

*Univerzitet u Novom Sadu,
Fakultet tehničkih nauka*

SEMINARSKI RAD

Nastavni predmet: Tehnologije i sistemi eUprave

Naziv teme: Zavod za statistiku

Student,

Ana Domonji SR 46/2021

Jun, 2024

1. Sažetak	3
2. Ključne reči.....	3
3. Uvod	4
4. Srodna rešenja i pregled korišćenih tehnologija.....	5
4.1. Srodna istraživanja	5
4.2. Korišćene tehnologije.....	5
5. Specifikacija zahteva	7
5.1. Specifikacija funkcionalnih zahteva	7
5.2. Specifikacija nefunkcionalnih zahteva	12
6. Specifikacija dizajna.....	13
7. Implementacija.....	15
8. Demonstracija	22
9. Zaključak	33
10. Literatura.....	34

1. Sažetak

U ovom radu je opisan sistem Zavoda za statistiku, koji omogućava pregled i generisanje statističkih podataka na osnovu informacija od drugih sistema. Za realizaciju projekta korišćeni su UML dijagrami klasa i slučajeva korišćenja, programski jezik Go [1] za backend, Angular [2] za frontend, dockerizacija servisa, single sign-on za autentifikaciju korisnika i mikroservisna arhitektura za komunikaciju između servisa. Primenom predloženog rešenja omogućena je efikasna obrada i pregled statističkih podataka kao što su broj registrovanih vozila, tipovi prekršaja i saobraćajne nesreće po tipu i stepenu. Korisnici mogu pregledati dostupne statističke podatke i podnositi zahteve za dodatne informacije.

2. Ključne reči

- veb aplikacija
- statistička analiza podataka
- mikroservisna arhitektura
- generisanje izveštaja
- autentifikacija korisnika

3. Uvod

Razvoj savremenog društva donosi sa sobom sve veći obim podataka koji se svakodnevno generišu i koriste u različitim oblastima. Jedan od ključnih izazova u ovom kontekstu je prikupljanje, obrada i analiza tih podataka kako bi se dobile korisne informacije za donošenje informisanih odluka. Upravo tim problemom bavi se Zavod za statistiku, čiji je zadatak da prikuplja podatke iz različitih izvora, kao što su Ministarstvo unutrašnjih poslova (MUP) i saobraćajna policija, i da na osnovu tih podataka generiše relevantne statističke izveštaje.

Problem koji se rešava kroz ovaj projekat je efikasno i precizno prikupljanje, obrađivanje i analiza velikih količina podataka iz različitih izvora. Bez adekvatne aplikacije ili sistema za statističku analizu, prikupljanje i obrada ovih podataka bi bila izuzetno teška i sklona greškama. Manualno rukovanje podacima može dovesti do netačnih informacija, što utiče na donošenje loših odluka.

Motivacija za rešavanje ovog problema leži u potrebi za tačnim, pravovremenim i relevantnim podacima koji su ključni za donošenje odluka u javnoj upravi, planiranju infrastrukture, kao i za istraživačke i analitičke svrhe. Sa tačnim i ažuriranim statistikama, moguće je bolje planirati, predvideti i reagovati na različite situacije.

Istorijski gledano, problem prikupljanja i obrade statističkih podataka postoji otkako postoje administrativni i upravljački sistemi. Tradicionalni pristupi uključivali su ručno prikupljanje podataka, što je bilo veoma naporno, vremenski zahtevno i sklono greškama. S razvojem tehnologije, pojavljivali su se različiti sistemi i softverska rešenja koja su omogućila automatizaciju ovih procesa, ali su često bila fragmentisana i nisu nudila sveobuhvatno rešenje.

Ostatak ovog rada je organizovan kao što je objašnjeno u nastavku. U drugom poglavlju su predstavljena srodna istraživanja koja se bave prikupljanjem i analizom statističkih podataka i korišćene tehnologije u izradi projekta. U trećem poglavlju je data specifikacija zahteva za sistem. Četvrto poglavlje obuhvata specifikaciju dizajna sistema. Implementacija rešenja je predstavljena u petom poglavlju. Šesto poglavlje prikazuje demonstraciju funkcionalnosti sistema. Zaključak je dat u sedmom poglavlju, dok osmo poglavlje sadrži literaturu korišćenu u radu.

4. Srodna rešenja i pregled korišćenih tehnologija

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

4.1. Srodna istraživanja

Ovaj odeljak donosi pregled aplikacija namenjenih prikupljanju i analizi statističkih podataka.

Republički zavod za statistiku Srbije

Republički zavod za statistiku Srbije [3] je glavna institucija u Srbiji za prikupljanje, obradu i analizu statističkih podataka. Na zvaničnom sajtu su dostupni razni statistički podaci, izveštaji i publikacije koje obuhvataju demografske, ekonomske, socijalne i druge statistike. Dobre strane ovog rešenja uključuju visok nivo detaljnosti i obuhvatnosti podataka, kao i redovno ažuriranje informacija. Loša strana može biti složenost pretrage i navigacije kroz obimne podatke, što može otežati korisnicima da brzo pronađu specifične informacije.

Statistički institut Uneska (UNESCO Institute for Statistics)

Statistički institut Uneska [4] je globalna referenca za prikupljanje i analizu statističkih podataka vezanih za obrazovanje, nauku, kulturu i komunikaciju. Institut pruža podatke koji podržavaju kreiranje politika i donošenje odluka na međunarodnom nivou. Prednost ovog rešenja je njegova međunarodna pokrivenost i standardizovani pristup prikupljanju podataka, što omogućava poređenje među zemljama. Nedostatak može biti fokus na specifične oblasti, što ograničava upotrebu podataka za šire ekonomske ili socijalne analize.

4.2. Korišćene tehnologije

Ovaj odeljak pruža pregled i objašnjenja tehnologija pomoću kojih je moguće rešiti problem statističke obrade podataka.

Go (Golang)

Go [2] je programski jezik razvijen od strane Google-a [5], poznat po svojoj efikasnosti i jednostavnosti. Pogodan je za razvoj visokoperformantnih backend sistema, zahvaljujući svom dizajnu koji omogućava lako rukovanje paralelnim operacijama. Go je posebno koristan za mikroservisnu arhitekturu zbog svojih karakteristika kao što su brzina izvršavanja i minimalna potrošnja resursa.

Angular

Angular [1] je platforma za razvoj veb aplikacija razvijena od strane Google-a [5]. Omogućava izradu dinamičkih, responzivnih i skalabilnih veb aplikacija. Jedna od ključnih prednosti Angular-a je modularna arhitektura koja omogućava razvoj kompleksnih aplikacija kroz jednostavno upravljanje komponentama.

Docker

Docker [6] je platforma za kontejnerizaciju aplikacija koja omogućava jednostavno pakovanje, distribuciju i pokretanje softverskih aplikacija u izolovanim okruženjima. Ovo je posebno korisno za razvoj i implementaciju mikroservisa, jer Docker omogućava lako skaliranje i upravljanje servisima.

Single Sign-On (SSO)

Single Sign-On (SSO) je metod autentifikacije koji omogućava korisnicima da se prijave u više aplikacija sa jednim skupom akreditiva. Ovo značajno poboljšava korisničko iskustvo i sigurnost, smanjujući potrebu za upravljanjem više lozinki.

5. Specifikacija zahteva

U ovom poglavlju su objašnjeni funkcionalni i nefunkcionalni zahtevi softverskog rešenja predstavljenog u ovom radu.

5.1. Specifikacija funkcionalnih zahteva

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

Slučajevi korišćenja

Tabela 1 prikazuje opis slučaja korišćenja **“Pretraga i pregled statističkih podataka”**.

Naziv	Pretraga i pregled statističkih podataka
Učesnici	Građanin
Preduslovi	Građanin je pristupio sistemu
Koraci	1. Građanin unosi kriterijume pretrage 2. Sistem vrši pretragu na osnovu unetih kriterijuma 3. Građanin pregleda rezultate pretrage 4. Građanin analizira statističke podatke 5. Sistem prikazuje rezultate pretrage
Rezultat	Građanin ima uvid u željene statističke podatke
Izuzeci	Nisu pronađeni podaci koji odgovaraju unetim kriterijumima

Tabela 1 - Opis slučaja korišćenja “Pretraga i pregled statističkih podataka”

Tabela 2 prikazuje opis slučaja korišćenja **“Zahtevanje dodatnih statističkih podataka”**.

Naziv	Zahtevanje dodatnih statističkih podataka
Učesnici	Građanin
Preduslovi	Građanin je pristupio sistemu
Koraci	1. Građanin bira opcije za slanje zahteva za dodatnim podacima 2. Građanin popunjava zahtev 3. Građanin čeka odgovor
Rezultat	Građaninu su dostavljani dodatni statistički podatci koji su traženi
Isozaki	Zahtev nije ispunjen

Tabela 2 - Opis slučaja korišćenja “Zahtevanje dodatnih statističkih podataka”

Tabela 3 prikazuje opis slučaja korišćenja “**Dostavljanje odgovora građaninu**”.

Naziv	Dostavljanje odgovora građaninu
Učesnici	Zaposleni
Preduslovi	Zaposleni je pristupio sistemu
Koraci	1. Zaposleni prima zahtev od građanina 2. Zaposleni analizira zahtev i priprema odgovor 3. Zaposleni šalje odgovor građaninu
Rezultat	Građanin dobija odgovor na svoj zahtev.
Izuzeci	Odgovor nije moguće pripremiti ili dostaviti

Tabela 3 - Opis slučaja korišćenja “Dostavljanje odgovora građaninu”

Tabela 4 prikazuje opis slučaja korišćenja “**Generisanje izveštaja o broju registrovanih vozila**”.

Naziv	Generisanje izveštaja o broju registrovanih vozila
Učesnici	Zaposleni
Preduslovi	Zaposleni ima pristup podacima o registraciji vozila od MUP-a
Koraci	1. Zaposleni bira opciju za generisanje izveštaja 2. Zaposleni definiše parametre izveštaja 3. Zaposleni pristupa podacima od MUP-a 4. Zaposleni generiše izveštaj 5. Zaposleni pregleda generisani izveštaj 6. Zaposleni objavljuje izveštaj
Rezultat	Zaposleni je generisao izveštaj o broju registrovanih vozila
Izuzeci	Problem u generisanju izveštaja

Tabela 4 - Opis slučaja korišćenja “Generisanje izveštaja o broju registrovanih vozila”

Tabela 5 prikazuje opis slučaja korišćenja “**Generisanje izveštaja o tipu prekršaja**”.

Naziv	Generisanje izveštaja o tipu prekršaja
Učesnici	Zaposleni
Preduslovi	Zaposleni ima pristup podacima o registraciji vozila od MUP-

	a
Koraci	1. Zaposleni bira opciju za generisanje izveštaja 2. Zaposleni definiše parametre izveštaja 3. Zaposleni pristupa podacima od Saobraćajne Policije 4. Zaposleni generiše izveštaj 5. Zaposleni pregleda generisani izveštaj 6. Zaposleni objavljuje izveštaj
Rezultat	Zaposleni je generisao izveštaj o tipu prekršaja
Izuzeci	Problem u generisanju izveštaja

Tabela 5 - Opis slučaja korišćenja “Generisanje izveštaja o tipu prekršaja”

Tabela 6 prikazuje opis slučaja korišćenja “**Generisanje izveštaja o tipu saobraćajne nesreće**”.

Naziv	Generisanje izveštaja o tipu saobraćajne nesreće
Učesnici	Zaposleni
Preduslovi	Zaposleni ima pristup podacima o registraciji vozila od Saobraćajne Policije
Koraci	1. Zaposleni bira opciju za generisanje izveštaja 2. Zaposleni definiše parametre izveštaja 3. Zaposleni pristupa podacima od Saobraćajne Policije 4. Zaposleni generiše izveštaj 5. Zaposleni pregleda generisani izveštaj 6. Zaposleni objavljuje izveštaj
Rezultat	Zaposleni je generisao izveštaj o tipu saobraćajne nesreće
Izuzeci	Problem u generisanju izveštaja

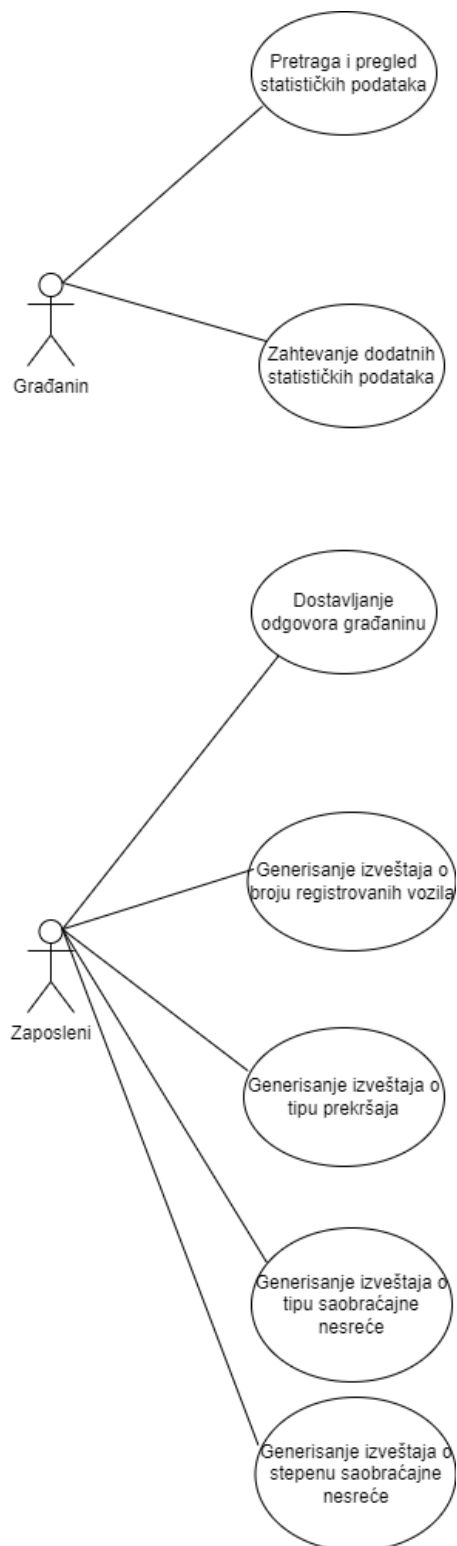
Tabela 6 - Opis slučaja korišćenja “Generisanje izveštaja o tipu saobraćajne nesreće”

Tabela 7 prikazuje opis slučaja korišćenja “**Generisanje izveštaja o stepenu saobraćajne nesreće**”.

Naziv	Generisanje izveštaja o stepenu saobraćajne nesreće
Učesnici	Zaposleni
Preduslovi	Zaposleni ima pristup podacima o registraciji vozila od Saobraćajne Policije
Koraci	1. Zaposleni bira opciju za generisanje izveštaja 2. Zaposleni definiše parametre izveštaja 3. Zaposleni pristupa podacima od Saobraćajne Policije 4. Zaposleni generiše izveštaj 5. Zaposleni pregleda generisani izveštaj

	6. Zaposleni objavljuje izveštaj
Rezultat	Zaposleni je generisao izveštaj o stepenu saobraćajne nesreće
Izuzeci	Problem u generisanju izveštaja

Tabela 7 - Opis slučaja korišćenja “Generisanje izveštaja o stepenu saobraćajne nesreće”



Slika 1 – UML dijagram slučaja korišćenja

5.2. Specifikacija nefunkcionalnih zahteva

Pomoću specifikacije nefunkcionalnih zahteva se definišu svojstva softverskog rešenja koja su potrebna da bi se dati problem rešio, ali ne predstavljaju njegove funkcionalnosti.

1. **Kontejnerizacija**

Sve mikroservise i baze podataka potrebno je pokrenuti kao Docker kontejnere i koristiti Docker Compose alat.

2. **Otpornost na parcijalne otkaze sistema**

U slučaju da neki servis trenutno nije dostupan, svi ostali servisi treba da nastavu da rade normalno i podrže funkcionalnosti koje nisu zavisne od navedenog servisa.

3. **Upotrebljivost:** Sistem mora biti jednostavan za korišćenje, sa intuitivnim korisničkim interfejsom.

6. Specifikacija dizajna

Ovo poglavlje objašnjava dizajn softverskog rešenja za Zavod za statistiku.

Arhitektura sistema

Arhitektura sistema je koncipirana kao mikroservisna arhitektura koja omogućava modularnost, skalabilnost i jednostavno održavanje. Sistem se sastoji od više komponenti koje međusobno komuniciraju preko API interfejsa.

- **Korisnički interfejs (frontend):** Omogućava korisnicima interakciju sa sistemom kroz web aplikaciju.
- **Autentifikacioni servis:** Upravljanje autentifikacijom korisnika koristeći Single Sign-On (SSO).
- **Servis za pretragu i pregled podataka:** Obrada zahteva korisnika za pretragu i pregled statističkih podataka.
- **Servis za generisanje izveštaja:** Kreiranje izveštaja na osnovu definisanih parametara i podataka iz baza.
- **Baza podataka:** Čuvanje svih relevantnih podataka za sistem.

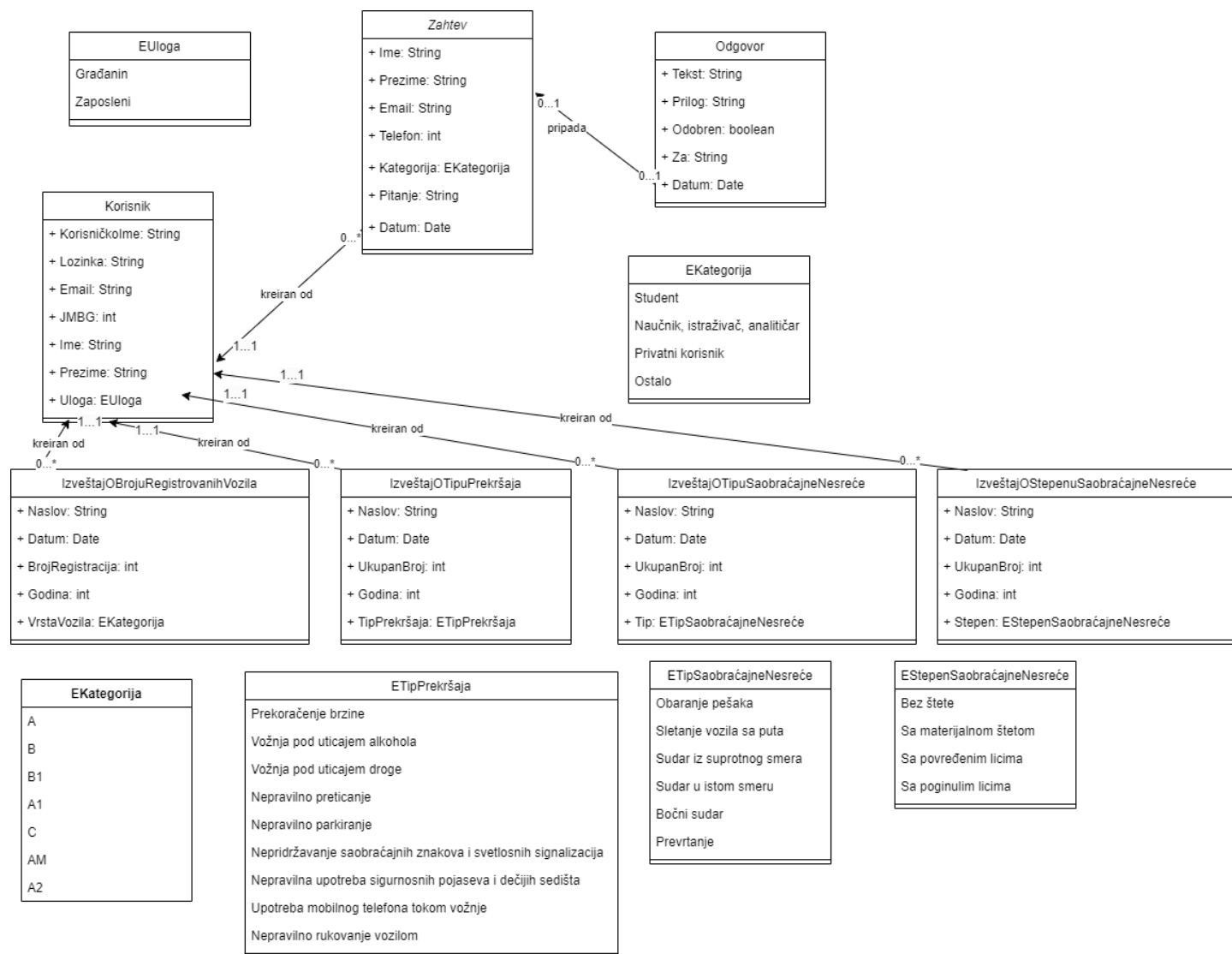
Komponente sistema su prikazane na dijagramu klasa, što je prikazano na slici 2. Dijagram klasa prikazuje ključne entitete sistema Zavoda za statistiku i njihove međusobne odnose. Centralni deo dijagrama čini klasa "Korisnik", koja reprezentuje korisnike sistema i sadrži njihove lične podatke i podatke za autentifikaciju na sistem. Svaki korisnik ima tačno jednu ulogu, što je modelovano vezom sa enumeracijom "EUloga" koja definiše različite uloge korisnika, poput "Zaposleni" ili "Građanin".

Korisnici mogu podnositi zahteve za dodatnim podacima, što je predstavljeno klasom "Zahtev". Svaki zahtev sadrži informacije o datumu i opisu zahteva. Na ove zahteve može biti odgovoreno putem klase "Odgovor", koja sadrži datum i tekst odgovora. Kardinalitet između "Zahtev" i "Odgovor" pokazuje da svaki zahtev može imati najviše jedan odgovor.

Za generisanje izveštaja, dijagram obuhvata klase kao što su "IzveštajORegistracijiVozila", "IzveštajOTipuPrekršaja", "IzveštajOTipuSaobraćajneNesreće" i "IzveštajOSTepenSaobraćajneNesreće". Ove klase beleže podatke specifične za svaki tip izveštaja, poput broja vozila, tipa prekršaja, tipa nesreće i stepena nesreće. Svaki zaposleni može kreirati više ovih izveštaja.

Kategorije izveštaja i prekršaja su modelovane putem enumeracija "EKategorija", "ETipPrekršaja", "ETipSaobraćajneNesreće" i "ESTepenSaobraćajneNesreće". Ove enumeracije definišu različite kategorije, tipove prekršaja i nesreća, i stepene nesreća, čime se omogućava standardizacija i klasifikacija podataka u sistemu.

Ovaj dijagram klase daje jasan uvid u strukturu sistema i načine na koje su podaci povezani i obrađeni, čime se postiže efikasna statistička obrada i generisanje izveštaja.



Slika 2 – UML dijagram klase

7. Implementacija

Ovo poglavlje prikazuje način na koji su implementirane neke od funkcija sistema za Zavod za statistiku. Objašnjena je implementacija ključnih komponenti sistema, uključujući kreiranje izveštaja o broju registrovanih vozila, autentifikaciju korisnika, obradu odgovora na zahteve i dobavljanje izveštaja o saobraćajnim nesrećama.

Implementacija kreiranja izveštaja o broju registrovanih vozila

U ovom odeljku je predstavljena implementacija funkcije za kreiranje izveštaja o broju registrovanih vozila prikazana na listingu 1.

U ovoj funkciji se obrađuju zahtevi za kreiranje izveštaja o broju registrovanih vozila. Na početku se preuzimaju parametri kategorije i godine iz URL-a i proverava se validnost godine. Potom se preuzima token iz zaglavlja zahteva kako bi se autentifikovao korisnik pomoću funkcije `GetCurrentUserFromAuthService`. Ako korisnik nema odgovarajuću ulogu (zaposleni), vraća se greška.

Zatim se obrađuje telo zahteva kako bi se preuzeo naslov izveštaja i preuzima se ukupan broj registrovanih vozila za zadatu kategoriju i godinu pomoću funkcije `GetVehiclesCountAndYearVehiclesService`. Nakon toga se kreira novi izveštaj sa trenutnim vremenom i jedinstvenim identifikatorom, koji se zatim čuva u bazi podataka. Ako je sve uspešno, vraća se odgovor sa porukom o uspešnom kreiranju izveštaja.

```

func (r *ReportRegisteredVehiclesHandler) CreateReport(c *gin.Context) { 1 usage  ± Ana Domonji
    rw := c.Writer
    h := c.Request

    category := c.Param( key: "category")
    yearStr := c.Param( key: "year")
    year, err := strconv.Atoi(yearStr)
    if err != nil {
        errorMsg := map[string]string{"error": "Invalid year parameter"}
        errorMessage.ReturnJSONError(rw, errorMsg, http.StatusBadRequest)
        return
    }

    token := h.Header.Get( key: "Authorization")
    user, err := r.GetCurrentUserFromAuthService(token, rw)
    if err != nil { return }

    if user.UserRole != domain.Employee {
        errorMessage.ReturnJSONError(rw, map[string]string{"error": "Unauthorized. You are not an employee."}, http.StatusUnauthorized)
        return
    }
    var requestBody struct {
        Title string `json:"title" binding:"required"`
    }

    if err := c.BindJSON(&requestBody); err != nil {
        c.JSON(http.StatusBadRequest, gin.H{"error": "Failed to parse request body"})
        return
    }

    totalNumber, err := r.GetVehiclesCountAndYearVehiclesService(token, domain.Category(category), year)
    if err != nil {
        c.JSON(http.StatusBadRequest, gin.H{"error": "Failed to obtain delicts information. Try again later."})
        return
    }

    currentDateTime := primitive.NewDateTimeFromTime(time.Now())
    id := primitive.NewObjectID()
    newReport := &domain.ReportRegisteredVehicle{
        ID:          id,
        Category:    domain.Category(category),
        Title:       requestBody.Title,
        Date:        currentDateTime,
        TotalNumber: totalNumber,
        Year:        year,
    }

    err, _ = r.service.Create(newReport)
    if err != nil {
        c.JSON(http.StatusBadRequest, gin.H{"error": "Failed to create report"})
        return
    }

    c.JSON(http.StatusCreated, gin.H{"message": "Report successfully saved", "report": newReport})
}

```

Listing 1 - Kreiranje izveštaja o broju registrovanih vozila

Implementacija autentifikacije korisnika

U ovom odeljku je predstavljena implementacija funkcije za autentifikaciju korisnika putem eksternog servisa za autentifikaciju, koja se koristi u funkciji za kreiranje izveštaja o broju registrovanih vozila i prikazana je na listingu 2.

```
func (r *ReportRegisteredVehiclesHandler) GetCurrentUserFromAuthService(token string, rw http.ResponseWriter) (*domain.User, error) { 1 usage
    url := "http://auth-service:8085/api/users/currentUser"

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

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

    statusCode := resp.StatusCode
    if statusCode != http.StatusOK {
        errorMessage.ReturnJSONError(rw, map[string]string{"error": "Unauthorized."}, http.StatusUnauthorized)
        return nil, errors.New( text: "Unauthorized")
    }

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

    return &domain.User{UserRole: responseUser.LoggedInUser.UserRole}, nil
}
```

Listing 2 - Autentifikacija korisnika putem eksternog servisa

Ova funkcija koristi token iz zaglavlja zahteva da bi se autentifikovao korisnik putem eksternog servisa za autentifikaciju. Koristi se context za postavljanje vremenskog ograničenja na zahtev. Ako je autorizacija uspešna, funkcija vraća korisnika sa njegovom ulogom.

Implementacija preuzimanja broja registrovanih vozila

U ovom odeljku je predstavljena implementacija funkcije za preuzimanje broja registrovanih vozila za određenu kategoriju i godinu, koja se koristi u funkciji za kreiranje izveštaja o broju

registrovanih vozila i prikazana je na listingu 3. Takođe je predstavljena implementacija funkcije za izvršavanje zahteva sa autentifikacijom.

```
func (r *ReportRegisteredVehiclesHandler) GetVehiclesCountAndYearVehiclesService(token string, category domain.Category, year int) (int, error) { 1 usage ± Ana Domonji
    baseURL := fmt.Sprintf( format: "http://vehicles-service:8080/api/vehicle/get/category/%s/year/%d", url.QueryEscape(string(category)), year)

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

    resp, err := r.performAuthorizationRequestWithContext( method: "GET", ctx, token, baseURL)
    if err != nil {
        if errors.Is(ctx.Err(), context.DeadlineExceeded) { return 0, errors.New( text: "vehicles service is not available") }
        return 0, errors.New( text: "error performing request")
    }
    defer resp.Body.Close()

    var vehicles []domain.Delict
    if err := json.NewDecoder(resp.Body).Decode(&vehicles); err != nil {
        return 0, errors.New( text: "failed to retrieve registered vehicles")
    }

    return len(vehicles), nil
}

func (r *ReportRegisteredVehiclesHandler) performAuthorizationRequestWithContext(method string, ctx context.Context, token string, url string) (*http.Response, error) {
    req, err := http.NewRequest(method, url, body: nil)
    if err != nil { return nil, err }
    req.Header.Set( key: "Authorization", token)

    client := &http.Client{}
    resp, err := client.Do(req.WithContext(ctx))
    if err != nil { return nil, err }

    return resp, nil
}
```

Listing 3 - Preuzimanje broja registrovanih vozila za određenu kategoriju i godinu

Ova funkcija preuzima broj registrovanih vozila za određenu kategoriju i godinu. Koristi se URL za servis koji vraća podatke o vozilima i postavlja se vremensko ograničenje za zahtev. Ako je zahtev uspešan, dekodiraju se podaci iz odgovora i vraća se broj vozila.

Funkcija performAuthorizationRequestWithContext kreira i izvršava HTTP zahtev sa zadatim metodom, kontekstom i tokenom za autentifikaciju. Ako je zahtev uspešan, vraća se odgovor.

Implementacija odgovora na zahteve

U ovom odeljku je predstavljena implementacija funkcije za kreiranje odgovora na zahteve korisnika prikazana na listingu 4.

```

func (r *ResponseHandler) Create(c *gin.Context) { // Ana Domonji
    rw := c.Writer
    h := c.Request

    token := h.Header.Get(key: "Authorization")
    user, err := r.GetCurrentUserFromAuthService(token, rw)
    if err != nil { return }

    if user.UserRole != domain.Employee {
        errorMessage.ReturnJSONError(rw, map[string]string{"error": "Unauthorized. You are not an employee."}, http.StatusUnauthorized)
        return
    }

    text := c.PostForm(key: "text")

    acceptedStr := c.PostForm(key: "accepted")
    accepted, err := strconv.ParseBool(acceptedStr)
    if err != nil {
        c.JSON(http.StatusBadRequest, gin.H{"error": "Invalid value for accepted"})
        return
    }

    sendTo := c.PostForm(key: "send_to")

    file, err := c.FormFile(name: "attachment")
    var attachmentPath string
    if err == nil {
        attachmentPath = filepath.Join(elem...: "/tmp", file.Filename)
        if err := c.SaveUploadedFile(file, attachmentPath); err != nil {
            c.JSON(http.StatusInternalServerError, gin.H{"error": "Failed to save attachment"})
            return
        }
    }

    currentDateTime := primitive.NewDateTimeFromTime(time.Now())
    id := primitive.NewObjectID()
    newResponse := &domain.Response{
        ID:        id,
        Text:       text,
        Attachment: attachmentPath,
        Accepted:   accepted,
        SendTo:     sendTo,
        Date:       currentDateTime,
    }

    err, _ = r.service.Create(newResponse)
    if err != nil {
        c.JSON(http.StatusBadRequest, gin.H{"error": "Failed to create response"})
        return
    }

    err = r.sendEmail(newResponse.Text, newResponse.Attachment, newResponse.SendTo)
    if err != nil {
        errorMessage.ReturnJSONError(rw, fmt.Sprintf(format: "Error sending email: %s", err), http.StatusInternalServerError)
        return
    }

    c.JSON(http.StatusCreated, gin.H{"message": "Response successfully saved", "response": newResponse})
}

```

Listing 4 - Kreiranje odgovora na zahteve korisnika

Ova funkcija obrađuje zahteve za kreiranje odgovora na zahteve korisnika. Nakon autentifikacije korisnika, proverava se da li korisnik ima ulogu zaposlenog. Zatim se obrađuju parametri iz zahteva, uključujući tekst odgovora, status prihvatanja i email adresu za slanje odgovora. Ako je priložen fajl, on se čuva na serveru. Novi odgovor se kreira i čuva u bazi podataka, a zatim se šalje email sa odgovorom korisniku, kao što je prikazano na listingu 5.

```
func (r *ResponseHandler) sendEmail(text string, attachment string, email string) error {
    m := gmail.NewMessage()
    m.SetHeader( field: "From", smtpEmail)
    m.SetHeader( field: "To", email)
    m.SetHeader( field: "Subject", value...: "Odgovor na zahtev - Institut za statistiku.")

    m.SetBody( contentType: "text/plain", text)

    if attachment != "" {
        m.Attach(attachment)
    }

    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 5 - Slanje emaila sa odgovorom

Ova funkcija šalje email sa tekstom odgovora i priloženim fajlom na zadatu email adresu. Koristi se “gmail” paket za kreiranje i slanje email poruke. Ako je slanje uspešno, vraća se nil, inače se vraća greška.

Implementacija preuzimanja izveštaja o saobraćajnim nesrećama

U ovom odeljku je predstavljena implementacija funkcije za preuzimanje izveštaja o saobraćajnim nesrećama po tipu nesreće i godini koja je prikazana na listingu 6.

```

func (r *ReportCarAccidentTypeHandler) GetAllByCarAccidentTypeAndYear(c *gin.Context) { 1 usage  ± Ana Domonji
    rw := c.Writer
    carAccidentType := c.Param(key: "carAccidentType")

    yearStr := c.Param(key: "year")
    year, err := strconv.Atoi(yearStr)
    if err != nil {
        errorMsg := map[string]string{"error": "Invalid year parameter"}
        errorMessage.ReturnJSONError(rw, errorMsg, http.StatusBadRequest)
        return
    }

    reports, err := r.service.GetAllByCarAccidentTypeAndYear(domain.CarAccidentType(carAccidentType), year)
    if err != nil {
        errorMsg := map[string]string{"error": "Failed to retrieve reports from the database."}
        errorMessage.ReturnJSONError(rw, errorMsg, http.StatusInternalServerError)
        return
    }

    jsonResponse, err := json.Marshal(reports)
    if err != nil {
        errorMsg := map[string]string{"error": "Error marshaling JSON."}
        errorMessage.ReturnJSONError(rw, errorMsg, http.StatusInternalServerError)
        return
    }
    rw.Header().Set(key: "Content-Type", value: "application/json")
    rw.WriteHeader(http.StatusOK)
    rw.Write(jsonResponse)
}

```

Listing 6 - Preuzimanje izveštaja o saobraćajnim nesrećama po tipu nesreće i godini

Ova funkcija preuzima izveštaje o saobraćajnim nesrećama po tipu nesreće i godini. Na početku se preuzimaju parametri iz URL-a i proverava se validnost godine. Zatim se preuzimaju izveštaji iz baze podataka za zadati tip nesreće i godinu. Ako je zahtev uspešan, podaci se vraćaju kao JSON odgovor.

8. Demonstracija

Ovo poglavlje prikazuje način korišćenja aplikacije za Zavod za statistiku. Prikazani su scenariji u korišćenju softvera koji rešavaju problem prikupljanja, analize i pregleda statističkih podataka.

Prijava na sistem

Prilikom pokretanja aplikacije korisniku se prikazuje stranica za prijavu, kao što je ilustrovano na slici 3. Korisnik unosi svoje kredencijale kako bi se prijavio u sistem.



[Registrujte se](#) [Prijavite se](#)



Prijava

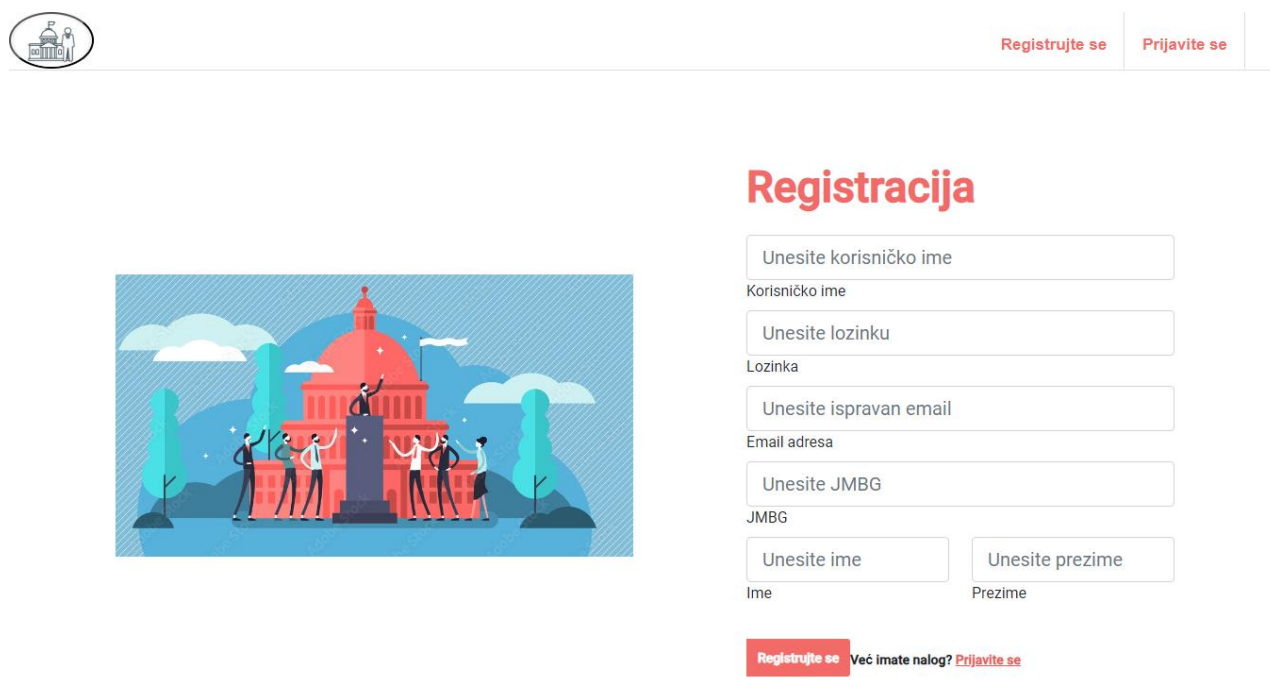
Unesite ispravan email
Email

Unesite lozinku
Lozinka

[Prijavite se](#) [Nemate nalog? Registrujte se](#)

Slika 3 - Prijava na sistem

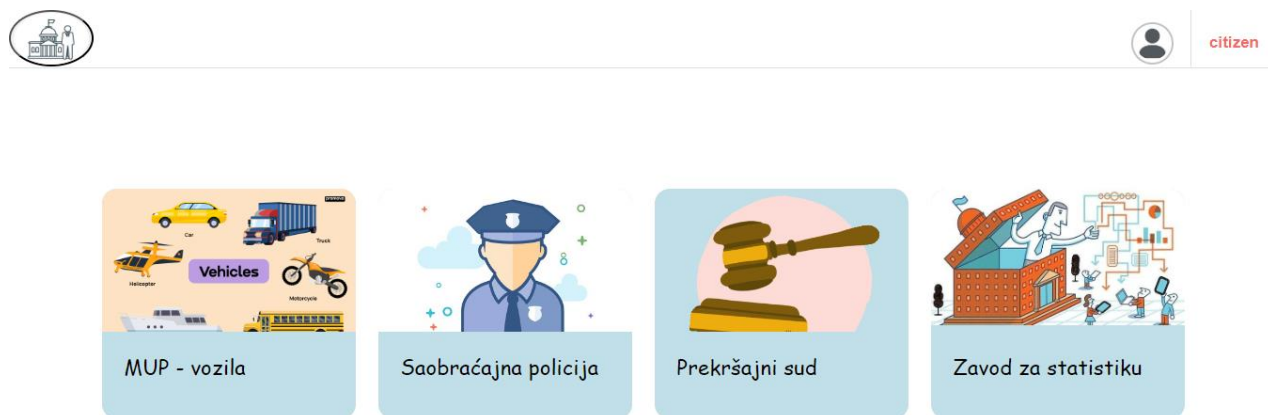
Ako korisnik nema nalog, može se registrovati putem forme za registraciju prikazane na slici 4.



Slika 4 – Registracija na sistem

Početna stranica

Nakon uspešne prijave, korisnik je preusmeren na početnu stranicu gde može birati između četiri sistema: MUP - vozila, Saobraćajna policija, Prekršajni sud i Zavod za statistiku. Početna stranica je prikazana na slici 5.



Slika 5 - Početna stranica

Zavod za statistiku - Izveštaj o broju registrovanih vozila

Korisnik zatim bira Zavod za statistiku, što ga vodi do početne stranice zavoda, prikazane na slici 6. Ova stranica prikazuje tabelu sa izveštajima o broju registrovanih vozila, koja sadrži naslov, period za koji je izveštaj, kategoriju vozila i datum ažuriranja izveštaja. Takođe je prikazan grafički prikaz godine sa najviše registrovanih vozila i kategorije vozila sa najviše registrovanih.

Na ovoj stranici korisnik može birati između izveštaja o registrovanim vozilima, stepenu saobraćajnih nesreća, tipu saobraćajnih nesreća i tipu prekršaja.



Slika 6 - Izveštaj o broju registrovanih vozila

Izveštaji o saobraćajnim nesrećama

Kada korisnik izabere "Stepen saobraćajnih nesreća", prikazuje se stranica kao na slici 7, koja sadrži tabelu sa izveštajima o broju saobraćajnih nesreća po stepenu, uključujući naslov, period,

stepen nesreće i datum ažuriranja izveštaja. Pored toga, tu je i grafički prikaz godine sa najviše nesreća i stepena sa najviše nesreća.



Zahtevi



employee

ZAVOD ZA STATISTIKU

E-UPRAVA

Registrovana vozila

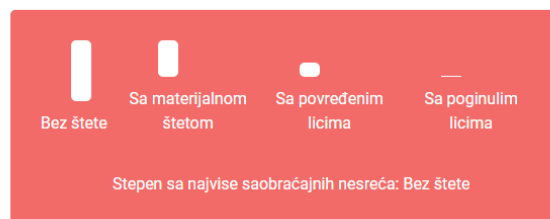
Stepen saobraćajnih nesreća

Tip saobraćajnih nesreća

Prekršaji

IZVEŠTAJ O STEPENU SAOBRAĆAJNE NESREĆE

Naslov	Period	Stepen nesreće				Datum ažuriranja
		Bez štete	Sa materijalnom štetom	Sa povređenim licima	Sa poginulim licima	
Izveštaj o stepenu saobraćajne nesreće 2024	2024	25	17	6	-	May 28, 2024
Izveštaj o stepenu saobraćajne nesreće 2023	2023	45	25	10	1	Dec 28, 2023



Slika 7 - Izveštaj o stepenu saobraćajne nesreće

Zatim kada korisnik izabere "Tip saobraćajnih nesreća", prikazuje se stranica kao na slici 8, koja sadrži tabelu sa izveštajima o broju saobraćajnih nesreća po tipu, uključujući naslov, period, tip nesreće i datum ažuriranja izveštaja. Uz tabelu, dostupan je grafički prikaz godine sa najviše nesreća i tipa sa najviše nesreća.



Zahtevi



employee

ZAVOD ZA STATISTIKU

E-UPRAVA

Registrovana vozila

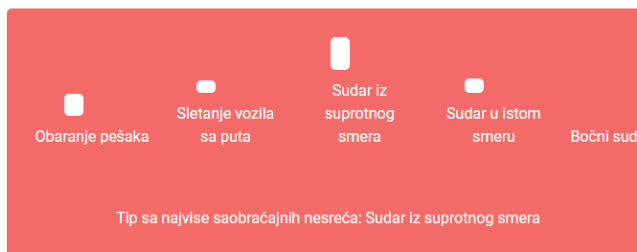
Stepen saobraćajnih nesreća

Tip saobraćajnih nesreća

Prekršaji

IZVEŠTAJ O TIPU SAOBRAĆAJNE NESREĆE

Naslov	Period	Tip nesreće						Datum ažuriranja
		Obaranje pešaka	Sletanje vozila sa puta	Sudar iz suprotnog smera	Sudar u istom smeru	Bočni sudar	Prevrtanje	
Izveštaj o tipu saobraćajne nesreće 2024	2024	3	5	6	1	-	-	May 23, 2024
Izveštaj o tipu saobraćajne nesreće 2023	2023	15	5	21	11	-	5	Dec 23, 2023



Slika 8 - Izveštaj o tipu saobraćajne nesreće

Izveštaji o prekršajima

Izborom "Tip prekršaja", prikazuje se stranica kao na slici 9, koja sadrži tabelu sa izveštajima o broju prekršaja po tipu, uključujući naslov, period, tip prekršaja i datum ažuriranja izveštaja. Dodatno, grafički je prikazana godina sa najviše prekršaja i tip sa najviše prekršaja.



Zahtevi



employee

ZAVOD ZA STATISTIKU

E-UPRAVA

Registrovana vozila

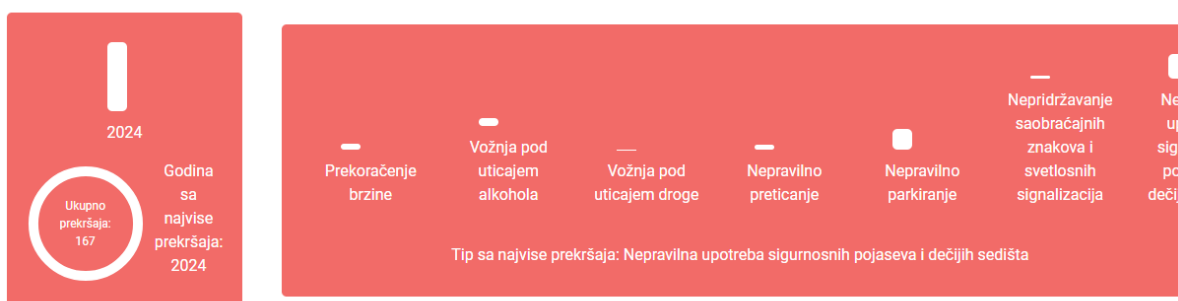
Stepen saobraćajnih nesreća

Tip saobraćajnih nesreća

Prekršaji

IZVEŠTAJ O TIPU PREKRŠAJA

Naslov	Period	Tip prekršaja									Datum ažuriranja
		Prekoračenje brzine	Vožnja pod uticajem alkohola	Vožnja pod uticajem droge	Nepravilno preticanje	Nepravilno parkiranje	Nepridržavanje saobraćajnih znakova i svetlosnih signalizacija	Nepravilna upotreba sigurnosnih pojaseva i dečijih sedišta	Upotreba mobilnog telefona tokom vožnje	Nepravilno rukovanje vozilom	
Izveštaj o tipu prekršaja 2024	2024	15	20	1	13	50	8	60	-	-	May 29, 2024



Slika 9 - Izveštaj o tipu prekršaja

Slanje zahteva za dodatnim statističkim podacima

Korisnik može poslati zahtev za dodatnim statističkim izveštajima ili postaviti pitanje putem forme prikazane na slici 10.

PITAJTE NAS

Ukoliko niste uspjeli da pronađete potrebne podatke i informacije na našem sajtu, možete nam uputiti zahtev popunjavanjem jednog od ponuđenih formulara. Na vaš zahtev biće odgovoreno što je pre moguće i u skladu sa zakonskim rokovima.

Ana

Domonji

domonjianna@gmail.com

0653535628

STUDENT

Dobar dan, Pisem rad na fakultetu i potrebni su mi dodatni statistički pod

ODUSTANI POŠALJI

Slika 10 - Slanje zahteva

Generisanje izveštaja od strane zaposlenih

Zaposleni imaju pristup formama za generisanje različitih izveštaja. Na slici 11 je prikazana forma za generisanje izveštaja o kategoriji vozila, koja sadrži naslov, godinu za koju se generiše izveštaj i kategoriju vozila.

GENERIŠI IZVEŠTAJ O KATEGORIJI VOZILA

Naslov

Izvestaj o broju registrovanih 2024 godine

Kategorija vozila

B1

Godina

2024

ODUSTANI KREIRAJ

Slika 11 - Generisanje izveštaja o kategoriji vozila

Na slici 12 je forma za generisanje izveštaja o stepenu saobraćajne nesreće, koja sadrži naslov, godinu za koju se generiše izveštaj i stepen saobraćajne nesreće.

GENERIŠI IZVEŠTAJ O STEPENU SAOBRAĆAJNE NESREĆE

Naslov

Izveštaj o stepenu saobraćajne nesreće 2024

Stepen saobraćajne nesreće

Bez štete

Godina

2024

ODUSTANI KREIRAJ

Slika 12 - Generisanje izveštaja o stepenu saobraćajne nesreće

Na slici 13 može se videti forma za generisanje izveštaja o tipu saobraćajne nesreće, koja sadrži naslov, godinu za koju se generiše izveštaj i tip saobraćajne nesreće.

GENERIŠI IZVEŠTAJ O TIPU SAOBRAĆAJNE NESREĆE

Naslov

Izveštaj o tipu saobraćajne nesreće

Stepen saobraćajne nesreće

Bočni sudar

Godina

2024

ODUSTANI KREIRAJ

Slika 13 - Generisanje izveštaja o tipu saobraćajne nesreće

Slika 14 je prikazuje formu za generisanje izveštaja o tipu prekršaja, koja sadrži naslov, godinu za koju se generiše izveštaj i tip prekršaja.

GENERIŠI IZVEŠTAJ O TIPU PREKRŠAJA

Naslov

Izveštaj o tipu prekršaja

Tip prekršaja

Upotreba mobilnog telefona tokom vožnje

Godina

2024

ODUSTANI KREIRAJ

Slika 14 - Generisanje izveštaja o tipu prekršaja

Pregled zahteva građana

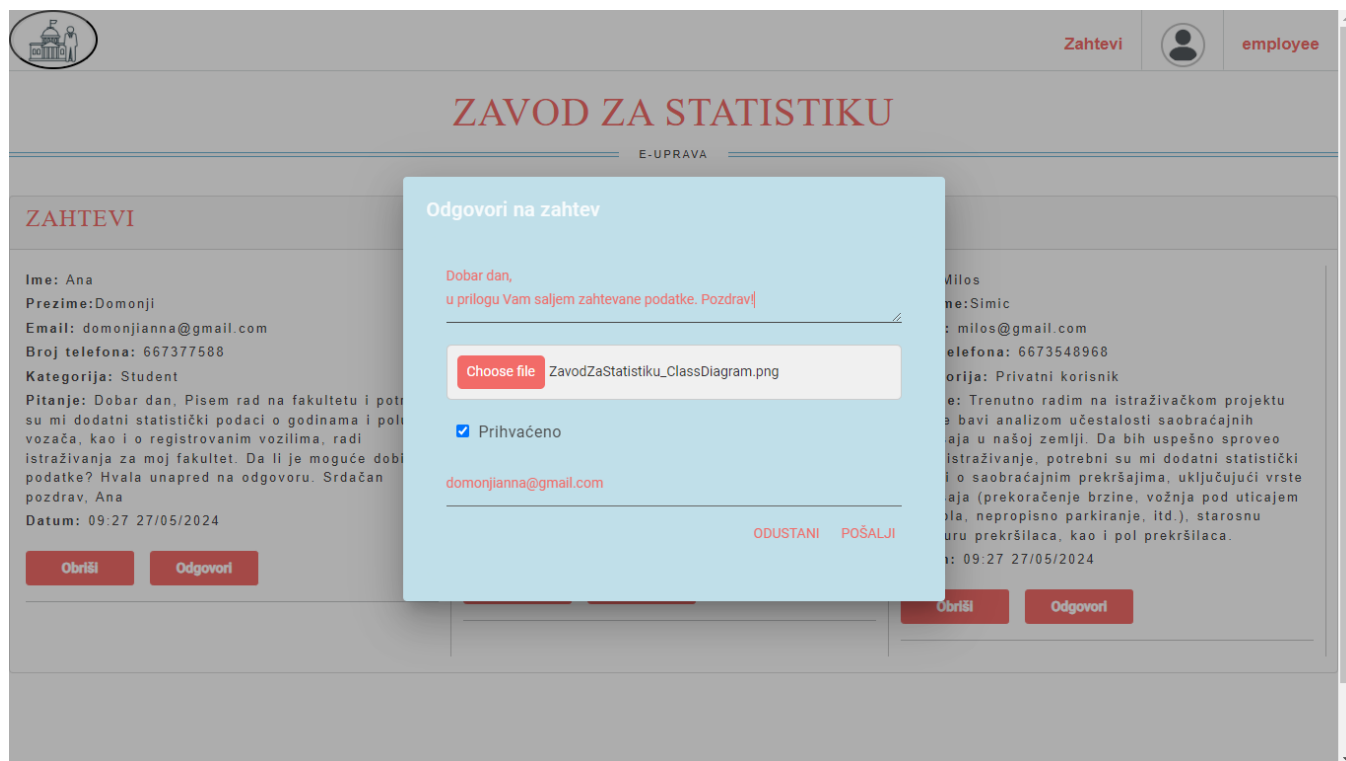
Na slici 15 je prikazan pregled svih zahteva koje su građani poslali. Zaposleni mogu pregledati, brisati ili odgovarati na ove zahteve.

	Zahtevi		employee
ZAVOD ZA STATISTIKU			
E-UPRAVA			
ZAHTEVI			
<p>Ime: Ana Prezime: Domonji Email: domonjianna@gmail.com Broj telefona: 667377588 Kategorija: Student Pitanje: Dobar dan, Pisem rad na fakultetu i potrebni su mi dodatni statistički podaci o godinama i polu vozača, kao i o registrovanim vozilima, radi istraživanja za moj fakultet. Da li je moguće dobiti te podatke? Hvala unapred na odgovoru. Srdačan pozdrav, Ana Datum: 09:27 27/05/2024</p> <p>Obrisi</p> <p>Odgovori</p>	<p>Ime: Milica Prezime: Mikic Email: mikicmilica@gmail.com Broj telefona: 6672345688 Kategorija: Student Pitanje: Poštovani, Zovem se Milica i trenutno radim na istraživačkom projektu koji se bavi analizom uticaja starosti vozila na učestalost saobraćajnih nezgoda. U sklopu mog rada, potrebni su mi dodatni statistički podaci o starosti vozila, vrsti vozila (putnička, teretna, itd.), kao i o broju registrovanih saobraćajnih nezgoda po vrsti vozila. Datum: 09:27 27/05/2024</p> <p>Obrisi</p> <p>Odgovori</p>	<p>Ime: Milos Prezime: Simic Email: milos@gmail.com Broj telefona: 6673548968 Kategorija: Privatni korisnik Pitanje: Trenutno radim na istraživačkom projektu koji se bavi analizom učestalosti saobraćajnih prekršaja u našoj zemlji. Da bih uspešno sproveo svoje istraživanje, potrebni su mi dodatni statistički podaci o saobraćajnim prekršajima, uključujući vrste prekršaja (prekoračenje brzine, vožnja pod uticajem alkohola, nepropisno parkiranje, itd.), starosnu strukturu prekršilaca, kao i pol prekršilaca. Datum: 09:27 27/05/2024</p> <p>Obrisi</p> <p>Odgovori</p>	

Slika 15 - Pregled zahteva

Odgovaranje na zahteve

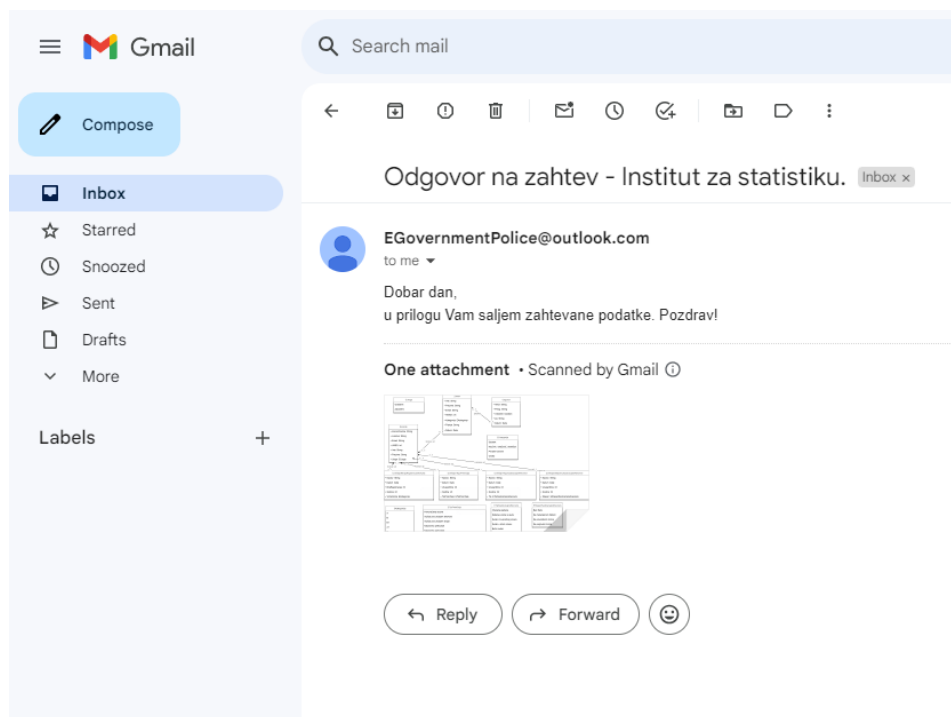
Na slici 16 je forma za odgovaranje na zahtev, gde zaposleni unosi odgovor, status prihvatanja zahteva i može priložiti fajl sa dodatnim informacijama.



Slika 16 - Odgovaranje na zahtev

Prijem odgovora putem emaila

Slika 17 je prikaz odgovora na zahtev putem emaila.



Slika 17 - Odgovor putem emaila

Ova demonstracija prikazuje kako korisnici i zaposleni mogu koristiti aplikaciju za prikupljanje, analizu i pregled statističkih podataka, kao i za komunikaciju i razmenu informacija putem sistema Zavoda za statistiku.

9. Zaključak

U ovom radu je predstavljen projekat Zavoda za statistiku, razvijen korišćenjem tehnologija kao što su Go[2] i Angular[1], i implementacijom mikroservisne arhitekture uz Docker[6] kontejnerizaciju. Sistem omogućava efikasno prikupljanje, analizu i pregled statističkih podataka, a takođe omogućava korisnicima da podnose zahteve za dodatnim informacijama i dobiju odgovore od zaposlenih.

Kroz implementaciju različitih funkcionalnosti, kao što su generisanje izveštaja o broju registrovanih vozila, tipu prekršaja, tipu i stepenu saobraćajnih nesreća, demonstrirano je kako softversko rešenje može značajno olakšati proces statističke obrade podataka. Upotreba SSO sistema omogućava bezbednu autentifikaciju korisnika.

Ovaj projekat pokazuje kako tehnologija može biti iskorišćena za unapređenje procesa statističke obrade podataka, pružajući korisnicima pouzdane informacije i olakšavajući rad zaposlenima.

10. Literatura

- [1] Angular. 2024. Angular Documentation, <https://angular.dev/>.
- [2] Go. 2024. Go Programming Language, <https://go.dev/>.
- [3] Republički zavod za statistiku Srbije. 2024. <https://www.stat.gov.rs/>.
- [4] UNESCO Institute for Statistics. 2024. <https://uis.unesco.org/>.
- [5] Google. 2024. <https://www.google.com/>.
- [6] Docker. 2024. <https://www.docker.com/>.