

# Medical servis u okviru sistema eUprava

---

Seminarski rad

Kristina Milošević  
Fakultet tehničkih nauka  
Univerzitet u Novom Sadu  
milosevic.sr14.2022@uns.ac.rs

## Sažetak

U ovom radu opisan je medical servis, kao deo integrisanog sistema eUprava, čija je svrha digitalizacija zdravstvenih procesa i olakšavanje komunikacije između učenika i doktora. Sistem je osmišljen tako da predstavlja mikroservisnu arhitekturu koja omogućava razmenu podataka između tri servisa: Auth, Medical i School.

Auth servis se koristi za autentifikaciju i autorizaciju korisnika putem Single Sign-On mehanizma, ali i putem standardnog načina prijave i registracije.

Medical servis omogućava podnošenje i obradu medicinskih zahteva, pregled kartona pacijenta i elektronsko izdavanje potvrda i uverenja.

Medical servis je povezan sa School servisom, čime se omogućava elektronsko dostavljanje medicinskih potvrda obrazovnim ustanovama, obaveštenje učeniku o kreiranim opravdanjima, kao i slanje zahteva za medicinski pregled i opravdanje

Za realizaciju je korišćen Go za backend implementaciju, Angular za frontend deo aplikacije uz korišćenje Tailwind CSS za stilizaciju i Postgres za čuvanje podataka. .

Primena predloženog rešenja omogućava učenicima jednostavno podnošenje i praćenje medicinskih zahteva, smanjuje potrebu za fizičkim odlaskom u zdravstvene ustanove i unapređuje efikasnost zdravstvenog sistema. Razvijen je prototip i objašnjeno kako korisnici obično koriste aplikaciju.

## Ključne reči

mikroservisna arhitektura, eUprava, medical servis, auth servis, school servis

## 1. Uvod

Jedan od ključnih problema u javnoj upravi je složenost i sporost procesa vezanih za zdravstvenu dokumentaciju i komunikaciju sa doktorima. Tradicionalni pristupi uključuju odlazak u ambulantu, ručno popunjavanje formulara i čekanje u redovima. To je vremenski zahtevno i podložno greškama, naročito u svetu u kom danas živimo.

Cilj medical servisa u okviru aplikacije eUprava jeste digitalizacija ovih procesa i omogućavanje učenicima da na jednom mestu podnose zahteve doktorima, pristupaju svojim medicinskim dokumentima i elektronski komuniciraju sa zdravstvenim ustanovama.

Ostatak rada je organizovan na sledeći način: u drugom poglavlju prikazana su srodna rešenja i tehnologije, u trećem poglavlju specifikacija zahteva, četvrto poglavlje opisuje dizajn, peto implementaciju, šesto demonstraciju rada sistema, a sedmo zaključak i pravce budućeg razvoja.

## 2. Srodna rešenja i korišćene tehnologije

### Uvod

U ovom poglavlju dat je pregled postojećih rešenja u oblasti digitalizacije zdravstvenih usluga, koja su poslužila kao inspiracija za razvoj našeg sistema. Takođe, prikazane su tehnologije koje omogućavaju implementaciju mikroservisnih arhitektura i funkcionalnosti sistema poput Medical servisa unutar integrisanog sistema eUprava.

---

### 2.1 Srodna rešenja

Ovaj odeljak pruža pregled aplikacija i sistema koji se bave sličnim problemima, odnosno elektronskim izdavanjem zdravstvenih dokumenata i digitalizacijom zdravstvenih procesa.

#### **eUprava – Zdravlje [1]**

Sistem *eUprava* omogućava građanima pristup elektronskim uslugama u oblasti zdravlja, uključujući izdavanje zdravstvenih kartica i drugih relevantnih usluga. Prednost ovog sistema je što integriše više usluga u jedan portal, a mana to što korisnički interfejs ponekad može biti neintuitivan za nove korisnike.

#### **MyChart – Elektronski zdravstveni portal [2]**

*MyChart* omogućava pacijentima pristup zdravstvenim podacima, zakazivanje pregleda i komunikaciju sa lekarima. Prednost je jednostavna upotreba i široka primena, dok je mana to što je ograničen na zdravstvene ustanove koje koriste takve sisteme.

---

### 2.2 Korišćene tehnologije

Ovaj odeljak pruža pregled tehnologija koje omogućavaju razvoj i implementaciju funkcionalnosti ovog sistema.

**Go [3]** je programski jezik koji omogućava backend implementaciju i podržava mikroservisnu arhitekturu. Njegova prednost je visoka brzina izvršavanja, konkurentnost i jednostavnost.

**Angular [4]** je frontend framework za izgradnju interaktivnih i responzivnih korisničkih interfejsa. Prednost je bogat skup funkcionalnosti.

**Tailwind CSS [5]** omogućava brzo i fleksibilno stilizovanje veb aplikacija, posebno za kreiranje responzivnog dizajna. Prednost je brz razvoj i velika prilagodljivost.

**PostgreSQL [6]** je sistem za upravljanje relacionim bazama podataka. Prednost je pouzdanost, podrška za kompleksne upite i skalabilnost.

**Single Sign-On (SSO) [7]** je mehanizam za autentifikaciju i autorizaciju korisnika, implementiran preko *Auth* servisa upotrebom Firebase Authentication. Prednost je jednostavno upravljanje pristupom i sigurnost, dok mana može biti zavisnost od spoljnog servisa.

### 3. Specifikacija zahteva

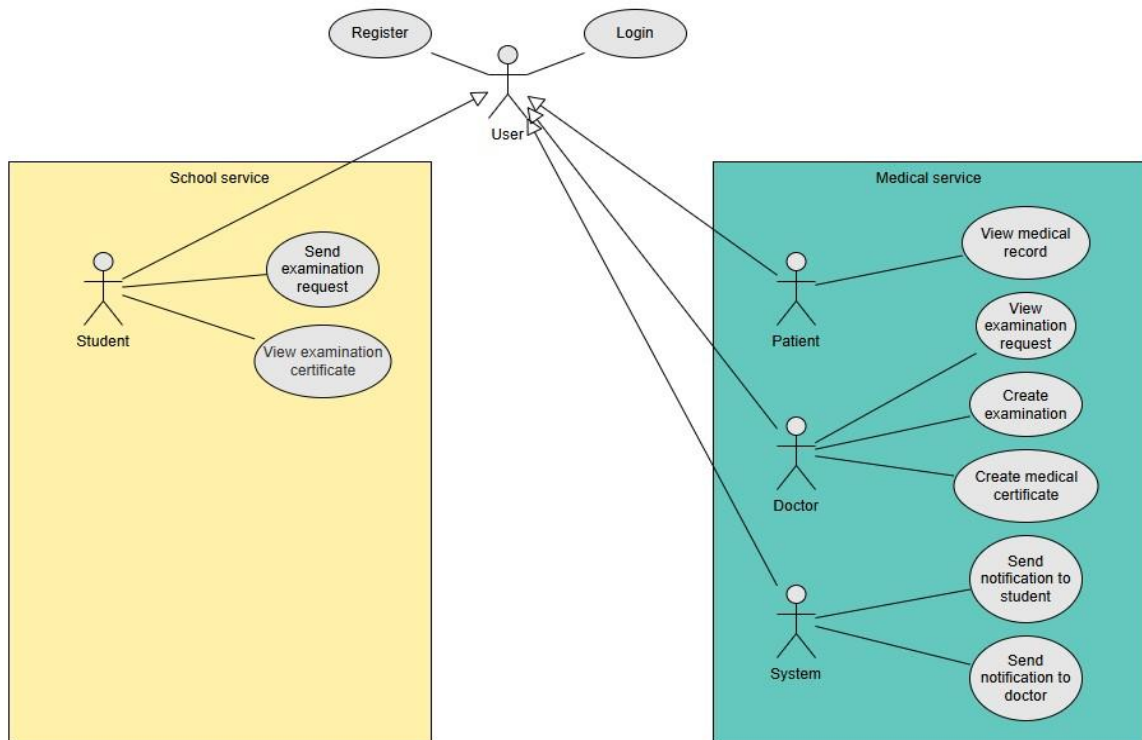
#### Uvod

U ovom poglavlju predstavljeni su funkcionalni i nefunkcionalni zahtevi softverskog rešenja razvijenog u okviru Medical servisa sistema eUprava. Takođe je opisan dizajn sistema i način organizacije komponenti.

#### 3.1 Specifikacija funkcionalnih zahteva

U ovom odeljku opisani su funkcionalni zahtevi koje softversko rešenje mora da ispunjava, odnosno šta korisnici mogu da rade i koje operacije sistem podržava. Funkcionalni zahtevi predstavljeni su kroz UML dijagram slučajeva korišćenja, kao što je prikazano na slici 1.

**Slika 1 – UML dijagram slučajeva korišćenja Medical servisa**



**Tabela 1 – Opis slučaja korišćenja „Prijavljivanje“**

**Naziv**

**Prijavljivanje**

<b>Naziv</b>	<b>Prijavljivanje</b>
Učesnici	Korisnik
Preduslovi	–
Koraci	<ol style="list-style-type: none"> <li>1. Korisnik bira opciju za prijavu</li> <li>2. Unosi korisničko ime i lozinku ili koristi SSO</li> <li>3. Potvrđuje unos</li> </ol>
Rezultat	Korisnik je prijavljen na sistem
Izuzeci	Pogrešno korisničko ime ili lozinka

**Tabela 2 – Opis slučaja korišćenja „Registracija“**

<b>Naziv</b>	<b>Registracija</b>
Učesnici	Novi korisnik
Preduslovi	–
Koraci	<ol style="list-style-type: none"> <li>1. Korisnik bira opciju za registraciju</li> <li>2. Popunjava lične podatke i podatke za prijavu</li> <li>3. Potvrđuje unos</li> <li>4. Sistem kreira nalog</li> </ol>
Rezultat	Korisnik je registrovan i može se prijaviti na sistem
Izuzeci	Uneti podaci su nepotpuni ili već postoje u bazi

**Tabela 3 – Opis slučaja korišćenja „Podnošenje medicinskog zahteva“**

<b>Naziv</b>	<b>Podnošenje medicinskog zahteva</b>
Učesnici	Korisnik (učenik), School servis, Medical servis
Preduslovi	Korisnik je prijavljen
Koraci	<ol style="list-style-type: none"> <li>1. Korisnik unosi podatke o zahtevu</li> </ol>

<b>Naziv</b>	<b>Podnošenje medicinskog zahteva</b>
	2. Sistem potvrđuje prijem 3. Zahtev se čuva u bazi
Rezultat	Zahtev je evidentiran i prosleđen na obradu
Izuzeci	Nepotpuni ili nevalidni podaci

**Tabela 4 – Opis slučaja korišćenja „Pregled medicinskog zahteva“**

<b>Naziv</b>	<b>Pregled medicinskog zahteva</b>
Učesnici	Korisnik (doktor), Medical servis
Preduslovi	Korisnik je prijavljen
Koraci	1. Korisnik bira opciju za pregled zahteva 2. Sistem prikazuje sve informacije o zahtevu, uključujući status obrade
Rezultat	Korisnik može videti status i detalje medicinskog zahteva
Izuzeci	Zahtev ne postoji ili korisnik nema pravo pristupa

**Tabela 5 – Opis slučaja korišćenja „Kreiranje pregleda“**

<b>Naziv</b>	<b>Kreiranje pregleda</b>
Učesnici	Korisnik (učenik), Medical servis
Preduslovi	Korisnik je prijavljen i podneo zahtev za pregled
Koraci	1. Korisnik unosi zahtev za medicinski pregled i označava da li mu je potrebno opravdanje 2. Sistem registruje zahtev i prosleđuje ga doktoru 3. Doktor obavlja pregled i unosi rezultate u sistem
Rezultat	Pregled je evidentiran u sistemu, uključujući odluku o opravdanju ako je potrebno
Izuzeci	Nepotpuni podaci u zahtevu ili greška u sistemu

**Tabela 6 – Opis slučaja korišćenja „Izdavanje opravdanja“**



<b>Naziv</b>	<b>Izdavanje opravdanja</b>
Učesnici	Medical servis, School servis, Korisnik (učenik)
Preduslovi	Medicinski pregled je u toku; korisnik je označio da mu je potrebno opravdanje
Koraci	<ol style="list-style-type: none"> <li>1. Lekar odlučuje da li će izdati opravdanje</li> <li>2. Medical servis kreira opravdanje</li> <li>3. Notifikacija se šalje School servisu i korisniku</li> <li>4. School servis evidentira opravdanje</li> </ol>
Rezultat	Opravdanje je dostupno za opravdavanje časova učenika
Izuzeci	Problem u komunikaciji između servisa ili neodobravanje opravdanja

**Tabela 7 – Opis slučaja korišćenja „Pregled medicinskog kartona“**

<b>Naziv</b>	<b>Pregled medicinskog kartona</b>
Učesnici	Korisnik (pacijent), Medical servis
Preduslovi	Korisnik je prijavljen
Koraci	<ol style="list-style-type: none"> <li>1. Korisnik bira opciju za pregled kartona</li> <li>2. Sistem prikazuje sve medicinske podatke korisnika</li> </ol>
Rezultat	Korisnik može videti svoje zdravstvene podatke
Izuzeci	Korisnik nema pristup podacima ili je karton prazan

**Tabela 8 – Opis slučaja korišćenja „Slanje i dobavljanje opravdanja“**

<b>Naziv</b>	<b>Slanje i dohvat opravdanja</b>
Učesnici	School servis, Medical servis, Korisnik
Preduslovi	Opravdanje je kreirano
Koraci	<ol style="list-style-type: none"> <li>1. School servis šalje zahtev za dobavljanje opravdanja</li> <li>2. Medical servis vraća podatke o opravdanju</li> <li>3. School servis prikazuje opravdanje korisniku</li> </ol>
Rezultat	Opravdanje je dostupno učeniku i školi

<b>Naziv</b>	<b>Slanje i dohvrat opravdanja</b>
Izuzeci	Problem u komunikaciji između servisa

**Tabela 9 – Opis slučaja korišćenja „Slanje notifikacija pacijentu“**

<b>Naziv</b>	<b>Slanje notifikacija</b>
Učesnici	Medical servis, School servis, Korisnik
Preduslovi	Kreirano opravdanje
Koraci	1. Medical servis generiše notifikaciju 2. School servis prosleđuje notifikaciju korisniku
Rezultat	Korisnik dobija obaveštenje o događaju
Izuzeci	Problemi u komunikaciji ili dostavi notifikacije

**Tabela 10 – Opis slučaja korišćenja „Slanje notifikacija doktoru“**

<b>Naziv</b>	<b>Slanje notifikacija</b>
Učesnici	Medical servis, Korisnik
Preduslovi	Kreiran zahtev za pregled
Koraci	1. Medical servis generiše notifikaciju 2. Medical servis prosleđuje notifikaciju korisniku
Rezultat	Korisnik dobija obaveštenje o događaju
Izuzeci	Problemi u dostavi notifikacije

---

### 3.2 Specifikacija nefunkcionalnih zahteva

Nefunkcionalni zahtevi definišu osobine sistema koje su važne za njegovo pravilno funkcionisanje, ali ne predstavljaju konkretne funkcionalnosti.

**API Gateway implementiran pomoću Nginx-a:** Sistem koristi Nginx kao API gateway radi efikasnog usmeravanja zahteva između servisa i skalabilnosti arhitekture.

**Rate limiter:** Sistem ograničava broj zahteva korisnika u određenom vremenskom intervalu kako bi se zaštitio od preopterećenja i omogućila stabilnost rada servisa.

**RBAC (Role-Based Access Control):** Sistem koristi kontrolu pristupa zasnovanu na ulogama radi bezbednosti i zaštite podataka, pri čemu korisnici imaju ograničene funkcionalnosti u skladu sa svojom ulogom.

**Docker Compose:** Sistem je kontejnerizovan pomoću Docker Compose-a radi jednostavnog pokretanja i održavanja servisa u razvoju i produkciji.

**Bezbednost:** Podaci korisnika moraju biti zaštićeni, pri čemu se koristi hešovanje lozinke.

**Upotrebljivost:** Interfejs mora biti intuitivan i jednostavan za korišćenje.

## 4. Specifikacija dizajna

Ovo poglavlje objašnjava organizaciju sistema i to kako su komponente raspoređene. Sistem je orijentisan kao mikroservisna arhitektura sa sledećim komponentama:

- **Auth servis** – odgovoran za autentifikaciju i autorizaciju korisnika (SSO i standardna prijava/registracija).
- **Medical servis** – rukuje podnošenjem, pregledom i obradom medicinskih zahteva, pregledom medicinskog kartona, izdavanjem opravdanja i slanjem notifikacija korisnicima.
- **School servis** – posreduje u komunikaciji sa frontend-om, vrši rate limiting i prosleđuje zahteve Medical servisu.

Komunikacija između servisa je asinhrona, obezbeđena notifikacijama i zahtevima za dobavljanje podataka, kako bi se omogućila pouzdana razmena informacija.

### Modeli korišćeni u Medical servisu

Medical servis koristi sledeće ključne modele (klase):

- **Patient (Pacijent)** – predstavlja jednog od korisnika sistema. Sadrži jedinstveni identifikator korisnika iz Auth servisa (UserId), referencu na doktora (DoctorId) i listu medicinskih potvrda (MedicalCertificates).
- **Doctor (Doktor)** – predstavlja jednog od korisnika sistema sa ulogom doktora. Sadrži jedinstveni identifikator korisnika (UserId), tip doktora (Type) i listu pacijenata koje prati (Patients).
- **MedicalRecord (Medicinski karton)** – evidencija o zdravstvenom stanju pacijenta, uključujući alergije, hronične bolesti, poslednje ažuriranje i listu pregleda (Examinations) i zahteva za pregled (Requests).
- **Request (Zahtev)** – modeluje zahteve za pregled koje korisnici podnose. Sadrži referencu na medicinski karton (MedicalRecordId), doktora koji je zadužen (DoctorId), tip pregleda (Type), status zahteva (Status), informaciju o potrebi medicinske potvrde (NeedMedicalCertificate) i listu pregleda (Examinations).
- **Examination (Pregled)** – predstavlja obavljene medicinske preglede, uključujući dijagnozu (Diagnosis), terapiju (Therapy) i dodatne beleške (Note). Povezan je sa zahtevom (RequestId) i medicinskim kartonom (MedicalRecordId).
- **MedicalCertificate (Opravdanje)** – evidencija izdatih opravdanja, uključujući referencu na zahtev (RequestId), pacijenta (PatientId), doktora (DoctorId), datum izdavanja (Date), tip potvrde (Type) i eventualne napomene (Note).
- **Notification (Notifikacija)** – modeluje obaveštenja koja sistem šalje korisnicima, uključujući identifikator korisnika (UserId), sadržaj poruke (Message), status pročitano/nepročitano (Read) i datum kreiranja (CreatedAt).

### Relacije između modela

- Jedan **Patient** može imati više **MedicalCertificate** i tačno jednog **Doctor**.
- Jedan **Doctor** može pratiti više **Patient**.

- Jedan **MedicalRecord** je povezan sa tačno jednim pacijentom, i može imati više **Request** i **Examination**.
- Svaki **Request** pripada jednom **MedicalRecord**, jednom **Doctor**.
- Svaka **MedicalCertificate** pripada jednom **Patient** i jednom **Doctor**, i povezana je sa tačnim **Request**.
- **Notification** je vezan za jednog korisnika i može se generisati prilikom različitih događaja u sistemu (npr. kreiranje zahteva, izdavanje potvrde).

## 5. Implementacija

Ovo poglavlje prikazuje način na koji su implementirane ključne funkcionalnosti sistema za Medical servis unutar sistema eUprava. Fokus je na backend logici, upravljanju modelima, asinhronoj komunikaciji između servisa i kreiranju notifikacija.

Sistem je podeljen u sledeće delove za potrebe implementacije:

- **Autentifikacija i autorizacija (Auth servis)**
- **Upravljanje medicinskim zahtevima i kartonima (Medical servis)**
- **Prosleđivanje i rate limiting zahteva (School servis)**

### 5.1 Implementacija Medical servisa

U ovom odeljku prikazana je implementacija ključnih funkcija Medical servisa, uključujući podnošenje zahteva, kreiranje pregleda i izdavanje opravdanja.

#### 5.1.1 Kreiranje medicinskog pregleda

##### Opis funkcionalnosti:

Ova funkcija omogućava kreiranje novog medicinskog pregleda. Prilikom kreiranja, status pripadajućeg medicinskog zahteva (Request) se automatski menja u FINISHED.

---

#### Listing 1 – Kreiranje medicinskog pregleda

```
func CreateExamination(exam *models.Examination) error {
    tx := database.DB.Begin()

    if err := tx.Create(exam).Error; err != nil {
        tx.Rollback()
        return err
    }
    if err := tx.Model(&models.Request{}).
        Where("id = ?", exam.RequestId).
        Update("status", models.FINISHED).Error; err != nil {
        tx.Rollback()
        return err
    }
    return tx.Commit().Error
}
```

Ova servisna funkcija prima referencu na objekat Examination, kreira novi pregled u bazi i menja status odgovarajućeg zahteva. Ukoliko nkea operacija ne uspe, transakcija se poništava.

---

## Listing 2 – HTTP handler za kreiranje pregleda

```
func CreateExamination(w http.ResponseWriter, r *http.Request) {
    var exam models.Examination
    if err := json.NewDecoder(r.Body).Decode(&exam); err != nil {
        http.Error(w, "Invalid request body", http.StatusBadRequest)
        return
    }

    if err := services.CreateExamination(&exam); err != nil {
        http.Error(w, "Failed to create examination",
http.StatusInternalServerError)
        return
    }
    w.WriteHeader(http.StatusCreated)
    json.NewEncoder(w).Encode(exam)
}
```

Ovaj handler prima HTTP POST zahtev sa JSON podacima o pregledu, dekodira ih u strukturu Examination, poziva servisnu funkciju i vraća odgovarajući HTTP odgovor.

### 5.1.2 Kreiranje opravdanja i notifikacije

#### Opis funkcionalnosti:

Ova funkcija omogućava kreiranje novog opravdanja na osnovu prethodnog medicinskog zahteva. Prilikom kreiranja, sistem automatski povezuje potvrdu sa pacijentom i šalje notifikaciju učeniku putem School servisa.

---

## Listing 3 – Kreiranje opravdanja i notifikacije učeniku

```
func CreateMedicalCertificateHandler(w http.ResponseWriter, r
*http.Request) {
    var cert models.MedicalCertificate
    if err := json.NewDecoder(r.Body).Decode(&cert); err != nil {
        http.Error(w, err.Error(), http.StatusBadRequest)
        return
    }

    if err := services.CreateMedicalCertificate(&cert); err != nil {
        http.Error(w, err.Error(), http.StatusInternalServerError)
    }
}
```

```

        return
    }
    w.WriteHeader(http.StatusCreated)
    json.NewEncoder(w).Encode(cert)
}

```

HTTP handler prima POST zahtev sa podacima o potvrdi, dekodira JSON u strukturu MedicalCertificate, poziva servisnu funkciju i vraća odgovor klijentu.

```

func CreateMedicalCertificate(cert *models.MedicalCertificate) error {
    var req models.Request
    if err := database.DB.First(&req, cert.RequestId).Error; err != nil {
        return err
    }

    var record models.MedicalRecord
    if err := database.DB.First(&record, req.MedicalRecordId).Error;
err != nil {
        return err
    }
    cert.PatientId = record.PatientId
    if err := database.DB.Create(cert).Error; err != nil {
        return err
    }
    var patient models.Patient
    if err := database.DB.First(&patient, cert.PatientId).Error; err !=
nil {
        return err
    }
    message := fmt.Sprintf("New Medical Certificate created of type %s
dated %s", cert.Type, cert.Date)
    if err := NotifyStudent(patient.UserId, message); err != nil {
        log.Printf("[NotifyStudent] Failed to notify student %d: %v",
patient.UserId, err)
    } else {
        log.Printf("[NotifyStudent] Notification sent to student %d",
patient.UserId)
    }
    return nil
}

func NotifyStudent(userId uint, message string) error {
    notif := map[string]interface{}{
        "studentId": userId,

```



```

        "message": message,
    }

    body, _ := json.Marshal(notif)
    resp, err := http.Post("http://school-service:8083/notifications",
"application/json", bytes.NewBuffer(body))
    if err != nil {
        return err
    }
    defer resp.Body.Close()
    if resp.StatusCode != http.StatusOK {
        return fmt.Errorf("school-service returned status %d",
resp.StatusCode)
    }
    return nil
}

```

Servisna funkcija CreateMedicalCertificate kreira potvrdu u bazi, povezuje je sa pacijentom i koristi NotifyStudent funkciju za slanje obaveštenja učeniku putem School servisa.

---

### 5.1.3 Odobravanje i odbijanje zahteva

#### Opis funkcionalnosti:

Ova funkcija omogućava doktoru da donese odluku o medicinskom zahtevu pacijenta. Zahtev prelazi iz statusa REQUESTED u APPROVED ili REJECTED, čime se modeluje proces odlučivanja u medicinskom kontekstu.

---

#### Listing 4 – Odobravanje zahteva

```

func ApproveRequest(w http.ResponseWriter, r *http.Request) {
    idStr := mux.Vars(r)["id"]
    id, _ := strconv.Atoi(idStr)

    req, err := services.UpdateRequestStatus(uint(id), models.APPROVED)
    if err != nil {
        http.Error(w, "Request not found", http.StatusNotFound)
        return
    }
    json.NewEncoder(w).Encode(req)
}

```

Handler pronalazi zahtev po ID-ju, menja status u APPROVED i čuva promene u bazi. Slična funkcija može se koristiti za odbijanje zahteva (status REJECTED).

---

#### 5.1.4 Pretraga, filtriranje, sortiranje i paginacija zahteva

##### Opis funkcionalnosti:

Kako bi doktori mogli da efikasno rade sa velikim brojem pacijenata, Medical servis implementira napredne funkcionalnosti:

- filtriranje po statusu zahteva,
- pretraga po pacijentu,
- sortiranje tako da se REQUESTED zahtevi prikazuju prvi,
- paginacija radi preglednijeg prikaza.

---

##### Listing 5 – Pretraga, filtriranje i sortiranje zahteva

```
func GetApprovedRequestsByDoctor(w http.ResponseWriter, r *http.Request) {
    idStr := mux.Vars(r)["id"]
    doctorId, _ := strconv.Atoi(idStr)

    pageStr := r.URL.Query().Get("page")
    page := 1
    if p, err := strconv.Atoi(pageStr); err == nil && p > 0 {
        page = p
    }
    search := r.URL.Query().Get("search")
    reqType := r.URL.Query().Get("type")
    const pageSize = 5
    requests, totalPages, err :=
services.GetRequestsByDoctorWithStudentPaginatedCustomFilters(
    uint(doctorId),
    page,
    pageSize,
    string(models.APPROVED),
    search,
    reqType,
    false,
)
    if err != nil {
        http.Error(w, "Failed to fetch approved requests",
http.StatusInternalServerError)
        return
    }
}
```

```
}  
w.Header().Set("Content-Type", "application/json")  
json.NewEncoder(w).Encode(map[string]interface{}{  
    "requests": requests,  
    "totalPages": totalPages,  
})  
}
```

Ova funkcionalnost omogućava lekarima da brzo pronađu tražene zahteve i efikasno upravljaju podacima.

## 6. Demonstracija

Ovo poglavlje prikazuje način korišćenja aplikacije eUprava kroz nekoliko tipičnih scenarija. Svaki scenario prikazuje rešavanje jednog od problema koji se javljaju u okviru medicinskih zahteva i školskog sistema.

### 6.1. Prijavljivanje u sistem

Nakon pokretanja aplikacije, korisniku se prikazuje početna stranica za prijavu (slika 1). Na ovoj stranici potrebno je uneti korisničko ime i lozinku za standardnu prijavu ili se bira opcija za prijavu preko Google naloga. Uspešna prijava otvara početni meni sa dostupnim opcijama u zavisnosti od tipa korisnika (učenik ili doktor).

The image shows a white login card with a soft shadow. At the top, the word 'Login' is centered in a bold, teal font. Below it, there are two input fields. The first is labeled 'Email' and contains the text 'example@mail.com'. The second is labeled 'Password' and contains the text 'Password'. Below these fields is a teal button with the text 'Login' in white. Underneath the button is the word 'or' in a small, grey font. At the bottom of the card is a yellow button with the text 'Continue with Google' in black.

(Slika 1 – Stranica za prijavu u sistem)

---

### 6.2. Podnošenje zahteva za pregled

Učenik, nakon prijave, može pristupiti formi za podnošenje novog zahteva za pregled (slika 2).

Potrebno je popuniti osnovne informacije i potvrditi slanjem zahteva. Nakon uspešnog slanja, zahtev prelazi u status „REQUESTED“.

### Create Request

Doctor

-- Select Doctor --

Type

-- Select Type --

☐ Request Medical Certificate

Submit Request

*(Slika 2 – Forma za podnošenje zahteva za pregled)*

---

### 6.3. Pregled i obrada zahteva od strane lekara

Doktor ima prikaz liste svih primljenih zahteva (slika 3). Zahtevi su sortirani tako da se najpre prikazuju oni koji su u statusu „REQUESTED“. Doktor može da donese odluku – odobriti ili odbiti pregled.

eUprava	Patient Requests	Approved Requests	🔔 🔍 Logout
---------	------------------	-------------------	------------

Patient Requests			
🔍 Search by patient name...	Filter by types ▾	Filter by status ▾	<input checked="" type="checkbox"/> Sort by Pending
Patient	Type	Status	Actions
Janja Ilic	SPECIALIST	Pending	Approve Reject
Janja Ilic	REGULAR	Pending	Approve Reject
Janja Ilic	URGENT	Pending	Approve Reject
Janja Ilic	URGENT	Pending	Approve Reject
Janja Ilic	URGENT	Pending	Approve Reject
Miomira Jovanovic	SPECIALIST	Rejected	
Janja Ilic	REGULAR	Approved	



(Slika 3 – Lista zahteva kod doktora)

## 6.4. Kreiranje medicinskog pregleda

Kada doktor obavi pregled, unosi podatke u sistem (slika 4). Nakon potvrde unosa, status zahteva se automatski menja u „FINISHED“.

eUprava

Patient RequestsApproved Requests

Logout

**Patient Medical Record**

Name: Miomira JovanovicJMBG: 0506001778655

Birth Date: 05.06.2001.

Gender: F

Allergies: None

Chronic Diseases: None

**Examination Form**

Diagnosis

Enter diagnosis here...

Therapy

Enter therapy here...

Notes

Enter any additional notes...

Save Examination

Medical Certificate

(Slika 4 – Forma za unos podataka o pregledu)

---


## 6.5. Izdavanje potvrde

Na osnovu završenog pregleda doktor može izdati medicinsku potvrdu (slika 5). Potvrda se povezuje sa nalogom učenika i automatski se šalje obaveštenje u školskom servisu. Učenik u svom interfejsu dobija poruku o novoj potvrdi.

### Medical Certificate Form


Date

10/02/2025



Type

REGULAR



Note

Enter additional notes if needed...

Save Certificate

(Slika 5 – Forma za izdavanje potvrde)

---

## 6.6. Pretraga i filtriranje zahteva

Kako bi se olakšao rad doktora, u okviru liste zahteva dostupna je pretraga po imenu učenika, filtriranje po tipu i statusu i prikaz u stranicama (slika 6). Na taj način doktor može brzo pronaći relevantne podatke i efikasno upravljati većim brojem zahteva.



### Patient Requests

REGULAR

Pending

☐ Sort by Pending

Patient	Type	Status	Actions
Janja Ilic	REGULAR	Pending	<div>ApproveReject</div>

Prev

1

Next

(Slika 6 – Primer pretrage i filtriranja zahteva)

## 7. Zaključak

Ovo poglavlje sumira realizaciju softverskog rešenja za Medical servis u okviru sistema eUprava, analizira postignute ciljeve i razmatra mogućnosti daljeg unapređenja.

U radu je prikazan razvoj i funkcionalnost Medical servisa, uključujući kreiranje medicinskih pregleda, izdavanje opravdanja, odobravanje i odbijanje zahteva, kao i pretragu i filtriranje podataka. Prikazano rešenje omogućava doktorima i učenicima da jednostavno upravljaju medicinskim zahtevima i evidencijom, čime se olakšavaju administrativni procesi i smanjuju greške u vođenju podataka.

### Dobre strane:

- Sistem omogućava jednostavno praćenje statusa medicinskih zahteva i pregleda.
- Automatsko slanje notifikacija učenicima smanjuje potrebu za dodatnom komunikacijom.
- Implementirane funkcionalnosti za pretragu, filtriranje i paginaciju olakšavaju doktorima rad sa velikim brojem zahteva.

### Loše strane i mogućnosti unapređenja:

- Sistem još ne podržava napredne analitike i izveštavanje, što bi moglo biti korisno za upravljanje medicinskim procesima.
- Uvođenje dodatnih mehanizama autentifikacije i bezbednosnih funkcija moglo bi dodatno poboljšati sigurnost podataka.

### Poređenje sa srodnim rešenjima:

U poređenju sa komercijalnim ili većim zdravstvenim sistemima, prikazano rešenje pruža jednostavniji i pregledniji interfejs, a osnovne funkcionalnosti su prilagođene za svakodnevnu upotrebu malog broja korisnika. Veća komercijalna rešenja nude širi spektar funkcionalnosti, ali uz složeniji interfejs i potrebe za obukom korisnika.

### Dalji pravci za unapređenje:

- Proširenje sistema tako da kreiranje pregleda i medicinskih potvrda bude omogućeno svim registrovanim pacijentima, a ne samo učenicima.
- Integracija sa kalendarima i zakazivanje termina – omogućavanje pacijentima da direktno rezervišu termine pregleda i dobijaju automatske podsetnike.
- Poboljšanje sistema notifikacija, uključujući email ili SMS obaveštenja.
- Prilagođavanje interfejsa za osobe sa posebnim potrebama – bolja pristupačnost sistema za sve korisnike.

## Literatura

- [1] eUprava. 2025. Zdravlje. Preuzeto 02.10.2025. sa <https://euprava.gov.rs/životna-oblast/17>
- [2] MyChart. Preuzeto 02.10.2025. sa <https://www.mychart.org>
- [3] Go Programming Language. Preuzeto 02.10.2025. sa <https://golang.org/>
- [4] Angular. Preuzeto 02.10.2025. sa <https://angular.io/>
- [5] Tailwind CSS. Preuzeto 02.10.2025. sa <https://tailwindcss.com/>
- [6] PostgreSQL Global Development Group. Preuzeto 02.10.2025. sa <https://www.postgresql.org/>
- [7] Firebase. Preuzeto 02.10.2025. sa <https://firebase.google.com/products/auth>