

SVEUČILIŠTE U ZAGREBU  
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

DIPLOMSKI RAD br. 1006

**SUSTAV ZA PRAĆENJE I UPRAVLJANJE ZAUZETOŠĆU  
PARKIRNIH MJESTA U STVARNOM VREMENU**

Tin Prvčić

Zagreb, srpanj 2025.

SVEUČILIŠTE U ZAGREBU  
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

DIPLOMSKI RAD br. 1006

**SUSTAV ZA PRAĆENJE I UPRAVLJANJE ZAUZETOŠĆU  
PARKIRNIH MJESTA U STVARNOM VREMENU**

Tin Prvčić

Zagreb, srpanj 2025.

## **DIPLOMSKI ZADATAK br. 1006**

Pristupnik: **Tin Prvčić (0036535899)**

Studij: Računarstvo

Profil: Programsko inženjerstvo i informacijski sustavi

Mentor: izv. prof. dr. sc. Alan Jović

Zadatak: **Sustav za praćenje i upravljanje zauzetošću parkirnih mjesta u stvarnom vremenu**

### Opis zadatka:

Današnji sustavi za praćenje parkinga sve su češće temeljeni na pametnim rješenjima koja kombiniraju senzore za detekciju zauzetosti, vizualnu signalizaciju i udaljeni prikaz podataka u stvarnom vremenu. Kroz ovaj diplomski rad potrebno je osmisлити i implementirati prototipni sustav za praćenje i upravljanje zauzetošću parkirališnih mjesta. Sustav treba omogućiti praćenje zauzetosti pojedinih mjesta kroz mobilnu aplikaciju, u stvarnom vremenu. Dodatno, sustav kroz aplikaciju autoriziranim korisnicima treba omogućiti i upravljanje statusom zauzeća cijelog parkirališta. Parkiralište treba, uz pristup putem mobilne aplikacije, imati i indikator zauzeća ostvaren kroz svjetlosni indikator, kao i dodatan LED zaslon koji pokazuje ukupan broj slobodnih mjesta. Uz to, svaki senzor treba imati svoj svjetlosni indikator zauzeća. Logički raspored senzora i podatke o senzorima (lokaciju, naziv pridijeljen senzoru) trebaju se moći uređivati kroz mobilnu aplikaciju. Programska podrška za senzore treba biti ostvarena neovisno o poslužitelju, a na način da s poslužiteljem komunicira putem REST sučelja. U teoretskom dijelu rada, potrebno je opisati metode i tehnologije koje se koriste u sličnim rješenjima u praksi te ih kvalitativno usporediti s ostvarenim sustavom.

Rok za predaju rada: 4. srpnja 2025.



# Sadržaj

|                                                  |           |
|--------------------------------------------------|-----------|
| <b>1. Uvod</b>                                   | <b>3</b>  |
| <b>2. Tehnologije detekcije vozila</b>           | <b>4</b>  |
| 2.1. Ultrazvučni senzori                         | 4         |
| 2.2. Infracrveni senzori                         | 5         |
| 2.3. Magnetni senzori                            | 5         |
| 2.4. Kamere                                      | 5         |
| 2.5. Primjena umjetne inteligencije              | 6         |
| 2.6. Zaključak                                   | 7         |
| <b>3. Opis i arhitektura razvijenog rješenja</b> | <b>8</b>  |
| <b>4. Sklopovsko rješenje</b>                    | <b>11</b> |
| 4.1. Raspberry Pi 3B+                            | 11        |
| 4.2. Senzori udaljenosti – HC-SR04               | 12        |
| 4.3. Svjetlosni indikatori                       | 13        |
| 4.4. LED zaslon – MAX7219 4-in-1                 | 13        |
| 4.5. Eksperimentalna pločica                     | 14        |
| 4.6. Cjelokupni fizički sloj sustava             | 14        |
| <b>5. Poslužiteljski dio sustava</b>             | <b>16</b> |
| 5.1. Korištene i razmatrane tehnologije          | 16        |
| 5.1.1. NestJS                                    | 16        |
| 5.1.2. TypeORM                                   | 17        |
| 5.1.3. SQLite                                    | 17        |
| 5.1.4. PostgreSQL                                | 17        |

|                                                                                 |           |
|---------------------------------------------------------------------------------|-----------|
| 5.2. Struktura i organizacija aplikacije . . . . .                              | 18        |
| 5.3. Autentifikacija i autorizacija . . . . .                                   | 18        |
| 5.4. Komunikacija s ostatkom sustava . . . . .                                  | 19        |
| 5.5. Baza podataka . . . . .                                                    | 19        |
| 5.6. Višekorisnički rad i sigurnost . . . . .                                   | 20        |
| 5.7. Razvoj i ispitivanje . . . . .                                             | 20        |
| <b>6. Mobilna aplikacija . . . . .</b>                                          | <b>21</b> |
| 6.1. Korištene tehnologije . . . . .                                            | 21        |
| 6.1.1. React Native . . . . .                                                   | 21        |
| 6.1.2. Expo . . . . .                                                           | 22        |
| 6.1.3. SWR . . . . .                                                            | 22        |
| 6.1.4. Expo Router . . . . .                                                    | 22        |
| 6.1.5. expo-secure-store . . . . .                                              | 22        |
| 6.2. Autentifikacija i autorizacija . . . . .                                   | 23        |
| 6.3. Struktura i organizacija aplikacije . . . . .                              | 23        |
| <b>7. Budući razvoj . . . . .</b>                                               | <b>27</b> |
| 7.1. Podrška za PostgreSQL kao alternativu SQLiteu . . . . .                    | 27        |
| 7.2. ZigBee komunikacija za lokalne sustave . . . . .                           | 27        |
| 7.3. Dodatne mogućnosti uređivanja senzora i parkirališta iz mobilne aplikacije | 28        |
| 7.4. Upravljanje korisnicima unutar mobilne aplikacije . . . . .                | 28        |
| 7.5. Komunikacija WebSocketom za mobilnu aplikaciju i LED zaslon . . . . .      | 29        |
| 7.6. Vizualna karta parkirališta . . . . .                                      | 29        |
| 7.7. Povezivanje s vanjskim sustavima za naplatu parkinga . . . . .             | 30        |
| <b>8. Zaključak . . . . .</b>                                                   | <b>31</b> |
| <b>Literatura . . . . .</b>                                                     | <b>33</b> |
| <b>Sažetak . . . . .</b>                                                        | <b>35</b> |
| <b>Abstract . . . . .</b>                                                       | <b>36</b> |

# 1. Uvod

Razvoj velikih gradova i povećanje broja vozila na cestama svakodnevno rezultiraju sve većim izazovima u domeni upravljanja parkiralištima. Vozači, koji se sve više oslanjaju na svoja vozila, svaki dan gube vrijeme tražeći slobodno mjesto, što dodatno opterećuje promet, a samim time povećava potrošnju goriva (i ubrzava habanje vozila), pa i negativno utječe na njihov ekološki otisak. Zbog toga se svakodnevno razvijaju pametni sustavi za upravljanje parkiralištima koji spajaju suvremene tehnologije – senzore, bežičnu komunikaciju i informacijske sustave – i tako pomažu korisnicima u lakšem pronalasku parkirnih mjesta za svoja vozila, a samim time rasterećuju promet i smanjuju ekološki utjecaj prometa na velike gradove.

Ovaj rad bavi se razvojem i implementacijom sustava za praćenje i upravljanje zauzetošću parkirališnih mjesta. Sustav koristi ultrazvučne senzore za provjeru zauzetosti svakog parkirnog mjesta, svjetlosne indikatore za lokalni prikaz stanja nekog parkirnog mjesta, kao i LED zaslon i dodatan svjetlosni indikator za prikaz ukupnog broja slobodnih mjesta i stanja cijelog parkirališta. Osim lokalnog dijela sustava, razvijen je i REST poslužitelj koji služi kao centralni izvor podataka o parkiralištu, a uz njega i mobilna aplikacija. Koristeći mobilnu aplikaciju, korisnici mogu u stvarnom vremenu provjeriti stanje parkirališta, dok ovlašteni korisnici mogu upravljati i konfiguracijom senzora i stanjem cjelokupnog parkirališta.

Uz to, rad analizira tehnologije koje se koriste u postojećim rješenjima i uspoređuje ih s rješenjima primijenjenim u ovom implementiranom sustavu. Cilj rada je demonstrirati primjenu jednostavne i cijenom pristupačne tehnologije za izgradnju skalabilnog sustava za pametno upravljanje parkiralištem.

## **2. Tehnologije detekcije vozila**

Kod projektiranja sustava za praćenje zauzetosti parkirališnih mjesta, najbitnija odluka je izbor tehnologije senzora. Svaka tehnologija i svaki pristup imaju svoje prednosti i mane, pa ta odluka ne utječe samo na točnost koju sustav postiže, nego i na cijenu, pouzdanost i kompleksnost održavanja sustava, ali i na mogućnosti integracije sustava u već postojeću infrastrukturu.

Kroz ovo poglavlje analizirat će se najčešće vrste senzora koje se koriste u sustavima pametnog parkiranja: ultrazvučni senzori, infracrveni senzori (IR), magnetni senzori i kamere. Na kraju poglavlja dat će se i osvrt na korištenje umjetne inteligencije radi dodatnog poboljšanja točnosti detekcije koju sustav postiže.

### **2.1. Ultrazvučni senzori**

Ultrazvučni senzori koriste visokofrekventne zvučne valove kojima mjere udaljenost do najbližeg objekta ispred senzora [1]. Najčešće su montirani na strop garaže iznad svakog parkirnog mjesta, pa stalno mjere prisutnost vozila ispod sebe.

Glavna prednost ovakvih senzora je jednostavnost implementacije i niska cijena u odnosu na ostale tehnologije. Zbog toga su ultrazvučni senzori najčešći odabir, osobito u zatvorenim parkiralištima gdje su uvjeti kontrolirani pa konzistentno postižu veliku preciznost.

No, ova vrsta senzora nije savršena – površine od kojih se svjetlo neobično odbija (poput lima automobila) i razlike u visini različitih vozila povećavaju vjerojatnost pojave netočnih mjerenja, u slučaju da se vozilo ispod senzora ne nalazi na sasvim ispravnom mjestu.

## **2.2. Infracrveni senzori**

Infracrveni senzori detektiraju promjene u razini infracrvenog zračenja. Postoje dvije vrste: aktivni (emitiraju infracrveno zračenje i detektiraju refleksiju tog zračenja) i pasivni (detektiraju samo promjene u infracrvenom zračenju objekta u koji je senzor uperen) [2].

Glavna uporaba infracrvenih senzora u kontekstu prometa je detekcija vozila u pokretu, no postoje i implementacije koje omogućuju detekciju prisutnosti vozila na određenom parkirnom mjestu. Točnost takvih senzora jako ovisi o vanjskim uvjetima: temperaturi, sunčevoj svjetlosti i vremenskim prilikama. To znači da imaju dosta ograničenu primjenu na otvorenim parkiralištima.

S druge strane, najveća prednost infracrvenih senzora je niska potrošnja energije i jednostavnost implementacije, ali i to dolazi nauštrb pouzdanosti u nepovoljnim uvjetima.

## **2.3. Magnetni senzori**

Magnetni senzori detektiraju promjene u magnetskom polju i pretvaraju ih u električni signal [3]. Ovakvi senzori uglavnom su ugrađeni u pod ispod svakog parkirnog mjesta, gdje mjere promjene u magnetskom polju.

To znači da su dosta pogodni za vanjska parkirališta, s obzirom na to da nisu osjetljivi na vanjske uvjete. Dodatno, moguće ih je ugraditi na način da budu nevidljivi, a i energetski su efikasni, pa mogu biti pogonjeni baterijama.

No, imaju i svojih mana. Osjetljivi su na elektromagnetske smetnje, a moraju se i kalibrirati na fizičkoj lokaciji, što može predstavljati izazove kod ugradnje i održavanja. Također, ne funkcioniraju idealno u prepoznavanju vozila koja nisu ispravno parkirana.

## **2.4. Kamere**

Sustavi koji se temelje na kamerama uglavnom koriste računalni vid kako bi detektirali zauzetost nekog parkirnog mjesta. Uz to, omogućuju i druge funkcionalnosti, poput

prepoznavanja registarskih oznaka ili brojanja vozila, a ujedno mogu poslužiti i kao sigurnosne kamere.

Najveća prednost kamera je mogućnost postavljanja na način da jedna kamera pokriva više parkirnih mjesta odjednom, pa se smanjuje broj potrebnih kamera za pokriće cijelog parkirališta (naravno, ako oblik parkirališta i raspored parkirnih mjesta to dopuštaju).

S druge strane, kamere su osjetljive na vremenske uvjete (primarno svjetlost), zahtijevaju povremeno čišćenje leće, a i cjenovno su manje pristupačne, i u nabavci i kod održavanja. Također, za obradu videa potrebno je imati i lokalno računalo, ili pak sve slati u oblak. Unatoč tome, napredak u metodama računalnog vida omogućuje pouzdanu analizu slike i u slabijim uvjetima vidljivosti, zbog čega kamere postaju sve bolji odabir za ovu primjenu.

## **2.5. Primjena umjetne inteligencije**

U skorije vrijeme popularnost umjetne inteligencije raste, kako u svim sferama društva, tako i u sustavima za detekciju vozila. Primjenom umjetne inteligencije na detekciju vozila možemo dopuniti informacije koje dobijemo od klasičnih senzora na različite načine:

- detekcijom zauzetosti parkirnog mjesta iz videozapisa
- prepoznavanjem registarskih oznaka (engl. *Automatic Number Plate Recognition*, ANPR) iz videozapisa
- klasifikacijom tipa i klase vozila iz videozapisa
- predviđanjem slobodnih mjesta na temelju povijesnih podataka

Korištenjem lokalnih modela i modela u oblaku moguće je postići zavidnu razinu točnosti, ali takvi sustavi zahtijevaju znatno veću računalnu snagu.

## 2.6. Zaključak

Odabir vrste senzora i tehnologije za detekciju vozila mora biti usklađen s ciljem sustava, vrstom parkirališta, budžetom i postojećim ograničenjima infrastrukture. Ultrazvučni i magnetni senzori pokazuju se kao pouzdana i cijenom pristupačna opcija za detekciju zauzetosti pojedinog parkirnog mjesta, dok sustavi koji se temelje na kamerama i umjetnoj inteligenciji pružaju veću razinu fleksibilnosti i velik broj dodatnih funkcionalnosti, ali su složeniji i skuplji za implementaciju.

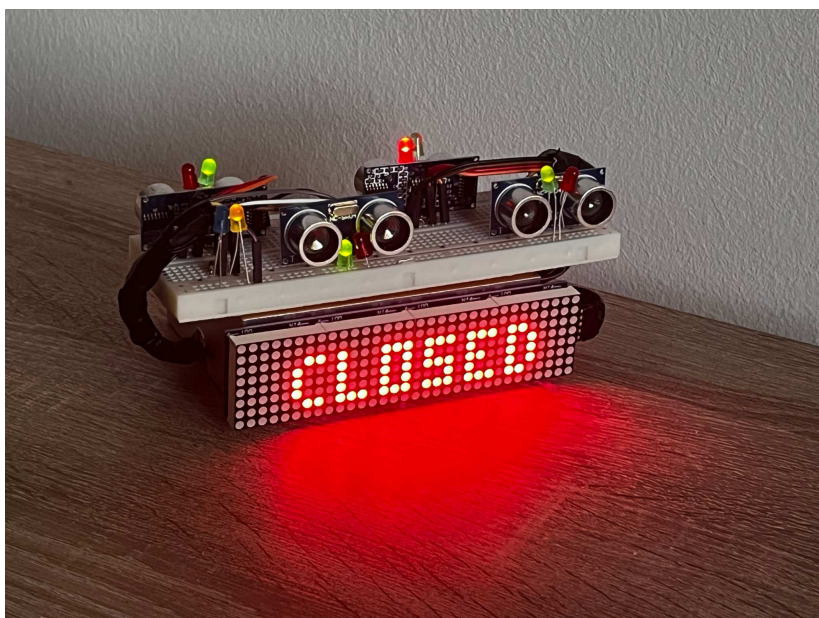
U složenijim sustavima moguće je implementirati više tehnologija odjednom – na primjer, ultrazvučne senzore za detekciju zauzetosti i kamere za dodatnu potvrdu i sigurnosni nadzor. Prije razvoja sustava potrebno je odrediti zahtjeve i potrebe, zatim pronaći ravnotežu između preciznosti, cijene i održivosti sustava, pa odabrati tehnologiju koja najbolje odgovara specifičnom slučaju. Za potrebe ovog rada, kao tehnologija za detekciju zauzetosti pojedinog parkirnog mjesta koristit će se ultrazvučni senzori, primarno radi svoje cijene i jednostavnosti.

### **3. Opis i arhitektura razvijenog rješenja**

Sustav razvijen u ovom radu omogućuje praćenje i upravljanje zauzetošću parkirališta u stvarnom vremenu. Ultrazvučni senzori detektiraju zauzetost parkirnog mjesta i to prikazuju na svom svjetlosnom indikatoru, a zatim dojavljuju središnjem poslužitelju. Program koji upravlja LED zaslonom i globalnim svjetlosnim indikatorom redovito osvježava podatke s poslužitelja, pa te podatke prikazuje na LED zaslonu i indikatoru. Zadnji dio sustava je mobilna aplikacija, koja korisnicima omogućuje pregled svih senzora, kao i dodatne informacije o samom parkiralištu. Autorizirani korisnici kroz aplikaciju dodatno mogu upravljati stanjem cjelokupnog parkirališta, kao i konfiguracijom senzora. Fizički dio sustava (senzori, svjetlosni indikator i LED zaslon) vidljiv je na slikama 3.1. i 3.2., a mobilna aplikacija će biti detaljnije opisana u kasnijim poglavljima. Na slici 3.1. sustav je prikazan u stanju kada je parkiralište otvoreno, a 1 od ukupno 4 senzora detektira zauzeće na svom mjestu (zid koji se nalazi iza senzora je dovoljno blizu), pa je njegov indikator crven, dok su ostali zeleni. Parkiralište je otvoreno i nije cijelo popunjeno, pa globalni svjetlosni indikator to i pokazuje plavom bojom. Na slici 3.2. sustav je u stanju kada je parkiralište zatvoreno (ručno kroz mobilnu aplikaciju ili je van radnog vremena), pa na zaslonu piše CLOSED, a globalni indikator svijetli žuto. Senzori će uvijek detektirati i pokazivati svoje zauzeće, bez obzira na stanje parkirališta.



**Slika 3.1.** Sustav u stanju kad je parkiralište otvoreno



**Slika 3.2.** Sustav u stanju kad je parkiralište zatvoreno

Sustav je izgrađen arhitekturom koja se fokusira na skalabilnost i modularnost. Arhitektura podrazumijeva fizičke komponente (senzore, indikatore i zaslon), poslužitelja i mobilnu aplikaciju. Sustav je zamišljen na način da ga je moguće jednostavno proširiti i prilagoditi na različite modele rada – bilo lokalne (*on-premise*) koji podržavaju jednog klijenta, ili centralizirane (SaaS) koji podržavaju više klijenata.

Fizičkim dijelom sustava upravlja računalo Raspberry Pi 3B+ koje kontrolira senzore, svjetlosne indikatore i LED zaslon. Arhitektura dopušta razdvajanje svih senzora i zaslona na zasebna računala – kako bi to funkcioniralo i u primijenjenom sustavu – no za potrebe ovog rada svim sensorima upravlja jedno računalo, prvenstveno radi smanjenja troškova i jednostavnosti implementacije. Računalo upravlja sensorima kroz GPIO pinove, bez dodatnih razina apstrakcije, a za upravljanje LED zaslonom koristi se knjižnica `luma.led_matrix`. Programska podrška senzora i zaslona napisana je u programskom jeziku Python.

Poslužitelj je razvijen u tehnologiji Node.js uz radni okvir NestJS, a za pristup bazi podataka korištena je knjižnica TypeORM. U ovom sustavu korištena je baza podataka SQLite, no arhitektura dopušta korištenje PostgreSQL ili nekog drugog sustava za upravljanje bazama podataka, što bi bilo prikladnije za model u kojem je poslužitelj centraliziran i opslužuje više različitih klijenata.

Mobilna aplikacija izrađena je korištenjem tehnologije React Native, uz alat Expo. Aplikacija omogućuje prikaz i praćenje stanja parkirališta u stvarnom vremenu za sve korisnike, a ovlašteni korisnici uz to mogu i upravljati parkiralištem i sensorima.

Komunikacija između senzora i poslužitelja, kao i komunikacija između poslužitelja i mobilne aplikacije, odvija se putem sučelja REST. Sensori šalju svoje stanje poslužitelju pri svakoj promjeni, a LED zaslon i mobilna aplikacija periodički dohvaćaju podatke koristeći *polling* – tehniku u kojoj program povremeno dohvaća nove podatke s poslužitelja. Sustav koristi jednostavan i siguran model autentifikacije i autorizacije s jasno definiranim ulogama i dozvolama.

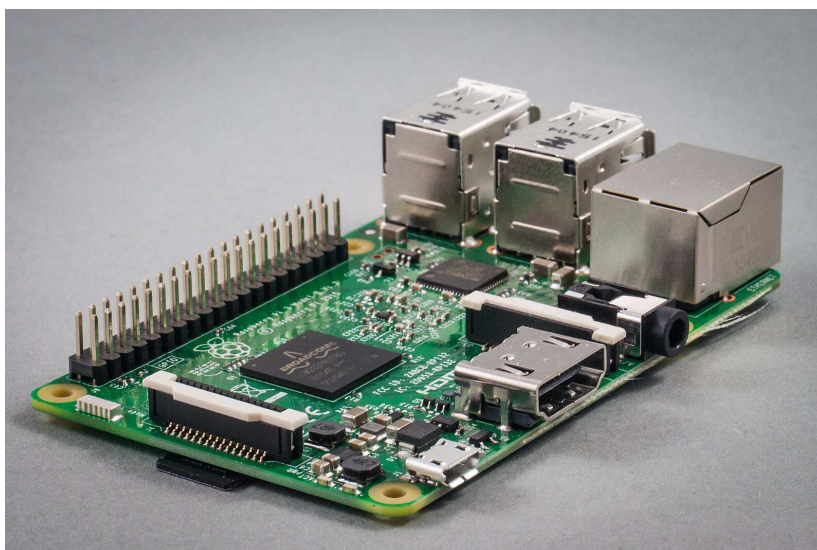
S ovim pristupom ostvarena je modularna cjelina u kojoj je svaki dio moguće neovisno razvijati, ispitivati i unaprjeđivati, bez ugrožavanja ostatka sustava.

## 4. Sklopovsko rješenje

Fizički sloj sustava temelji se na korištenju računala Raspberry Pi 3B+ koje upravlja svim ulazno-izlaznim komponentama sustava: LED zaslonom, svjetlosnim indikatorima i senzorima za detekciju udaljenosti. U implementiranom sustavu sve komponente su povezane na jedno računalo radi cijene i jednostavnosti implementacije, ali arhitektura sustava dopušta razdvajanje svih komponenti na vlastita računala, što je prikladnije za parkirališta s većim brojem parkirnih mjesta koja su fizički udaljenija.

### 4.1. Raspberry Pi 3B+

Raspberry Pi 3B+, prikazan na slici 4.1., popularno je računalo malih dimenzija na kojem je moguće izvoditi potpune operacijske sustave, poput sustava Raspberry Pi OS koji je službeno podržan na uređaju [4] i osmišljen specifično za tu primjenu. Računalo upravlja perifernim uređajima kroz GPIO (*General Purpose Input/Output*) pinove kroz koje šalje i prima digitalne signale [5].



Slika 4.1. Raspberry Pi 3B+ [6]

Raspberry Pi 3B+ ima podršku za WiFi, Bluetooth i žičane mreže, što mu omogućuje da jednostavno komunicira s vanjskim svijetom putem interneta ili lokalne mreže. U ovom radu koristi se za izvođenje Pythonovih skripti koje upravljaju senzorima, indikatorima i LED zaslonom, a uz to kroz lokalnu mrežu komunicira i s udaljenim poslužiteljem.

## 4.2. Senzori udaljenosti – HC-SR04

Za detekciju zauzetosti pojedinih parkirališnih mjesta koriste se četiri ultrazvučna senzora modela HC-SR04, prikazana na slici 4.2. Ovi senzori koriste ultrazvučne valove kako bi odredili udaljenost od predmeta [7]. Emitiraju signal i mjere vrijeme koje je zvuku potrebno da se vrati, na temelju čega se računa udaljenost do prepreke, odnosno vozila. Senzori mjere u rasponu od svega 2 centimetra do četiri metra, a precizni su u 2 centimetra razlučivosti. Svaki senzor zadužen je za jedno parkirališno mjesto, a svi su povezani na GPIO pinove računala Raspberry Pi preko eksperimentalne pločice.



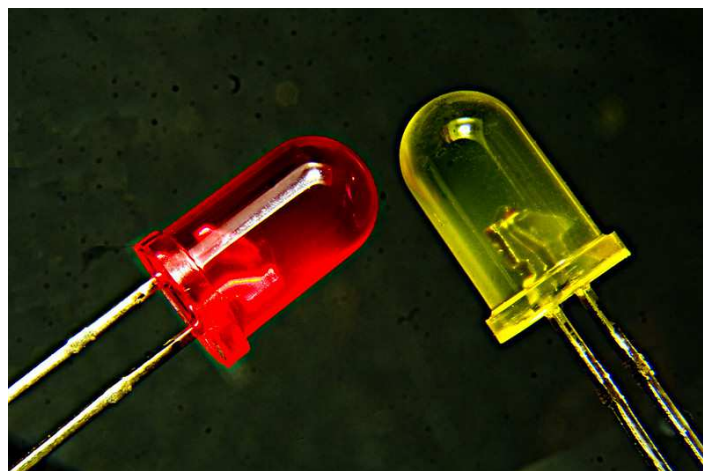
**Slika 4.2.** Ultrazvučni senzor HC-SR04 [8]

Radi pouzdanosti mjerenja, senzor svakih pola sekunde mjeri udaljenost do najbliže prepreke i pohranjuje zadnjih pet vrijednosti. Ako su barem tri vrijednosti ispod definirane granice, mjesto se smatra zauzetim. Na taj način izbjegavaju se lažno pozitivna

mjerenja uzrokovana šumom senzora, do kojeg dolazi relativno često kod ovog modela senzora. S druge strane, krivo očitavanje je i dalje moguće ako se točno ispod senzora nađe čovjek ili neki drugi objekt koji nije vozilo. No, to u praksi ne predstavlja problem, s obzirom na to da je mjerna površina senzora vrlo mala, a koristi se već navedeni algoritam za izbjegavanje šuma, koji će, ako čovjek samo prođe ispod senzora, spriječiti lažno očitavanje. Tek kada se stanje zauzetosti promijeni, ažurira se lokalni svjetlosni indikator i poslužitelju se preko HTTP zahtjeva na sučelje REST dojaviti novo stanje zauzetosti.

### 4.3. Svjetlosni indikatori

Svakom senzoru pridružen je par LED dioda (jedna crvena i jedna zelena) koje služe kao vizualni indikator zauzetosti parkirnog mjesta. LED diode spojene su na računalo preko eksperimentalne pločice i računalo njima upravlja izravno kroz GPIO pinove, kao što je slučaj i kod senzora koji mjere udaljenost. Na identičan način implementiran je i globalni indikator, koji također koristi dvije LED diode za prikaz stanja cijelog parkirališta. Vrsta korištenih LED dioda vidljiva je na slici 4.3.

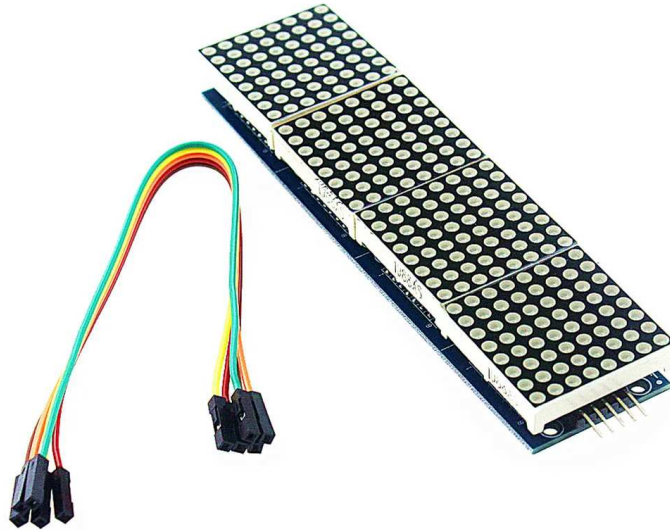


Slika 4.3. LED diode [9]

### 4.4. LED zaslon – MAX7219 4-in-1

Za prikaz ukupnog broja slobodnih mjesta korišten je LED zaslon modela MAX7219 4-in-1 koji je prikazan na slici 4.4. Zaslon sadrži četiri LED matrice razlučivosti 8×8 u liniji, što mu omogućuje ispis cijelih riječi. Zaslon je spojen izravno na Raspberry Pi, a njime se upravlja korištenjem Pythonove knjižnice `luma.led_matrix`. Knjižnica služi

za upravljanje zaslonima s LED matricom kroz sučelje SPI (*Serial Peripheral Interface*) sučelje [10].



**Slika 4.4.** LED zaslon MAX7219 4-in-1 [11]

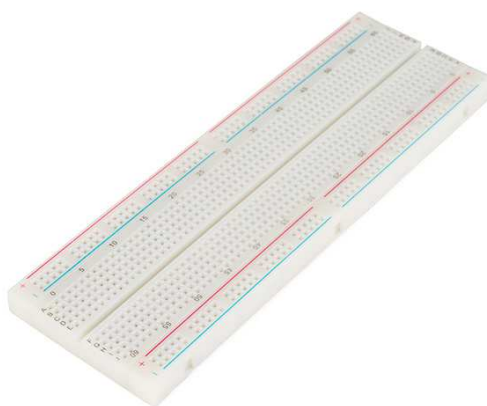
Pythonova skripta koja upravlja LED zaslonom svake sekunde izvrši HTTP zahtjev na sučelje REST poslužitelja, koji joj odgovara s podacima o parkiralištu (je li parkiralište otvoreno i koliko je mjesta slobodno), a skripta te podatke ispisuje na zaslon i ažurira globalni svjetlosni indikator. Na taj način omogućen je stalni pregled dostupnosti parkirališta u stvarnom vremenu.

## 4.5. Eksperimentalna pločica

Svi senzori i LED indikatori spojeni su na eksperimentalnu pločicu, prikazanu na slici 4.5., a koriste se i otpornici tamo gdje je to potrebno. Sve komponente napajaju se izravno kroz računalo Raspberry Pi. Komponente su spojene tako da je moguće jednostavno proširenje i održavanje.

## 4.6. Cjelokupni fizički sloj sustava

Korištenjem lako dostupnih i jeftinih senzora, zaslona i računala implementiran je funkcionalan sustav za praćenje zauzetosti parkirališta. Fokus na modularnu arhitekturu



**Slika 4.5.** Eksperimentalna pločica [12]

osigurava jednostavnu nadogradnju i daljnji razvoj sustava, ali i mogućnost razdvajanja fizičkih komponenti na više uređaja, ako se javi potreba za većim brojem uređaja ili većom fizičkom udaljenosti među uređajima.

## 5. Poslužiteljski dio sustava

Poslužiteljski dio sustava izrađen je korištenjem radnog okvira NestJS koji se temelji na jeziku TypeScript, a omogućuje modularan razvoj skalabilnih aplikacija. Pritom koristi već uvriježene i dokazane oblikovne obrasce. Komunikacija s bazom podataka odvija se korištenjem knjižnice TypeORM, a odabrana baza podataka je SQLite, zbog njene jednostavnosti. No, s obzirom na to da je sama aplikacija neovisna o bazi podataka koja se koristi, za korištenje u višekorisničkom (SaaS) načinu rada predviđa se sustav za upravljanje bazama podataka PostgreSQL. Komunikacija između senzora, mobilne aplikacije i poslužitelja odvija se putem sučelja REST, a poslužitelj je ujedno i posrednik u toj komunikaciji.

### 5.1. Korištene i razmatrane tehnologije

#### 5.1.1. NestJS

NestJS je radni okvir za razvoj efikasnih, skalabilnih i pouzdanih poslužiteljskih aplikacija koji se temelji na Node.js-u [13]. Njegova glavna prednost dolazi iz toga što kombinira funkcionalnosti popularnih jezika i radnih okvira – koristi koncepte OOP-a, inspiriran je arhitekturom Angulara, ali podržava i funkcionalnu paradigmu. NestJS je radni okvir koji postavlja dosta stroga pravila razvoja, kojima potiče modularan pristup razvoju gdje se aplikacija dijeli u module s jasnim granicama i ulogama, što olakšava razvoj i kasnije održavanje sustava. Temelji se na mnogim konceptima poput ubacivanja ovisnosti (engl. *dependency injection*), posredničke arhitekture (engl. *middleware*), presretača HTTP putanja (engl. *interceptor*) i čuvara HTTP putanja (engl. *guard*), koji uvelike olakšavaju izgradnju kompleksnih poslužiteljskih sustava. Također ima odličnu podršku za rad s bazama podataka i GraphQL-om, komunikaciju WebSocketom, ispitivanje i integraciju s manjim knjižnicama, što znači da je odličan (i čest) odabir za profe-

sionalne i kompleksne projekte.

### 5.1.2. TypeORM

TypeORM je popularna ORM (*Object-Relational Mapping*) knjižnica koja se koristi u TypeScriptu i JavaScriptu. Omogućuje razvoj aplikacija u objektno-orijentiranom stilu, u kojem klase u kodu predstavljaju tablice u bazi podataka, a njihove međusobne relacije u potpunosti odgovaraju strukturi baze podataka. TypeORM podržava više različitih baza podataka, uključujući PostgreSQL, SQLite i MySQL [14], što omogućuje fleksibilnost prilikom odabira baze podataka. Osim osnovnog povezivanja modela i tablica, knjižnica nudi alate za generiranje migracija baze podataka, upravljanje relacijama, automatsko dohvaćanje podataka koji su povezani u bazi (tim relacijama), validaciju i rad s transakcijama. U kontekstu ovog projekta, TypeORM je korišten kako bi se pojednostavio rad s bazom podataka, a kasnije olakšalo proširenje ili prelazak na kompleksnija rješenja, bez značajnih promjena u implementaciji samog sustava.

### 5.1.3. SQLite

SQLite je vrlo jednostavna baza podataka koja dolazi bez zasebnog poslužitelja – cijela baza podataka nalazi se u jednoj datoteci. Zbog toga je SQLite privlačan odabir za razvoj, ispitivanje i jednostavnije aplikacije koje ne zahtijevaju veliku količinu podataka ili višekorisnički rad. U ovom sustavu, SQLite je poslužio kao baza podataka korištena tijekom razvoja sustava, ali i kao baza koja se koristi u lokalnom modelu rada, gdje klijent ima svoj vlastiti sustav. Iako je vrlo jednostavna, podržava većinu SQL standarda, rad s relacijama i rad s indeksima, što ju čini dobrim odabirom za jednostavnije aplikacije. S obzirom na to da sustav ne ovisi o specifičnostima SQLitea, moguć je izravan prelazak na kompleksnija rješenja poput PostgreSQL-a za višekorisnički model rada.

### 5.1.4. PostgreSQL

PostgreSQL jedan je od popularnijih sustava za upravljanje relacijskim bazama podataka. Otvorenog je koda [15], a poznat je i po pouzdanosti, stabilnosti i podršci za napredne značajke jezika SQL. Često se koristi u produkcijskom okruženju jer omogućuje rad s velikim količinama podataka, ima dobru podršku za transakcije i omogućuje istovremeni

rad više korisnika bez da dođe do gubitka performanci. PostgreSQL podržava definiranje vlastitih funkcija, napredne vrste indeksiranja, tip podataka JSON, replikaciju i sva ACID svojstva. U kontekstu ovog projekta, planira se korištenje PostgreSQL-a u višekorisničkom modelu rada zbog njegove brzine, pouzdanosti i stabilnosti. Dodatna prednost je to što je potpuno kompatibilan s TypeORM-om, pa je moguće vrlo jednostavno proširiti sustav sa SQLite baze podataka korištene u trenutačnoj implementaciji sustava na PostgreSQL.

## 5.2. Struktura i organizacija aplikacije

Aplikacija je organizirana i podijeljena prema značajkama, a svaka je značajka sustava implementirana unutar vlastitog modula. Takav pristup olakšava održavanje, ispitivanje i kasniju nadogradnju aplikacije. Trenutačno su implementirani sljedeći moduli: AuthModule, SensorModule, UserModule, ParkingLotModule i SeederModule. Ovdje SeederModule služi isključivo za punjenje baze podataka početnim podacima.

Sučelje REST aplikacije sastoji se od nekoliko glavnih sučelja koja obuhvaćaju osnovne funkcionalnosti sustava. Neka od tih sučelja su:

- POST /sensor/:id/occupancy – ažuriranje statusa zauzetosti senzora
- GET /parking-lot/:id/info – dohvaćanje osnovnih informacija o parkiralištu
- GET /user/me – dohvaćanje podataka o trenutačno prijavljenom korisniku
- POST /auth/login – prijava korisnika i izdavanje autentifikacijskog tokena

Ovakav pristup omogućio je podjelu aplikacije u logičke cjeline, što znači da je kod jednostavniji i lakše ga je održavati, a samim time lakši je daljnji razvoj.

## 5.3. Autentifikacija i autorizacija

Sustav autentifikacije temelji se na tokenima JWT (JSON Web Token) s ograničenim trajanjem – odabrano trajanje je 8 sati. Autorizacija je izvedena prema modelu RBAC (*Role-Based Access Control*) s tri razine pristupa: gost (može samo pregledavati podatke), administrator (može upravljati stanjem cijelog parkirališta) i superuser (može mijenjati kon-

figuraciju senzora). Senzori se identificiraju preko jedinstvenog identifikatora (UUID) i sigurnog tokena koji im se dodjeljuje tijekom inicijalne konfiguracije. Tako je ostvarena sigurnost komunikacije i pristupa poslužitelju sa senzora i ostalih komponenti sustava.

## 5.4. Komunikacija s ostatkom sustava

Svaki put kada senzori detektiraju promjenu svog stanja, poslužitelju šalju zahtjev HTTP POST. Globalni svjetlosni indikator i ekran za prikaz stanja koriste *polling* – svake sekunde dohvaćaju trenutno stanje s poslužitelja. Mobilna aplikacija koristi sličan pristup, ali osvježava svoje podatke nešto rjeđe, svake 3 sekunde. Iako u trenutnoj implementaciji ne postoji, jedna od opcija proširenja sustava jest implementacija WebSoketa za komunikaciju ili primjena energetski efikasnijih protokola poput protokola ZigBee ili Z-Wave, što bi bilo korisno, pogotovo u lokalnim implementacijama. Kad poslužitelj primi podatke od senzora, validira identitet senzora koristeći sigurnosni token i ažurira stanje senzora u bazi podataka. Broj slobodnih mjesta računa se direktno iz baze, što je izvedivo zbog relativno malog broja senzora u sustavu – u trenutnom sustavu ih je četiri, no u realnim uvjetima ne očekuje se više od par tisuća senzora – što ne predstavlja problem za relacijske baze podataka.

## 5.5. Baza podataka

Baza podataka sadrži tri osnovne tablice: `users`, `sensor` i `parking_lot`. Tablica `users` sadrži informacije o ulozi korisnika, korisničkom imenu i pripadnosti određenom parkiralištu. Tablica `sensor` sadrži podatke o lokaciji i trenutnom stanju senzora i povezanosti senzora s parkiralištem. Tablica `parking_lot` definira osnovne informacije o parkiralištu – naziv, radno vrijeme i veze prema korisnicima i sensorima. Relacije između entiteta ostvarene su korištenjem TypeScriptovih dekoratora `@ManyToOne` i `@OneToMany` koje koristi TypeORM, čime se omogućuje jednostavno i lako korištenje povezanih podataka.

## 5.6. Višekorisnički rad i sigurnost

Iako demonstracija sustava ima samo jedno parkiralište i poklapa se s lokalnim modelom rada, sustav je razvijen imajući na umu podršku za višekorisnički rad. U slučaju SaaS modela, svaki klijent predstavljen je kao jedan zapis u tablici `parking_lot`, dok u lokalnom modelu sustav radi izolirano za jednog korisnika (pa će tablica `parking_lot` imati samo jedan zapis). Ovakva arhitektura omogućuje jednostavnu prilagodbu sustava različitim poslovnim modelima.

S gledišta sigurnosti, korisnička sučelja REST zaštićena su autentifikacijom putem JWT-a, dok se senzori identificiraju putem sigurnog tokena. Osjetljivi podaci, kao tajni ključ za generiranje tokena JWT, pohranjeni su u datoteci `.env`, da bi se izbjeglo njihovo izlaganje u kodu.

## 5.7. Razvoj i ispitivanje

Razvoj aplikacije odvijao se lokalno, putem Postmana i sličnih alata za ispitivanje sučelja REST. Aplikacija se pokreće standardnom naredbom `npm run start:dev`, a konfiguracijski i osjetljivi podaci definirani su u datoteci `.env`. Iako u trenutačnoj verziji nisu implementirani automatizirani ispitni slučajevi, struktura aplikacije dopušta jednostavno dodavanje takvih ispitnih slučajeva u budućim fazama razvoja. Ispitivanje je obavljeno ručno.

## 6. Mobilna aplikacija

Mobilna aplikacija je dio sustava kroz koji korisnici mogu interagirati sa sustavom i namijenjena je krajnjim korisnicima i administratorima. Omogućuje uvid u stanje parkirališta u stvarnom vremenu i upravljanje njegovim radom (za autentificirane korisnike). Aplikacija je razvijena u tehnologiji React Native uz pomoć alata Expo, čime su osigurane podrška za rad na Android i iOS platformama, brzina i jednostavnost razvoja, a pritom je očuvano korisničko iskustvo nativnih aplikacija svake platforme. Za dohvat podataka koristi se knjižnica SWR, koja koristeći *polling* mehanizam osigurava svježinu podataka, dok je sigurnost osjetljivih podataka i autentifikacijskih tokena osigurana spremanjem tih informacija u zaštićenu lokalnu pohranu.

### 6.1. Korištene tehnologije

#### 6.1.1. React Native

React Native je radni okvir koji omogućuje izradu mobilnih aplikacija za više platformi odjednom, korištenjem jedinstvenog izvornog koda pisanog u JavaScriptu, odnosno TypeScriptu. U pozadini koristi knjižnicu React, a uz to omogućuje direktno preslikavanje Reactovih komponenti na nativne elemente sučelja [16]. Na taj način postiže izgled i korisničko iskustvo nativnih aplikacija, a i performance koje su vrlo bliske onima u nativnim aplikacijama. U ovom projektu korišten je zbog jednostavnosti razvoja (jedinstveni izvorni kod na obje platforme), velike podrške zajednice i brzog razvoja. Također podržava i web platformu, pa je kroz buduća unaprjeđenja moguće prilagoditi aplikaciju da radi i na webu.

### 6.1.2. Expo

Expo je skup alata i usluga koje dodatno pojednostavljaju razvoj i ispitivanje React Native aplikacija. Omogućuje razvoj aplikacije bez potrebe za lokalnim alatima za prevođenje (Xcode i Android Studio), nudi velik broj gotovih modula i sučelja prema nativnim funkcionalnostima svake platforme [17], poput knjižnice `expo-secure-store`, a uz to i omogućuje jednostavno prevođenje i distribuciju aplikacija. U ovom sustavu je korišten radi lakšeg lokalnog razvoja, ali i radi buduće jednostavnosti distribucije aplikacije na trgovinu svake od platformi.

### 6.1.3. SWR

SWR (stale-while-revalidate) je knjižnica za dohvat i upravljanje podacima koja kombinira lokalno spremanje podataka i automatsko osvježavanje podataka. U ovoj aplikaciji koristi se za osvježavanje podataka s poslužitelja, jednom u sekundi kroz *polling*, čime je postignuto skoro trenutačno ažuriranje stanja senzora. Iako podržava i složenije načine rada, u trenutačnoj implementaciji koristi se na najjednostavniji mogući način, bez ručnog upravljanja lokalnim podacima.

### 6.1.4. Expo Router

Za navigaciju unutar aplikacije koristi se Expo Router – deklarativni sustav za definiranje zaslona i navigiranje između definiranih zaslona koji se temelji na strukturi datoteka u izvornom kodu aplikacije. Na taj način izbjegnuta je ručna konfiguracija pojedinih zaslona i pojednostavljena organizacija većih aplikacija s više zaslona. U ovom projektu korišten je za jednostavnu navigaciju između svega tri zaslona, no vrlo je skalabilan pa je pogodan i za budući razvoj.

### 6.1.5. expo-secure-store

Za pohranu osjetljivih korisničkih podataka (autentifikacijskih tokena JWT) korištena je knjižnica `expo-secure-store`. Na uređajima s Androidom knjižnica koristi `SharedPreferences` kao spremište, a dodatno šifrira podatke korištenjem nativnog sustava `KeyStore` uređaja s Androidom [18]. Na uređajima s iOS-om koristi uslugu `Keychain` [18] koja sama štiti podatke. Na taj način je ostvareno sigurno lokalno spremanje osjetljivih poda-

taka.

## 6.2. Autentifikacija i autorizacija

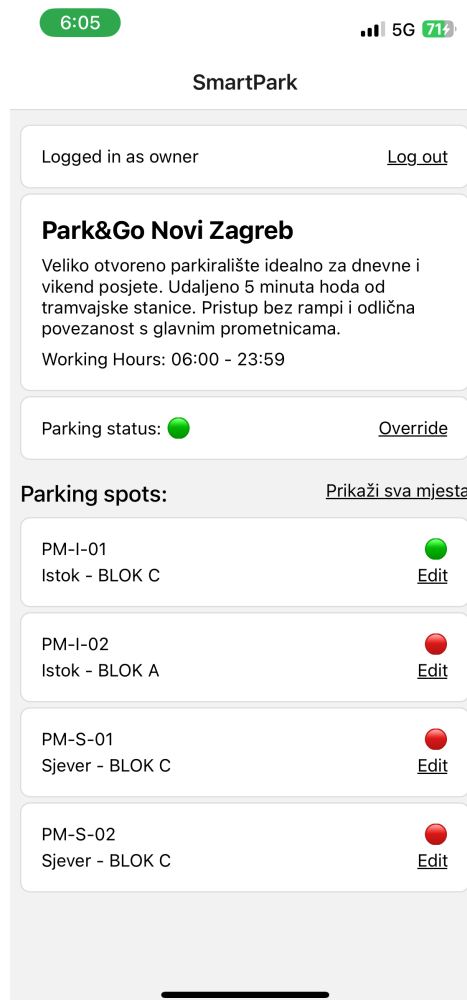
Sustav koristi tokene JWT za autentifikaciju korisnika. Korisnici prilikom prijave dobivaju svoj token koji se sprema u sigurnu pohranu, zatim se automatski dodaje u zaglavlje svih zahtjeva prema poslužitelju. Korištenjem modela RBAC, razlikujemo tri razine pristupa:

- Gost – može pregledavati stanje senzora i parkirališta
- Administrator – dodatno može mijenjati globalno stanje parkirališta (otvoreno/zatvoreno)
- Superuser – uz sve prethodne ovlasti, može uređivati osnovne podatke o senzorima

## 6.3. Struktura i organizacija aplikacije

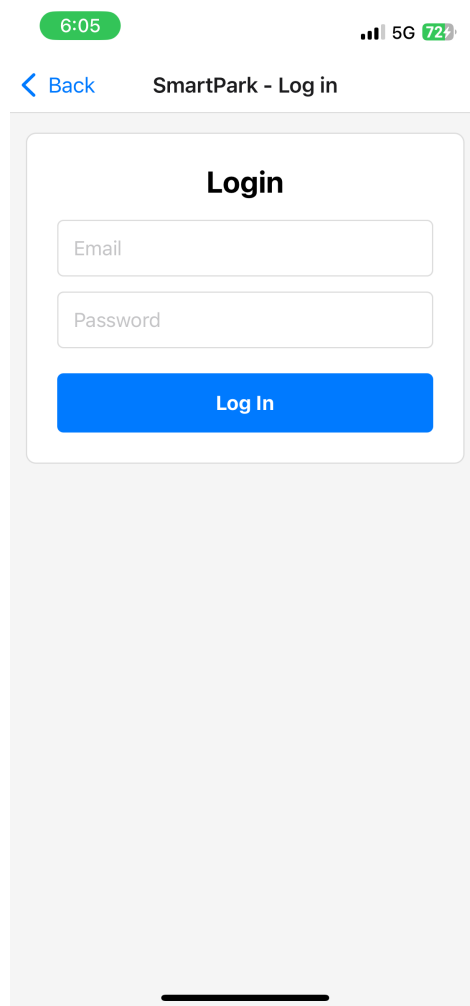
Aplikacija je organizirana prema funkcionalnim cjelinama i koristi Reactov Context API za upravljanje autentifikacijom i autorizacijom korisnika. Trenutačna struktura sastoji se od tri osnovna zaslona:

- Pregled parkirališta – zaslon, vidljiv na slici 6.1., prikazuje osnovne podatke o parkiralištu (ime i opis), radno vrijeme, stanje parkirališta (otvoreno/zatvoreno), kao i popis parkirališnih mjesta s prikazom statusa svakog mjesta. Uz to, ako je korisnik prijavljen, pokazuje i njegovo korisničko ime, kao i mogućnost odjave. Ako korisnik nije prijavljen, daje mu se mogućnost da se prijavi. Mjestu su dodatno pridruženi naziv i lokacija. Prijavljeni korisnici mogu mijenjati stanje cjelokupnog parkirališta, ali i uređivati osnovne podatke o senzorima, ovisno o razini pristupa koju imaju. Također je moguće filtrirati mjesta tako da se prikažu samo slobodna mjesta



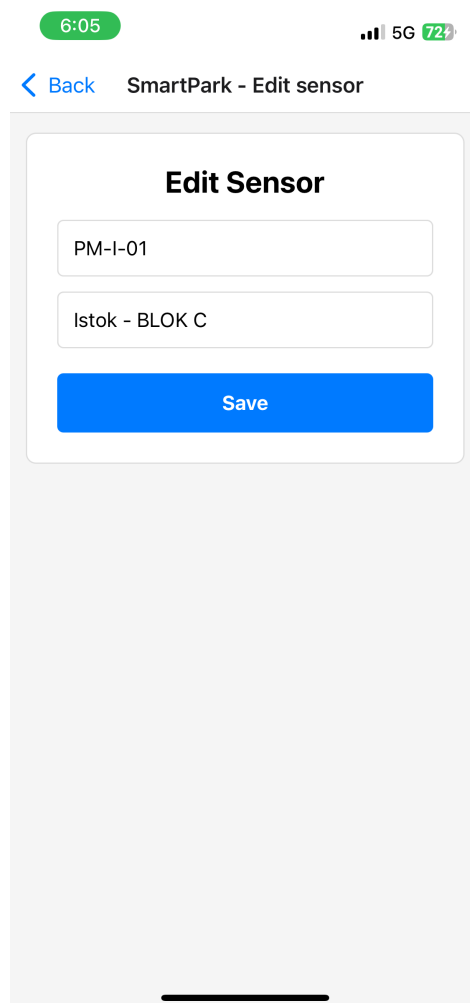
**Slika 6.1.** Zaslon pregleda parkirališta

- Zaslon prijave, prikazan na slici 6.2. – omogućuje prijavu korisnika upisivanjem korisničkog imena i zaporkе. Nakon uspješne prijave, token JWT se pohranjuje lokalno pomoću knjižnice expo-secure-store, ali se i privremeno zadržava u memoriji aplikacije radi bržeg pristupa



**Slika 6.2.** Zaslون korisničke prijave

- Uređivanje senzora – zaslon dostupan samo najvećoj razini autentificiranih korisnika (Superuser), koji omogućuje uređivanje imena i lokacije pojedinih senzora. Važno je napomenuti da svi korisnici mogu koristiti aplikaciju i bez prijave, ali mogu samo čitati podatke, no ne mogu ništa mijenjati. Zaslون je vidljiv na slici 6.3.



**Slika 6.3.** Zaslon uređivanja podataka o senzoru

U trenutačnoj implementaciji aplikacija ne podržava izvanmrežni rad, naprednije upravljanje podacima (poput optimističnih izmjena) niti sofisticirani prikaz grešaka, no to su mogućnosti koje su predviđene za kasnije faze razvoja.

## 7. Budući razvoj

Iako ovaj sustav već omogućuje osnovne funkcionalnosti praćenja i upravljanja zauzeću parkirališta pomoću senzora, mobilne aplikacije i poslužitelja, postoji mnogo prilika za proširenje i poboljšanje sustava, kroz koje bi se ostvarila šira funkcionalnost i veća fleksibilnost sustava. Neki od prijedloga u nastavku su temeljeni na ograničenjima postojeće implementacije, a neki proizlaze iz pretpostavljenih potreba i želja korisnika u produkcijskom okruženju.

### 7.1. Podrška za PostgreSQL kao alternativu SQLiteu

Postojeći poslužiteljski dio sustava koristi bazu podataka SQLite, no PostgreSQL je, radi svoje skalabilnosti i pouzdanosti, bolji odabir ako se radi o višekorisničkom modelu rada, gdje postoji jedan centralizirani poslužitelj koji opskrbljuje više klijenata. PostgreSQL nudi naprednije značajke poput sigurnosnih kopija, koje ostvaruje pomoću dnevnika i inkrementalnih kopija, ili bolje skalabilnosti ostvarene kroz replikaciju i *sharding*. Samim time, nudi i bolje performance.

S obzirom na to da aplikacija koristi TypeORM za rad s bazom podataka, implementacija PostgreSQL baze kao alternative SQLiteu je trivijalna, kada se za tim javi potreba.

### 7.2. ZigBee komunikacija za lokalne sustave

U trenutačnoj verziji sustava senzori s poslužiteljem komuniciraju preko HTTP protokola, koristeći sučelje REST poslužitelja. Za primjenu na većim parkiralištima, gdje će se fizički poslužitelj nalaziti na lokaciji parkirališta, planira se implementacija protokola ZigBee za komunikaciju poslužitelja i senzora. ZigBee je bežični protokol koji je energetski učinkovit [19], a omogućuje uspostavljanje stabilne MESH mreže između uređaja na

kraćoj udaljenosti, što ga čini pogodnim za lokalne implementacije gdje na maloj lokaciji postoji velik broj parkirališnih senzora.

Integracijom ZigBeejevih modula na uređaje senzora i upotrebom ZigBeejevog koodinatora na poslužitelju, senzori će moći međusobno komunicirati i slati podatke prema lokalnom poslužitelju putem protokola ZigBee. To će dodatno omogućiti i konfiguraciju sustava u zatvorenom načinu rada, gdje će cijeli sustav funkcionirati automatizirano, bez mobilne aplikacije ili pristupa vanjskoj mreži.

### **7.3. Dodatne mogućnosti uređivanja senzora i parkirališta iz mobilne aplikacije**

U trenutačnoj implementaciji aplikacija podržava uređivanje osnovnih podataka senzora (ime i lokacija) i izmjenu statusa parkirališta (otvoreno/zatvoreno). Planira se proširenje te funkcionalnosti i omogućavanje potpunog uređivanja i dodavanja senzora iz same mobilne aplikacije, bez potrebe za ručnom konfiguracijom kroz bazu podataka. To je nešto kompleksnija značajka, s obzirom na to da treba biti moguće jednostavno konfigurirati i same senzore, pa i oni moraju imati neko sučelje za konfiguraciju. Nastavno na to, planirana je i mogućnost stvaranja vlastitog parkirališta u aplikaciji, nakon čega će korisnici moći samostalno uvesti sustav u svoje parkiralište, bez vanjske tehničke podrške.

### **7.4. Upravljanje korisnicima unutar mobilne aplikacije**

U postojećem sustavu upravljanje korisnicima (dodavanje novih korisnika, promjena uloga) je ograničeno na ručni unos korisnika u bazu podataka. Planira se implementacija upravljanja korisnicima unutar aplikacije, radi povećanja pristupačnosti i fleksibilnosti sustava.

Funkcionalnosti potrebne za ostvarenje ove značajke:

- Prikaz popisa svih korisnika
- Dodavanje i uklanjanje korisnika
- Promjena korisničkih uloga (gost, administrator, superuser)

- Resetiranje zaporce

Implementacijom ove značajke omogućilo bi se fleksibilnije upravljanje većim brojem korisnika u velikim parkirališnim sustavima.

## 7.5. Komunikacija WebSocketom za mobilnu aplikaciju i LED zaslon

Pristup REST koji se temelji na mehanizmu *polling* nije idealno rješenje, posebice kada je potrebno reagirati na promjene stanja u stvarnom vremenu. Za potrebe trenutačne sinkronizacije između navedenih dijelova sustava planira se implementacija komunikacije preko WebSocketa. Na taj način omogućit će se:

- Dvosmjerna komunikacija između mobilne aplikacije (ili LED zaslona) i poslužitelja
- Veća efikasnost i manji promet u usporedbi s *pollingom*
- Brža, gotovo trenutačna promjena podataka u korisničkom sučelju

WebSockete će koristiti mobilna aplikacija za prikaz ažuriranih podataka o stanju parkirnih mjesta, a LED zaslon će korištenjem WebSocketa moći automatski prikazati broj slobodnih mjesta bez potrebe za *pollingom*.

## 7.6. Vizualna karta parkirališta

Jedna od bitnijih funkcionalnosti koja bi znatno poboljšala postojeće korisničko iskustvo je vizualna karta parkirališta. Ova značajka obuhvaća vizualni interaktivni prikaz svih senzora, njihovih pozicija i statusa u stvarnom vremenu, unutar aplikacije.

Za ostvarenje ovog prikaza potrebno je omogućiti:

- Unos i prikaz više karti parkirališta, podijeljenih po sekcijama (npr. kat garaže)
- Unos pripadnosti karti i pozicije na karti za svaki senzor prilikom inicijalne konfiguracije

- Prikaz statusa senzora na karti korištenjem ikona u boji

Ova funkcionalnost može se izvesti korištenjem SVG ili PNG slika unutar aplikacije preko koje bi se pozicionirali senzori. Alternativno, moguće je razviti potpuno prilagodljivu komponentu aplikacije s mogućnosti dodavanja, pozicioniranja i konfiguriranja senzora izravno na karti, kroz aplikaciju.

Vizualna karta bila bi posebno korisna na većim parkiralištima koja se sastoje od više sekcija, gdje bi korisnici jednostavno kroz mobilnu aplikaciju mogli pronaći sebi najbliže mjesto na karti, a zatim tamo i parkirati.

## **7.7. Povezivanje s vanjskim sustavima za naplatu parkinga**

Da bi sustav bio korisniji i privlačniji klijentima s komercijalnim parkiralištima, potrebno je razmotriti integraciju s postojećim sustavima za naplatu parkiranja, bilo da se radi o aparatima za izdavanje karata, mobilnim aplikacijama za plaćanje ili automatskim rampama. Jedan od potencijalnih načina implementacije ove značajke podrazumijeva izradu standardiziranog sučelja REST s kojim bi drugi sustavi mogli komunicirati, no vjerojatniji put razvoja je integracija najpopularnijih metoda naplate izravno u ovaj sustav.

## 8. Zaključak

Sustavi za praćenje i upravljanje zauzetošću parkirališta uvelike olakšavaju korisnicima pronalazak parkinga za svoje vozilo, a samim time štede vrijeme i gorivo, smanjuju problem prometa u velikim urbanim sredinama i smanjuju ekološki učinak prometa na velike gradove.

Kroz ovaj rad implementiran je funkcionalni sustav za praćenje i upravljanje zauzetošću parkirališta. Sustav se sastoji od fizičkog sloja (ultrazvučnih senzora i uređaja Raspberry Pi, uz dodatak LED zaslona i svjetlosnih indikatora), središnjeg poslužitelja koji je izrađen korištenjem tehnologija NestJS i TypeORM, i mobilne aplikacije razvijene uz pomoć radnog okvira React Native i alata Expo. Komunikacija između svih komponenti implementiranog sustava odvija se preko sučelja REST, a informacije o stanju parkirališta i parkirnih mjesta krajnji korisnici mogu pratiti u stvarnom vremenu.

Sustav je razvijen s naglaskom na modularnost, skalabilnost i mogućnosti proširenja. Korištenjem TypeORM-a za pristup bazi podataka postignuta je mogućnost lakog prebacivanja sustava sa SQLite baze podataka, predviđene za implementaciju sustava na jednoj lokaciji, za jednog klijenta, na konkretnija rješenja poput PostgreSQL-a pogodnija za implementaciju sustava u okruženju u kojem će ga koristiti veći broj klijenata i korisnika. Implementacijom kontrole pristupa RBAC i uporabom sigurnosnih komunikacijskih mehanizama (tokeni JWT i sigurni tokeni za identifikaciju senzora), ostvarena je visoka razina sigurnosti, koja je primjerena i za ozbiljnije produkcijske sustave.

Napravljena je i kvalitativna analiza različitih vrsta detekcije zauzetosti, uključujući kamere i ultrazvučne, infracrvene i magnetne senzore. Naposljetku su se, za potrebe ovog rada, ultrazvučni senzori pokazali kao najbolji izbor, zbog cjenovne pristupačnosti, jednostavnosti integracije i lake dostupnosti. Iako je bio moguć i kompleksniji izbor koji

omogućuje dodatne funkcionalnosti, ovaj izbor postoji i u komercijalnim sustavima, a pokazao se prikladnim za ovaj rad, uz prostor za buduću nadogradnju u slučaju da se sustav implementira u stvarnim uvjetima.

Sustav je ručno ispitan u lokalnom okruženju gdje se pokazao stabilnim i pouzdanim u radu, bez grešaka. Kombinacijom korištenih senzora i algoritama omogućeno je filtriranje šuma i ispravna detekcija zauzetosti, a mobilna aplikacija omogućuje jednostavan pregled i upravljanje parkiralištem, ovisno o razini pristupa koju korisnik ima.

Dodatno, postoji više smjerova u kojima je moguće razvijati sustav: implementacija komunikacije WebSocketom, podrška za protokol i mrežu ZigBee, nadogradnja korisničkog sučelja mobilne aplikacije i integracija s vanjskim sustavima za naplatu parkinga. Na taj način dodatno bi se povećala korisnost sustava u stvarnom, produkcijskom okruženju.

Zaključno, ovaj rad pokazuje da je moguće, korištenjem cijenom pristupačnih i lako dostupnih komponenti, razviti pouzdan sustav za praćenje i upravljanje zauzetosti parkirališta koji se može iskoristiti kao temelj za budući razvoj i komercijalnu primjenu.

## Literatura

- [1] MaxBotix: How Ultrasonic Sensors Work, <https://maxbotix.com/blogs/blog/how-ultrasonic-sensors-work>, [mrežno; stranica posjećena: lipanj 2025.].
- [2] GSTiR: Active Infrared Vs. Passive Infrared, <https://www.gst-ir.net/news-events/infrared-knowledge/491.html>, [mrežno; stranica posjećena: lipanj 2025.].
- [3] ABLIC: What is a Magnetic Sensor?, <https://www.ablic.com/en/semicon/products/sensor/magnetism-sensor-ic/intro3/>, [mrežno; stranica posjećena: lipanj 2025.].
- [4] Raspberry Pi: Raspberry Pi OS, <https://www.raspberrypi.com/software/>, [mrežno; stranica posjećena: lipanj 2025.].
- [5] Raspberry Pi: GPIO Pins, <https://projects.raspberrypi.org/en/projects/physical-computing/1>, [mrežno; stranica posjećena: lipanj 2025.].
- [6] Wikimedia Commons: Raspberry Pi 3, [https://commons.wikimedia.org/wiki/File:9815\\_-\\_Raspberry\\_Pi\\_3.jpg](https://commons.wikimedia.org/wiki/File:9815_-_Raspberry_Pi_3.jpg), [mrežno; stranica posjećena: lipanj 2025.].
- [7] Loomen: HC-SR04 UZV senzor, <https://arhiva-2022.loomen.carnet.hr/mod/book/view.php?id=2421964>, [mrežno; stranica posjećena: lipanj 2025.].
- [8] SparkFun Electronics: Ultrasonic Distance Sensor - HC-SR04, <https://www.flickr.com/photos/sparkfun/48439643861>, [mrežno; stranica posjećena: lipanj 2025.].
- [9] Tudor Barker: LED (Light Emitting Diode) Red and Yellow, <https://www.flickr.com/photos/tudedude/3421613669>, [mrežno; stranica posjećena: lipanj 2025.].
- [10] Luma.LED\_Matrix: Display driver for MAX7219, WS2812, APA102, <https://luma-led-matrix.readthedocs.io/en/latest/intro.html>, [mrežno; stranica posjećena: li-

panj 2025.].

- [11] Zagdima: 2x4 Led matrix clock Case, <https://www.thingiverse.com/thing:2841332>, [mrežno; stranica posjećena: lipanj 2025.].
- [12] SparkFun Electronics: Breadboard, <https://www.flickr.com/photos/sparkfun/11715275493/in/photostream/>, [mrežno; stranica posjećena: lipanj 2025.].
- [13] NestJS, <https://nestjs.com/>, [mrežno; stranica posjećena: lipanj 2025.].
- [14] TypeORM, <https://typeorm.io/docs/getting-started/>, [mrežno; stranica posjećena: lipanj 2025.].
- [15] Postgres, <https://www.postgresql.org/>, [mrežno; stranica posjećena: lipanj 2025.].
- [16] React Native: Core Components and Native Components, <https://reactnative.dev/docs/intro-react-native-components>, [mrežno; stranica posjećena: lipanj 2025.].
- [17] Expo: Introduction, <https://docs.expo.dev/get-started/introduction/>, [mrežno; stranica posjećena: lipanj 2025.].
- [18] Expo: Expo SecureStore, <https://docs.expo.dev/versions/latest/sdk/securestore/>, [mrežno; stranica posjećena: lipanj 2025.].
- [19] Lumories: ZigBee za kuće i poslovne objekte, <https://www.lumories.hr/inspiracija/sto-je-zigbee>, [mrežno; stranica posjećena: lipanj 2025.].

## Sažetak

### Sustav za praćenje i upravljanje zauzetošću parkirnih mjesta u stvarnom vremenu

Tin Prvčić

U sklopu ovog rada implementiran je sustav za praćenje i upravljanje zauzetošću parkirnih mjesta u stvarnom vremenu. U radu se prvo kvalitativno uspoređuju najčešće tehnologije koje se koriste za detekciju vozila i bira se jedna od njih za razvoj sustava. Sustav se sastoji od ultrazvučnih senzora povezanih na uređaj Raspberry Pi, poslužitelja izrađenog u radnom okviru NestJS i mobilne aplikacije izrađene koristeći radni okvir React Native i alat Expo. Komponente sustava međusobno komuniciraju putem sučelja REST, a podaci su dostupni u stvarnom vremenu. Kod autentifikacije korisnika korištena je implementacija tokena JWT, dok je kontrola pristupa implementirana prema modelu RBAC. Prilikom razvoja sustava, naglasak je stavljen na modularnost i mogućnost proširenja sustava, radi lakših poboljšanja sustava u budućnosti. Razmotrena su moguća poboljšanja, neka od kojih su podrška za PostgreSQL, dodavanje vizualne karte parkirališta u mobilnu aplikaciju i integracija protokola ZigBee i WebSocket u sustav.

**Ključne riječi:** pametno parkiranje, ultrazvučni senzor udaljenosti, mobilna aplikacija, React Native, Expo, NestJS, Raspberry Pi

# Abstract

## Real-time parking occupancy monitoring and management system

Tin Prvčić

This thesis implements a system for monitoring and managing parking space occupancy in real time. First, the thesis dives into a qualitative comparison of technologies commonly used for detecting vehicles, and then a technology is chosen to be used for this system. The system consists of ultrasonic distance sensors connected to a Raspberry Pi computer, a server endpoint made using NestJS framework, and a mobile application developed with React Native and Expo. Parts of the system communicate with each other through a REST interface, and the data can be monitored in real time. For authentication, an implementation of JWT tokens is used, while access control implements the RBAC model. The system was developed with a focus on its modularity and extensibility to allow for easier improvements in the future. Possible improvements were considered, including support for PostgreSQL, adding a map of the parking lot to the application, and integrating ZigBee and WebSocket protocols into the system.

**Keywords:** smart parking, ultrasonic distance sensor, mobile application, React Native, Expo, NestJS, Raspberry Pi