Projektna naloga 4

Sistemska administracija

Kazalo:

Uvod

Cilj projekta je izdelati spletno platformo in mobilno aplikacijo, ki voznikom pomaga najti prosta parkirna mesta. Prav tako omogoča izračun oddaljenosti do parkirnega mesta ter poda ustrezne informacije o njem (cena, tip parkirnega mesta, enota plačila...). S to funkcionalnostjo uporabniki aplikacije lahko izognejo gneči in prihranijo čas pri iskanju prostih parkirnih mest v Ljubljani.

Ime projekta:

ParkingMate

Ime skupine:

SVD

Člani skupine:

David Jaćović

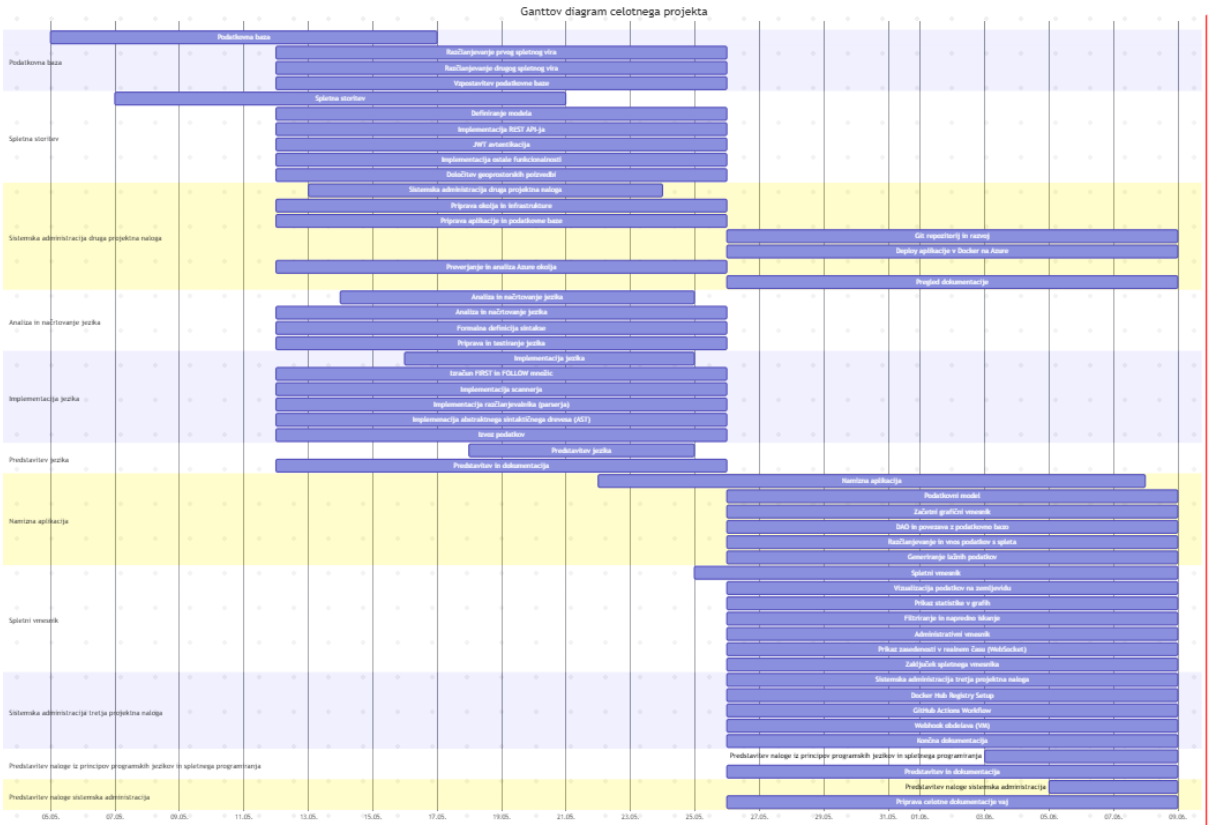
Vojin Keser

Sara Vesković

Povezava do github repozitorija z kodo:

https://github.com/davidjacovic/parkingmate-studentski-projekt

Ganttov dijagram



```
gantt
    title Ganttov diagram celotnega projekta
    dateFormat  YYYY-MM-DD
    axisFormat  %d.%m.

    section Podatkovna baza
    Podatkovna baza                      :db, 2025-05-05, 2025-05-17
    Razčlanjevanje prvog spletnog vira              :db1, 2025-05-12, 2025-05-26
    Razčlanjevanje drugog spletnog vira             :db2, 2025-05-12, 2025-05-26
    Vzpostavitev podatkovne baze                   :db3, 2025-05-12, 2025-05-26

    section Spletna storitev
    Spletna storitev                    :api, 2025-05-07, 2025-05-21
    Definiranje modela                 :api1, 2025-05-12, 2025-05-26
    Implementacija REST API-ja            :api2, 2025-05-12, 2025-05-26
    JWT avtentikacija                 :api3, 2025-05-12, 2025-05-26
    Implementacija ostale funkcionalnosti            :api4, 2025-05-12, 2025-05-26
    Določitev geoprostorskih poizvedbi           :api5, 2025-05-12, 2025-05-26

    section Sistemska administracija druga projektna naloga
    Sistemska administracija druga projektna naloga :sys1, 2025-05-13, 2025-05-24
    Priprava okolja in infrastrukture :sys1a, 2025-05-12, 2025-05-26
    Priprava aplikacije in podatkovne baze       :sys1b, 2025-05-12, 2025-05-26
    Git repozitorij in razvoj         :sys1c, 2025-05-26, 2025-06-09
    Deploy aplikacije v Docker na Azure            :sys1d, 2025-05-26, 2025-06-09
    Preverjanje in analiza Azure okolja          :sys1e, 2025-05-12, 2025-05-26
    Pregled dokumentacije             :sys1f, 2025-05-26, 2025-06-09


    section Analiza in načrtovanje jezika
    Analiza in načrtovanje jezika                    :lang, 2025-05-14, 2025-05-25
    Analiza in načrtovanje jezika          :lang1, 2025-05-12, 2025-05-26
    Formalna definicija sintakse     :lang2, 2025-05-12, 2025-05-26
    Priprava in testiranje jezika               :lang3, 2025-05-12, 2025-05-26

    section Implementacija jezika
    Implementacija jezika             :impl, 2025-05-16, 2025-05-25
    Izračun FIRST in FOLLOW množic             :impl1, 2025-05-12, 2025-05-26
    Implementacija scannerja          :impl2, 2025-05-12, 2025-05-26
    Implementacija razčlanjevalnika (parserja)            :impl3, 2025-05-12, 2025-05-26
    Implemenacija abstraktnega sintaktičnega drevesa (AST)              :impl4, 2025-05-12, 2025-05-26
    Izvoz podatkov                    :impl5, 2025-05-12, 2025-05-26

    section Predstavitev jezika
    Predstavitev jezika               :pres1, 2025-05-18, 2025-05-25
    Predstavitev in dokumentacija        :pres1a, 2025-05-12, 2025-05-26

    section Namizna aplikacija
    Namizna aplikacija                :desk, 2025-05-22, 2025-06-08
    Podatkovni model                  :desk1, 2025-05-26, 2025-06-09
    Začetni grafični vmesnik              :desk2, 2025-05-26, 2025-06-09
    DAO in povezava z podatkovno bazo            :desk3, 2025-05-26, 2025-06-09
    Razčlanjevanje in vnos podatkov s spleta         :desk4, 2025-05-26, 2025-06-09
    Generiranje lažnih podatkov       :desk5, 2025-05-26, 2025-06-09


    section Spletni vmesnik
    Spletni vmesnik                   :web, 2025-05-25, 2025-06-09
    Vizualizacija podatkov na zemljevidu          :web1, 2025-05-26, 2025-06-09
    Prikaz statistike v grafih              :web2, 2025-05-26, 2025-06-09
    Filtriranje in napredno iskanje           :web3, 2025-05-26, 2025-06-09
    Administrativni vmesnik                    :web4, 2025-05-26, 2025-06-09
    Prikaz zasedenosti v realnem času (WebSocket):web5, 2025-05-26, 2025-06-09
    Zaključek spletnega vmesnika               :web6, 2025-05-26, 2025-06-09

    section Sistemska administracija tretja projektna naloga
    Sistemska administracija tretja projektna naloga :sys2, 2025-05-26, 2025-06-09
    Docker Hub Registry Setup          :sys2a, 2025-05-26, 2025-06-09
    GitHub Actions Workflow           :sys2b, 2025-05-26, 2025-06-09
    Webhook obdelava (VM)             :sys2c, 2025-05-26, 2025-06-09
    Končna dokumentacija              :sys2d, 2025-05-26, 2025-06-09

    section Predstavitev naloge iz principov programskih jezikov in spletnega programiranja
    Predstavitev naloge iz principov programskih jezikov in spletnega programiranja          :pres2, 2025-06-03, 2025-06-09
    Predstavitev in dokumentacija           :pres2a, 2025-05-26, 2025-06-09

    section Predstavitev naloge sistemska administracija
    Predstavitev naloge sistemska administracija      :presys3, 2025-06-05, 2025-06-09
    Priprava celotne dokumentacije vaj     :presys4, 2025-05-26, 2025-06-09
```

Primeri uporabe

Problem

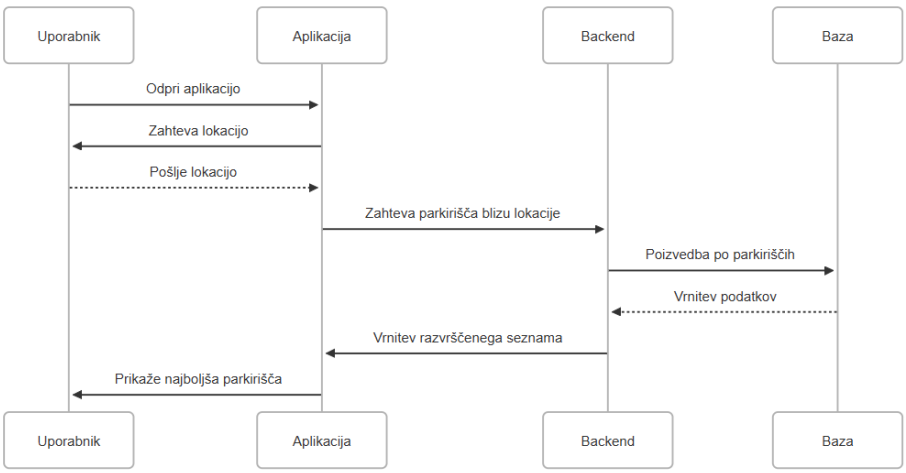
Problem, ki jo rešuje naša programska rešitev *Parking Mate*, je izboljšanje preglednosti in upravljanja mestnih parkirišč. V mestih pogosto prihaja do slabe informiranosti uporabnikov o prostih parkirnih mestih, nepravilno nastavljenih tarifah ter neučinkovitega izkoriščanja parkirnih kapacitet. To vodi do povečane gneče, nepotrebnega kroženja po mestu, izgube časa in večjega onesnaženja. Naš sistem omogoča pregledno vizualizacijo zasedenosti parkirišč, učinkovito upravljanje tarif ter nudi vpogled v uporabniške ocene in izkušnje, kar pripomore k boljši organizaciji in uporabniški izkušnji.

Primeri:

1. Iskanje parkirišča glede na trenutno lokacijo

Uporabnik zažene aplikacijo → sistem pridobi njegovo lokacijo → vrne seznam bližnjih parkirišč z razvrstitvijo po uporabnosti.

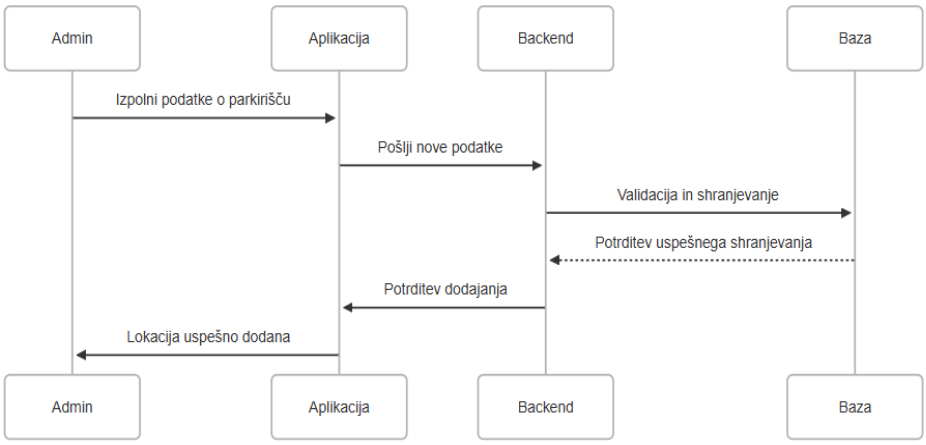
```
sequenceDiagram
    participant Uporabnik
    participant Aplikacija
    participant Backend
    participant Baza

    Uporabnik->>Aplikacija: Odpri aplikacijo
    Aplikacija->>Uporabnik: Zahteva lokacijo
    Uporabnik-->>Aplikacija: Pošlje lokacijo
    Aplikacija->>Backend: Zahteva parkirišča blizu lokacije
    Backend->>Baza: Poizvedba po parkiriščih
    Baza-->>Backend: Vrnitev podatkov
    Backend->>Aplikacija: Vrnitev razvrščenega seznama
    Aplikacija->>Uporabnik: Prikaže najboljša parkirišča
```



2. Dodajanje novega parkirišča (admin funkcija)

Administrator doda novo parkirišče → sistem preveri podatke → shrani novo lokacijo v bazo → uporabniki jo lahko nato vidijo.

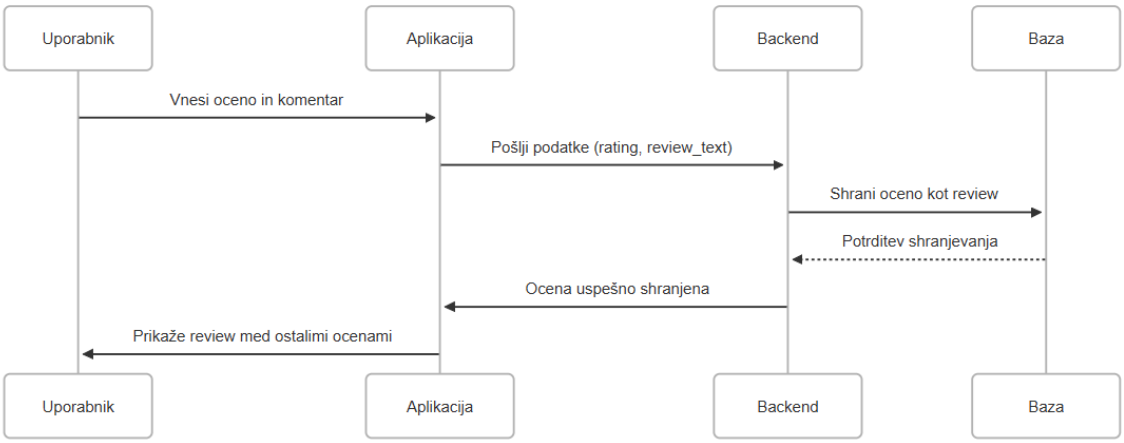
```
sequenceDiagram
    participant Admin
    participant Aplikacija
    participant Backend
    participant Baza

    Admin->>Aplikacija: Izpolni podatke o parkirišču
    Aplikacija->>Backend: Pošlji nove podatke
    Backend->>Baza: Validacija in shranjevanje
    Baza-->>Backend: Potrditev uspešnega shranjevanja
    Backend->>Aplikacija: Potrditev dodajanja
    Aplikacija->>Admin: Lokacija uspešno dodana
```



3. Oddaja ocene za parkirišče

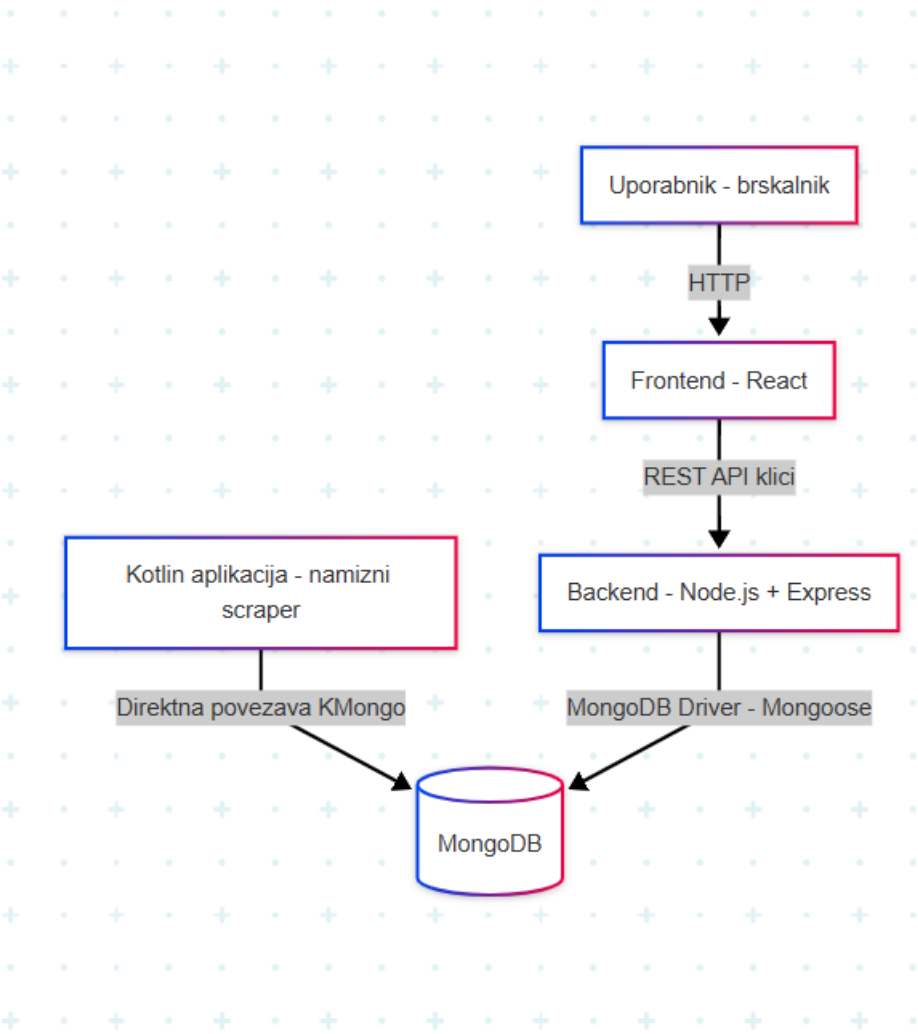
Uporabnik oceni parkirišče → ocena se pošlje v bazo → ocena se izpiše kot pregled (review) parkirišča.

```
sequenceDiagram
    participant Uporabnik
    participant Aplikacija
    participant Backend
    participant Baza

    Uporabnik->>Aplikacija: Vnesi oceno in komentar
    Aplikacija->>Backend: Pošlji podatke (rating, review_text)
    Backend->>Baza: Shrani oceno kot review
    Baza-->>Backend: Potrditev shranjevanja
    Backend->>Aplikacija: Ocena uspešno shranjena
    Aplikacija->>Uporabnik: Prikaže review med ostalimi ocenami
```


Uporabnik
Aplikacija
Backend
Baza
Vnesi oceno in komentar
Pošlji podatke (rating, review_text)
Shrani oceno kot review
Potrditev shranjevanja
Ocena uspešno shranjena
Prikaže review med ostalimi ocenami
Uporabnik
Aplikacija
Backend
Baza

Arhitektura programske rešitve

```
flowchart TD
    Uporabnik[Uporabnik - brskalnik] -->|HTTP| ReactApp[Frontend - React]
    ReactApp -->|REST API klici| Backend[Backend - Node.js + Express]
    KotlinApp[Kotlin aplikacija - namizni scraper] -->|Direktna povezava KMongo| MongoDB[(MongoDB)]
    Backend -->|MongoDB Driver - Mongoose| MongoDB
```



Uporabljene tehnologije

Uporabljen programski jezik:

- JavaScript – za razvoj zaledne storitve (Node.js + Express)
- Kotlin – za namizno aplikacijo (scraper)
- JavaScript (React) – za spletni uporabniški vmesnik

Podatkovna baza:

- MongoDB – nerelacijska NoSQL podatkovna baza, uporabljena za shranjevanje vseh entitet (uporabniki, parkirišča, ocene, vozila ...)

Komunikacija in protokoli:

- HTTP – za komunikacijo med brskalnikom, frontend aplikacijo in backend strežnikom
- MongoDB TCP protokol – za povezavo med backendom (in scraperjem) ter bazo
- Frontend deluje na vratih 3000
- Backend na vratih 3002
- MongoDB posluša na privzetih vratih 27017

Spletni strežnik:

- Express.js – spletni strežnik na Node.js, uporablja REST API za komunikacijo med frontendom in backendom

Knjižnice in API-ji:

Za Node.js (backend):

- Express.js – za upravljanje HTTP zahtevkov in definiranje API poti
- Mongoose – ODM knjižnica za enostavno delo z MongoDB

Za Kotlin scraper:

- KMongo – knjižnica za enostavno povezovanje Kotlin kode z MongoDB brez vmesnega API-ja
- Kotlinx.serialization – za serializacijo objektov

Za React (frontend):

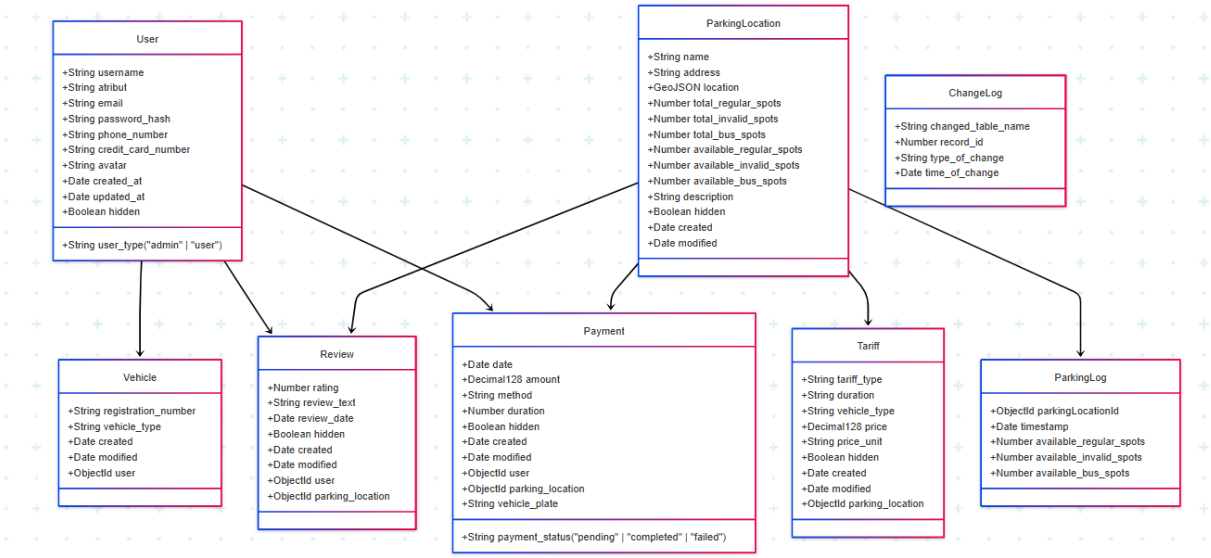
- Leaflet.js – za prikaz zemljevida in izris poti
- Axios/Fetch – za komunikacijo z backendom uporabili

Razlogi za izbiro

Express in React omogočata hitro gradnjo skalabilnih REST sistemov. MongoDB omogoča shranjevanje fleksibilnih strukturiranih/nestrukturiranih podatkov. Kotlin je enostaven za uporabo na namizju in ponuja dober dostop do MongoDB prek Kmongo

Razredni dijagram

Razredni dijagram jezika Node.js



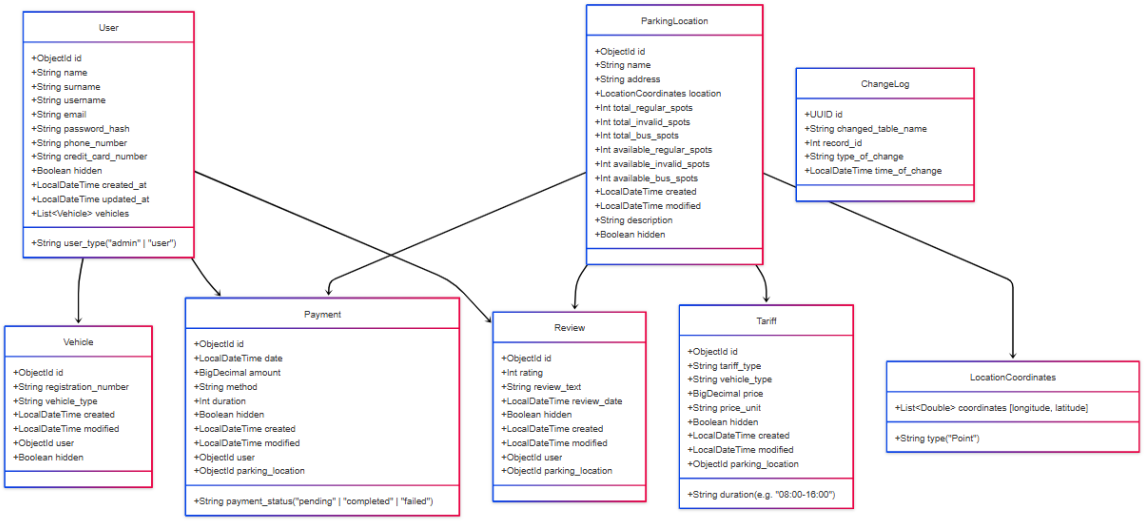

```
classDiagram
direction TB
    class User {
        +String username
        +String atribut
        +String email
        +String password_hash
        +String phone_number
        +String credit_card_number
        +String avatar
        +Date created_at
        +Date updated_at
        +Boolean hidden
        +String user_type("admin" | "user")
    }

    class Vehicle {
        +String registration_number
        +String vehicle_type
        +Date created
        +Date modified
        +ObjectId user
    }
```

```
class Review {
    +Number rating
    +String review_text
    +Date review_date
    +Boolean hidden
    +Date created
    +Date modified
    +ObjectId user
    +ObjectId parking_location
}

class Tariff {
    +String tariff_type
    +String duration
    +String vehicle_type
    +Decimal128 price
    +String price_unit
    +Boolean hidden
    +Date created
    +Date modified
    +ObjectId parking_location
}
```

```
class Payment {
    +Date date
    +Decimal128 amount
    +String method
    +Number duration
    +Boolean hidden
    +Date created
    +Date modified
    +ObjectId user
    +ObjectId parking_location
    +String vehicle_plate
    +String payment_status("pending" | "completed" | "failed")
}

class ChangeLog {
    +String changed_table_name
    +Number record_id
    +String type_of_change
    +Date time_of_change
}

User --> Vehicle
User --> Review
ParkingLocation --> Review
ParkingLocation --> Tariff
ParkingLocation --> ParkingLog
User --> Payment
ParkingLocation --> Payment
```

```
class ParkingLocation {
    +String name
    +String address
    +GeoJSON location
    +Number total_regular_spots
    +Number total_invalid_spots
    +Number total_bus_spots
    +Number available_regular_spots
    +Number available_invalid_spots
    +Number available_bus_spots
    +String description
    +Boolean hidden
    +Date created
    +Date modified
}

class ParkingLog {
    +ObjectId parkingLocationId
    +Date timestamp
    +Number available_regular_spots
    +Number available_invalid_spots
    +Number available_bus_spots
}
```

Razredni dijagram jezika Kotlin




```
classDiagram

class User {
    +ObjectId id
    +String name
    +String surname
    +String username
    +String email
    +String password_hash
    +String phone_number
    +String credit_card_number
    +String user_type ("admin" | "user")
    +Boolean hidden
    +LocalDateTime created_at
    +LocalDateTime updated_at
    +List~Vehicle~ vehicles
}

class Vehicle {
    +ObjectId id
    +String registration_number
    +String vehicle_type
    +LocalDateTime created
    +LocalDateTime modified
    +ObjectId user
    +Boolean hidden
}

class Review {
    +ObjectId id
    +Int rating
    +String review_text
    +LocalDateTime review_date
    +Boolean hidden
    +LocalDateTime created
    +LocalDateTime modified
    +ObjectId user
    +ObjectId parking_location
}

class Tariff {
    +ObjectId id
    +String tariff_type
    +String duration (e.g. "08:00-16:00")
    +String vehicle_type
    +BigDecimal price
    +String price_unit
    +Boolean hidden
    +LocalDateTime created
    +LocalDateTime modified
    +ObjectId parking_location
}

class Payment {
    +ObjectId id
    +LocalDateTime date
    +BigDecimal amount
    +String method
    +String payment_status ("pending" | "completed" | "failed")
    +Int duration
    +Boolean hidden
    +LocalDateTime created
    +LocalDateTime modified
    +ObjectId user
    +ObjectId parking_location
}

class LocationCoordinates {
    +String type ("Point")
    +List~Double~ coordinates [longitude, latitude]
}
```

```
class ParkingLocation {
    +ObjectId id
    +String name
    +String address
    +LocationCoordinates location
    +Int total_regular_spots
    +Int total_invalid_spots
    +Int total_bus_spots
    +Int available_regular_spots
    +Int available_invalid_spots
    +Int available_bus_spots
    +LocalDateTime created
    +LocalDateTime modified
    +String description
    +Boolean hidden
}

class ChangeLog {
    +UUID id
    +String changed_table_name
    +Int record_id
    +String type_of_change
    +LocalDateTime time_of_change
}

%% Relationships
User --> Vehicle
User --> Payment
User --> Review
ParkingLocation --> Review
ParkingLocation --> Tariff
ParkingLocation --> Payment
ParkingLocation --> LocationCoordinates
```

V Node.js definiramo modele z mongoose.Schema, dostop do MongoDB pa gre prek API-ja. Relacije so določene z ref, logika pa je večinoma zunaj modela. V Kotlinu uporabljamo data class in KMongo, kjer scraper piše direktno v bazo brez API-ja. Oba hranita iste podatke, le način dostopa je drugačen.

Varnost programske rešitve

Za zagotavljanje varnosti spletne aplikacije *ParkingMate* smo implementirali več zaščitnih mehanizmov, ki varujejo tako strežniško infrastrukturo kot tudi podatke uporabnikov.

Uporaba požarnega zidu

Na strežniškem okolju (Azure VM) je konfiguriran požarni zid (firewall), ki omogoča dostop le do potrebnih vrat:

- Port 3000 – za frontend (React aplikacija)
- Port 3002 – za backend (Node.js + Express)

Vsa druga vrata so zaprta za preprečevanje nepooblaščenega dostopa.

Omejitve uporabnikov

Uporabniški dostop je omejen glede na avtorizacijo:

- Gostje lahko dostopajo le do javnih podatkov o parkiriščih.
- Registrirani uporabniki lahko ocenjujejo parkirišča, dodajajo ocene in dostopajo do svojega profila.
- Administratorji imajo razširjen dostop za urejanje podatkov, dodajanje lokacij ter pregled uporabniških informacij.

Vse operacije, ki zahtevajo spremembe podatkov, so zaščitene z ustreznimi preverjanji pravic uporabnika.

Varnostne vloge uporabnikov

Uporabniki so razdeljeni v naslednje vloge:

- USER – navaden uporabnik
- ADMIN – administrator, ki ima nadzor nad vsemi entitetami v sistemu

Dostop do občutljivih API-jev je omogočen le uporabnikom z vlogo ADMIN. Vloge se preverjajo na strani strežnika pred vsako varnostno kritično operacijo.

Varovanje podatkov

Za varovanje podatkov uporabljamo večplastni pristop:

- Gesla se šifrirajo z algoritmom bcrypt pred shranjevanjem v bazo (hash + salt).
- JWT (JSON Web Token) se uporablja za varno avtentikacijo uporabnikov in zaščito API klicev.

- MongoDB je dostopna le preko preverjene povezave, z ločenim geslom in po potrebi IP omejitvami.