Fleet Management System - Tehnička dokumentacija

1. Uvod

Fleet Management System je web aplikacija dizajnirana za efikasno upravljanje voznim parkom kompanije PONG. Cilj sistema je eliminisati ručno vođenje evidencije vozila i putnih naloga te omogućiti ovlaštenim korisnicima jednostavan pristup svim relevantnim informacijama.

Aplikacija omogućava:

- Administraciju vozila (dodavanje, ažuriranje, brisanje).
- Evidenciju putnih naloga sa detaljima o vozaču, ruti i statusu putovanja.
- Generisanje izvještaja o korištenju vozila unutar definisanog vremenskog perioda.
- Različite nivoe pristupa korisnicima, pri čemu administratori imaju punu kontrolu nad podacima, dok obični korisnici mogu unositi putne naloge i pregledati podatke.

Sistem je izgrađen kao moderna web aplikacija, podijeljena na **frontend i backend**, koji međusobno komuniciraju putem REST API-ja.

Ključne prednosti sistema uključuju:

- Digitalizacija procesa upravljanja voznim parkom.
- Povećana tačnost i dostupnost podataka.
- Efikasno praćenje statusa vozila i putnih naloga.
- Smanjenje administrativnih grešaka i ručnog rada.

U nastavku dokumentacije detaljno su opisane korištene tehnologije, funkcionalnosti, arhitektura sistema, API endpointi i proces instalacije.

2. Tehnologije

Fleet Management System je razvijen koristeći moderni **JavaScript** ekosistem, baziran na **React.js** za frontend i **Node.js** s **Express.js** za backend. Podaci se pohranjuju u **PostgreSQL** bazu podataka, hostanu na **Neon.tech**. Aplikacija je distribuirana preko servisa **Vercel** (frontend) i **Railway** (backend).

2.1 Frontend

Frontend je razvijen kao **Single Page Application (SPA)** koristeći:

- **React.js** – JavaScript biblioteka za izgradnju interaktivnog korisničkog interfejsa.
- **Tailwind CSS** – Utility-first CSS framework za brzu i modernu stilizaciju.
- **React Router** – Upravljanje rutiranjem unutar aplikacije.
- **Axios** – HTTP klijent za slanje zahtjeva prema backend API-ju.

Frontend aplikacija se hostuje na **Vercel**, što omogućava **automatski deployment** iz GitHub repozitorija.

2.2 Backend

Backend aplikacija je bazirana na **Node.js** i **Express.js**, uz sljedeće tehnologije:

- **Express.js** – Minimalistički web framework za API razvoj.
- **PostgreSQL** – Relacijska baza podataka za pohranu podataka.
- **JSON Web Token (JWT)** – Koristi se za autentifikaciju i autorizaciju korisnika.
- **CORS middleware** – Omogućava siguran pristup API-ju s različitih domena.
- **Railway** – Cloud hosting za backend servis.

2.3 Baza podataka

Sistem koristi **PostgreSQL**, moćnu relacijsku bazu podataka, hostovanu na **Neon.tech**, koji omogućava: automatski backup i skalabilnost, visoku dostupnost baze podataka i sigurnu i brzu obradu upita.

Baza sadrži tri ključne tabele:

- **Vehicles** – Sadrži podatke o vozilima.
- **Trips** – Evidencija putnih naloga.
- **User** – Podaci o korisniku (admin).

2.4 Hosting i Deployment

- **Frontend (React.js) → Vercel** – Automatski deployment direktno iz GitHub-a. Link
- **Backend (Node.js, Express.js) → Railway** – Cloud hosting backend aplikacije.
- **Baza podataka (PostgreSQL) → Neon.tech** – Cloud baza podataka s automatskim skaliranjem.

Ova tehnologijska arhitektura omogućava **visoke performanse, fleksibilnost i jednostavno održavanje sistema**.

3. Funkcionalnosti

Fleet Management System omogućava administraciju voznog parka, evidenciju putnih naloga i generisanje izvještaja. Sistem koristi validaciju podataka kako bi osigurao tačnost unosa i spriječio nepravilnosti.

3.1 Autentifikacija i autorizacija

Sistem koristi JWT autentifikaciju kako bi osigurao siguran pristup API-ju.

Korisnici imaju različite nivoe pristupa:

- Administrator može upravljati vozilima i putnim nalozima.
- Obični korisnik može kreirati i pregledati putne naloge, ali ne može uređivati vozila.

Prijava korisnika vrši se putem login forme, a sistem provjerava ispravnost kredencijala prije dodjeljivanja pristupa.

3.2 Upravljanje vozilima

Administratori imaju potpunu kontrolu nad podacima o vozilima i mogu:

- Dodavati nova vozila unosom podataka kao što su marka, model, broj šasije, snaga motora i vrsta goriva.
- Ažurirati postojeća vozila.
- Brisati vozila ako nemaju aktivne putne naloge.
- Pregledati listu svih vozila sa svim relevantnim podacima.

Obični korisnici mogu samo pregledati vozila, bez mogućnosti izmjena.

Validacija prilikom unosa vozila:

- Provjera da broj šasije i broj motora nisu već uneseni u sistem.
- Godina proizvodnje mora biti između 1900 i trenutne godine.
- Snaga motora (kW i HP) mora biti veća od 0.

3.3 Upravljanje putnim nalozima

Putni nalozi predstavljaju evidenciju putovanja svakog vozila.

Obični korisnici mogu:

- Dodavati nove putne naloge.
- Pregledati listu svojih putnih naloga.

Administratori dodatno mogu:

- Ažurirati status putnog naloga.
- Brisati putne naloge ako nisu u statusu "Završen".

Statusi putnog naloga:

- Evidentiran – Nalog je kreiran, ali još nije potvrđen.
- Potvrđen – Vozilo je preuzeto i nalog je aktivan.
- Odbijen – Nalog nije odobren.
- Završeno – Putovanje je završeno.

Validacija prilikom unosa putnog naloga:

- Sva polja su obavezna.
- Početak perioda mora biti prije kraja perioda.

- Broj putnika mora biti veći od 0.
- Ime vozača smije sadržavati samo slova i razmake.
- Polazna i dolazna lokacija ne mogu biti iste.
- Mora se odabrati vozilo iz liste dostupnih vozila.
- Nije moguće kreirati novi putni nalog za vozilo koje je već rezervisano (ako postoji aktivan nalog u statusu "Evidentiran" ili "Potvrđen").

3.4 Generisanje izvještaja o korištenju vozila

Sistem omogućava pregled iskorištenosti vozila u određenom vremenskom periodu.

Izvještaj se generiše u formi tabele koja prikazuje relevantne podatke o korištenju vozila. Korisnici mogu preuzeti izvještaj u PDF formatu za arhiviranje ili dalju analizu.

4. Arhitektura sistema

Fleet Management System je baziran na modernoj arhitekturi koja obuhvata frontend, backend i bazu podataka. Sistem koristi REST API za komunikaciju između frontend i backend slojeva, čime se omogućava modularnost i lahko održavanje.

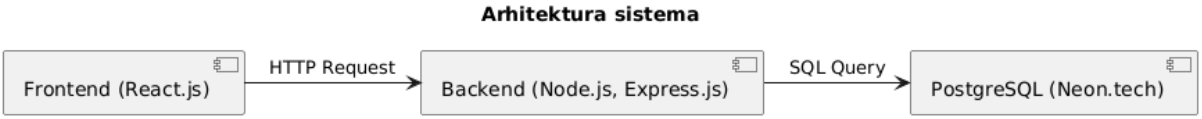
4.1 Opis arhitekture

Arhitektura sistema je podijeljena u tri osnovna sloja:

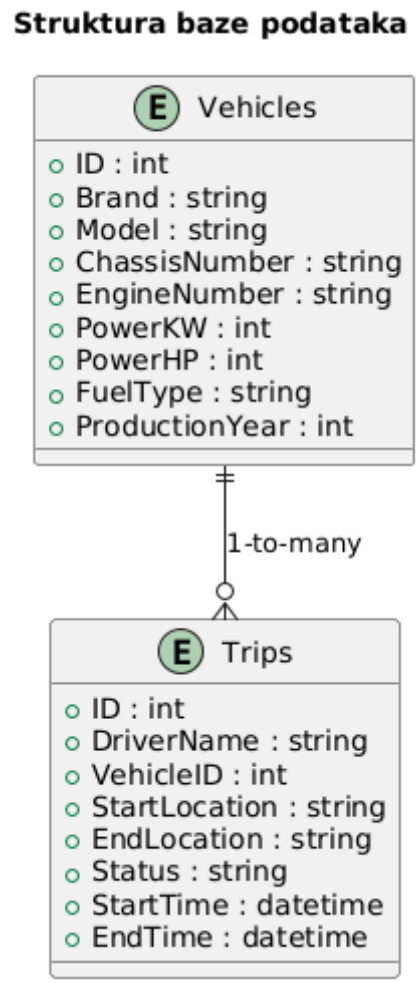
1. **Frontend (React.js, Tailwind CSS)**
    - Implementiran korištenjem React.js za dinamičko renderovanje korisničkog interfejsa.
    - Koristi Tailwind CSS za brzu i jednostavnu stilizaciju elemenata.
    - Komunicira sa backend-om putem REST API poziva.
    - Deployan na Vercel platformi, omogućavajući brzu i sigurnu isporuku korisnicima.
2. **Backend (Node.js, Express.js)**
    - Backend je razvijen u Node.js frameworku sa Express.js za upravljanje HTTP zahtjevima.
    - Implementira autentifikaciju i autorizaciju koristeći JWT (JSON Web Token).
    - Održava REST API koji omogućava CRUD operacije nad vozilima i putnim nalozima.
    - Deployan na Railway platformi radi skalabilnosti i lahkoće održavanja.
3. **Baza podataka (PostgreSQL – Neon.tech)**
    - PostgreSQL baza podataka hostana na Neon.tech.
    - Sadrži dvije glavne tabele: vehicles (za evidenciju vozila) i trips (za putne naloge).

- Koristi relacione odnose kako bi omogućila povezivanje podataka o vozilima i putnim nalozima.

4.2 Dijagram arhitekture



4.3 ERD dijagram



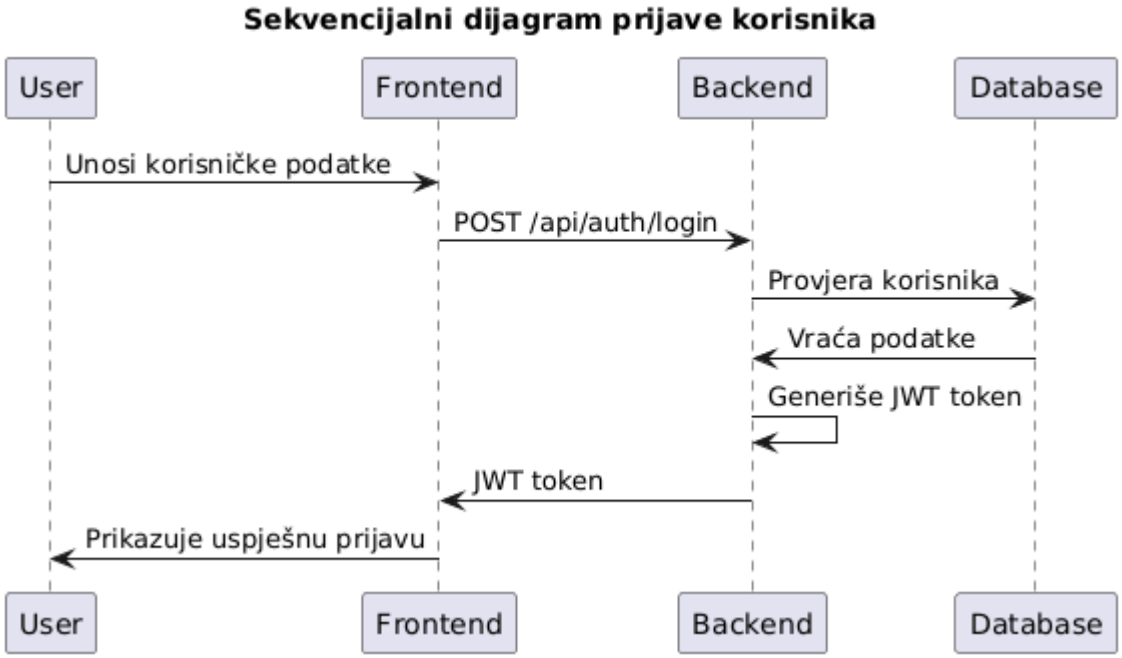
4.4 Opis komunikacije između slojeva

- **Frontend** šalje HTTP zahtjeve prema backendu koristeći REST API.
- **Backend** obrađuje zahtjeve, autentificira korisnike putem JWT tokena, te komunicira s bazom podataka radi izvršavanja CRUD operacija.
- **Baza podataka** čuva sve informacije o vozilima i putnim nalozima te omogućava dohvaćanje i manipulaciju podacima.

4.5 Sigurnosne mjere

- JWT autentifikacija osigurava pristup samo ovlaštenim korisnicima.

- Role-based Access Control (RBAC) diferencira privilegije između administratora i običnih korisnika.
- PostgreSQL koristi enkripciju podataka i sigurnosne mjere za zaštitu podataka.



Ova arhitektura omogućava efikasno i skalabilno rješenje za upravljanje voznim parkom, uz jednostavno održavanje i mogućnost budućeg proširenja funkcionalnosti.

5. Instalacija i pokretanje

5.1 Kloniranje projekta

Za preuzimanje i pokretanje projekta, koristite sljedeće komande:

git clone https://github.com/sejooo1/fleet-management.git

cd fleet-management

5.2 Pokretanje backenda

cd backend

npm install

node index.js

5.3 Pokretanje frontenda

cd frontend

npm install

npm start

6. API Endpointi

6.1 Autentifikacija

Metoda	Ruta	Opis
POST	/api/auth/login	Prijava korisnika

6.2 Vozila

Metoda	Ruta	Opis
GET	/api/vehicles	Dohvatanje svih vozila
POST	/api/vehicles	Dodavanje vozila (admin)
PUT	/api/vehicles/:id	Ažuriranje vozila (admin)
DELETE	/api/vehicles/:id	Brisanje vozila (admin)

6.3 Putni nalozi

Metoda	Ruta	Opis
GET	/api/trips	Dohvatanje svih putnih naloga
POST	/api/trips	Dodavanje putnih naloga
PUT	/api/trips/:id/status	Ažuriranje statusa naloga (admin)
DELETE	/api/trips/:id	Brisanje putnog naloga (admin)

6.4 Kreiranje izvještaja

Metoda	Ruta	Opis
GET	/api/reports/usage	Kreiranje izvještaja

7. Zaključak

Fleet Management System omogućava efikasno upravljanje voznim parkom i nudi jednostavno korisničko iskustvo putem modernog frontend dizajna i pouzdane backend logike. Daljnja poboljšanja mogu uključivati dodavanje notifikacija i bolju analitiku izvještaja.