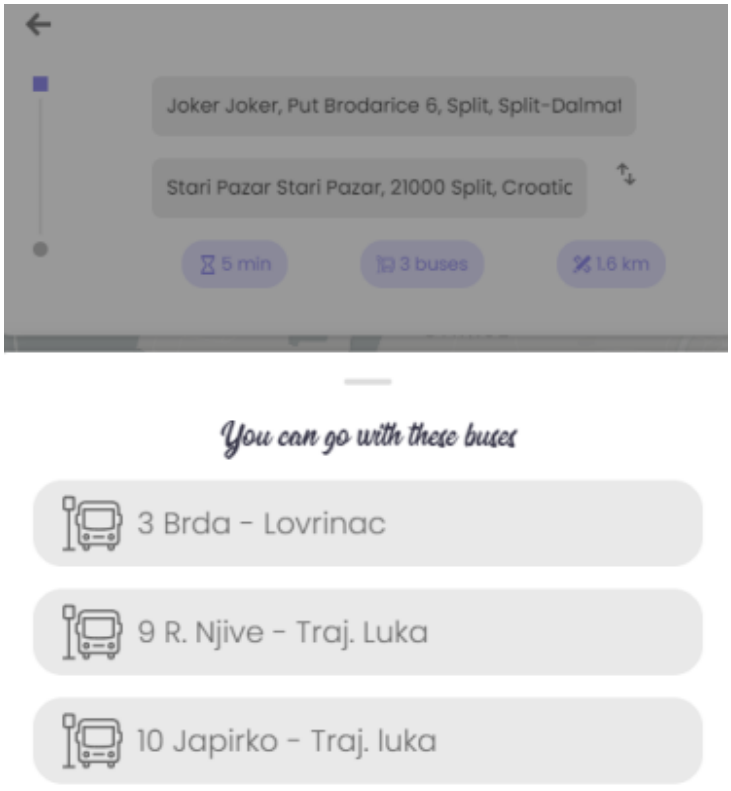
Lokacija autobusa nešto je složeniji dio posla kada govorimo o izradi baza. Svaka autobusna stanica ima naravno određenu poziciju, svoju geografsku dužinu i širinu. Upravo na taj način smo „ubacivali“ stanice na našu mapu.

Preko platfortme Google Maps bismo uzeli točne koordinate svake stanice, kojih ima više od 200, te bismo iste te koordinate unosili kao vrijednost „geopoint“ u svakoj autobusnoj stanici.

Slika 3.6. Unos lokacije autobusne stanice

Baza Autobusnih Linija

Svaka autobusna linija ima svoje ime, broj te nekakvu rutu kojom se svakodnevno služi. Isto tako, svaka linija ima i svoj raspored vožnji. Upravo od ovih komponenti smo i izradili našu bazu autobusnih linija.



Slika 3.7. Odabir autobusa kojim želimo ići

Ime autobusne linije naravno dolazi od rute kojom se autobus kreće, pa tako na primjer imamo autobus „Ravne Njive – Traj. Luka“, što bi značilo da taj autobus kreće s gradskih kotara ravnih njiva, prolazi gradom te dolazi do trajektne luke kao odredišta, na kojem se bus okreće i vozi sličnom rutom za nazad.

Broj autobusne linije služi kako bise građani lakše snalazili te kako bi već u daljini prepoznali autobus koji dolazi na njihovu stanicu. Broj autobusnih linija je izrađen po uzoru na već autobusima dane brojčane vrijednosti, pa tako na primjer, ranije spomenuti autobus „Ravne Njive – Traj. Luka“ pored svog naziva ima i broj 9. Sada ta autobusna linija izgleda ovako:

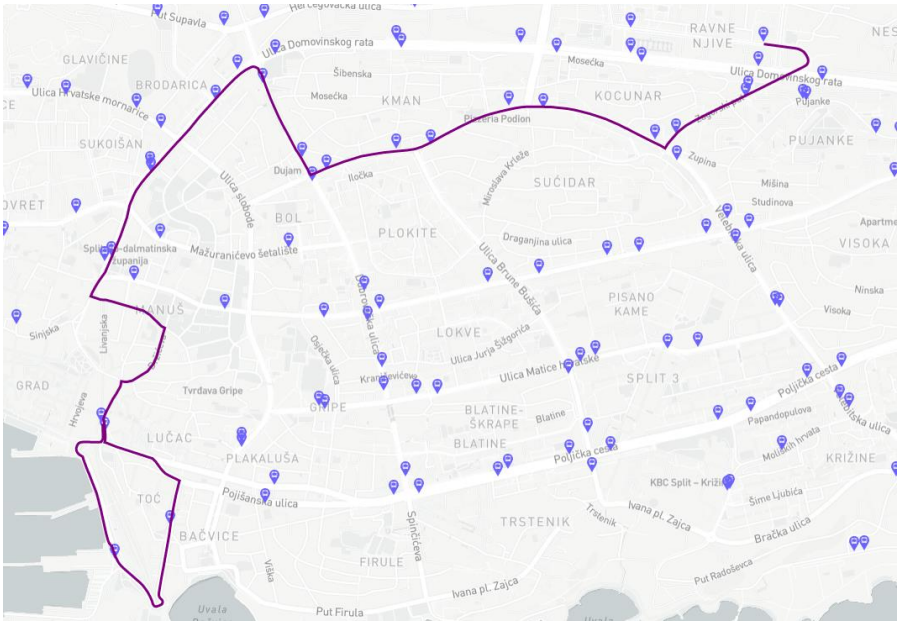
„9 Ravne Njive – Traj. Luka“.

Ruta svakog autobusa isplanirana je za svaki autobus te u bazi autobusnih linija, ispod svakog autobusa, nalazimo polje (string) autobusnih stanica kojima taj autobus prolazi.Na taj način, odabirom određenog autobusa na mapi naše aplikacije, prikazuje se jasna ruta kojom taj autobus prolazi, a upravo ta ruta nastaje povezivanjem stanica kojima taj autobus prolazi.

Firebase.js

```
const handleNormal = (potrebneStanice) => {
    const bus_stations = potrebneStanice.map(
      (stanica) => stanice.find(({ ime }) => stanica === ime)?.lokacija
    );
    const coords = bus_stations
      .map((station) => station.longitude + "," + station.latitude)
      .join(";");

    (async () => {
      const finalRoute = await matchRoute(coords);
      setApiCoords(finalRoute.geometry.coordinates)
      setDuzinaPuta(finalRoute.distance)
    })()
  }
```

Slika 3.8. Generiranje rute autobusne linije



Slika 3.9. Ruta autobusa „9 Ravne Njive – Traj. Luka“

O nama

Viktor Bilokapić

Entuzijastični zaljubljenik u kodove i utege.

Ekspert u UX/UI dizajnu. Iskusan u izgradnji full stack web aplikacija i softvera, CMS sustavi, SEO

~ Čovjek bez vizije za budućnost se uvijek vraća u prošlost.

Tarik Bešić

Iskusan u React JS, Node JS I kreiranju korisničkih sučelja. Ekspert u izradi web aplikacija i algoritama.

~ Poraz ne postoji, ili pobijediš ili naučiš.

Antonio Nikolić

Veliki fan programiranja na računalu, ali i bez njega.

Iskusan u izradi web aplikacija I baza podataka (SQL, NOSQL, React JS, Node JS)

~ Prilike su često prerušene u veliki napor, tako da ih većina ne prepozna.