Web Aplikacija Za Planiranje Punjenja Električnih Automobila

1. Opis sistema

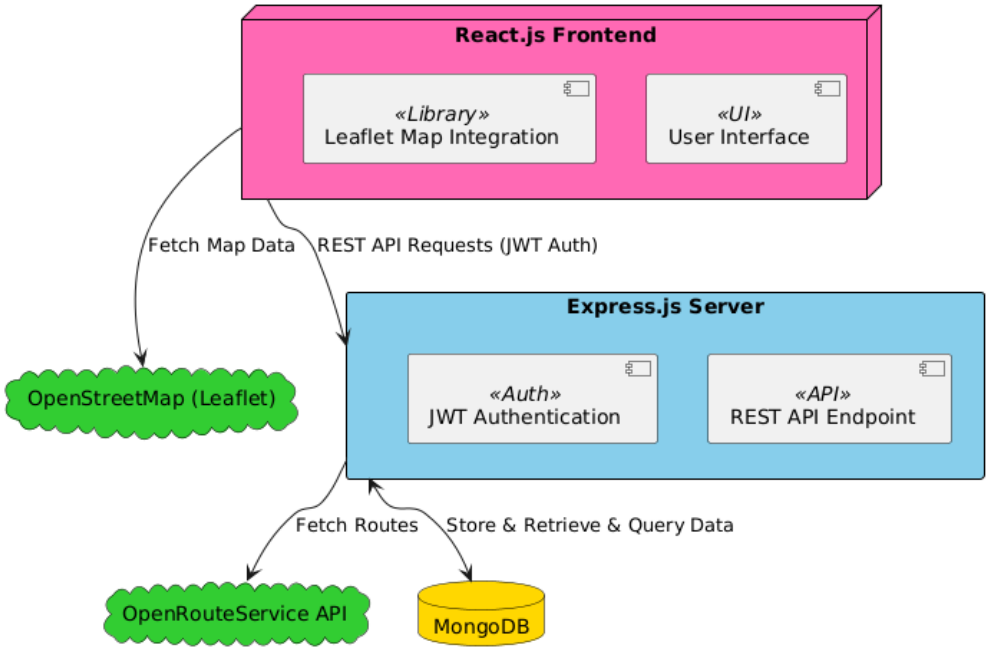
Aplikacija omogućava korisnicima planiranje punjenja električnih automobila tokom putovanja. Funkcionalnosti uključuju:

- Pregled dostupnih punjača
- Rezervaciju punjača
- Planiranje optimalne rute
- Praćenje statusa baterije vozila
- Administraciju korisnika i punjača

1.1 Ciljevi projekta

- Omogućiti efikasno planiranje punjenja.
- Pružiti administrativne alate za upravljanje mrežom punjača i korisnicima.

2. Arhitektura aplikacije



Slika 1 – Arhitektura aplikacije

Aplikacija koristi arhitekturu klijent-server:

- **Frontend**: React.js
- **Backend**: Node.js sa Express.js frameworkom
- **Baza podataka**: MongoDB
- **Eksterni servisi**: OpenRouteService za navigaciju, OpenStreetMap (Leaflet) za prikaz mape

Arhitektura obuhvata:

1. **Korisnički interfejs (Frontend)** – prikazuje interaktivnu mapu, korisničke forme i simulacije.
2. **Server (Backend)** – procesira zahteve, autentifikuje korisnike i komunicira sa bazom.
3. **Baza podataka** – skladišti informacije o korisnicima, vozilima i punjačima.

3. Korišćene tehnologije

Frontend:

- React.js
- Leaflet za mapu
- Axios za API komunikaciju

Backend:

- Node.js sa Express.js
- JWT za autentifikaciju
- Mongoose za rad sa MongoDB
- Axios za OpenRouteService API

Baza podataka:

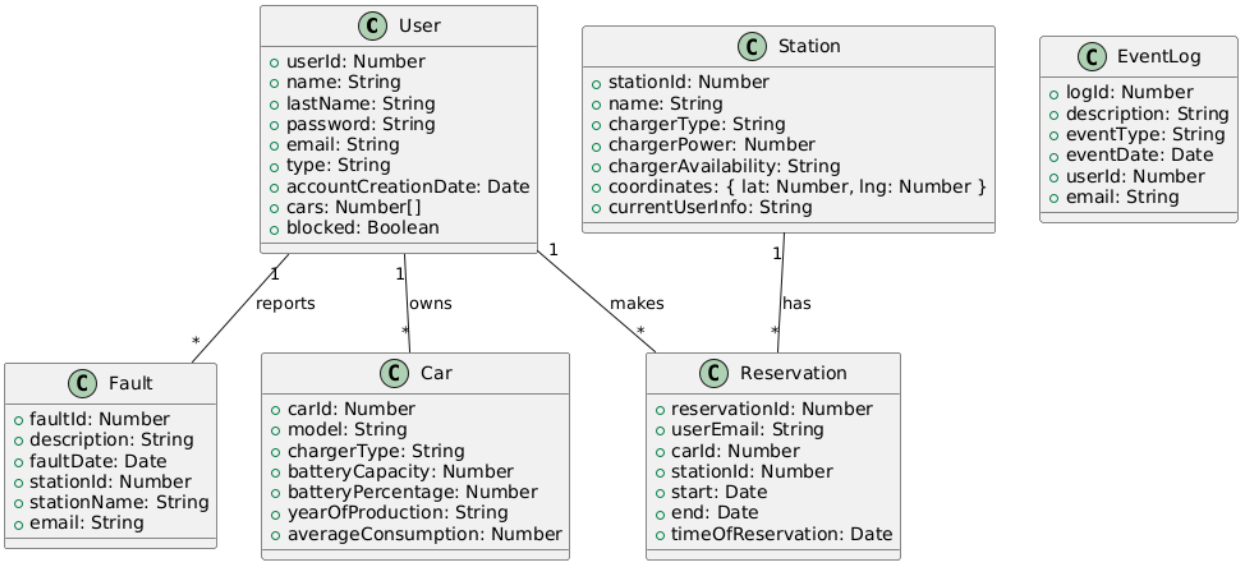
- MongoDB
- Mongoose biblioteka za modelovanje podataka

Eksterni servisi:

- OpenRouteService za generisanje ruta
- OpenStreetMap za prikaz mape

4. Struktura baze podataka

Aplikacija koristi sledeće MongoDB kolekcije:



Slika 2 – Modeli podataka

4.1 Korisnik (User)

- userId (number) - indentifikacioni broj
- name (string) - Ime korisnika
- lastName (string) - Prezime korisnika
- password (string) - Šifra korisnika

- email (string) - Email korisnika, koristi se za autentifikaciju
- type (string) - Tip korisnika (User ili Administrator)
- accountCreationDate (datetime) - Datum i vreme kreiranja naloga
- cars (Object array Car[]) - Lista korisnikovih automobila
- blocked (bool) - Da li je korisnik blokiran

4.2 Automobil (Car)

- carId (number) - indentifikacioni broj
- model (string) - Model automobila
- chargerType (string) - Vrsta utičnice koju automobil koristi (Type2, CHAdeMo, CCS...)
- batteryCapacity (number) - Kapacitet baterije u kWh
- batteryPercentage (number) - Trenutna napunjenost baterije u procentima
- yearOfProduction (string) - Godina proizvodnje automobila
- averageConsumption (number) - Prosečna potrošnja energije u kWh/100 km

4.3 Rezervacija (Reservation)

- reservationId (number) - indentifikacioni broj
- userEmail (string) - Email korisnika koji pravi rezervaciju (Strani ključ User.email)
- carId (number) - Automobil korišćen za rezervaciju (Strani ključ Car.carId)
- stationId (number) - ID stanice (Strani kljuc Station.stationId)
- start (datetime) - Datum i vreme početka rezervacije
- end (datetime) - Datum i vreme završetka rezervacije
- timeOfReservation (datetime) - Datum i vreme kreiranja rezervacije

4.4 Punjač (Station)

- stationId (number) - indentifikacioni broj
- name (string) - Naziv stanice
- chargerType (string) - Tip utičnice punjača (Type2, CHAdeMo, CCS)
- chargerPower (number) - Snaga punjača u kW
- chargerAvailability (string) - Status dostupnosti punjača (available, occupied, preparing, faulted)
- FaultReports (string array) - Reportaža kvarnosti od korisnika stanice
- coordinates (object) - Geografske koordinate stanice: lat (Number) - Geografska širina lng (Number) - Geografska dužina
- currentUserInfo (User object) - Informacije o korisniku trenutno priključenom na punjač

4.5 Log događaja (EventLog)

- logId (number) - Identifikacioni broj loga

- description (string) - Opis događaja
- eventType (string) - Tip događaja ("Rezervacija", "Greška", "Informacija")
- eventDate (datetime) - Datum i vreme događaja
- userId (number) - ID korisnika koji je generisao događaj (Strani ključ User.userId)
- email (string) - Email korisnika

4.6 Prijavljena greška (Fault)

- faultId (number) - Identifikacioni broj greške
- description (string) - Opis problema
- stationId (number) - Indentifikacioni broj stanice na kojoj je zabeležena greška
- stationName (string) - Ime stanice
- email (string) - Email korisnika koji je prijavio grešku

5. Korišćeni upiti

5.1 Stanice (Station)

- **Dobavljanje svih stanica:** Station.find()
- **Dobavljanje stanica za goste (samo ID, ime i koordinate):** Station.find().select("id name coordinates")
- **Dodavanje nove stanice:** Kreiranje i čuvanje nove stanice new Station({...}).save()
- **Brisanje stanice po ID-u:** Station.findOne({ stationId }) → Station.deleteOne({ stationId })
- **Ažuriranje dostupnosti stanice:** Station.findOneAndUpdate({ stationId }, { $set: { chargerAvailability: availability } }, { new: true })
- **Dobavljanje svih stanica sa brojem prijavljenih kvarova:**

```
const stations = await Station.find();
const faultCounts = await Fault.aggregate([
  { $group: { _id: "$stationId", count: { $sum: 1 } } }
]);
```

5.2 Korisnici (User)

- **Dobavljanje korisnika po emailu:** `User.findOne({ email })`
- **Dobavljanje korisnika po ID-u:** `User.findOne({ userId })`
- **Ažuriranje profila korisnika:** `User.findOne({ userId })` → izmena podataka → `user.save()`
- **Promena lozinke:** `User.findOne({ email })` → provera lozinke → ažuriranje → `user.save()`
- **Dobavljanje svih korisnika tipa 'User':** `User.find({ type: "User" })`
- **Blokiranje i deblokiranje korisnika:**

```
const user = await User.findOne({ userId });
user.blocked = !user.blocked;
await user.save();
```

- **Izmena korisničkih podataka:** `User.findOne({ userId })` → izmena podataka → `user.save()`
- **Brisanje korisnika po ID-u:** `User.findOneAndDelete({ userId })`

5.3 Kvarovi (Fault)

- **Prijavljivanje kvara:**
  - Provera postojanja stanice: `Station.findOne({ stationId })`
  - Provera postojanja korisnika: `User.findOne({ email })`
  - Dodavanje kvara: Kreiranje i čuvanje novog kvara `new Fault({...}).save()`
- **Dobavljanje svih kvarova za određenu stanicu:** `Fault.find({ stationId })`
- **Brisanje prijavljenog kvara:** `Fault.findOneAndDelete({ faultId })`

5.4 Automobili (Car)

- **Dodavanje novog automobila:**
  - Provera korisnika: `User.findOne({ userId })`
  - Kreiranje i čuvanje automobila: `new Car({...}).save()`
  - Dodavanje automobila korisniku: `user.cars.push(newCar.carId), user.save()`

5.5 Događaji (EventLog)

- **Dobavljanje svih logova:** `EventLog.find()`
- **Filtriranje logova prema tipu događaja, korisniku ili datumu:**

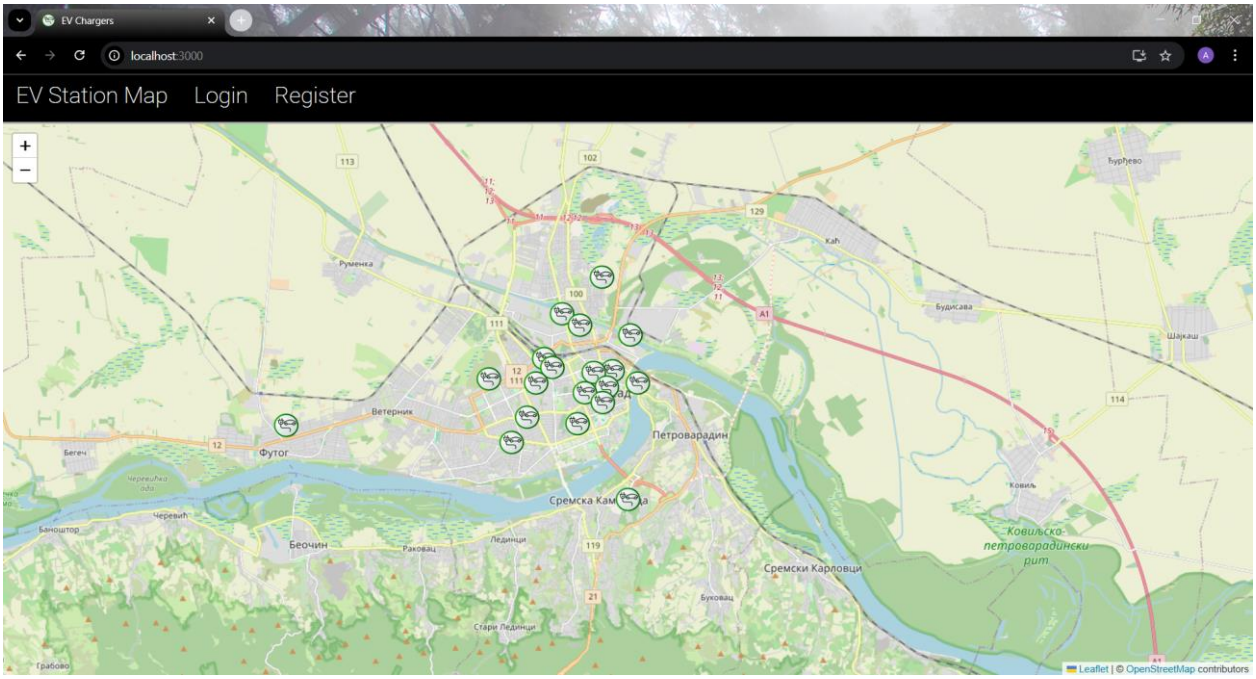
```
const filter = {};
if (eventType) filter.eventType = eventType;
if (userId) filter.userId = Number(userId);
if (email) filter.email = email;
if (startDate || endDate) {
  filter.eventDate = {};
  if (startDate) filter.eventDate.$gte = new Date(startDate);
  if (endDate) filter.eventDate.$lte = new Date(endDate);
}
const logs = await EventLog.find(filter);
```

- **Dodavanje logova prilikom izmena u sistemu:** `EventLog.create({ description, eventType, userId, email })`
- **Logovanje neuspešne prijave:** `EventLog.create({ description: "Invalid login attempt", eventType: "error", userId })`
- **Logovanje uspešne prijave:** `EventLog.create({ description: "User logged in successfully", eventType: "info", userId, email })`

6. UI Prikazi

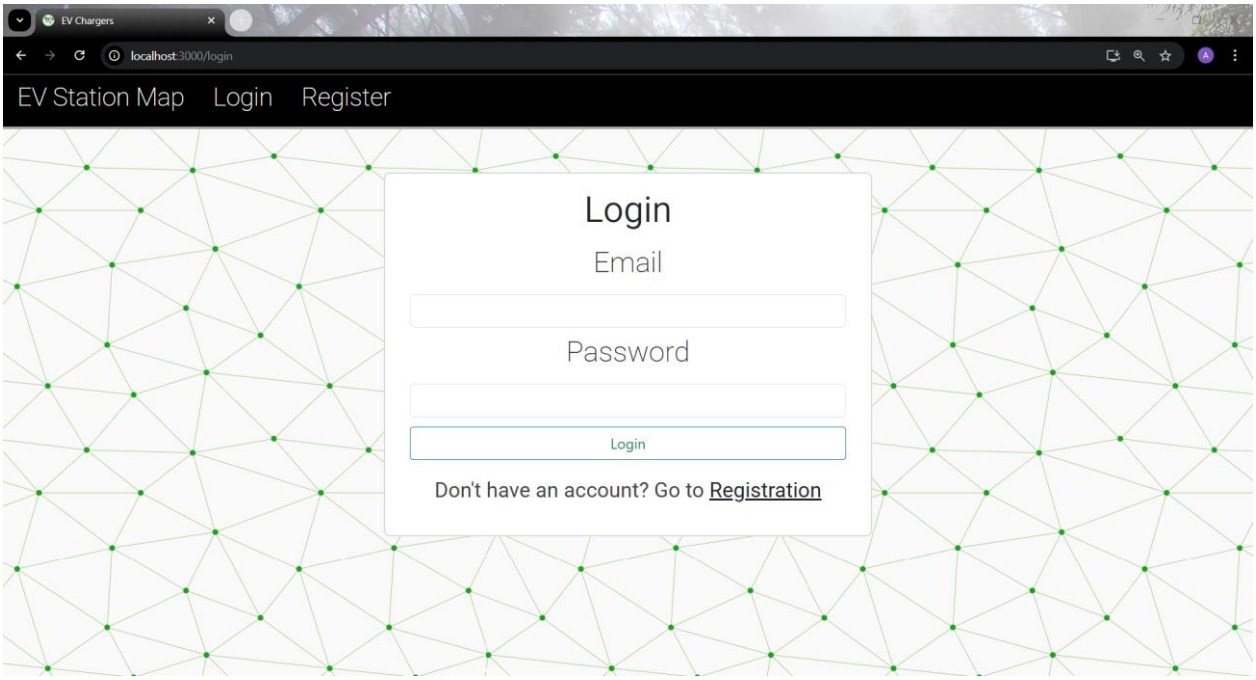
6.1 Neulogovani korisnik

Pregled mape sa punjačima (gost korisnik):

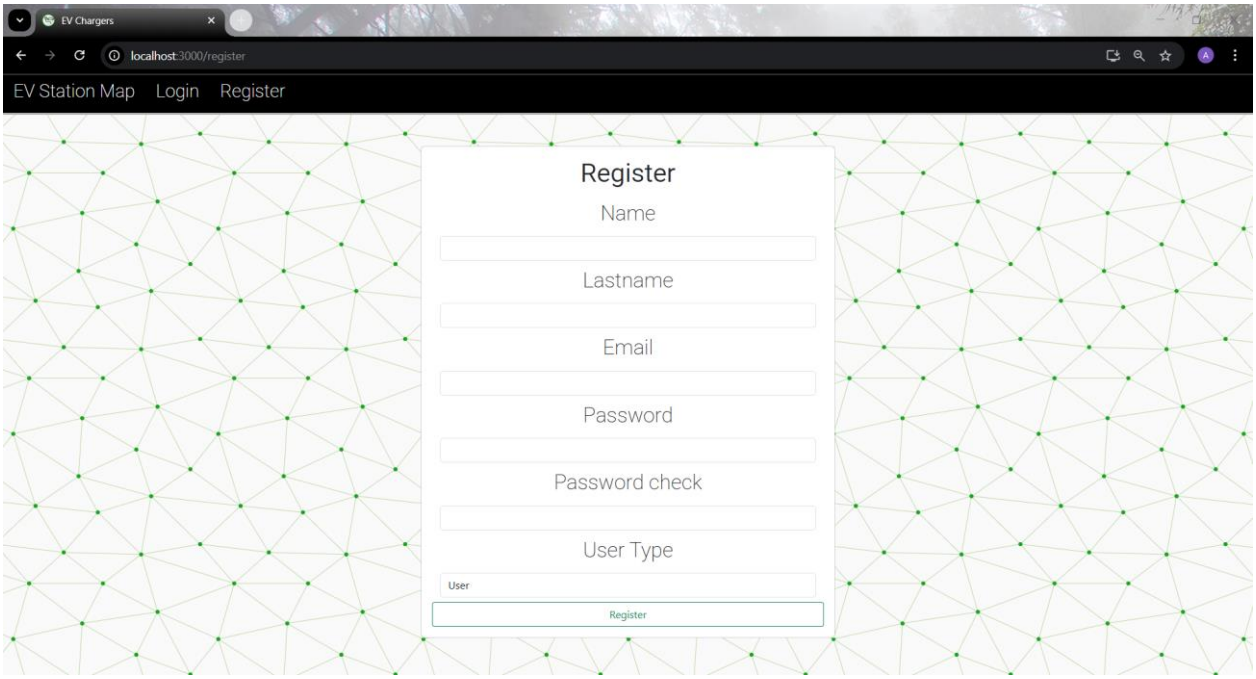


Slika 3 – EV Station Map (Guest)

Login i Register stranice:



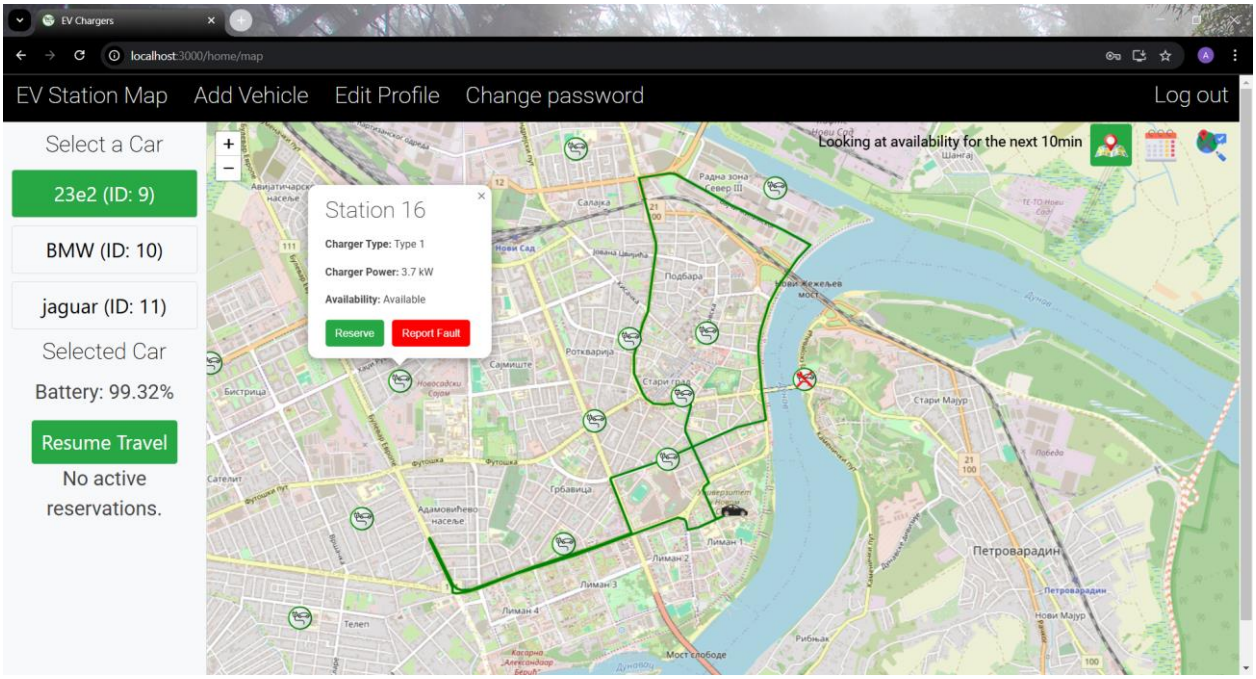
Slika 4 - Login



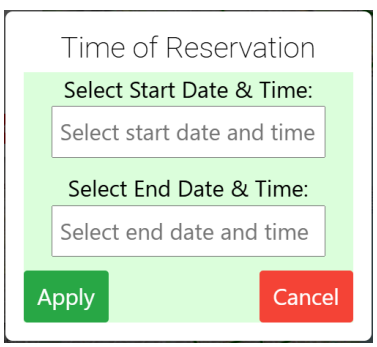
Slika 5 – Register

6.2 Ulogovani korisnik

Stranica za Rezervaciju Punjača:

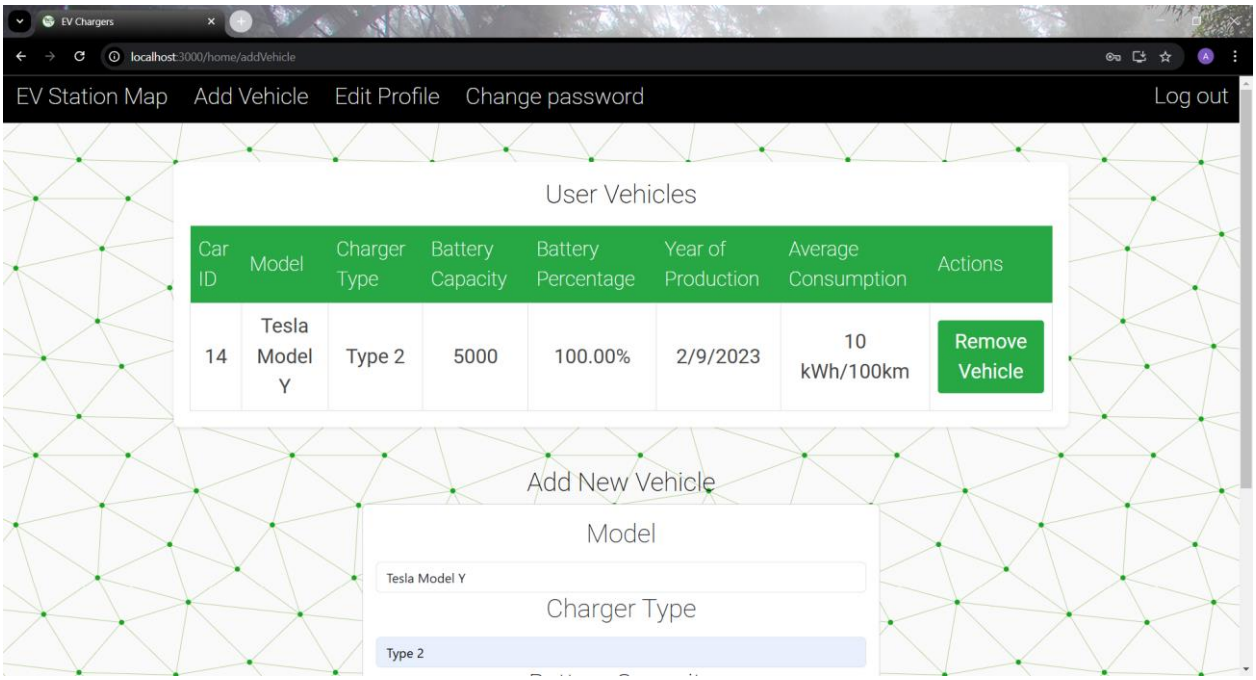


Slika 6 - Ev Station Map



Slika 7 – Reservation Date Modal

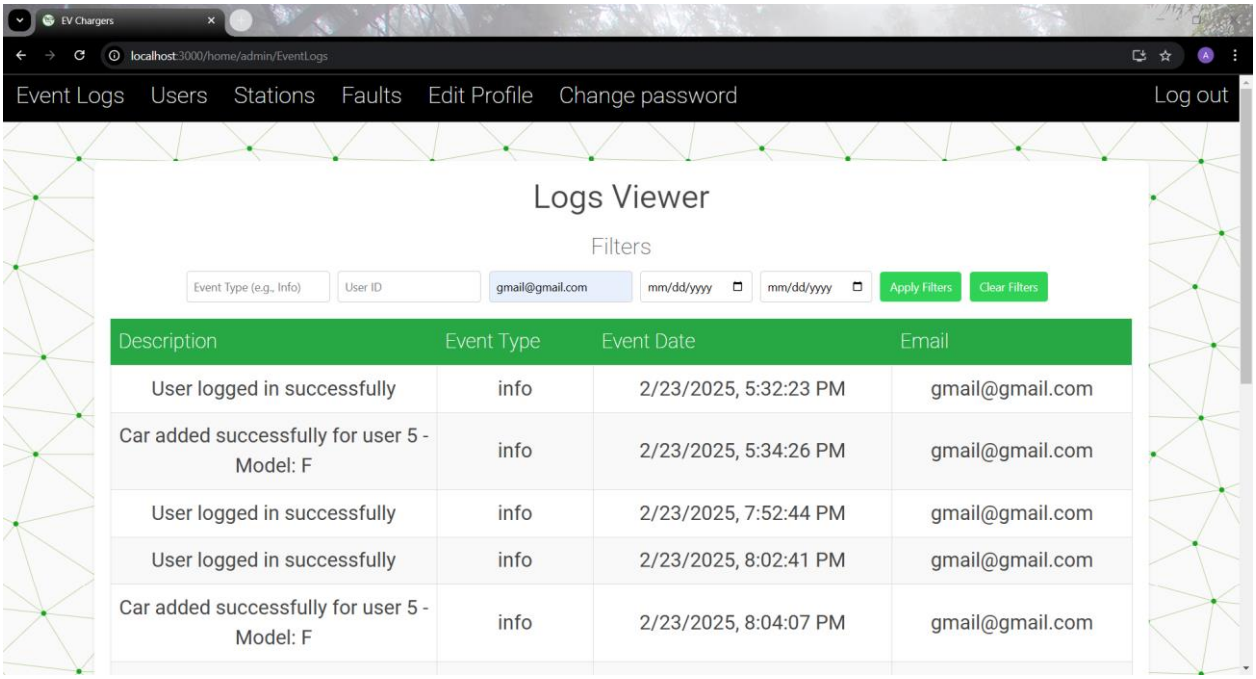
Upravljanje korisnikovim vozilima:



Slika 8 – Add Vehicle

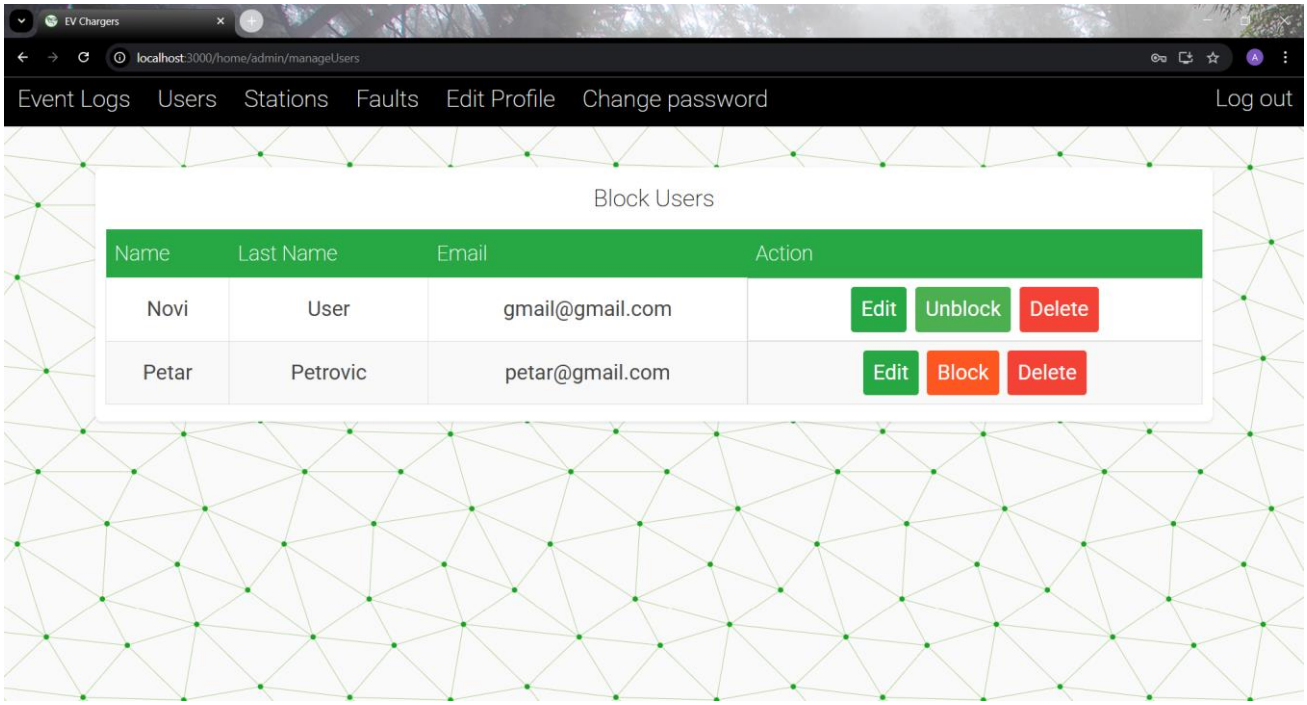
6.3 Administratorski korisnik

Administratorski pregled sistemskih zapisa:



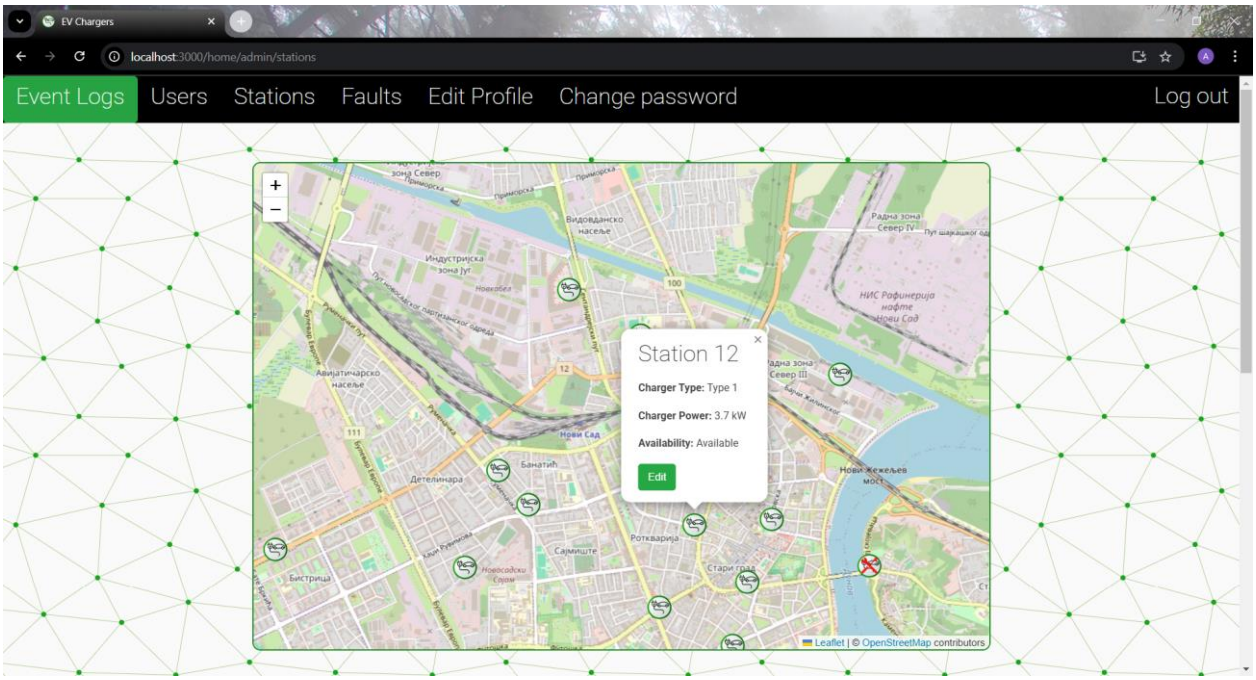
Slika 9 – Event Logs

Administratorski pregled korisnika:

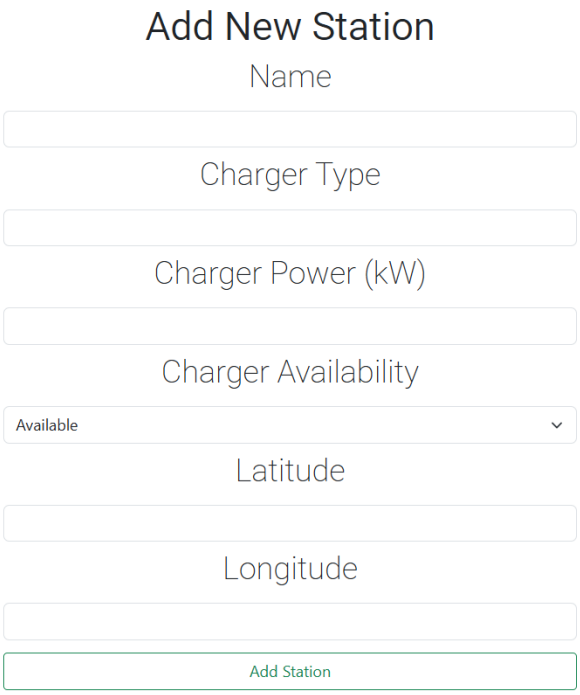


Slika 10 - Users

Administratorsko upravljanje i dodavanje stanica:

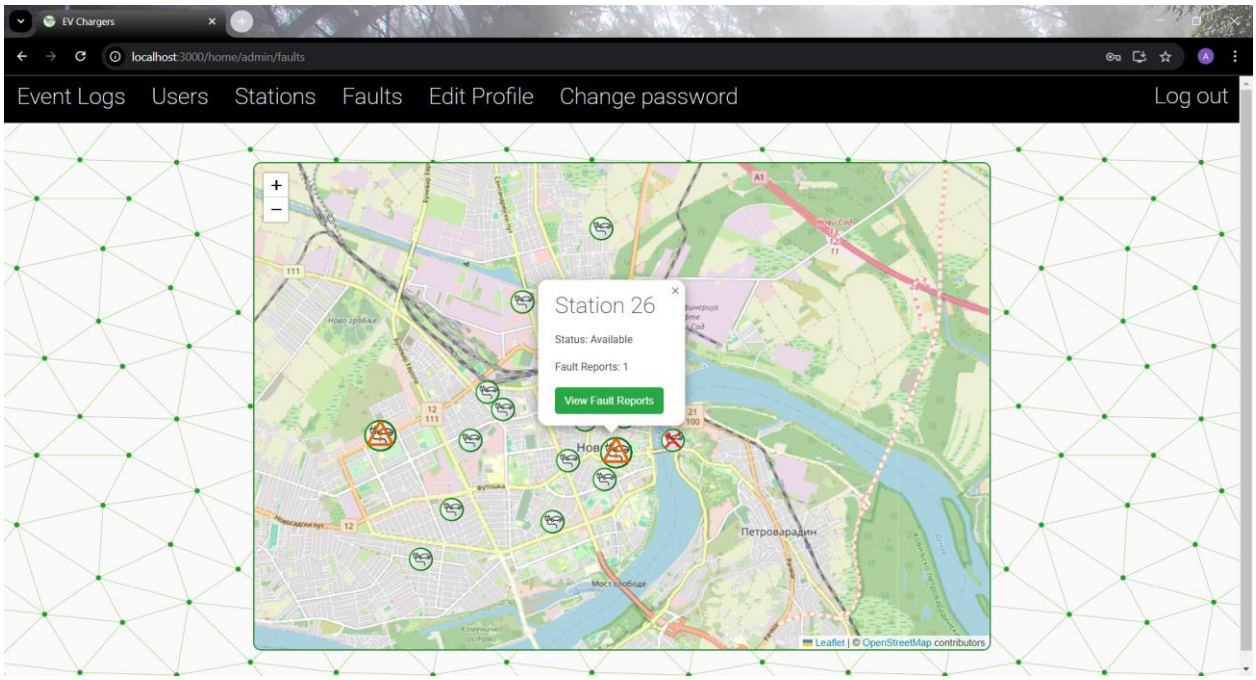


Slika 11 - Stations

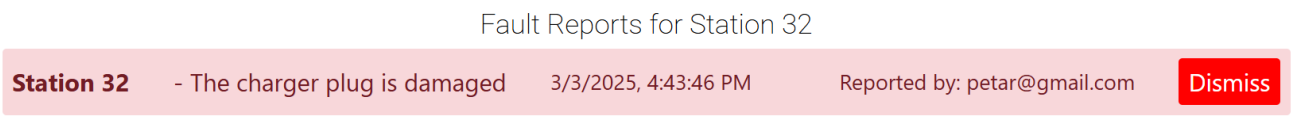


Slika 12 – Add Station Form

Administratorski pregled prijava kvarova na stanicama:



Slika 13 - Faults



Slika 14 – Fault Report

7. Funkcionalnosti Projekta

7.1 Funkcionalnsti pristupne administratoru

- **Upravljanje punjačima** - izmena dostupnosti, dodavanje novih, brisanje punjača.
- **Upravljanje kvarovima** – Pregled kvarova, uklanjanje kvarova.
- **Upravljanje registrovanim korisnicima** – Brisanje, blokiranje, izmena profila.
- **Evidencija događaja** - Administratorski prikaz sistema.

7.2 Funkcionalnsti pristupne ulogovanom korisniku

- **Izmena profila** - Ažuriranje podataka korisničkog naloga.
- **Pregled punjača** - Mapa sa prikazom svih dostupnih punjača.
- Prijava greške na punjaču.
- **Upravljanje automobilima** - Dodavanje i brisanje vozila.
- **Simulacija putovanja** - Praćenje lokacije korisnika.
- **Optimalna ruta** - OpenRouteService integracija za navigaciju.
- **Simulacija praženja baterije** - Procena potrošnje energije.
- **Simulacija rezervacije** - Pravila rezervacije punjača.
- **Simulacija punjenja vozila** - Automatsko prekidanje po završetku punjenja.
- **Najbliži punjač** - Geolokacijska usluga za pronalaženje punjača.

7.3 Funkcionalnosti dostupne gostu

- Pregled punjača na mapi bez interakcije.
- Registracija i logovanje.

8. Dokumentacija Koda

Istaknuti i uprošteni delovi koda koji prikazuju bitne funkcionalnosti aplikaije.

8.1 Rutiranje (React Router) i Autorizacija

Za navigaciju unutar aplikacije koristi se `react-router-dom`. Rute se definišu u `App.js`.

```
import { BrowserRouter as Router, Routes, Route } from "react-router-dom";
import PrivateRoute from "./routes/PrivateRoute";
import Login from "./pages/Login";
import Home from "./pages/Home";
import MapView from "./pages/MapView";

function App() {
```

```
 return (
  <Router>
   <Routes>
    <Route path="/login" element={<Login />} />
    <Route path="/home" element={<PrivateRoute allowedRoles={["User", "Admin"]}><Home
/></PrivateRoute>} />
    <Route path="/map" element={<PrivateRoute allowedRoles={["User"]}><MapView
/></PrivateRoute>} />
   </Routes>
  </Router>
 );
}

export default App;
```

`PrivateRoute` osigurava da samo prijavljeni korisnici sa odgovarajućom ulogom mogu pristupiti određenim stranicama.

Uloge korisnika se proveravaju pomoću **JWT tokena** unutar PrivateRoute.js.

```
import React from "react";
import { Navigate, useLocation } from "react-router-dom";
import { getUserFromLocalStorage } from "../Model/User";

export default function PrivateRoute({ children, allowedRoles }) {
 const location = useLocation();
 const decodedToken = JSON.parse(localStorage.getItem("token"));

 if (!decodedToken || !decodedToken.exp || decodedToken.exp * 1000 < Date.now()) {
  if (allowedRoles.includes("Guest")) return children;
  return <Navigate to="/login" />;
 }

 const role = decodedToken.user_role;
 const user = getUserFromLocalStorage();

 if (allowedRoles[0] === "Guest") return <Navigate to="/home/profile" />;
 if (role === "User" && user.blocked) return <Navigate to="/unauthorized" />;
 if (!allowedRoles.includes(role)) return <Navigate to="/unauthorized" />;

 return children;
}
```

PrivateRoute proverava da li postoji token. Ako token ne postoji ili je istekao, korisnik se preusmerava na stranicu za prijavu.
Uloga korisnika (`user_role`) se koristi za kontrolu pristupa određenim funkcionalnostima. Ako je korisnik blokiran ili nema odgovarajuću ulogu, onda se preusmerava na unauthorized stranicu.

8.2 Prikaz mape sa Leaflet i OpenRouteService

Za vizualizaciju rute i mapa koristi se **Leaflet**, dok se za generisanje ruta koristi **OpenRouteService API**.

```
import { MapContainer, TileLayer, Polyline } from "react-leaflet";
import "leaflet/dist/leaflet.css";
import { getRoute } from "../services/RouteService";
import { stationIcon } from "./Assets/MapIcons";
import { GetStationsGuest } from "../Services/StationService";


function MapView() {
 const [route, setRoute] = useState([]);
 const [stations, setStations] = useState([]);

 useEffect(() => {
  getRoute().then(data => setRoute(response.data));
  GetStationsGuest().then(data => setStations(response.data));

 }, []);

 return (
  <MapContainer center={[45.2671, 19.8335]} zoom={13}>
   <TileLayer url="https://{s}.tile.openstreetmap.org/{z}/{x}/{y}.png" />
{stations.map((station) => (
     <Marker
      key={station.stationId}
      position={[station.coordinates.lat, station.coordinates.lng]}
      icon={stationIcon}
     >
      <Popup>{station.name}</Popup>
     </Marker>
   ))}
   <Polyline positions={route} color="green" />
  </MapContainer>
 );
}
```

Leaflet mapa prikazuje OpenStreetMap pločice.
Polyline prikazuje rutu na osnovu podataka dobijenih sa backend servisa.
getRoute() funkcija komunicira sa backendom kako bi dobila koordinate rute.
GetStationsGuest() dobavlja stanice za prikaz preko **Marker** elementa

Kako server dobavlja rute sa OpenRouteService:

```
const response = await axios.post(
```

```
    `https://api.openrouteservice.org/v2/directions/driving-car`,
    { coordinates: [start, end], radiuses: [500, 500] },
    { headers: { Authorization: process.env.ApiKey } }
   );
    const encodedGeometry = response.data.routes[0].geometry;
    res.status(200).json({ route: encodedGeometry });
```

8.3 Komunikacija Frontend-Backend-DataBase

Za komunikaciju sa backendom koristi se **`axios`** biblioteka. Servisni fajl `UserService.js` sadrži funkciju za prijavu korisnika.

```
import axios from "axios";

export const LoginUser = async (data) => {
 try {
  const response = await axios.post(`${process.env.REACT_APP_API_URL}/auth/login `, data);
  localStorage.setItem("token", JSON.stringify(response.data.token));
  return response;
 } catch (error) {
  return error;
 }
};
```

Ako je prijava uspešna, JWT token se čuva u `localStorage`.
Ako prijava nije uspešna, vraća se odgovarajuća greška.

Provera postojanja tokena pri procesiranju zahteva na serveru.

```
const token = req.headers.authorization?.split(" ")[1];
```

Konekcija sa bazom:

Na backend strani, server se povezuje sa **MongoDB** bazom podataka pomoću mongoose biblioteke.

```
const mongoose = require("mongoose");

mongoose
 .connect("mongodb://localhost:27017/ev-charging", {
  useNewUrlParser: true,
  useUnifiedTopology: true,
 })
 .then(() => console.log("Connected to MongoDB"))
 .catch((err) => console.error("MongoDB connection error:", err));
```

9. Potencijalna Unapređenja

9.1 Bezbednost i optimizacija

- **Skladištenje tokena -** JWT token se trenutno čuva u `localStorage`, što nije najsigurnija opcija jer može biti podložna XSS napadima. Može se unarediti upotrebom `httpOnly` kolačića sa `Secure` i `SameSite` atributima.
- **Rate limiting & brute-force zaštita -** Bilo bi korisno ograničiti broj neuspješnih pokušaja prijave kako bi se sprečili brute-force napadi.
- **Bolji API pozivi -** Umesto direktnog korišćenja `fetch` ili `axios`, mogao bi implementirati **React Query** ili **SWR** za keširanje odgovora i poboljšanje performansi.
- **Optimizacija renderovanja -** Moguće je optimizovati komponente tako što će se koristiti `React.memo`, `useMemo`, i `useCallback` gde je potrebno.

9.2 Funkcionalnosti

- **Email obaveštenja -** Implementacija sistema koji korisnicima šalje email kada im rezervacija bude aktivirana ili završena. Možeš koristiti **Nodemailer** na backendu.
- Dodavanje **grafičkog prikaza statistike** (broj punjenja po danu, prosečno trajanje punjenja, najčešće korišćeni punjači).
- Korisnici mogu videti **istoriju svojih punjenja** sa podacima o lokaciji, vremenu punjenja, potrošenoj energiji i ceni
- Korisnici mogu ostaviti **ocenu (1-5 zvezdica) i komentar** za svaki punjač kako bi drugi korisnici znali koji su najpouzdaniji.