# Seminarski Rad

**Fakultet:** Fakultet Tehnickih Nauka u Novom Sadu
**Smer:** Softverske i informacione tehnologije

## Tema: Dizajn i implementacija mikroservisa za upravljanje saobraćajne policije

**Profesor:**
Marko Markovic

**Student:**
Miodrag

Ugrica,

SR7/2022

**Mesto, Godina**
Novi Sad, 2025.

## Sažetak

U ovom projektu prezentovana je implementacija softverskog sistema i njenog dizajna. Sistem je razvijen radi podrške saobraćajne policije. Programski jezik Java je upotrebljen za izradu logike obrade podataka koja se čuva u MongoDB, dok je za vizuelni prikaz i interakciju sa korisnicima primenjen Angular okvir. Kako bi aplikacija mogla da se pokrene, Docker platforma je korišćena za kontejnerizaciju aplikacije. Implementacijom ovog rešenja omogućeno je evidentiranje saobraćajnih kazni, upravljanje podacima o vlasnicima vozila i pregled relevantnih informacija u okviru aplikacije.

Ključne reči: mikroservisna veb aplikacija, obrada kazni, saobraćajna policija

---

***Sadržaj***

# 1. Uvod

U savremenom društvu digitalizacija je postala proces koji je nemoguće izbeći. Obuhvata gotovo sve strukture života i rada. Državne institucije, uključujući i policiju, suočavaju se sa sve većom obavezom da svoje zadatke obavljaju brže, efikasnije i pouzdanije. Tradicionalni načini evidentiranja i obrade podataka često su spori, zastareli, podložni greškama i zahtevaju mnogo resursa. Upravo iz tog razloga, digitalna rešenja predstavljaju logičan korak unapređenja. Kako se digitalizuju bankarstvo, zdravstvo i obrazovanje, tako je i u oblasti policijskih poslova neophodno uvođenje informacionih sistema koji omogućavaju efikasnije upravljanje kaznama, vozilima i vlasnicima. Ovaj seminarski rad ima za cilj da prikaže jedno takvo rešenje kroz aplikaciju *Traffic Police*, koje je razvijeno kao primer modernog pristupa digitalizaciji policijskih procesa.

# 2. Korišćene tehnologije

U ovom poglavlju prikazan je skup alata i programskih okvira na kojima se zasniva izgrađeno rešenje.

Java je višenamenski jezik zasnovan na objektno-orijentisanom paradigmom, poznat po pouzdanosti, mogućnosti izvođenja na različitim platformama i širokoj upotrebi u razvoju serverskih sistema. U okviru ovog projekta iskorišćen je za kreiranje poslovnih pravila i implementaciju serverskih servisa koji vrše razmenu podataka sa bazom i korisničkim interfejsom.

MongoDB je nerelacioni sistem za upravljanje podacima, koji informacije skladišti u formi dokumenata sličnih JSON formatu. Njegova prednost ogleda se u mogućnosti rada sa nestrukturisanim i delimično strukturisanim sadržajem, što ga čini pogodnim za čuvanje informacija o prekršajima, vozilima i vlasnicima.

Angular je razvojni okvir za izgradnju klijentske strane veb aplikacija, čiji je razvoj inicirala i održava kompanija Google. Zasniva se na jeziku TypeScript i omogućava konstruisanje naprednih aplikacija sa jednom stranicom. U ovom radu Angular je upotrebljen za oblikovanje korisničkog interfejsa, kroz koji policijski službenici imaju pristup podacima i funkcionalnostima sistema.

Docker je softverska platforma namenjena kontejnerizaciji, to jest izolovanom pokretanju aplikacija zajedno sa svim potrebnim zavisnostima. Na ovaj način postiže se jednostavnija instalacija i održavanje sistema u različitim okruženjima.

Za istovremeno upravljanje više kontejnera korišćen je dodatni alat Docker Compose.

# 3. Specifikacija zahteva

## 3.1 Funkcionalni zahtevi

U ovom odeljku predstavljeni su funkcionalni zahtevi koje razvijeno softversko rešenje treba da ispuni. Funkcionalni zahtevi definišu šta sistem treba da omogućava krajnjim korisnicima, odnosno koje operacije i scenarije korišćenja mora da podrži. Oni predstavljaju osnovu za dalju implementaciju i testiranje, jer jasno opisuju ponašanje sistema u tipičnim situacijama.

Korisnici sistema su policijski službenici različitih činova (LOW, MEDIUM, HIGH), pri čemu svaki rang ima različit nivo ovlašćenja. Na primer, policajac ranga LOW može unositi prekršaje, MEDIUM ima dodatne mogućnosti poput dodeljivanje prekršaja drugom policajcu, dok HIGH ima administratorske privilegije kao što su dodavanje unapređenje policajaca ili suspenzija istih.

Za potrebe ovog rada, funkcionalni zahtevi su prikazani kroz tabelu (Tabela 1) koja sadrži opis slučajeva korišćenja, uključujući njihove učesnike, preduslove, korake, rezultat i izuzetke.

| Naziv | Učesnici | Preduslovi | Koraci | Rezultat | Izuzeci |
|---|---|---|---|---|---|
| Pregled svih policajaca | Policajac (HIGH) | Korisnik je ulogovan kao korisnik sa najvišim činom | 1. Korisnik bira opciju „Policajci“. 2. Sistem dohvaća podatke iz baze. | Lista policajaca se prikazuje korisniku. | Policajac ne postoji; greška pri dohvatanju podataka. |
| Suspenzija policajca | Policajac (HIGH) | Policajac postoji u bazi | 1.Administrator bira policajca. 2. Sistem menja status na „isSuspended = true“. | Policajac je suspendovan. | Policajac već suspendovan. |

| | | | | | |
|---|---|---|---|---|---|
| Promocija policajca | Policajac (HIGH) | Policajac postoji u bazi | 1.Administrator bira policajca. 2. Sistem menja rang (LOW → MEDIUM → HIGH). | Policajac je unapređen. | Policajac ima najviši rang |
| Pregled svih kazni | Policajac (HIGH) | Postoje kazne u sistemu | 1. Policajac bira opciju „Kazne“. 2. Sistem dohvaća sve kazne iz baze. | Lista kazni se prikazuje. | Greška pri dohvatanju podataka; prazna lista. |
| Prikaz svih kazni koje su povezane sa prekršajem dodeljenim određenom policijskom službeniku | Policajac (LOW/MEDIUM) | Postoje kazne u sistemu. | 1. Policajac bira opciju „Kazne“. 2. Sistem dohvaća odgovarajuće kazne iz baze. | Lista kazni se prikazuje | Greška pri dohvatanju podataka; prazna lista. |
| Označavanje kazne kao plaćene | Policajac | Kazna postoji u sistemu | 1. Policajac bira kaznu. 2. Sistem menja status „isPaid = true“. | Kazna je označena kao plaćena. | Kazna je već plaćena. |
| Evidencija prekršaja | Policajac | Policajac je prijavljen | 1. Policajac otvara formu. 2. Unosi podatke. 3. Ako tip nije odabran, sistem dodeljuje „MINOR“. | Novi prekršaj je sačuvan u bazi. | Nevalidni podaci. |
| Dodela policajca prekršaju | Policajac (MEDIUM/HIGH) | Prekršaj postoji u sistemu | 1. Policajac bira prekršaj. 2. Bira policajca. 3. Sistem ažurira podatke. | Prekršaj je dodeljen policajcu. | Greška baze. |

| | | | | | |
|---|---|---|---|---|---|
| Pregled statusa vozila | Policajac | Vozilo postoji u sistemu | 1. Policajac bira vozilo.<br>2. Sistem dohvaća podatke. | Prikazuje se status vozila (ukradeno ili ne). | Greška baze. |
| Prijava vozila kao ukradenog | Policajac | Vozilo postoji u sistemu | 1. Policajac bira vozilo.<br>2. Označava status „stolen = true“. | Vozilo je prijavljeno kao ukradeno. | Vozilo već prijavljeno; vozilo ne postoji. |
| Pregled dnevne statistike | Policajac | Policajac je prijavljen | 1. Policajac bira opciju „Statistika“.<br>2. Sistem prikazuje dnevne prekršaje | Prikazuje se broj prekršaja po dnevnom nivou. | Greška pri obradi podataka; nema podataka. |
| Izvoz podataka u PDF/CSV | Policajac (HIGH) | Postoje evidentirani prekršaji | 1. Policajac bira period i format.<br>2. Sistem kreira fajl. | Dokument se generiše i preuzima. | Nepostojeći format; nema podataka u tom periodu. |

Tabela 1 – Prikaz slučajeva korišćenja

## 3.2 Nefunkcionalni zahtevi

Softversko rešenje mora biti razvijeno na način koji obezbeđuje modularnost i skalabilnost, zbog čega je izabrana mikroservisna arhitektura. Upravljanje korisničkim nalozima i proces autentifikacije realizuje se kroz jedinstveni sistem prijave, čime se postiže veća sigurnost i jednostavnije održavanje. Komunikacija među servisima ostvaruje se isključivo preko REST interfejsa, uz obavezno obezbeđivanje više različitih razmena podataka tokom rada. Celokupna aplikacija se distribuira i pokreće unutar kontejnerskog okruženja zasnovanog na Docker tehnologiji, čime se postiže lakše upravljanje verzijama. Posebna pažnja posvećena je bezbednosti, korisničke lozinke se ne čuvaju u izvornom obliku, već u heširanom formatu, dok se pristup zaštićenim delovima sistema odobrava isključivo korisnicima sa validnim JWT tokenom.

# 4. Specifikacija dizajna

Sistem je implementiran kao mikroservisna arhitektura, sa tri nezavisna, ali međusobno povezana servisa. Svaki servis je razvijen u programskom jeziku Java i obavlja specifične funkcionalnosti unutar sistema:
Servis za upravljanje vozilima
Ovaj servis je odgovoran za evidenciju vozila, podatke o vlasnicima, tehničke karakteristike i status vozila. Servis omogućava CRUD operacije nad bazom podataka vozila i pruža API-je za interakciju sa drugim servisima, posebno sa saobraćajnom policijom.

Servis za saobraćajnu policiju
Servis saobraćajne policije upravlja podacima o prekršajima i kaznama, kontroli saobraćaja i povezivanju evidentiranih vozila sa odgovarajućim prekršajima. Komunikacija sa servisom vozila se ostvaruje preko definisanih REST API-ja, čime se omogućava sigurna i efikasna razmena podataka.

Autentifikacioni servis
Ovaj servis pruža funkcionalnosti autentifikacije i autorizacije korisnika sistema. On osigurava da samo ovlašćeni korisnici mogu da pristupe određenim funkcionalnostima i servisima. Servis koristi standardne sigurnosne protokole i omogućava centralizovano upravljanje korisničkim pravima i ulogama.

Svaki servis je dizajniran da bude nezavisan i skalabilan, što omogućava jednostavno održavanje i nadogradnju sistema u budućnosti. Integracija između servisa je realizovana preko REST API-ja i standardnih JSON formata, čime se obezbeđuje interoperabilnost i fleksibilnost sistema.

# 5. Implementacija

## 5.1 Struktura projekta

Realizacija rešenja ostvarena je upotrebom Spring Boot okvira, koji je dozvolio lako formiranje REST usluga i povezivanje sa skladištem podataka MongoDB. Svaki servis implementiran je kao odvojena Spring Boot aplikacija, sa vlastitim konfiguracionim fajlovima i autonomnim build procesom.

Struktura aplikacije je tipična struktura Spring Boot aplikacije, gde postoji tri sloja: controller, service, repo. Kontroler izlaže Rest API krajnjim korisnicima, ali i drugim servisima. Servisni sloj uspostavlja granice aplikacije i opseg dostupnih funkcija. On sadrži poslovnu logiku aplikacije, upravljanje transakcijama i koordiniranje odgovora tokom rada. Repozitorijum je sloj koji izoluje pristup

podacima i pruža centralizovani pristup upravljanju aktivnostima podataka.

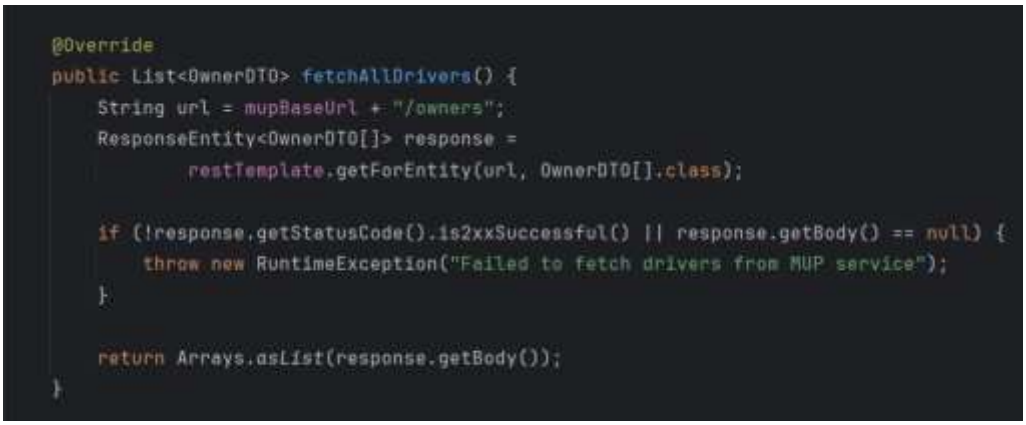
Primer REST kontroler, koji je prikazan na slici 1, implementiran je pomoću anotacija @RestController, čime se označava da klasa izlaže REST API metode, i @RequestMapping("/api/fines") kojim se definiše putanja ka resursima.
Metode unutar klase kontroler su obeležene anotacijama kao što su GetMapping, kojim se mapira HTTP Get zahtev, i PatchMapping, koji mapira HTTP Patch zahtev. Svaka metoda vraća rezultat u formi objekta ResponseEntity, čime se obezbeđuje jasan odgovor klijentu, zajedno sa HTTP status kodom.

```
@RestController
@RequestMapping("/api/fines")
public class FineController {

    @Autowired
    private TrafficPoliceService trafficPoliceService;

    @GetMapping
    public ResponseEntity<List<Fine>> getAllFines() {
        return ResponseEntity.ok(trafficPoliceService.getAllFines());
    }

    @PatchMapping("/{id}")
    public ResponseEntity<Void> updateFine(@PathVariable String id) {
        trafficPoliceService.markFineAsPaid(id);
        return ResponseEntity.ok().build();
    }

    @GetMapping("/unpaid/{jmbg}")
    public ResponseEntity<List<Fine>> getUnpaidFines(@PathVariable String jmbg) {
        return ResponseEntity.ok(trafficPoliceService.findUnpaidFinesByDriverId(jmbg));
    }

}
```

Slika 1 – Primer REST kontrolera

## 5.2 Međuservisna komunikacija

Za komunikaciju između mikroservisa korišćen je RestTemplate, mehanizam koji omogućava da jedan servis uspostavi HTTP vezu i pozove REST API drugog servisa. Na ovaj način obezbeđena je razmena podataka i koordinacija funkcionalnosti među nezavisnim komponentama sistema. Na primer, na slici 2 prikazana je metoda u okviru servisa saobraćajne policije koja upućuje zahtev ka MUP servisu i kao odgovor dobija listu vozila.



```java
@Override
public List<OwnerDTO> fetchAllDrivers() {
    String url = mupBaseUrl + "/owners";
    ResponseEntity<OwnerDTO[]> response =
            restTemplate.getForEntity(url, OwnerDTO[].class);

    if (!response.getStatusCode().is2xxSuccessful() || response.getBody() == null) {
        throw new RuntimeException("Failed to fetch drivers from MUP service");
    }

    return Arrays.asList(response.getBody());
}
```

Slika 2 – Medjuservisna komunikacija

### 5.3 Docker

Svi servisi su kontejnerizovani korišćenjem Docker-a. Za orkestraciju i zajedničko pokretanje servisa koristi se Docker Compose, gde je definisan fajl docker-compose.yml sa konfiguracijom za svaki servis i MongoDB bazu podataka.

## 6. Demonstracija

Poglavlje daje prikaz rada aplikacije kroz primer upotrebe koji integriše delovanje svih servisa.

### 6.1. Scenario evidentiranja saobraćajnog prekršaja

Scenario započinje nakon što se policijski službenik uspešno prijavio u sistem putem Autentifikacionog servisa.

Pokretanje zahteva od strane policijskog službenika:
Policajac se prijavljuje na sistem i na svom panelu unosi podatke o vozilu (model, boja, marka i/ili registracija).

Provera vozila putem MUP servisa:
Sistem automatski šalje zahtev ka MUP Vozila servisu, koji vraća listu vozila koja odgovaraju kriterijumima. Policijski službenik na svom panelu vidi detalje o pojedinačnim vozilima.

Evidentiranje prekršaja u Saobraćajnoj policiji:
Policijski službenik unosi detalje o prekršaju (vreme kada se prekršaj desio,

lokacija i tip prekršaja). Sistem prosleđuje zahtev servisu Saobraćajne policije, koji čuva podatke o prekršaju, generiše novu kaznu i šalje email vozaču da je nastao prekršaj.

Pregled kazne od strane policijskog službenika:
Nakon što je kazna uspešno evidentirana, policajcu se na ekranu prikazuje potvrda o unetom prekršaju i informacija da je kazna povezana sa vozačem i vozilom. Status kazne (plaćeno/neplaćeno) može se pratiti u realnom vremenu.

## 7. Zaključak

Cilj ovog projekta bio je da prikaže fleksibilno i savremeno rešenje namenjeno saobraćajnoj policiji, gde su brzina, tačnost i ažurnost ključni faktori. Postojeća rešenja nisu u potpunosti prilagođena potrebama na terenu i često otežavaju efikasno obavljanje posla. Ova aplikacija uvodi modernizovan pristup, omogućava bržu obradu podataka, jednostavniji pristup informacijama i efikasniju komunikaciju među službenicima. Time se obezbeđuje pouzdan alat koji može odgovoriti na dinamične izazove i doprineti većoj bezbednosti u saobraćaju.

Nedostatak projekta ogleda se u tome što nisu obuhvaćeni krajnji korisnici, odnosno građani koji dobijaju kazne. Sistem bi se mogao unaprediti omogućavanjem pristupa korisnicima njihovim podacima, uključujući informacije o izdatim kaznama i druge relevantne detalje. Na taj način postigla bi se veća transparentnost u radu saobraćajne policije, olakšalo praćenje sopstvenih obaveza i omogućila brža komunikacija između građana i institucija. Pored toga, sistem bi mogao da podrži funkcionalnosti kao što su notifikacije o novim ili neplaćenim kaznama, mogućnost plaćanja preko interneta, kao i opcija za prijavu grešaka ili nepravilnosti, čime bi se dodatno unapredilo korisničko iskustvo i efikasnost celokupnog sistema.

## 8. Literatura


1. Oracle. (n.d.). Java documentation. Oracle. Preuzeto 26. 09. 2025. sa https://docs.oracle.com/en/java/

2. Docker. (n.d.). Docker documentation. Docker. Preuzeto 26. 09. 2025. sa https://docs.docker.com/

3. Angular. (n.d.). Overview. Angular. Preuzeto 26. 09. 2025. sa https://angular.dev/overview

4. Paraschiv, E. (16. 05. 2013.). Spring @RequestMapping annotation. Baeldung. Pregledao E. Martin. Preuzeto 26. 09. 2025. sa https://www.baeldung.com/spring-requestmapping

5. Baeldung. (11. 05. 2024.). Spring @Controller vs @RestController. Baeldung. Reviewed by M. Aibin. Preuzeto 26. 09. 2025. sa https://www.baeldung.com/spring-controller-vs-restcontroller

6. Microservices.io. (n.d.). Home. Preuzeto 26. 09. 2025. sa https://microservices.io/

7. MongoDB. (n.d.). MongoDB documentation. MongoDB. Preuzeto 26. 09. 2025. sa https://www.mongodb.com/docs/