

Univerzitet u Novom Sadu,  
Fakultet tehničkih nauka

# SEMINARSKI RAD

Nastavni predmet: Tehnologije i sistemi eUprave

Naziv teme: Aplikacija za upravljanje saobraćajnom policijom u okviru eUprave

Student,

Nataša Kotaranin SR 22/2021

Jun, 2024

<b>1. Sažetak .....</b>	<b>3</b>
<b>2. Ključne reči.....</b>	<b>4</b>
<b>3. Uvod .....</b>	<b>5</b>
<b>4. Srodna rešenja i pregled korišćenih tehnologija .....</b>	<b>5</b>
<b>4.1. Srodna rešenja .....</b>	<b>6</b>
<b>4.2. Korišćene tehnologije .....</b>	<b>6</b>
<b>5. Specifikacija zahteva .....</b>	<b>8</b>
<b>5.1. Specifikacija funkcionalnih zahteva .....</b>	<b>8</b>
<b>5.2. Specifikacija nefunkcionalnih zahteva .....</b>	<b>13</b>
<b>6. Specifikacija dizajna .....</b>	<b>14</b>
<b>7. Implementacija .....</b>	<b>16</b>
<b>8. Demonstracija .....</b>	<b>26</b>
<b>9. Zaključak .....</b>	<b>38</b>
<b>10. Literatura.....</b>	<b>39</b>

## 1. Sažetak

Ovaj rad prikazuje aplikaciju za upravljanje saobraćajnom policijom unutar sistema eUprave. Za implementaciju su korišćeni programski jezik Go [1] za serversku stranu, Angular [2] za korisnički interfejs, MongoDB [3] kao baza podataka, HDFS Hadoop [4] za skladištenje slika, Docker [5] za mikroservisnu arhitekturu i single sign-on (SSO) za korisničku autentifikaciju. Predloženo rešenje omogućava efektivnije i fleksibilnije upravljanje saobraćajnim prekršajima i nesrećama, kao i jednostavniju komunikaciju između policije i građana, kao i jednostavniju komunikaciju između saobraćajne policije i MUP-a. Razvijen je prototip sistema koji pruža funkcionalnosti kao što su evidentiranje prekršaja i saobraćajnih nesreća, generisanje PDF dokumenata [6], slanje email obaveštenja [7] i praćenje statusa prekršaja.

## **2. Ključne reči.**

- eUprava
- mikroservisna arhitektura
- saobraćajna policija
- autentifikacija
- skladištenje slika

### 3. Uvod

Uvođenje informacionih sistema u okviru eUprave predstavlja ključni korak ka modernizaciji i efikasnosti javnih usluga. Jedan od izazova sa kojim se suočavaju državne institucije širom sveta jeste efikasno upravljanje saobraćajnim prekršajima i nesrećama. Tradicionalni pristupi često su ograničeni sporim administrativnim procesima, nedostatkom transparentnosti i neadekvatnom upotrebom resursa.

Ovaj rad se bavi razvojem aplikacije za upravljanje saobraćajnom policijom unutar sistema eUprave. Cilj aplikacije je pružiti efikasan mehanizam za evidenciju, praćenje i rešavanje saobraćajnih incidenata, omogućavajući policajcima brz pristup relevantnim informacijama i građanima transparentno rešavanje prekršajnih postupaka.

Problemi u upravljanju saobraćajnim prekršajima i nesrećama postaju sve izraženiji sa porastom broja vozila i kompleksnošću urbanih saobraćajnih mreža. Tradicionalne metode često nisu adekvatne za efikasno upravljanje ovim izazovima, što dovodi do nezadovoljstva građana i administrativnih prepreka u radu saobraćajne policije.

Kroz implementaciju ove aplikacije, ne samo da se rešavaju trenutni problemi već se postavljaju temelji za buduće inovacije u oblasti eUprave. Dalje unapređenje digitalnih sistema za upravljanje saobraćajnom policijom neophodno je za poboljšanje bezbednosti u saobraćaju i efikasnije korišćenje javnih resursa.

U narednim poglavljima, detaljno će se razmatrati arhitektura aplikacije, specifikacija zahteva, proces dizajna, implementacija i demonstracija, sa ciljem da se pruži sveobuhvatan uvid u tehnološke aspekte i praktičnu primenu rešenja za unapređenje saobraćajne policije u okviru eUprave.

## **4. Srodna rešenja i pregled korišćenih tehnologija**

U ovom poglavlju je dat pregled postojećih rešenja za prikupljanje i analizu podataka relevantnih za saobraćajnu policiju, kao i tehnologije koje omogućavaju rešavanje problema statističke obrade podataka.

### **4.1. Srodna rešenja**

Ovaj odeljak donosi pregled aplikacija i sistema namenjenih prikupljanju i analizi saobraćajnih podataka, kao i korišćene tehnologije za unapređenje saobraćajnih sistema.

#### **Mumbai Traffic Police**

Mumbai Traffic Police (MTP) predstavlja primer napredne integracije tehnologije u upravljanju gradskim saobraćajem. MTP je implementirao napredne sisteme kao što su elektronski sistemi upravljanja prekršajima (E-Challan), video nadzor, kao i aplikacije za upravljanje saobraćajem kao što su MTP VMS i MTP SMC. Ovi sistemi omogućavaju brzu detekciju prekršaja i efikasno upravljanje saobraćajem, čime doprinose smanjenju saobraćajnih gužvi i poboljšanju bezbednosti na putevima. Detaljnije informacije o MTP-u mogu se pronaći na zvaničnom sajtu Mumbai Traffic Police [8].

#### **Tripura Traffic Police**

Tripura Traffic Police koristi modernu tehnologiju za upravljanje saobraćajem u gradu Agartali, uključujući instalaciju elektronskih semafora, video nadzora, i sistem za automatsko prepoznavanje registarskih tablica vozila (ANPR). Implementacija sistema poput inteligentnog sistema upravljanja saobraćajem (ITMS) i virtuelnog suda pokazuju njihovu posvećenost unapređenju efikasnosti u kontroli saobraćajnih prekršaja i upravljanju gradskim saobraćajem. Više informacija o Tripura Traffic Police može se naći na zvaničnom sajtu Tripura Traffic Police [9].

### **4.2. Korišćene tehnologije**

Ovaj odeljak pruža pregled aplikacija i sistema za prikupljanje i analizu saobraćajnih podataka, kao i tehnologija koje se koriste za unapređenje saobraćajnih sistema.

#### **Go (Golang)**

Go [1] je efikasan programski jezik razvijen od strane Google-a, poznat po jednostavnosti i sposobnosti za paralelno izvršavanje. Koristi se za razvoj backend mikroservisa u aplikaciji saobraćajne policije zbog svoje brzine izvršavanja i minimalne potrošnje resursa, što doprinosi skalabilnosti i performansama sistema.

#### **Angular**

Angular [2] je popularan framework za razvoj veb aplikacija, razvijen od strane Google-a. Omogućava dinamičke i responzivne korisničke interfejsse, što olakšava efikasno upravljanje korisničkim zahtevima prema backend servisima. Modularna arhitektura Angular-a doprinosi lakšem održavanju i proširivanju kompleksnih aplikacija kroz ponovnu upotrebu komponenti.

## **Docker**

Docker [5] je platforma za kontejnerizaciju aplikacija koja omogućava pakovanje, distribuciju i pokretanje softverskih aplikacija u izolovanim kontejnerima. Ova tehnologija olakšava razvoj i implementaciju mikroservisne arhitekture, pružajući lako skaliranje i upravljanje servisima uz konzistentnost okruženja od razvoja do produkcije.

## **Single Sign-On (SSO)**

Single Sign-On (SSO) je metodologija autentifikacije koja omogućava korisnicima da se prijave u više aplikacija sa jednim setom akreditiva. Uveden je SSO kako bi se poboljšalo korisničko iskustvo i povećala sigurnost sistema, eliminirajući potrebu za upravljanjem više lozinki.

## **MongoDB**

MongoDB [3] je NoSQL baza podataka koja se koristi zbog svoje fleksibilnosti i skalabilnosti. U sistemu saobraćajne policije, MongoDB služi za perzistenciju podataka vezanih za saobraćajne prekršaje i administrativne informacije, omogućavajući efikasno upravljanje podacima u realnom vremenu.

## **Hadoop HDFS**

Hadoop HDFS [4] se koristi za skladištenje slika i drugih binarnih podataka u sistemu. Ova tehnologija omogućava efikasno upravljanje velikim količinama podataka i osigurava njihovu dostupnost i pouzdanost, što je ključno za operacije saobraćajne policije koje zahtevaju sigurno čuvanje i brzu obradu podataka.

## **PDF dokument**

Za generisanje PDF dokumenata [6] koristi se biblioteka [github.com/jung-kurt/gofpdf](https://github.com/jung-kurt/gofpdf), koja omogućava dinamičko kreiranje i formatiranje PDF fajlova direktno iz aplikacije. Ova funkcionalnost je ključna za generisanje izveštaja i dokumentacije vezane za saobraćajne prekršaje.

## **Slanje emailova**

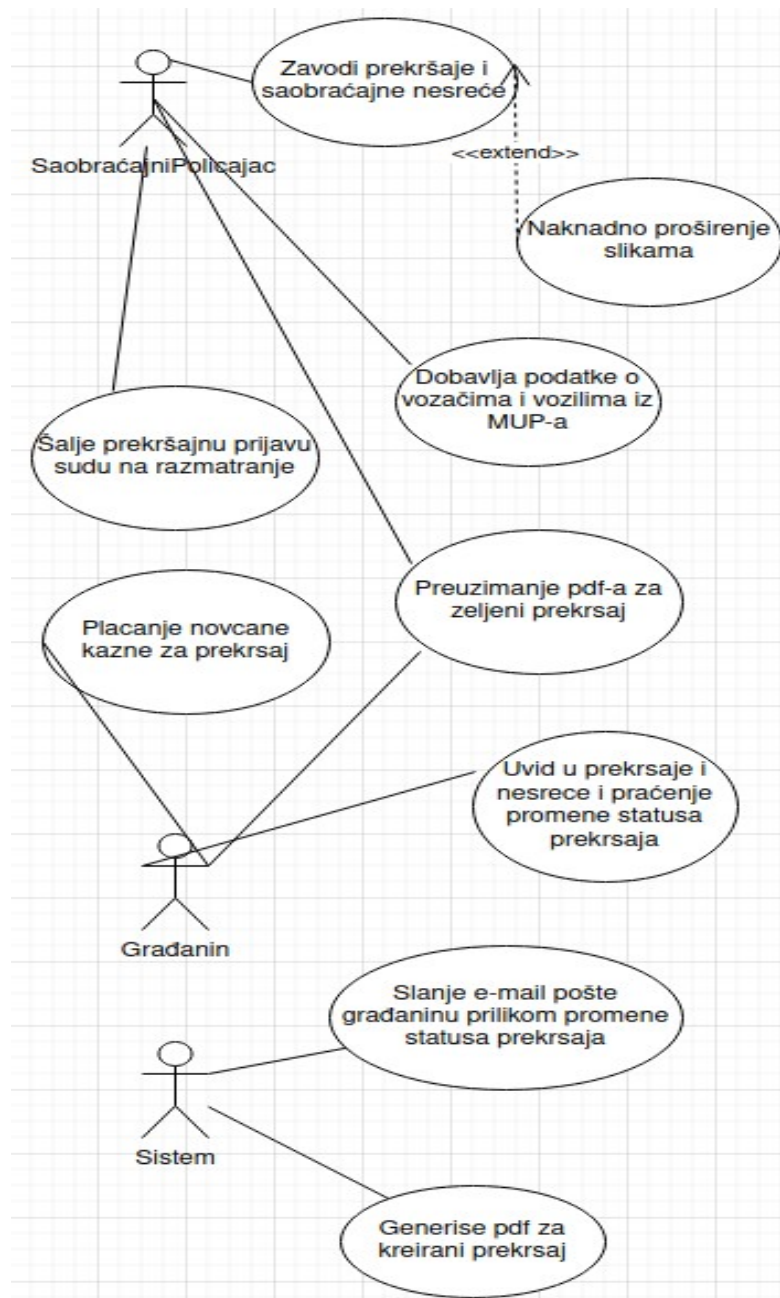
Za slanje emailova [7] koristi se biblioteka [gopkg.in/gomail.v2](https://github.com/gopkg.in/gomail.v2), koja pruža siguran i jednostavan način za slanje email poruka iz aplikacije, čime se korisnici obaveštavaju o važnim informacijama ili događajima vezanim za saobraćajnu policiju.

## 5. Specifikacija zahteva

U ovom poglavlju su detaljno razmotreni funkcionalni i nefunkcionalni zahtevi softverskog rešenja opisanog u ovom radu.

### 5.1. Specifikacija funkcionalnih zahteva

U ovom odeljku su opisani funkcionalni zahtevi koje je potrebno da ispunjava softversko rešenje za Saobraćajnu policiju. Funkcionalni zahtevi ovog softverskog rešenja su predstavljeni UML dijagramom slučaja korišćenja, kao što je prikazano na slici 1.



Slika 1 – UML dijagram slučaja korišćenja



## Slučajevi korišćenja

Tabela 1 prikazuje opis slučaja korišćenja “**Dobavlja podatke o vozačima i vozilima iz MUPa-**”.

Naziv	Dobavlja podatke o vozačima i vozilima iz MUPa-
Učesnici	Saobraćajni policajac
Preduslovi	saobraćajni policajac je pristupio sistemu i ima pristup podacima iz MUP-a
Koraci	1. policajac unosi podatke o vozaču i vozilu u MUP-u 2. dobijene podatke evidentira u prekršaj ili nesreću
Rezultat	podaci su uspešno pronađeni i evidentirani
Izuzeci	nisu pronađeni traženi podaci

Tabela 1 - Opis slučaja korišćenja “Dobavlja podatke o vozačima i vozilima iz MUPa-”

Tabela 2 prikazuje opis slučaja korišćenja “**Generiše pdf za kreirani prekršaj**”.

Naziv	Generiše pdf za kreirani prekršaj
Učesnici	Sistem
Preduslovi	sistem je funkcionalan
Koraci	1. policajac evidentira prekršaj 2. sistem prikuplja podatke i na osnovu njih generiše pdf za kreirani prekršaj
Rezultat	Pdf je uspešno sacuvan na file sistemu
Izuzeci	neuspešno generisanje pdf-a zbog pogrešnog formata unetih podataka

Tabela 2 - Opis slučaja korišćenja “Generiše pdf za kreirani prekršaj”

Tabela 3 prikazuje opis slučaja korišćenja “**Zavodi prekršaje i saobraćajne nesreće**”.

Naziv	Zavodi prekršaje i saobraćajne nesreće
Učesnici	SaobraćajniPolicajac
Preduslovi	saobraćajni policajac je pristupio sistemu i ima pristup podacima iz MUP-a
Koraci	1. policajac unosi sve potrebne podatke o prekršaju ili saobraćajnoj nesreći (uključujući i slike ako u zabeležene) 2. podaci su uspešno sačuvani u bazi
Rezultat	prekršaj/nesreća je uspešno evidentiran
Izuzeci	vozača ili vozilo nije moguće pronaći u MUP-u datoteka(slika) prekršaja nije odgovarajućeg formata datoteka je oštećena

Tabela 3 - Opis slučaja korišćenja “Zavodi prekršaje i saobraćajne nesreće”

Tabela 4 prikazuje opis slučaja korišćenja “**Naknadno proširenje slikama**”.

Naziv	Naknadno proširenje slikama
Učesnici	SaobraćajniPolicajac
Preduslovi	saobraćajni policajac je pristupio sistemu
Koraci	1. policajac unosi potrebne podatke o prekršaju ili saobraćajnoj nesreći (ID) 2. izbor opcije za dodavanje slike 3. otpremanje(upload) slike
Rezultat	slike su uspešno dodate
Izuzeci	datoteka nije odgovarajućeg formata/ datoteka je oštećena

Tabela 4 - Opis slučaja korišćenja “Naknadno proširenje slikama”

Tabela 5 prikazuje opis slučaja korišćenja “**Šalje prekršajnu prijavu sudu na razmatranje**”.

Naziv	Šalje prekršajnu prijavu sudu na razmatranje
Učesnici	SaobraćajniPolicajac
Preduslovi	saobraćajni policajac je pristupio sistemu

Koraci	1. policajac evidentira prekršaj koji ima status PoslatoNaSud 2. policajac šalje zavedeni prekršaj sudu
Rezultat	prekršaj je uspešno dostavljen sudu
Izuzeci	neuspešno slanje prekršaja

Tabela 5 - Opis slučaja korišćenja “Šalje prekršajnu prijavu sudu na razmatranje”

Tabela 6 prikazuje opis slučaja korišćenja **“Preuzimanje pdf-a za zeljeni prekršaj”**.

Naziv	Preuzimanje pdf-a za zeljeni prekršaj
Učesnici	saobraćajni policajac, Građanin
Preduslovi	saobraćajni policajac/građanin je pristupio sistemu
Koraci	1. korisnik pristupa stranici saobraćajne policije 2. bira zeljeni prekršaj iz liste svojih prekršaja 3. pregled detalja prekršaja i dugme za preuzimanje pdf-a 4. klikom na dugme pdf prekršaja biva preuzet
Rezultat	pdf je uspesno preuzet
Izuzeci	neuspešno preuzimanje pdf sadržaja

Tabela 6 - Opis slučaja korišćenja “Preuzimanje pdf-a za zeljeni prekršaj”

Tabela 7 prikazuje opis slučaja korišćenja **“Slanje e-mail pošte građaninu prilikom promene statusa Prekršaja”**.

Naziv	Slanje e-mail pošte građaninu prilikom promene statusa prekršaja
Učesnici	Sistem
Preduslovi	Sistem je funkcionalan
Koraci	1. policajac unosi sve potrebne podatke o građaninu i prekršaju /nesreci koji se za njega kreirao ili kada se promeni status prekršaja 2. sistem prikuplja podatke i formira email poruku 3. istem šalje email na navedenu email adresu građanina

Rezultat	email pošta je uspešno dostavljena građaninu
Izuzeci	uneta je pogrešna email adresa

Tabela 7 - Opis slučaja korišćenja “Slanje e-mail pošte građaninu prilikom promene statusa prekršaja”

Tabela 8 prikazuje opis slučaja korišćenja “**Uvid u prekršaje i nesrece i praćenje promene statusa prekršaja**”.

Naziv	Uvid u prekršaje i nesrece i praćenje promene statusa prekršaja
Učesnici	Građnin
Preduslovi	građnin je pristupio sistemu potrebno je da postoji makar jedan prekršaj
Koraci	1. građanin pristupa stranici saobraćajne policije 2. pregled liste evidentiranih prekršaja I nesreca 3. klikom na određeni prekršaj iz liste prikazuje se status prekršaja
Rezultat	uvid u status prekršaja
Izuzeci	-

Tabela 8 - Opis slučaja korišćenja “Uvid u prekršaje i nesrece i praćenje promene statusa prekršaja”

Tabela 9 prikazuje opis slučaja korišćenja “**Plaćanje novcane kazne za prekršaj**”.

Naziv	Plaćanje novcane kazne za prekršaj
Učesnici	Građnin
Preduslovi	građnin je pristupio sistemu potrebno je da postoji makar jedan prekršajkoji ima status NovcanaKaznaDodeljena
Koraci	1.građanin pristupa stranici saobraćajne policije 2. klikom na određeni prekršaj iz liste prikazuju se detalji prekršaja i dugme za plaćanje kazne 3. klikom na dugme menja se status prekršaja na NovcanaKaznaIsplacena
Rezultat	uspesno isplacena kazna
Izuzeci	neuspesno plaćanje kazne

Tabela 9 - Opis slučaja korišćenja “Plaćanje novcane kazne za prekršaj”

## 5.2. Specifikacija nefunkcionalnih zahteva

Ovi nefunkcionalni zahtevi i implementirane tehnologije pomažu u definisanju ključnih karakteristika sistema za saobraćajnu policiju, fokusirajući se na performanse, bezbednost i korisničko iskustvo.

### **Performanse**

Brzina odziva sistema za pretragu podataka o saobraćajnim prekršajima.  
Efikasnost generisanja izveštaja i obrade velikih količina podataka.

### **Bezbednost**

Zaštita podataka o saobraćajnim prekršajima od neovlašćenog pristupa.  
Implementacija sigurnosnih mehanizama poput enkripcije podataka.

### **Održavanje**

Struktura koda i dokumentacija radi lakšeg održavanja i brzog otklanjanja grešaka.

### **Upotrebljivost**

Intuitivan korisnički interfejs za jednostavnu navigaciju i upotrebu aplikacije.

### **Portabilnost**

Mogućnost pokretanja aplikacije na različitim platformama.

### **Kompatibilnost**

Integracija sa postojećim sistemima i bazama podataka.

### **Skalabilnost**

Mogućnost sistema da raste sa povećanjem broja korisnika i podataka.

### **PDF generisanje**

Biblioteka za generisanje dinamičkih PDF dokumenata direktno iz aplikacije.

### **Slanje emailova**

Biblioteka za sigurno slanje email poruka iz aplikacije.

## 6. Specifikacija dizajna

Ovo poglavlje objašnjava dizajn softverskog rešenja za upravljanje saobraćajnim prekršajima, uključujući organizaciju sistema i komunikaciju između ključnih komponenti.

### Arhitektura sistema

Sistem je organizovan kao mikroservisna arhitektura sa nekoliko ključnih komponenti koje efikasno rukuju različitim aspektima saobraćajnih prekršaja.

#### Klase sistema:

**Prekršaj:** Sadrži informacije o prekršajima, uključujući tip prekršaja, status (poput "NovčanaKaznaDodeljena", "NovčanaKaznaIsplaćena", "PoslatoNaSud"), datum i vreme, lokaciju, kazne i slike.

**Policajac:** Sadrži informacije o policajcima, uključujući ime, prezime, email, JMBG, lozinku i pol.

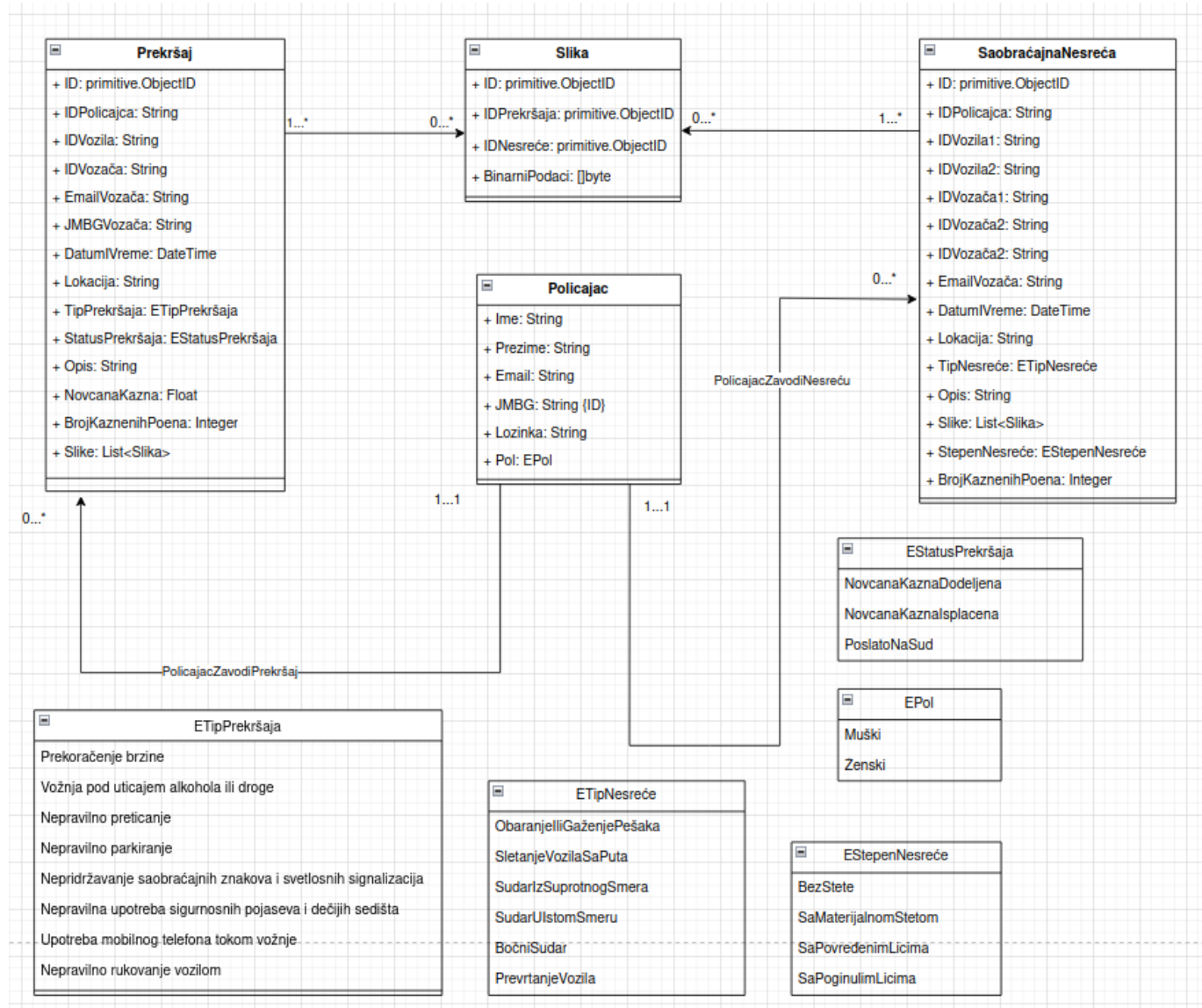
**Saobraćajna Nesreća:** Sadrži informacije o saobraćajnim nesrećama, uključujući tip nesreće, lokaciju, opis i slike.

**Slika:** Ova klasa sadrži informacije o slikama koje su povezane sa prekršajima i nesrećama.

#### Komunikacija između komponenti:

Komponente sistema međusobno komuniciraju putem definisanih API-ja i servisa. Na primer, klasa Prekršaj koristi servise za upravljanje slikama (klasa Slika) i podacima o policajcima (klasa Policajac) radi potpunog prikaza informacija o prekršaju.

Na slici 2 je pomoću UML dijagrama klasa predstavljen objektni model sistema.



Slika 2 - Dijagram klasa

## 7. Implementacija

Ovo poglavlje prikazuje način na koji su implementirane neke od funkcija sistema za Saobraćajnu policiju. Objašnjena je implementacija sledećih delova sistema: kreiranje prekršaja, dobavljanje svih prekršaja za policajca, plaćanje novčane kazne i slanje e-mail poruke o potvrdi isplaćene kazne, kreiranje saobraćajne nesreće.

### Implementacija kreiranja prekršaja

U ovom poglavlju je predstavljena implementacija kreiranja prekršaja. Ova funkcionalnost uključuje autentifikaciju korisnika, pozivajući drugi servis koji na osnovu tokena proverava da li je uloga korisnika koji pristupa API endpoint-u koji poziva ovu funkciju saobraćajni policajac. Zatim funkcija prihvata podatke o prekršaju, validirajući tip i status prekršaja. Nakon uspešno sačuvanog prekršaja u bazi podataka, poziva se funkcija koja će kreirati i poslati e-mail vozaču za kojeg je kreiran prekršaj. Sledi poziv funkcije koja kreira PDF dokument koji se čuva na file sistemu backend-a, koji će kasnije moći da se preuzme sa klijentske strane. Prilikom kreiranja prekršaja proverava se da li je uneti broj prekršajnih poena veći od nule, ukoliko jeste veći, poziva se API iz servisa MUP-a za vozila koji će uvećati broj prekršajnih poena vozaču koji je počinio prekršaj. Takođe, ukoliko je status prekršaja PoslatoNaSud, onda se poziva API servisa za prekršajni sud, koji će kreirati novi predmet sa prosleđenim prekršajem.

```
func (s *DelictHandler) CreateDelict(c *gin.Context) {
    rw := c.Writer
    h := c.Request
    token := h.Header.Get("Authorization")
    url := "http://auth-service:8085/api/users/currentUser"
    timeout := 5 * time.Second // Adjust the timeout duration as needed
    ctx, cancel := context.WithTimeout(context.Background(), timeout)
    defer cancel()
    resp, err := s.performAuthorizationRequestWithContext("GET", ctx, token, url)
    if err != nil {
        if ctx.Err() == context.DeadlineExceeded {
            errorMsg := map[string]string{"error": "Authorization service is not available."}
            errorMessage.ReturnJSONError(rw, errorMsg, http.StatusBadRequest)
            return
        }
        errorMsg := map[string]string{"error": "Error performing authorization request."}
        errorMessage.ReturnJSONError(rw, errorMsg, http.StatusBadRequest)
        return
    }
    defer resp.Body.Close()
    statusCode := resp.StatusCode
    if statusCode != 200 {
        errorMsg := map[string]string{"error": "Unauthorized."}
        errorMessage.ReturnJSONError(rw, errorMsg, http.StatusUnauthorized)
        return
    }
    decoder := json.NewDecoder(resp.Body)
    // Define a struct to represent the JSON structure
    var responseUser struct {
        LoggedInUser struct {
            ID          primitive.ObjectID `json:"id"`
            username    string             `json:"username"`
            email       string             `json:"email"`
            UserRole    data.UserRole      `json:"userRole"`
        } `json:"user"`
    }
    if err := decoder.Decode(&responseUser); err != nil {
        errorMsg := map[string]string{"error": "User object was not valid."}
        errorMessage.ReturnJSONError(rw, errorMsg, http.StatusUnauthorized)
        return
    }
    if responseUser.LoggedInUser.UserRole != data.TrafficPoliceman {
```



```

policemanID := responseUser.LoggedInUser.ID.Hex()
delict, exists := c.Get( key: "delict")
if !exists {
    errorMsg := map[string]string{"Error": " delict object was not valid."}
    errorMessage.ReturnJSONError(rw, errorMsg, http.StatusBadRequest)
    return
}
delictInsert, ok := delict.(domain.DelictCreate)
if !ok {
    errorMsg := map[string]string{"error": "Invalid type for delict."}
    errorMessage.ReturnJSONError(rw, errorMsg, http.StatusBadRequest)
    return
}
if !isValidDelictType(delictInsert.DelictType) {
    errorMessage.ReturnJSONError(rw, map[string]string{"error": "Invalid delict type."}, http.StatusBadRequest)
    return
}
if !isValidDelictStatus(delictInsert.DelictStatus) {
    errorMessage.ReturnJSONError(rw, map[string]string{"error": "Invalid delict status."}, http.StatusBadRequest)
    return
}
delictInsertDB, _, err := s.service.InsertDelict(&delictInsert, policemanID)
if err != nil {
    errorMsg := map[string]string{"error": "Database problem."}
    errorMessage.ReturnJSONError(rw, errorMsg, http.StatusBadRequest)
    return
}

err = s.sendDelictMail(delictInsertDB.Description, delictInsertDB.DriverEmail)
if err != nil {
    errorMessage.ReturnJSONError(rw, fmt.Sprintf( format: "Error sending email: %s", err), http.StatusInternalServerError)
    return
}

pdfFilePath, err := s.GenerateDelictPDF(delictInsertDB)
if err != nil {
    log.Printf( format: "Error generating PDF: %v\n", err)
    errorMsg := map[string]string{"error": "Failed to generate PDF report."}
    errorMessage.ReturnJSONError(rw, errorMsg, http.StatusInternalServerError)
    return
}
log.Printf( format: "Generated PDF saved at: %s", pdfFilePath)

```

Listing 1 - Kreiranje prekršaja

```

if delictInsert.NumberOfPenaltyPoints > 0 {
    driverID := delictInsert.DriverIdentificationNumber
    points := delictInsert.NumberOfPenaltyPoints
    updatePointsURL := fmt.Sprintf( format: "http://vehicles-service:8080/api/driver/updatePenaltyPoints/%s", driverID)

    updatePointsReqBody := struct {
        Points int64 `json:"points"`
    }{
        Points: points,
    }

    updatePointsReqBodyJSON, err := json.Marshal(updatePointsReqBody)
    if err != nil {
        errorMessage.ReturnJSONError(rw, fmt.Sprintf( format: "Error marshaling JSON: %s", err), http.StatusInternalServerError)
        return
    }

    updatePointsReq, err := http.NewRequest( method: "PATCH", updatePointsURL, bytes.NewBuffer(updatePointsReqBodyJSON))
    if err != nil {
        errorMessage.ReturnJSONError(rw, fmt.Sprintf( format: "Error creating update points request: %s", err), http.StatusInternalServerError)
        return
    }
    updatePointsReq.Header.Set( key: "Content-Type", value: "application/json")
    updatePointsReq.Header.Set( key: "Authorization", token)

    updatePointsResp, err := http.DefaultClient.Do(updatePointsReq)
    if err != nil || updatePointsResp.StatusCode != http.StatusOK {
        errorMessage.ReturnJSONError(rw, err: "Failed to update penalty points in vehicle driver service.", http.StatusInternalServerError)
        return
    }
    defer updatePointsResp.Body.Close()
}

if delictInsert.DelictStatus == domain.SentToCourt {

    courtURL := "http://court-service:8083/api/subject/create"
    subject := struct {
        ViolationID string `json:"violation_id"`
    }{
        ViolationID: delictInsertDB.ID.Hex(),
    }
    subjectJSON, err := json.Marshal(subject)
    if err != nil {
        errorMessage.ReturnJSONError(rw, fmt.Sprintf( format: "Error marshaling JSON: %s", err), http.StatusInternalServerError)
        return
    }
    courtReq, err := http.NewRequest( method: "POST", courtURL, bytes.NewBuffer(subjectJSON))
    if err != nil {
        errorMessage.ReturnJSONError(rw, fmt.Sprintf( format: "Error creating court request: %s", err), http.StatusInternalServerError)
        return
    }
    courtReq.Header.Set( key: "Content-Type", value: "application/json")
    courtReq.Header.Set( key: "Authorization", token)
    courtResp, err := http.DefaultClient.Do(courtReq)
    if err != nil || courtResp.StatusCode != http.StatusCreated {
        errorMessage.ReturnJSONError(rw, err: "Failed to create subject in court service.", http.StatusInternalServerError)
        return
    }
    defer courtResp.Body.Close()

    rw.WriteHeader(http.StatusCreated)
    jsonResponse, err1 := json.Marshal(delictInsertDB)
    if err1 != nil {
        errorMessage.ReturnJSONError(rw, fmt.Sprintf( format: "Error marshaling JSON: %s", err1), http.StatusInternalServerError)
        return
    }
    rw.Write(jsonResponse)
}
}

func (c *DelictHandler) GenerateTestPPE(cty *gin.Context) { // 1 ucena - 1 flower/1408

```

Listing 1 - Kreiranje prekršaja

## Implementacija dobavljanja svih prekršaja za policajca

```
func (s *DelictHandler) GetDelictsByPolicemanID(c *gin.Context) { 1 usage Ana Domonji +1
    rw := c.Writer
    h := c.Request

    token := h.Header.Get(key: "Authorization")
    url := "http://auth-service:8085/api/users/currentUser"

    timeout := 5 * time.Second
    ctx, cancel := context.WithTimeout(context.Background(), timeout)
    defer cancel()

    resp, err := s.performAuthorizationRequestWithContext(method: "GET", ctx, token, url)
    if err != nil {
        if ctx.Err() == context.DeadlineExceeded {
            errorMsg := map[string]string{"error": "Authorization service is not available."}
            errorMessage.ReturnJSONError(rw, errorMsg, http.StatusBadRequest)
            return
        }
        errorMsg := map[string]string{"error": "Error performing authorization request."}
        errorMessage.ReturnJSONError(rw, errorMsg, http.StatusBadRequest)
        return
    }
    defer resp.Body.Close()

    statusCode := resp.StatusCode
    if statusCode != 200 {
        errorMsg := map[string]string{"error": "Unauthorized."}
        errorMessage.ReturnJSONError(rw, errorMsg, http.StatusUnauthorized)
        return
    }

    decoder := json.NewDecoder(resp.Body)
    var responseUser struct {
        LoggedInUser struct {
            ID primitive.ObjectID `json:"id"`
            UserRole data.UserRole `json:"userRole"`
        } `json:"user"`
    }
    if err := decoder.Decode(&responseUser); err != nil {
        errorMsg := map[string]string{"error": "User object was not valid."}
        errorMessage.ReturnJSONError(rw, errorMsg, http.StatusUnauthorized)
        return
    }
}
```

Listing 2 – Dobavljanje svih prekršaja za policajca

```

func (s *DelictHandler) GetDelictsByPolicemanID(c *gin.Context) { 1 usage Ana Domonji +1
    rw := c.Writer
    h := c.Request

    token := h.Header.Get(key: "Authorization")
    url := "http://auth-service:8085/api/users/currentUser"

    timeout := 5 * time.Second
    ctx, cancel := context.WithTimeout(context.Background(), timeout)
    defer cancel()

    resp, err := s.performAuthorizationRequestWithContext(method: "GET", ctx, token, url)
    if err != nil {
        if ctx.Err() == context.DeadlineExceeded {
            errorMsg := map[string]string{"error": "Authorization service is not available."}
            errorMessage.ReturnJSONError(rw, errorMsg, http.StatusBadRequest)
            return
        }
        errorMsg := map[string]string{"error": "Error performing authorization request."}
        errorMessage.ReturnJSONError(rw, errorMsg, http.StatusBadRequest)
        return
    }
    defer resp.Body.Close()

    statusCode := resp.StatusCode
    if statusCode != 200 {
        errorMsg := map[string]string{"error": "Unauthorized."}
        errorMessage.ReturnJSONError(rw, errorMsg, http.StatusUnauthorized)
        return
    }

    decoder := json.NewDecoder(resp.Body)
    var responseUser struct {
        LoggedInUser struct {
            ID primitive.ObjectID `json:"id"`
            UserRole data.UserRole `json:"userRole"`
        } `json:"user"`
    }
    if err := decoder.Decode(&responseUser); err != nil {
        errorMsg := map[string]string{"error": "User object was not valid."}
        errorMessage.ReturnJSONError(rw, errorMsg, http.StatusUnauthorized)
        return
    }
}

```

Listing 2 – Dobavljanje svih prekršaja za policajca

## Implementacija plaćanja novčane kazne i slanje e-mail poruke o potvrdi isplaćene kazne

U ovom poglavlju je predstavljena implementacija plaćanja novčane kazne. Ova funkcionalnost takođe uključuje autentifikaciju korisnika, pozivajući drugi servis koji na osnovu tokena proverava da li je uloga korisnika koji pristupa API endpoint-u koji poziva ovu funkciju građanin. Ovoj funkciji se putem URL-a prosleđuje id prekršaja koji treba da bude isplaćen. Kada korisnik na klijentskoj strani aplikacije klikne na dugme za plaćanje kazne, ova funkcija će promeniti status sa NovčanaKaznaDodeljena na NovčanaKaznaIsplaćena i nakon isplate dugme za plaćanje više neće biti vidljivo. Nakon što je kazna uspešno isplaćena, poziva se funkcija koja će pomenutom građaninu poslati e-mail o potvrdi da je kazna plaćena zajedno sa opisom počinjenog prekršaja.

```
func (s *DelictHandler) PayFine(c *gin.Context) { 1 usage  Flower0408

    rw := c.Writer
    h := c.Request

    token := h.Header.Get(key: "Authorization")
    url := "http://auth-service:8085/api/users/currentUser"

    timeout := 5 * time.Second
    ctx, cancel := context.WithTimeout(context.Background(), timeout)
    defer cancel()

    resp, err := s.performAuthorizationRequestWithContext(method: "GET", ctx, token, url)
    if err != nil {
        if ctx.Err() == context.DeadlineExceeded {
            errorMsg := map[string]string{"error": "Authorization service is not available."}
            errorMessage.ReturnJSONError(rw, errorMsg, http.StatusBadRequest)
            return
        }
        errorMsg := map[string]string{"error": "Error performing authorization request."}
        errorMessage.ReturnJSONError(rw, errorMsg, http.StatusBadRequest)
        return
    }
    defer resp.Body.Close()

    statusCode := resp.StatusCode
    if statusCode != 200 {
        errorMsg := map[string]string{"error": "Unauthorized."}
        errorMessage.ReturnJSONError(rw, errorMsg, http.StatusUnauthorized)
        return
    }

    decoder := json.NewDecoder(resp.Body)
    var responseUser struct {
        LoggedInUser struct {
            Email string `json:"email"`
            UserRole data.UserRole `json:"userRole"`
        } `json:"user"`
    }
    if err := decoder.Decode(&responseUser); err != nil {
        errorMsg := map[string]string{"error": "User object was not valid."}
        errorMessage.ReturnJSONError(rw, errorMsg, http.StatusUnauthorized)
        return
    }
}
```

Listing 3 – Plaćanje novčane kazne i slanje e-mail poruke o potvrdi isplaćene kazne



```

func (s *DelictHandler) PayFine(c *gin.Context) { 1 usage  ± flower0408
}

if err := decoder.Decode(&responseUser); err != nil {
    errorMsg := map[string]string{"error": "User object was not valid."}
    errorMessage.ReturnJSONError(rw, errorMsg, http.StatusUnauthorized)
    return
}

if responseUser.LoggedInUser.UserRole != data.Citizen {
    errorMsg := map[string]string{"Unauthorized": "You are not a citizen."}
    errorMessage.ReturnJSONError(rw, errorMsg, http.StatusUnauthorized)
    return
}

delictId := c.Param(key: "id")
delict, err := s.service.GetDelictById(delictId, ctx)
if err != nil {
    errorMsg := map[string]string{"error": "Failed to retrieve delict from the database. No such delict."}
    errorMessage.ReturnJSONError(rw, errorMsg, http.StatusInternalServerError)
    return
}

if delict.DelictStatus != domain.FineAwarded {
    errorMsg := map[string]string{"error": "Delict is not in a payable status."}
    errorMessage.ReturnJSONError(rw, errorMsg, http.StatusBadRequest)
    return
}

delict.DelictStatus = domain.FinePaid
if err := s.service.UpdateDelictStatus(delict); err != nil {
    errorMsg := map[string]string{"error": "Failed to update delict status."}
    errorMessage.ReturnJSONError(rw, errorMsg, http.StatusInternalServerError)
    return
}

if err := s.sendPaymentConfirmationEmail(delict.Description, delict.DriverEmail); err != nil {
    errorMessage.ReturnJSONError(rw, fmt.Sprintf(format: "Error sending email: %s", err), http.StatusInternalServerError)
    return
}

rw.WriteHeader(http.StatusOK)
rw.Write([]byte(`{"message": "Fine paid successfully."}`))
}

func (s *DelictHandler) sendPaymentConfirmationEmail(Description, driverEmail string) error { 1 usage  ± flower0408
    m := gmail.NewMessage()
    m.SetHeader(field: "From", smtpEmail)
    m.SetHeader(field: "To", driverEmail)
    m.SetHeader(field: "Subject", value: "Potvrda isplate prekršaja")

    bodyString := fmt.Sprintf(format: "Vasa novcana kazna je uspesno isplacena.\n Opis isplacenog prekršaja:\n %s ", Description)
    m.SetBody(contentType: "text", bodyString)

    client := gmail.NewDialer(smtpServer, smtpServerPort, smtpEmail, smtpPassword)

    if err := client.DialAndSend(m); err != nil {
        log.Fatalf(format: "Failed to send mail because of: %s", err)
        return err
    }

    return nil
}
}

```

Listing 3 – Plaćanje novčane kazne i slanje e-mail poruke o potvrdi isplaćene kazne

## Implementacija kreiranja saobraćajne nesreće

U ovom poglavlju je predstavljena implementacija kreiranja saobraćajne nesreće. Ova funkcionalnost takođe uključuje autentifikaciju korisnika, pozivajući drugi servis koji na osnovu tokena proverava da li je uloga korisnika koji pristupa API endpoint-u koji poziva ovu funkciju saobraćajni policajac. Zatim se vrši validacija tipa i stepena saobraćajne nesreće, gde će se nakon uspešne validacije tih polja, podaci o nesreći sačuvati u bazi podataka. Prilikom kreiranja saobraćajne nesreće vrši se proverava da li je uneti broj prekršajnih poena veći od nule, ukoliko jeste veći poziva se API iz servisa MUP-a za vozila koji će uvećati broj prekršajnih poena vozaču koji je počinio saobraćajnu nesreću. Takođe, na kraju se poziva funkcija koja će kreirati i poslati e-mail poruku vozaču koji je počinio saobraćajnu nesreću. Primer ove implementacije je prikazan na listingu br. 4.

```
func (s *CarAccidentHandler) CreateCarAccident(c *gin.Context) { 1 usage  flower0408
    rw := c.Writer
    h := c.Request

    token := h.Header.Get(key: "Authorization")
    url := "http://auth-service:8085/api/users/currentUser"

    timeout := 5 * time.Second // Adjust the timeout duration as needed
    ctx, cancel := context.WithTimeout(context.Background(), timeout)
    defer cancel()

    resp, err := s.performAuthorizationRequestWithContext(method: "GET", ctx, token, url)
    if err != nil {
        if ctx.Err() == context.DeadlineExceeded {
            errorMsg := map[string]string{"error": "Authorization service is not available."}
            errorMessage.ReturnJSONError(rw, errorMsg, http.StatusBadRequest)
            return
        }

        errorMsg := map[string]string{"error": "Error performing authorization request."}
        errorMessage.ReturnJSONError(rw, errorMsg, http.StatusBadRequest)
        return
    }
    defer resp.Body.Close()

    statusCode := resp.StatusCode
    if statusCode != 200 {
        errorMsg := map[string]string{"error": "Unauthorized."}
        errorMessage.ReturnJSONError(rw, errorMsg, http.StatusUnauthorized)
        return
    }

    decoder := json.NewDecoder(resp.Body)

    // Define a struct to represent the JSON structure
    var responseUser struct {
        LoggedInUser struct {
            ID          primitive.ObjectID `json:"id"`
            username    string              `json:"username"`
            email       string              `json:"email"`
            UserRole    data.UserRole       `json:"userRole"`
        }
    }
```

```

        email    string    `json:"email"`
        UserRole data.UserRole `json:"userRole"`
    } `json:"user"`
}

if err := decoder.Decode(&responseUser); err != nil {
    errorMsg := map[string]string{"error": "User object was not valid."}
    errorMessage.ReturnJSONError(rw, errorMsg, http.StatusUnauthorized)
    return
}

if responseUser.LoggedInUser.UserRole != data.TrafficPoliceman {
    errorMsg := map[string]string{"Unauthorized": " You are not traffic policeman."}
    errorMessage.ReturnJSONError(rw, errorMsg, http.StatusUnauthorized)
    return
}

policemanID := responseUser.LoggedInUser.ID.Hex()

carAccident, exists := c.Get( key: "carAccident")
if !exists {
    errorMsg := map[string]string{"Error": " carAccident object was not valid."}
    errorMessage.ReturnJSONError(rw, errorMsg, http.StatusBadRequest)
    return
}

carAccidentInsert, ok := carAccident.(domain.CarAccidentCreate)
if !ok {
    errorMsg := map[string]string{"error": "Invalid type for carAccident."}
    errorMessage.ReturnJSONError(rw, errorMsg, http.StatusBadRequest)
    return
}

if !isValidCarAccidentType(carAccidentInsert.CarAccidentType) {
    errorMessage.ReturnJSONError(rw, map[string]string{"error": "Invalid car Accident type."}, http.StatusBadRequest)
    return
}

if !isValidCarAccidentDegreeOfAccident(carAccidentInsert.DegreeOfAccident) {
    errorMessage.ReturnJSONError(rw, map[string]string{"error": "Invalid degree Of Accident type."}, http.StatusBadRequest)
    return
}
}

```

Listing 4 – Kreiranje saobraćajne nesreće



```

carAccidentInsertDB, _, err := s.service.InsertCarAccident(&carAccidentInsert, policemanID)
if err != nil {
    errorMsg := map[string]string{"error": "Database problem."}
    errorMessage.ReturnJSONError(rw, errorMsg, http.StatusBadRequest)
    return
}

if carAccidentInsertDB.NumberOfPenaltyPoints > 0 {
    driverID := carAccidentInsertDB.DriverIdentificationNumber1
    points := carAccidentInsertDB.NumberOfPenaltyPoints
    updatePointsURL := fmt.Sprintf( format: "http://vehicles-service:8080/api/driver/updatePenaltyPoints/%s", driverID)

    updatePointsReqBody := struct {
        Points int64 `json:"points"`
    }{
        Points: points,
    }

    updatePointsReqBodyJSON, err := json.Marshal(updatePointsReqBody)
    if err != nil {
        errorMessage.ReturnJSONError(rw, fmt.Sprintf( format: "Error marshaling JSON: %s", err), http.StatusInternalServerError)
        return
    }

    updatePointsReq, err := http.NewRequest( method: "PATCH", updatePointsURL, bytes.NewBuffer(updatePointsReqBodyJSON))
    if err != nil {
        errorMessage.ReturnJSONError(rw, fmt.Sprintf( format: "Error creating update points request: %s", err), http.StatusInternalServerError)
        return
    }
    updatePointsReq.Header.Set( key: "Content-Type", value: "application/json")
    updatePointsReq.Header.Set( key: "Authorization", token)

    updatePointsResp, err := http.DefaultClient.Do(updatePointsReq)
    if err != nil || updatePointsResp.StatusCode != http.StatusOK {
        errorMessage.ReturnJSONError(rw, err: "Failed to update penalty points in vehicle driver service.", http.StatusInternalServerError)
        return
    }
    defer updatePointsResp.Body.Close()
}

err = s.sendCarAccidentMail(carAccidentInsertDB.Description, carAccidentInsertDB.DriverEmail)
if err != nil {
    err = s.sendCarAccidentMail(carAccidentInsertDB.Description, carAccidentInsertDB.DriverEmail)
    if err != nil {
        errorMessage.ReturnJSONError(rw, fmt.Sprintf( format: "Error sending email: %s", err), http.StatusInternalServerError)
        return
    }
}

rw.WriteHeader(http.StatusCreated)
jsonResponse, err1 := json.Marshal(carAccidentInsertDB)
if err1 != nil {
    errorMessage.ReturnJSONError(rw, fmt.Sprintf( format: "Error marshaling JSON: %s", err1), http.StatusInternalServerError)
    return
}
rw.Write(jsonResponse)
}

func isValidCarAccidentType(carAccidentType domain.CarAccidentType) bool { 1 usage  ± flower0408
    switch carAccidentType {
    case domain.KnockingDownPedestrians, domain.VehicleLandingFromRoad, domain.CollisionFromOppositeDirection, domain.CollisionInSameDirection, domain.SideCollision, domain.Rollover,
    default:
        return true
    }
    return false
}

func (s *CarAccidentHandler) sendCarAccidentMail(Description, email string) error { 1 usage  ± flower0408
    m := gmail.NewMessage()
    m.SetHeader( field: "From", smtpEmail)
    m.SetHeader( field: "To", email)
    m.SetHeader( field: "Subject", value: "EUprava obvestenje")

    bodyString := fmt.Sprintf( format: "Za Vas je kreirana saobraćajna nesreca sa opisom:\n %s \nStanje vase saobraćajne nesrece mozete pratiti na portalu EUprave https://localhost:429
    m.SetBody( contentType: "text", bodyString)

    client := gmail.NewDialer(smtpServer, smtpServerPort, smtpEmail, smtpPassword)

    if err := client.DialAndSend(m); err != nil {
        log.Fatalf( format: "Failed to send mail because of: %s", err)
        return err
    }

    return nil
}

```

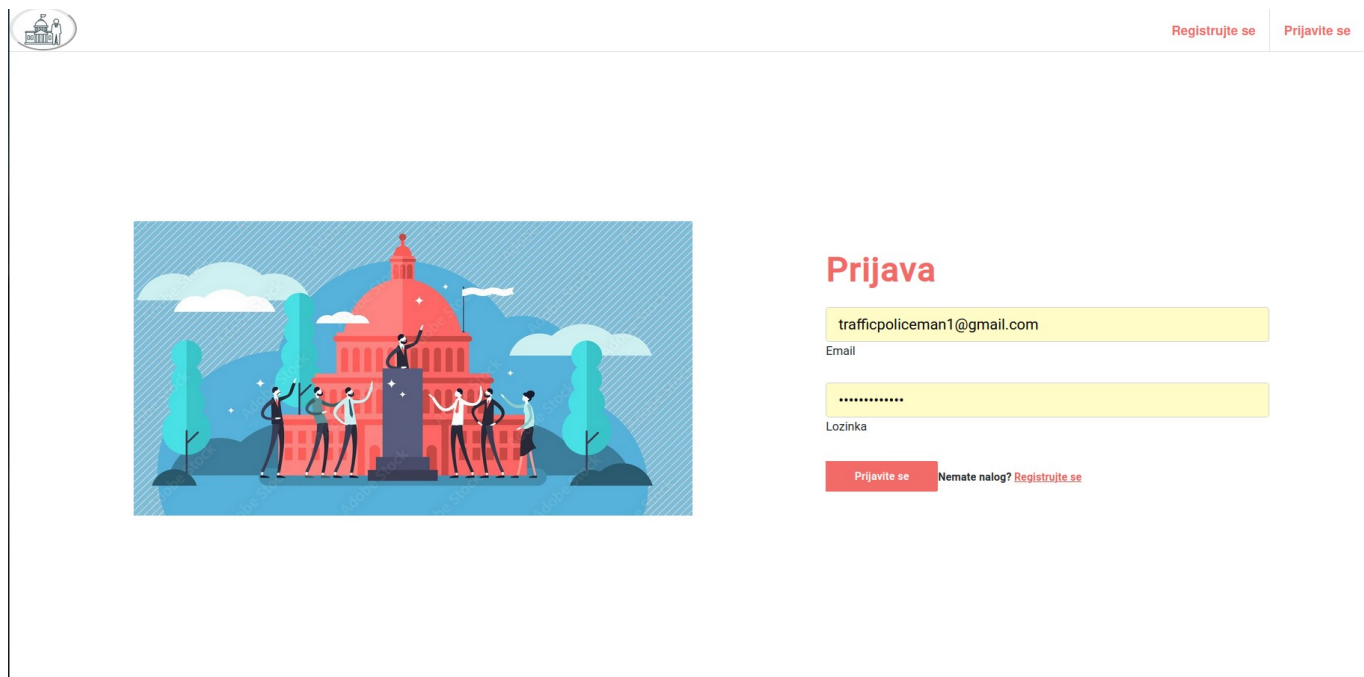
Listing 4 – Kreiranje saobraćajne nesreće

## 8. Demonstracija

Ovo poglavlje prikazuje način korišćenja aplikacije za Saobraćajnu policiju, pružajući uvid u korisnički interfejs i funkcionalnosti koje omogućavaju policajcima efikasno upravljanje saobraćajnim prekršajima i nesrećama.

### Prijava na sistem

Na početku korišćenja aplikacije, korisnik će se susresti sa stranicom za prijavljivanje, prikazanom na slici 3. Na ovoj stranici, korisnik unosi svoje korisničke podatke kako bi pristupio sistemu. Korisniku se prikazuje stranica za prijavu, kao što je ilustrovano.



Slika 3 - Prijava na sistem

### Prikaz početne stranice za saobraćajne policije za policajca

U ovom primeru je prikazana početna stranica kada se saobraćajni policajac sa svojim kredencijalima uloguje na sistem.



Slika 4 – Prikaz početne stranice za saobraćajne policije za policajca

## Kreiranje saobraćajnog prekršaja

Nakon što je saobraćajac pristupio početnoj stranici klikom na dugme za kreiranje prekršaja, otvara se stranica na kojoj je moguće proveriti podatke o vozaču koji je počinio prekršaj, tako što će pristupiti podacima iz MUP-a pretragom vozača u njihovoj bazi podataka preko unetog JMBG-a vozača, kao i pretragom vozačke dozvole na sličan način i pretragom vozila preko broja registarskih tablica. Nakon što se provere ovi podaci, unose se podaci o prekršaju, gde je moguće uneti slike, ali nije neophodno. Ovaj primer će biti prikazan na slikama 5, 6 i 7.

SAOBRAĆAJNA POLICIJA

E - UPRAVA

LISTA SVIH VOZAČA

PRETRAŽI VOZAČE PO JEDISTVENOM MATIČNOM BROJU:

1002004735045

Pretraži

Osveži

JMBG

1002004735045

Ime vozača

Pera

Ima prekršaj:

false

Kazneni poeni:

5

Prezime vozača

Peric

Datum Registracije

1992-09-02

Pol

Male

Slika 5 – Kreiranje saobraćajnog prekršaja  
(pretraga vozača u MUP-u)

## LISTA SVIH VOZACKIH DOZVOLA

PRETRAŽI VOZACKE DOZVOLE PO JMBG-U:

Pretraži

Osveži

JMBG vozaca

1002004735045

Mesto izdavanja

Smederevo

Broj dozvole

0b944706-09a4-4e4f-85d7-eb1be40d160b

Kategorije

B, C

## LISTA SVIH VOZILA

PREGLED SVIH REGISTROVANIH VOZILA (KLIKNITE OVDE)

PRETRAŽI VOZILO PO REGISTARSKOJ TABLICI:

Pretraži

Osveži

Registarska Tablica

BG155FG

Model Vozila

Fiat

Vlasnik Vozila

1002004735045

Datum Registracije

2024-06-26

Kategorija

C

Slika 6 – Kreiranje saobraćajnog prekršaja (pretraga vozačke dozvole i vozila u MUP-u)

## Kreiraj prekršaj

JMBG vozača:

1002004735045

Broj registarskih tablica:

BG155FG

Email vozača:

natasakotaranin@gmail.com

Lokacija:

Novi Sad

Opis:

Vozac je vozio nepropisnom brzinom dok je prolazio kroz semafor na Bulevaru Oslobođenja

Tip prekršaja:

Prekoracenje brzine

Status prekršaja:

Novcana kazna dodeljena

Novčana kazna:

5000

Broj kaznenih poena:

2

Upload image

Browse... 2 files selected.

Slike:

speeding2.jpg

speeding.jpeg

Kreiraj prekršaj

Slika 7 – Kreiranje saobraćajnog prekršaja

## Stranica za prikaz svih prekršaja

Nakon što je saobraćajac pristupio početnoj stranici klikom na dugme prikaži sve prekršaje, otvara se stranica na kojoj su prikazani svi prekršaji koje je taj saobraćajac kreirao. Klikom na određeni prekršaj, otvara se stranica sa detaljima tog prekršaja. Stranica za prikaz svih prekršaja izgleda identično i kod saobraćajca i kod građanina, s tim što će građaninu biti prikazani prekršaji koje je on počinio. Ova implementacija je prikazana na slici broj 8.



Slika 8 – Stranica za prikaz svih prekršaja



## Stranica za prikaz detalja prekršaja (saobraćajni policajac)

Kao što je prethodno pomenuto, klikom na određeni prekršaj, otvara se stranica sa detaljima tog prekršaja. U slučaju kada je saobraćajni policajac trenutno ulogovani korisnik, pored samih detalja o prekršaju, saobraćajac ima opciju da uploaduje dodatne slike koje su naknadno dobavljene za postojeći prekršaj. Pored toga, ima opciju da preuzme PDF dokument tog prekršaja na svoj file sistem. Ovo je prikazano na slici broj 9.

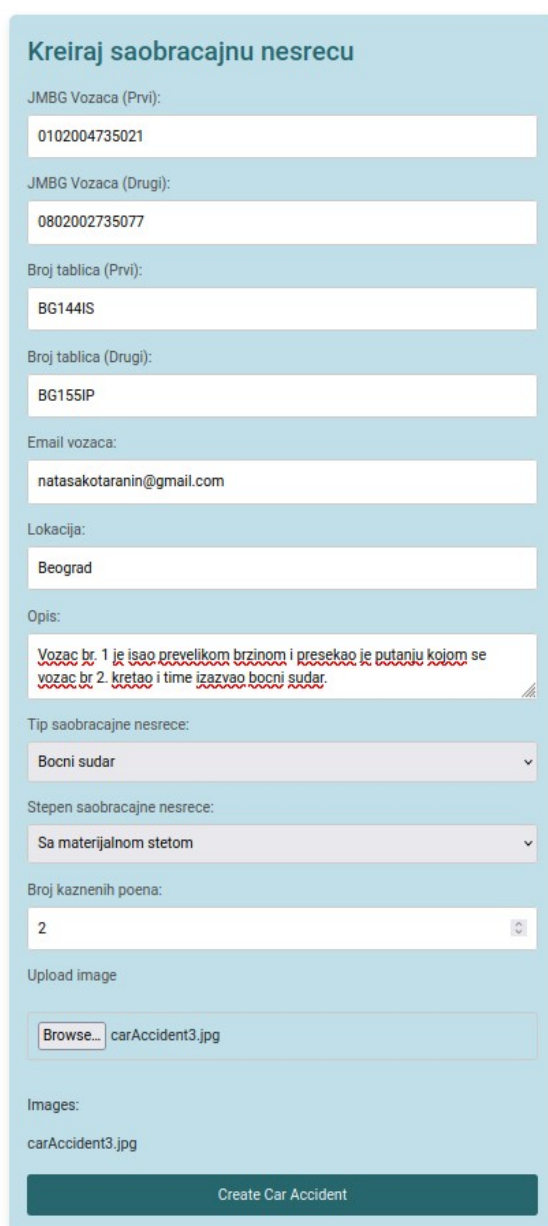


Slika 9– Stranica za prikaz detalja prekršaja (saobraćajni policajac)

## Kreiranje saobraćajne nesreće

Nakon što je saobraćajac pristupio početnoj stranici klikom na dugme za kreiranje saobraćajne nesreće, otvara se stranica na kojoj je moguće proveriti podatke o vozaču koji je počinio prekršaj, tako što će pristupiti podacima iz MUP-a pretragom vozača u njihovoj bazi podataka preko unetog JMBG-a vozača, kao i pretragom vozačke dozvole na sličan način i pretragom vozila preko broja registarskih tablica. Nakon što se provere ovi podaci, unose se podaci o nesreći gde je moguće uneti slike, ali nije neophodno. Ovaj primer će biti prikazan na slici 10.

Napomena – na slici će biti prikazana samo forma za kreiranje saobraćajne nesreće, pošto su forme za pretragu već prikazane kod kreiranja prekršaja.



The screenshot shows a web form titled "Kreiraj saobracajnu nesrecu" (Create traffic accident). The form is light blue with white input fields and a dark blue submit button. It contains the following fields and data:

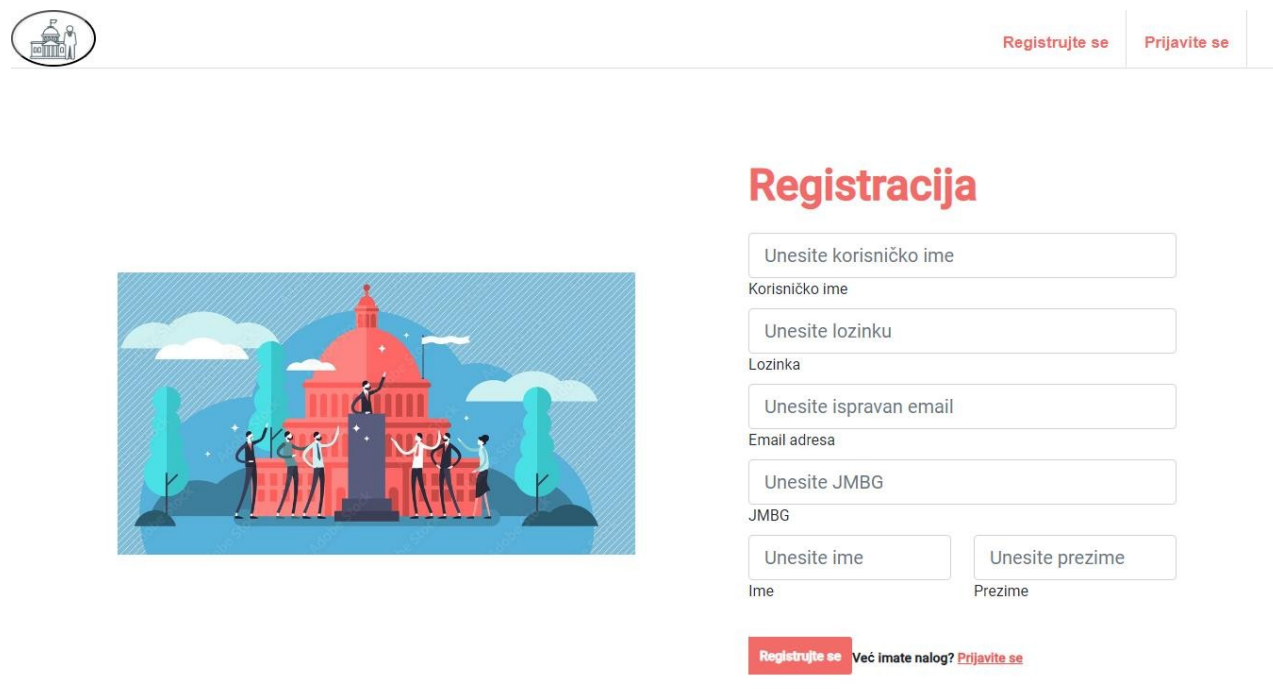
- JMBG Vozaca (Prvi): 0102004735021
- JMBG Vozaca (Drugi): 0802002735077
- Broj tablica (Prvi): BG144IS
- Broj tablica (Drugi): BG155IP
- Email vozaca: natakotaranin@gmail.com
- Lokacija: Beograd
- Opis: Vozac br. 1 je isao prevelikom brzinom i presekao je putanju kojom se vozac br 2. kretao i time izazvao bocni sudar.
- Tip saobracajne nesrece: Bocni sudar
- Stepen saobracajne nesrece: Sa materijalnom stetom
- Broj kaznenih poena: 2
- Upload image: Browse... carAccident3.jpg
- Images: carAccident3.jpg
- Submit button: Create Car Accident

Slika 10– Kreiranje saobraćajne nesreće



## Forma za registraciju

Ako građnin nema nalog, može se registrovati putem sledece forme - slika 11.



The screenshot shows a web page for registration. At the top left is a logo of a building. At the top right are two links: "Registrujte se" and "Prijavite se". The main heading is "Registracija" in red. Below it is a large illustration of a red building with a dome and people celebrating. To the right of the illustration is a registration form with the following fields: "Unesite korisničko ime" (Username), "Unesite lozinku" (Password), "Unesite ispravan email" (Valid email), "Unesite JMBG" (JMBG), "Unesite ime" (First name), and "Unesite prezime" (Last name). Below the form are two buttons: "Registrujte se" and "Već imate nalog? Prijavite se".

## Početna stranica

Nakon što se uspešno prijavi, korisnik biva preusmeren na početnu stranicu gde može izabrati jedan od sledećih sistema: Saobraćajna policija, Prekršajni sud, Zavod za statistiku ili MUP - vozila. Početna stranica je prikazana na slici 12.



Slika 12 - Početna stranica

## Prikaz početne stranice saobraćajne policije za građanina

U ovom primeru je prikazana pocetna stranica kada se gradjanin sa svojim kredencijalima uloguje na sistem.



Slika 13 – Prikaz početne stranice saobraćajne policije za građanina

## Stranica za prikaz detalja prekršaja (građanin)

Kao što je prethodno pomenuto, klikom na određeni prekršaj, otvara se stranica sa detaljima tog prekršaja. U slučaju kada je građanin trenutno ulogovani korisnik, pored samih detalja o prekršaju, građanin ima opciju da isplati novčanu kaznu ukoliko mu je dodeljena. Nakon što je kazna isplaćena, status će biti promenjen, a građanin će dobiti e-mail poruku o potvrđenoj isplati. Prikaz ovih funkcionalnosti je prikazan na slikama 14, 15 i 16. Pored toga, građanin ima opciju da preuzme PDF dokument tog prekršaja na svoj file sistem. Ovo je prikazano na slici broj 17.

## Vozac je vozio nepropisnom brzinom dok je prolazio kroz semafor na Bulevaru Oslobođenja

JMBG vozaca: 1002004735045

Broj registarskih tablica: BG155FG

Email vozaca: natasakotaranin@gmail.com

Tip prekršaja: Prekoracenje brzine

Status prekršaja: Novcana kazna dodeljena

Datum: Jul 1, 2024

Lokacija: Novi Sad

Novčana kazna: 5000

Broj kaznenih poena: 2



Preuzmi PDF

Upload image  No files selected.

Slika 14 – Stranica za prikaz detalja prekršaja (građanin)– pre plaćanja kazne

## Vozac je vozio nepropisnom brzinom dok je prolazio kroz semafor na Bulevaru Oslobođenja

JMBG vozaca: 1002004735045

Broj registarskih tablica: BG155FG

Email vozaca: natasakotaranin@gmail.com

Tip prekršaja: Prekoracenje brzine

Status prekršaja: Novcana kazna isplacena

Datum: Jul 1, 2024

Lokacija: Novi Sad

Novčana kazna: 5000

Broj kaznenih poena: 2



[Preuzmi PDF](#)

Slika 15 – Stranica za prikaz detalja prekršaja (građanin)– nakon plaćanja kazne

## Potvrda isplate prekršaja Примљено x



**EGovernmentPolice@outlook.com**

коме ја ▼

Vasa novcana kazna je uspesno isplacena.

Opis isplacenog prekršaja:

Vozac je vozio nepropisnom brzinom dok je prolazio kroz semafor na Bulevaru Oslobođenja

↩ Одговори

➡ Проследи



Slika 16 – Stranica za prikaz detalja prekršaja (građanin)– nakon plaćanja kazne

Izvestaj o prekršaju	
Detalji izvestaja	
ID prekršaja:	668207fa109b9ec70cf3dc7f
ID policajca:	664336d6c83a6c01fe332040
Broj vozacke dozvole:	BG155FG
Email vozaca:	natasakotaranin@gmail.com
JMBG vozaca:	1002004735045
Datum:	01.07.2024. 01:35:54
Lokacija:	Novi Sad
Opis:	Vozac je vozio nepropisnom brzinom dok je prolazio kroz semafor na Bulevaru Oslobođenja
Tip prekršaja:	Prekoracenje brzine
Status prekršaja:	Novcana kazna dodeljena
Novcana kazna:	5000.00
Broj kaznenih poena:	2

Generisano od strane eUprave

Slika 17– Stranica za prikaz detalja prekršaja (građanin)– PDF dokument

## 9. Zaključak

U ovom radu je predstavljeno softversko rešenje za upravljanje prekršajima u saobraćaju, koje omogućava efikasno evidentiranje, obradu i praćenje prekršaja od strane saobraćajnih policajaca. Demonstrirana je funkcionalnost sistema za kreiranje prekršaja, proveru podataka o vozačima kroz integraciju sa MUP-ovom bazom podataka, kao i sistem za prikaz i upravljanje prekršajima kako za policajce tako i za građane.

Sistem je uspešno adresirao potrebe za brzom proverom i unosom relevantnih informacija o prekršaju, što je doprinelo efikasnijem radu saobraćajnih policajaca. Kroz upotrebu tehnologije za generisanje PDF dokumenata i automatsko obaveštavanje putem e-maila, omogućeno je transparentno praćenje i dokumentovanje prekršaja.

Dobre strane ovog rešenja obuhvataju jednostavnost korišćenja i integraciju sa postojećim sistemima, što olakšava rad policajcima i poboljšava efikasnost procesa evidentiranja prekršaja. Međutim, identifikovane su i određene slabosti kao što su potreba za unapređenjem korisničkog interfejsa radi bolje prilagođenosti potrebama korisnika.

U poređenju sa sličnim rešenjima na tržištu, naše rešenje se ističe po svojoj prilagodljivosti lokalnim zakonima i procedurama saobraćajne policije, što ga čini idealnim izborom za implementaciju u nacionalnim saobraćajnim sistemima.

Dalji pravci za unapređenje obuhvataju implementaciju dodatnih funkcionalnosti poput elektronskog plaćanja kazni direktno kroz aplikaciju, kao i podršku za mobilne platforme radi veće dostupnosti građanima i policiji.

Ovim radom smo pokazali da je softversko rešenje za upravljanje prekršajima u saobraćaju korak napred ka efikasnijem i transparentnijem radu saobraćajne policije, ali i da postoje mogućnosti za dalje unapređenje u cilju još boljeg zadovoljenja potreba korisnika.

## 10. Literatura

- [1] Go Programming Language, <https://go.dev/>.
- [2] Angular , <https://angular.dev/>.
- [3] MongoDB, <https://www.mongodb.com/>.
- [4] HDFS Hadoop, <https://pkg.go.dev/github.com/colinmarc/hdfs>.
- [5] Docker, <https://www.docker.com/>.
- [6] PDF dokumenti, <https://github.com/jung-kurt/gofpdf>.
- [7] slanje email obaveštenja, <https://gopkg.in/gomail.v2>.
- [8] Mumbai Traffic Police,  
<https://trafficpolicemumbai.maharashtra.gov.in/about-department/introduction/>.
- [9] Tripura Traffic Police,  
<https://trafficpolice.tripura.gov.in/general-information>.