Dokumentacija projekta rt21

David Slatinek **Marcel Iskrač** **Vid Kreča**

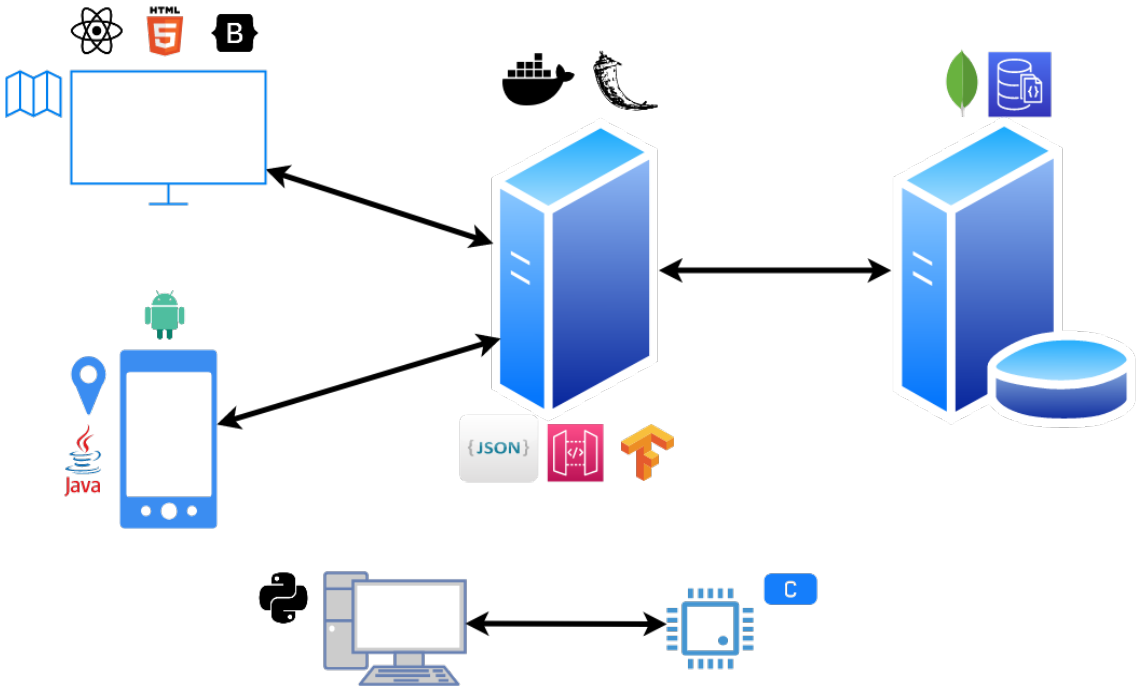
Maribor, januar 2022

Povzetek

Ta dokument predstavi dokumentacijo projekta rt21. Najprej predstavimo projekt kot celoto in njegove glavne enote. Na koncu predstavimo posamezno enoto, med katere spadajo podatkovna baza, spletni vmesnik za pridobivanje in shranjevanje podatkov z računalniškim vidom, Android aplikacija, spletna stran, algoritem za stiskanje zaporedja števil in mikrokrmilnik.

1 Enote projekta

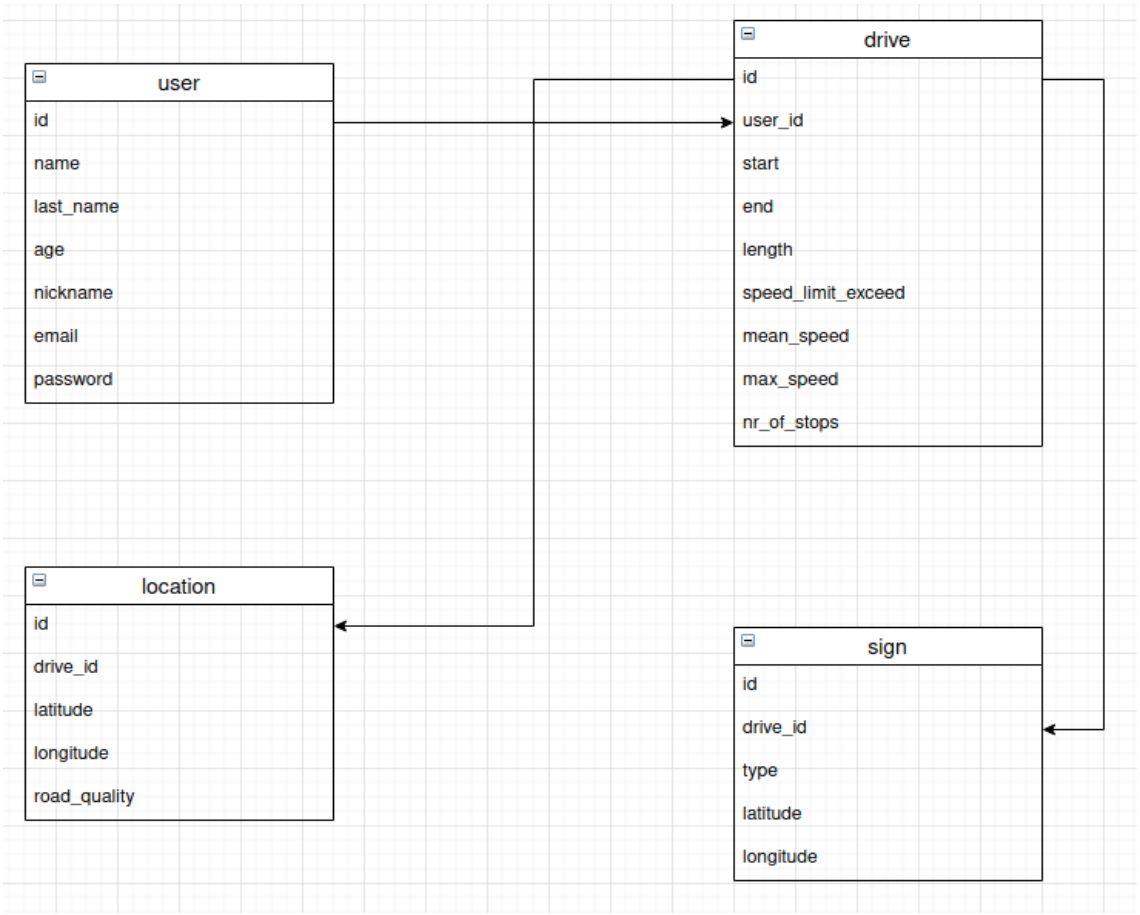
Projekt sestavlja 6 glavnih komponent: podatkovna baza, spletni vmesnik z računalniškim vidom, Android aplikacija, spletna stran, algoritem za stiskanje podatkov in mikrokrmilnik - povezave med enotami in uporabljene tehnologije prikazuje slika 1. [1]



Slika 1: Enote projekta.

1.1 Podatkovna baza

Za podatkovno bazo smo izbrali NoSQL tip baze, natančneje MongoDB. Za gostovanje podatkovne baze smo uporabili MongoDB Atlas, ki ima različne nivoje gostovanj, izbrali smo zastonj verzijo. V bazi shranjujemo podatke o uporabniku, njegovih vožnjah, lokacijah teh voženj in podatke o prometnih znakih. Slika 2 prikazuje ER diagram, ki smo ga naredili s spletno aplikacijo Diagrams.net. [1]



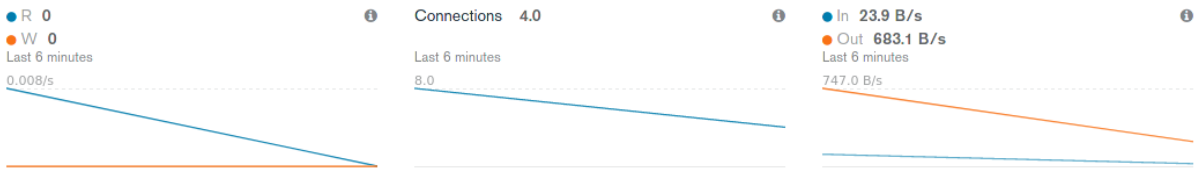
Slika 2: ER diagram.

1.1.1 Uporaba podatkovne baze

Podatkovna baza je namenjena shranjevanju in pridobivanju podatkov. Zaradi varnosti je onemogočen neposredni dostop - vse operacije se izvajajo preko ustreznih API klicev preko spletnega vmesnika.

Upravljanje s podatkovno bazo je namenjeno administratorju. Ko se ta prijavi na MongoDB Atlas ali preko terminala, kot je na primer MongoDB Compass, ima na voljo veliko možnosti, s katerimi lahko upravlja s podatkovno bazo. Vidi lahko vse zbirke, njihovo vsebino, lahko izbriše zapise, jih doda ali posodobi ...

Prav tako ima na voljo upravljanje s strežnikom - strežnik lahko vklopi/izklopi, vidi druge statistične podatke, na primer število povezav, omrežni promet, lokacijo strežnika - slika 3. Administrator lahko tudi upravlja dostop do baze - določa pravice uporabnikom, onemogoči dostop iz določenih IP-naslovov ...



Slika 3: Upravljanje s podatkovno bazo.

1.2 Spletni vmesnik z računalniškim vidom

API je narejen s programskim jezikom python oziroma z njegovim ogrodjem, flask in temelji na arhitekturi REST, ki vsebuje HTTP metode. API gostuje na platformi Heroku in deluje znotraj docker zabojnika. Za hitrejše delovanje smo uporabili strežnik *gunicorn*. Podatki se vračajo v obliki JSON. Kar se dotika varnosti, sistem vsebuje naslednje varnostne mehanizme: [1]

- Identifikacija - API ključ.
- Šifriranje prometa - TLS[1].

Podpiramo vse operacije CRUD:

- C - Create - ustvarjanje novih zapisov.
- R - Read - pridobivanje podatkov.
- U - Update - posodabljanje podatkov.
- D - Delete - brisanje podatkov.

Za prepoznavo prometnih znakov iz slike smo uporabili knjižnice TensorFlow, Keras in OpenCV - z njimi smo ustvarili konvolucijsko nevronsko mrežo. Ob prejemu slike API prepozna prometni znak iz slike, nato pa to vrednost pošlje nazaj k odjemalcu. Za shranjevanje gesla kot hash smo uporabili knjižnico Flask-Bcrypt, za povezovanje na bazo pa knjižnico Flask-PyMongo. Prav tako smo - zaradi avtomatizacije - napisali bash skripto, ki zgradi docker sliko in jo naloži na strežnik. API smo razvili z razvojnim okoljem PyCharm Professional, za testiranje delovanja smo uporabili program Postman. [1]

Za hitrejše delovanje smo uporabili knjižnico mpi4py, s katero porazdelimo delovanje programa na več računalnikov. Funkcionalnost deluje pri klicu metode za prepoznavo prometnih znakov iz slike - glavni program prejme zahtevek in slike, ki jih nato razdeli med ostale računalnike, nato pa prejme njihove odzive oziroma odgovore - slika 4.

```
SLAVE 1 - processing image '.\images\image2.jpg'
SLAVE 1 - processing image '.\images\image3.jpg'
MASTER - results_index: 0 for slave 1 and subset index 0, result: Yield
MASTER - results_index: 1 for slave 1 and subset index 1, result: Speed limit (60km/h)
MASTER - results_index: 2 for slave 1 and subset index 2, result: Priority road
MASTER - finished with results: ['Yield', 'Speed limit (60km/h)', 'Priority road']
```

Slika 4: Paralelna in porazdeljena obdelava slik.

1.2.1 Uporaba spletnega vmesnika

Spletni vmesnik deluje preko ustreznih API klicev. Ob prejemu zahtevka se pogleda njegova pravilnost in ustreznost API ključa. Če je zahtevek ustrezen, ključ pa pravilen, se API poveže na podatkovno bazo, kjer izvede ustrezno akcijo, nato pa vrne odziv. Če je zahtevek napačen ali pa je ključ nepravilen, API vrne opis napake z ustrezno kodo.

1.3 Android aplikacija

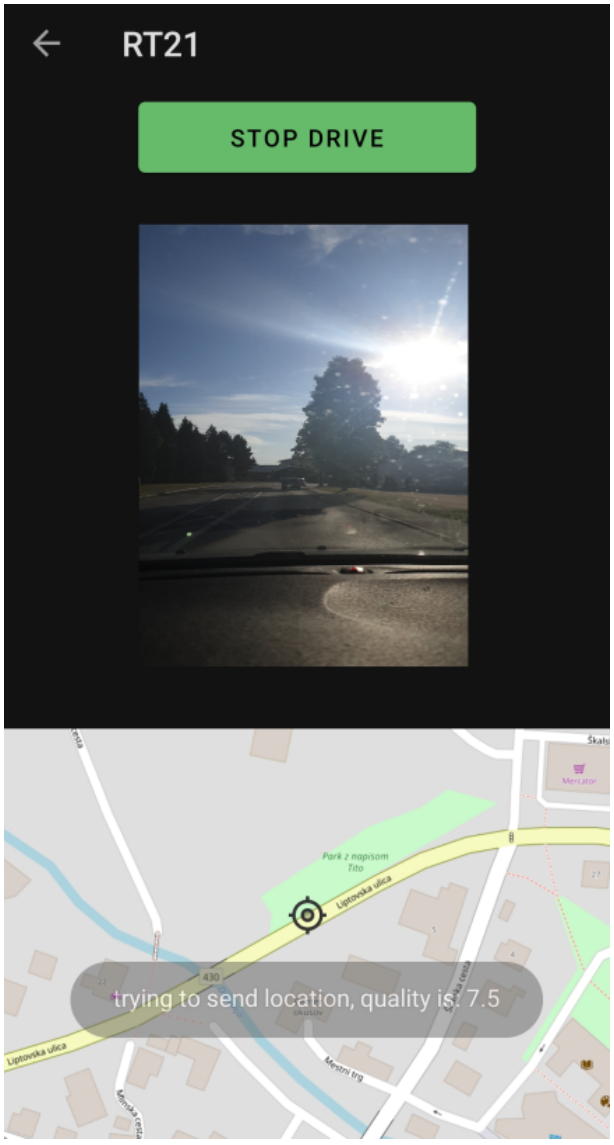
Android aplikacijo smo razvili v razvojnem okolju Android Studio ter s programskim jezikom Java. Glavna funkcionalnost aplikacije je zajem slike in podatkov iz senzorjev ter pošiljanje teh na strežnik. Dodatno smo uporabili knjižnico OpenStreetMap za prikaz trenutne lokacije. [1]
V aplikaciji smo uporabili naslednje senzorje:

[1]Transport Layer Security; poskrbi Heroku.

- Kamera - zajemanje slike za prepoznavo prometnih znakov.
- GPS - dostop do zemljepisne širine in dolžine.
- Žiroskop - kakovost ceste.
- Pospeškometer - spremljanje hitrosti.

1.3.1 Uporaba aplikacije

Po namestitvi in dodelitvi pravic aplikacija omogoča registracijo, prijavo in spremembo gesla uporabniškega računa. Po prijavi lahko uporabnik začne vožnjo - aplikacija začne zajemati podatke iz senzorjev in sliko iz kamere, ki jih nato pošilja na strežnik. Aplikacija prikazuje trenutno lokacijo na zemljevidu, dodatno še prikazuje ostale zajete podatke - slika 5.



Slika 5: Aplikacija med vožnjo.

1.4 Spletna stran

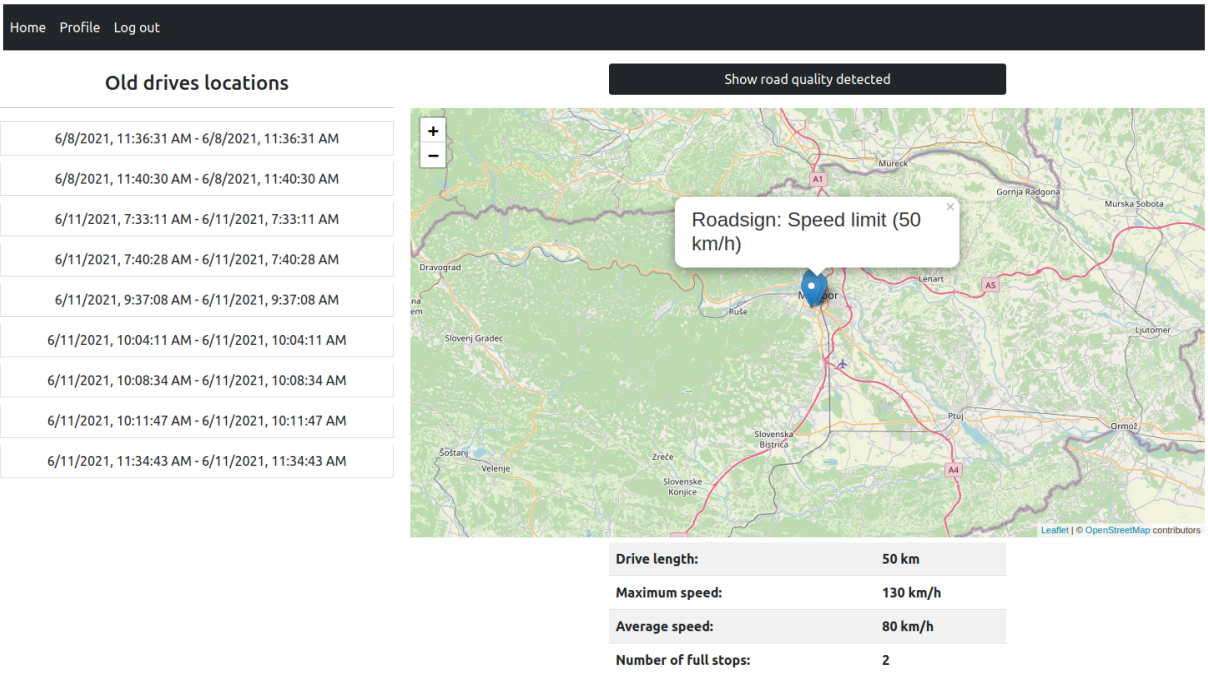
Spletno stran smo ustvarili s pomočjo JavaScript knjižnice React, HTML5 in CSS ter Boostrap knjižnice za enostavno oblikovanje elementov. React smo uporabili za postavitev in klic komponent aplikacije in za komunikacijo med API-jem in spletno stranjo. Dodatno smo uporabili knjižnico Leaflet za prikaz zemljevida in podatkov na njem. [1]

Spletna stran gostuje na platformi Heroku, znotraj docker zabojnika in podpira HTTPS protokol. Za hitrejše delovanje smo uporabili strežnik Nginx. Tudi tukaj smo - zaradi avtomatizacije - napisali bash skripto, ki zgradi docker sliko in jo naloži na strežnik. Spletno stran smo razvili z razvojnim okoljem WebStorm, PhpStorm in urejevalnikom Visual Studio Code. [1]

1.4.1 Uporaba spletne strani

Glavna funkcionalnost spletne strani je vizualizacija podatkov, poleg tega aplikacija omogoča še registracijo, prijavo in dostop do profilne strani.
Glavna stran prikaže osnovne podatke o projektu in služi kot vstopna točka. Na registracijski strani uporabnik vnese svoje osnovne podatke in se registrira, s čimer mu je omogočena uporaba storitev projekta. Na strani za prijavo se uporabnik identificira, nato pa lahko dostopa do profilne strani - lahko posodobi svoje podatke - in do strani za vizualizacijo podatkov. [2]

Stran z vizualizacijo predstavlja jedro spletne strani. Tukaj lahko uporabnik vidi svoje vožnje - datum začetka in konca, dolžino vožnje, hitrosti, zaznane prometne znake ..., celotna vožnja pa se tudi interaktivno prikaže na zemljevidu - slika 6. [2]



Slika 6: Prikaz podatkov na spletni strani.

1.5 Algoritem za stiskanje zaporedja števil

Algoritem za stiskanje zaporedja števil smo razvili s programskim jezikom python, v razvojnem okolju PyCharm Professional. Za testiranje smo uporabili program Postman. Algoritem je uporabljen v APIju. [1]

1.5.1 Uporaba algoritma

Ko spletni vmesnik dobi zahtevo, se izvedejo naslednje operacije: [1]

- Pridobijo se vrednosti kakovosti ceste iz podatkovne baze.
- Števila se zapišejo v datoteko.
- Izvede se stiskanje podatkov.
- Binaren zapis stisnjenih podatkov v datoteko.
- Pošiljanje datoteke odjemalcu.

Odjemalec lahko po prejemu datoteke izvede dekompresijo in uporabi števila v nadaljnji obdelavi. [1]

1.6 Mikrokrmilnik STM32F3DISCOVERY

Na mikrokrmilniku STM32F3DISCOVERY smo uporabili pospeškometer, s katerim na nizkonivojskem nivoju pridobimo podatek iz senzorja. Program za mikrokrmilnik smo razvili s programskim jezikom C, v razvojnem okolju STM32CubeIDE. Nato smo s pythonom v PyCharmu izdelali konzolno aplikacijo, ki pridobi ta podatek iz mikrokrmilnika preko USB povezave. Za prikaz grafa smo uporabili knjižnico matplotlib, za USB komunikacijo pa knjižnico pyserial. [1]

1.6.1 Uporaba mikrokrmilnika

Ko pritisnemo gumb na mikrokrmilniku, ta prebere vrednosti iz pospeškometra in jih pošlje preko USB povezave. Program na računalniku bere podatke iz serijskega vodila. Ko prejme podatke, jih izpiše na zaslon v obliki grafa.

Literatura

[1] David Slatinek, Marcel Iskrač in Vid Kreča. "Tehnična dokumentacija projekta rt21". 26. dec. 2021. (Pridobljeno 2. 1. 2022).

[2] David Slatinek, Marcel Iskrač in Vid Kreča. "Logična shema projekta rt21". 21. dec. 2021. (Pridobljeno 2. 1. 2022).