

# **Web aplikacija za simulaciju eUprave u zdravstvu**

by Andjela Lozanov



# Sažetak

U ovom radu je opisana simulacija elektronske uprave za zdravstvene usluge, koja transformiše način na koji građani Srbije pristupaju i Srbije pristupaju i koriste zdravstvene usluge, omogućavajući korisnicima efikasno zakazivanje medicinskih pregleda i praćenje terapija, kao i praćenje terapija, kao i sistematskih.

Za realizaciju je korišćena kombinacija modernih tehnologija kao što su Go i MongoDB, omogućavajući brzu i pouzdanu obradu podataka. obradu podataka.

Primenom predloženog rešenja, korisnici mogu jednostavno upravljati svojim zdravstvenim informacijama putem intuitivnog interfejsa, a i intuitivnog interfejsa, a i poboljšan je kvalitet zdravstvene zaštite.

Razvijen je prototip sistema i objašnjeno njegovo korišćenje, što doprinosi boljoj dostupnosti zdravstvenih usluga i efikasnijoj upotrebi. efikasnijoj upotrebi.

# Ključne reči

- elektronska uprava
- zdravstvene usluge
- zakazivanje pregleda
- Go programiranje
- MongoDB

# Uvod

Elektronska uprava za zdravstvene usluge postaje sve važnija zbog potrebe za efikasnijim upravljanjem zdravstvenim resursima. Tradicionalni pristupi zakazivanju pregleda, pristupu medicinskim podacima i upravljanju terapijama često su neefikasni i zahtevaju fizičko prisustvo pacijenata, što može biti problematično.. Ova aplikacija nudi centralizovano rešenje koje povezuje pacijente i lekare, omogućavajući brz pristup potrebnim informacijama i uslugama. Bez ovakvih rešenja, pacijenti se suočavaju s dugim čekanjima, komplikovanom papirologijom i ograničenom dostupnošću zdravstvenih resursa.

Istorijski gledano, procesi upravljanja zdravstvenim uslugama bili su manuelni i decentralizovani. Međutim, s razvojem tehnologije, potreba za efikasnijim i automatizovanim rešenjima postala je očigledna. U budućnosti, razvoj ovakvih sistema može znatno poboljšati kvalitet zdravstvene zaštite i omogućiti personalizovanu negu.

Ostatak rada organizovan je na sledeći način: u drugom poglavlju prikazane su korišćene tehnologije, treće opisuje implementaciju, četvrto evaluaciju sistema, dok peto donosi zaključke i buduće pravce razvoja.

# Srodna rešenja i pregled korišćenih tehnologija

U ovom poglavlju je dat pregled jednog od postojećih rešenja za simulaciju elektronske uprave u zdravstvu, kao i pregled tehnologija koje omogućavaju razvoj ovakvih sistema.

## Srodna rešenja

Postojeće aplikacije, kao što je **EUprava**, pružaju korisnicima mogućnost upravljanja zdravstvenim podacima, zakazivanja pregleda i integraciju s elektronskim zdravstvenim kartonima, čime se unapređuje efikasnost zdravstvene zaštite. **EUprava** omogućava jednostavno i sigurno upravljanje ličnim zdravstvenim podacima i zakazivanje termina kod lekara, što značajno smanjuje vreme čekanja i poboljšava kvalitet usluga.

## Pregled korišćenih tehnologija

Portal **eUprava [1]** koristi tehnologije kao što su **Java** i **C#** za backend, omogućavajući stabilno i efikasno upravljanje podacima. Na frontendu se koristi **JavaScript**, uz korišćenje **React** biblioteke, što omogućava savremeno i responzivno korisničko iskustvo. **Oracle** baza podataka se koristi za sigurno i efikasno skladištenje velikih količina osetljivih podataka.

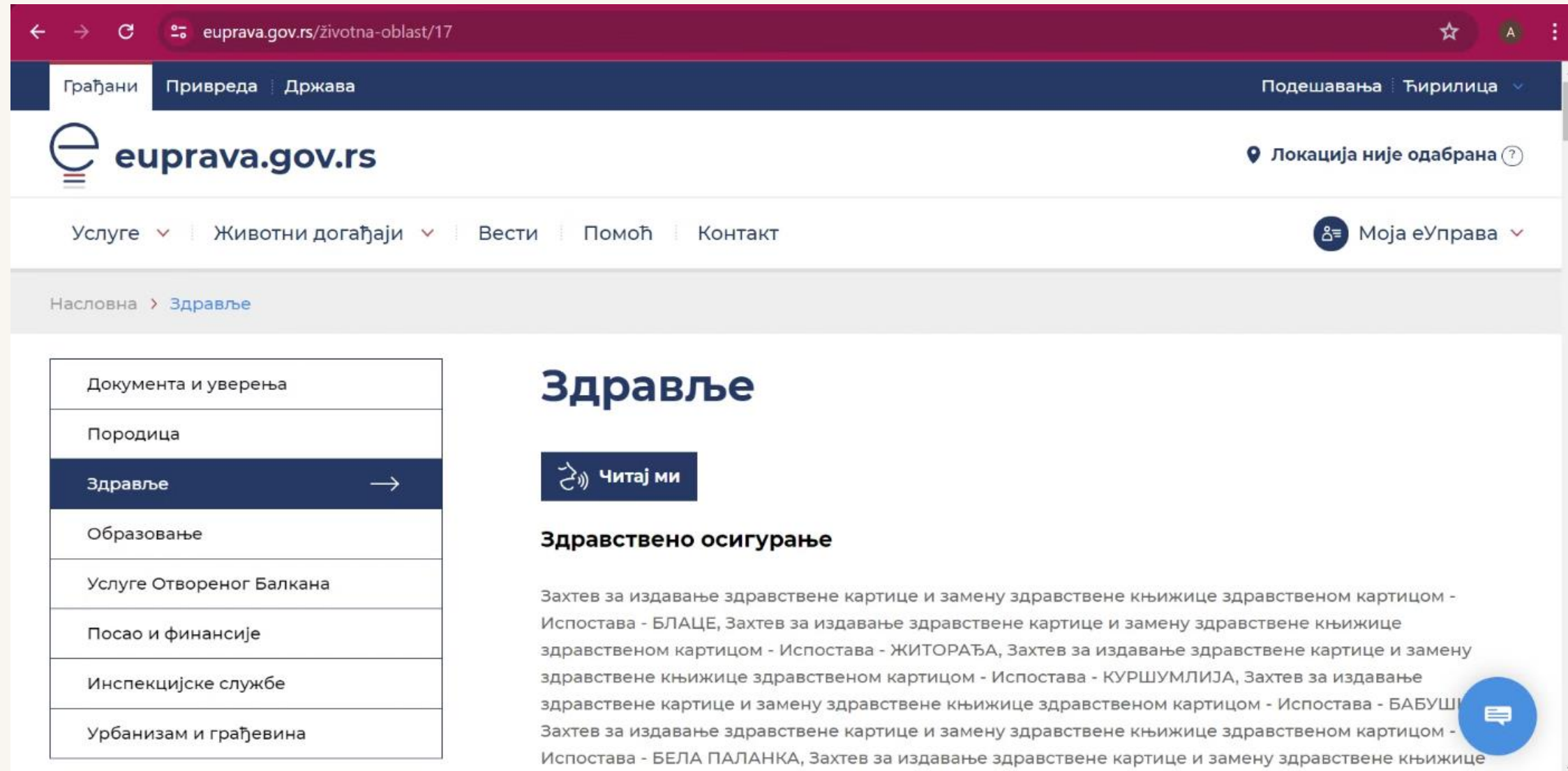
# Srodna rešenja

Ovaj odeljak donosi pregled aplikacija namenjenih olakšanju pristupa zdravstvenim uslugama putem elektronske administracije.

**eZdravlje [2]** je platforma koja omogućava građanima pristup raznim uslugama sličnim kao eUprave u oblasti zdravstva, putem interneta. U okviru zdravstvenog sektora, omogućava elektronsko zakazivanje lekarskih pregleda, uvid u zdravstvene kartone i izdavanje elektronskih recepata. Više o eZdravlju možete pročitati na [\[izvor\]](#) a kako izgleda možete videti na Slici broj 1. **eZdravlje** nudi značajne prednosti kao što su smanjenje administrativnih procedura i brži pristup zdravstvenim uslugama, što olakšava građanima održavanje sopstvenog zdravlja. Međutim, nedostatak može biti ograničen pristup za korisnike bez pristupa internetu ili sa slabom tehničkom podrškom.

*"Збогом чекању у редовима, гужвама на шалтерима или погрешним бројевима телефона. Електронска управа вам омогућује да све административне послове завршите далеко комфорније из свог дома, канцеларије, или чак у покрету, путем мобилних уређаја, тако да вам остаје више времена за ствари које су заиста битне."* [3]

# Srodna rešenja - EUprava



Slika 1 - Izgled aplikacije eZdravlje [2]

# Pregled korišćenih tehnologija

**Java i C#** su dva popularna programska jezika korišćena za backend razvoj. **Java [4]** je poznata po svojoj prenosivosti i stabilnosti, što je čini pogodnom za izgradnju velikih sistema. **C# [5]**, kao deo .NET okvira, omogućava brzu izradu aplikacija sa odličnom integracijom u Microsoft ekosistem. Ove tehnologije omogućavaju stabilno i efikasno upravljanje podacima.

**JavaScript [6]** sa **React** bibliotekom koristi se na frontend strani. **React [7]** omogućava razvoj modernih, responzivnih korisničkih interfejsa sa brzim renderovanjem. Ova biblioteka omogućava razvijanje interaktivnih aplikacija koje poboljšavaju korisničko iskustvo.

**Oracle baza podataka [8]** koristi se za sigurno skladištenje velikih količina podataka. Oracle baze su poznate po svojoj pouzdanosti i sposobnosti da upravljaju velikim količinama osetljivih podataka efikasno.



# Specifikacija zahteva

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

Neke od mogućnosti koje se očekuju od ovog rešenja problema iz perspektive u kojoj taj problem još nije rešen.

Ovo softversko rešenje bi trebalo da omogući zakazivanje, kao i otkazivanje, zdravstvenih i sistematskih pregleda, od strane doktora i studenata, kao i komunikaciju izmedju servera, kao sto su komunikacija sa Službom ishrane i Fakultetom. Potrebno je da sistem bude efikasan kako bi se izbeglo nezadovoljstvo korisnika i smanjila potreba za fizičkim odlaskom na određene lokacije.

# Specifikacija funkcionalnih zahteva

U ovom odeljku su opisani funkcionalni zahtevi koje je potrebno da ispunjava softversko rešenje za upravljanje zdravstvenim uslugama. Funkcionalni zahtevi ovog softverskog rešenja su predstavljeni UML dijagramom slučajeva korišćenja, kao što je prikazano na slici 2 (svaki slučaj korišćenja je opisan).

## 1. Održavanje liste termina pregleda:

- Sistem mora omogućiti doktorima da dodaju, ažuriraju i brišu termine pregleda.

## 2. Zakazivanje i otkazivanje pregleda:

- Korisnici (pacijenti) mogu zakazati i otkazati preglede putem aplikacije.

## 3. Deljenje terapija sa službom ishrane:

- Sistem treba omogućiti doktorima da unesu terapije koje se dele sa službom ishrane radi prilagođavanja obroka pacijentima.

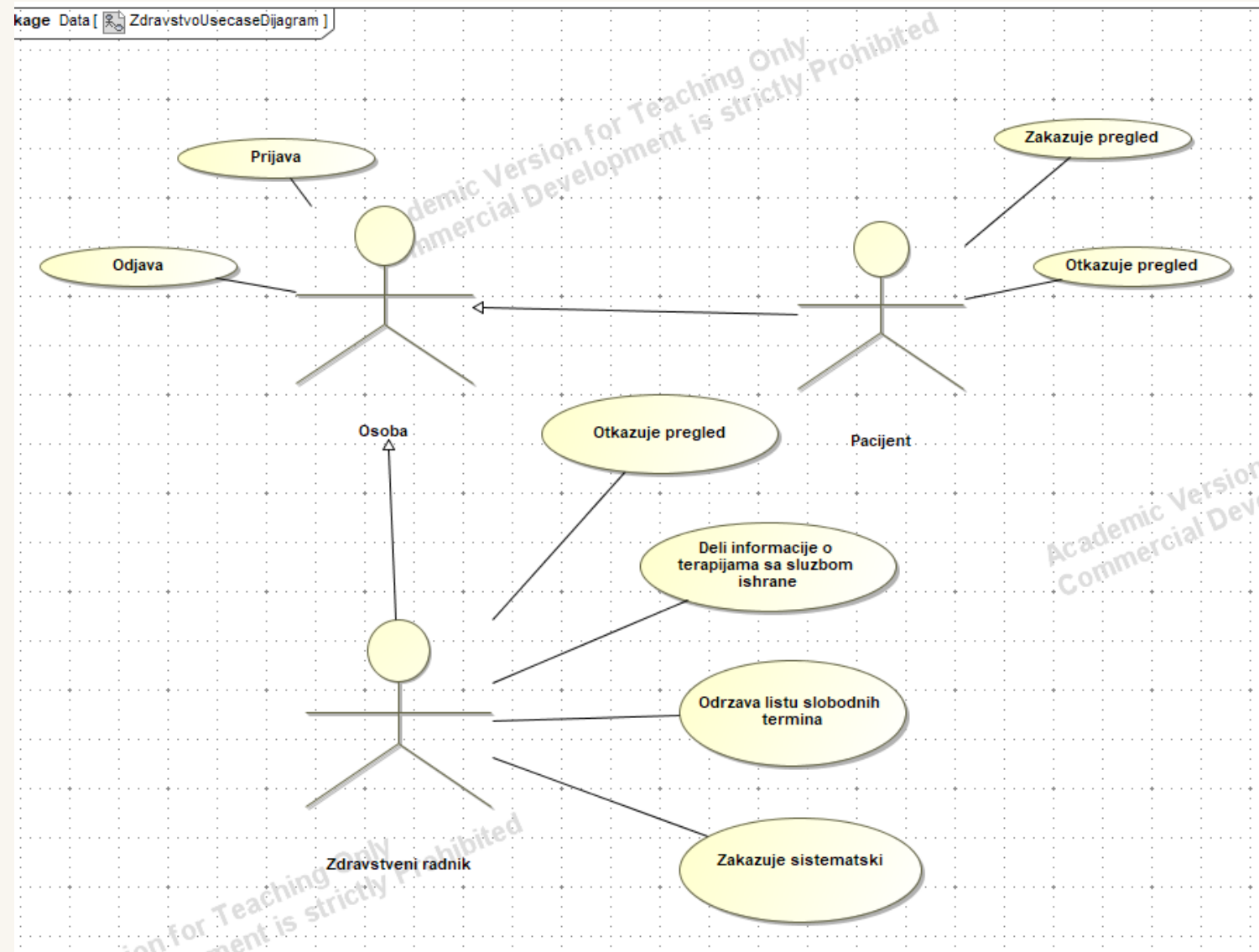
## 4. Zakazivanje sistematskih pregleda:

- Sistem mora automatski zakazivati sistematske preglede za korisnike na osnovu unapred definisanih kriterijuma.

## 5. Obaveštavanje fakulteta o sistematskim pregledima:

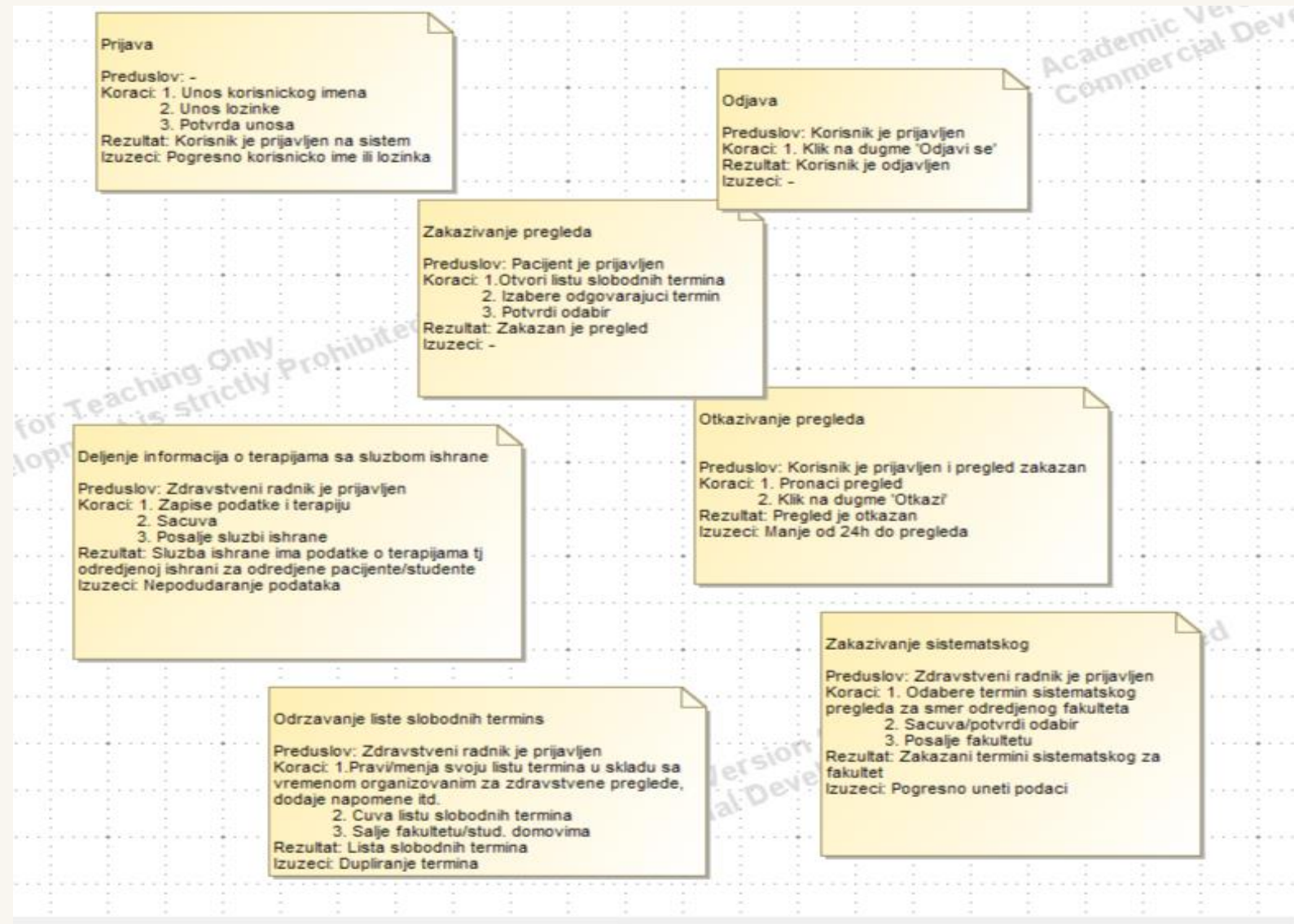
- Sistem treba slati automatska obaveštenja fakultetu o zakazanim sistematskim pregledima studenata.

# Dijagram slučajeve korišćenja



Slika 2 - Prikazuje sve interakcije korisnika sa sistemom i relevantne slučajeve korišćenja.

# Dijagram slučajeve korišćenja



Slika 2 – Tabele se koracima - Prikazuje sve interakcije korisnika sa sistemom i relevantne slučajeve korišćenja.

# Specifikacija nefunkcionalnih zahteva

Pomoću specifikacije nefunkcionalnih zahteva definišu se svojstva softverskog rešenja koja su ključna za rešavanje datog problema, ali ne direktno vezana za njegove funkcionalnosti.

## 1. Performanse - *brzina odziva Sistema*

Sistem mora odgovarati na korisničke zahteve u roku od dve sekunde.

## 2. Dostupnost - *dostupnost sistema*

Sistem mora biti dostupan 99% vremena, sa planiranim održavanjem van radnog vremena.

## 3. Bezbednost - *zaštita podataka*

Podaci korisnika moraju biti zaštićeni enkripcijom tokom prenosa i skladištenja.

## 4. Upotrebljivost - *lakoća korišćenja*

Interfejs mora biti intuitivan, omogućavajući korisnicima lako navigaciju i korišćenje funkcionalnosti sistema.

## 5. Podrška - *korisnička podrška*

Sistem treba da pruža podršku korisnicima kroz često postavljana pitanja (FAQ) i korisnički servis dostupan putem emaila.

# Specifikacija dizajna

Ovo poglavlje objašnjava dizajn softverskog rešenja za efikasno upravljanje zdravstvenim pregledima.

## Arhitektura sistema

Sistem je organizovan u sledeće komponente:

### 1. Korisnički interfejs

- Interfejs za doktore i studente omogućava zakazivanje, otkazivanje pregleda i pregledanje dostupnih terapija.

### 2. Backend server

- Server koji obrađuje zahteve korisnika, omogućava interakciju sa bazom podataka i upravlja komunikacijom sa eksternim servisima kao što su Služba ishrane i Fakultet.

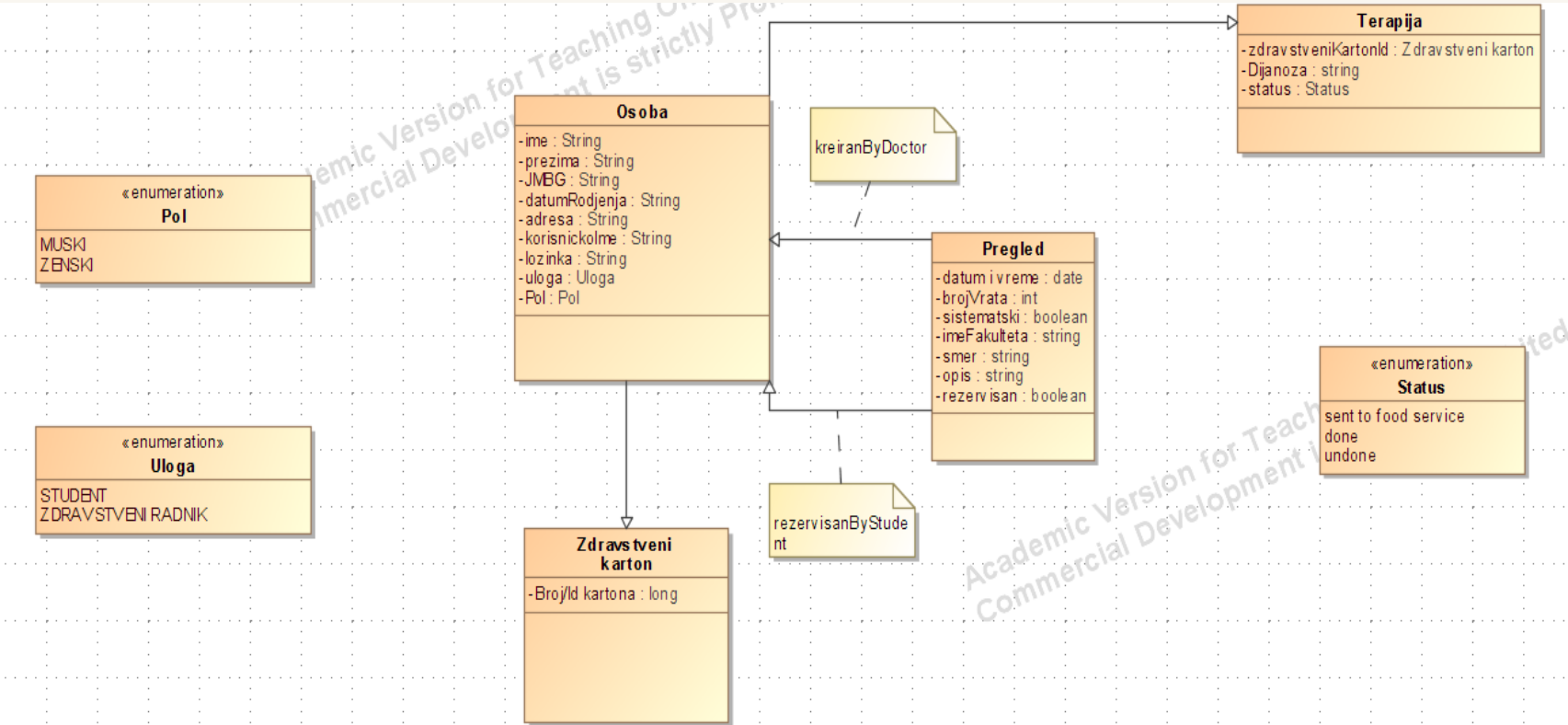
### 3. Baza podataka

- MongoDB baza podataka za skladištenje zdravstvenih kartona, podataka o pregledima i terapijama.

Sistem se sastoji od jasno definisanih komponenti koje omogućavaju efikasno upravljanje zdravstvenim pregledima. Komponente su dizajnirane da međusobno komuniciraju preko definisanih interfejsa, osiguravajući integritet podataka i brzu obradu zahteva korisnika.

# Specifikacija dizajna

Sistem se sastoji od jasno definisanih komponenti koje omogućavaju efikasno upravljanje zdravstvenim pregledima. Komponente su dizajnirane da međusobno komuniciraju preko definisanih interfejsa, osiguravajući integritet podataka i brzu obradu zahteva korisnika.



Slika 3 - UML dijagram klasa sistema za upravljanje zdravstvenim pregledima

Na slici 3 je prikazan UML dijagram klasa koji pruža jasnu strukturu objektnog modela podataka sistema za upravljanje zdravstvenim pregledima. Svaka klasa ima definisane atribute i metode koji omogućavaju efikasno upravljanje podacima i operacijama u sistemu.

# Implementacija

Ovo poglavlje prikazuje način na koji su implementirane neke od funkcija sistema za elektronsko upravljanje zdravstvom, u skladu sa Specifikacijom zahteva i dizajnom sistema.

- **Implementacija Backend-a (Go, Java)** U ovom odeljku će biti opisana implementacija serverske strane aplikacije. Prikazani su ključni aspekti kao što su obrada zahteva, komunikacija sa bazom podataka, u suštini struktura implementacije.
- **Implementacija Frontend-a (Angular, TypeScript)** Ovde će biti detaljno objašnjena implementacija klijentske strane aplikacije. Fokus je na korisničkom interfejsu, navigaciji, i integraciji sa backend-om putem REST API-ja.

Detaljno se opisuje način na koji su implementirani različiti zahtevi definisani specifikacijom sistema za zdravstvenu negu. U fokusu su funkcionalnosti koje su ključne za efikasno upravljanje podacima o studentima, zakazivanjem i upravljanjem pregleda, kao i manipulacijom zdravstvenim kartonima.



# Implementacija Backenda

**1. Inicijalizacija i konfiguracija:** Pri pokretanju aplikacije, sistem inicijalizuje sve potrebne resurse kao što su baza podataka i loggeri za praćenje stanja aplikacije kao što je prikazano na listingu broj 1.

```
port := os.Getenv( key: "HEALTHCARE_SERVICE_PORT")

timeoutContext, cancel := context.WithTimeout(context.Background(), 50*time.Second)
defer cancel()

logger := log.New(os.Stdout, prefix: "[res-api] ", log.LstdFlags)
storeLogger := log.New(os.Stdout, prefix: "[res-store] ", log.LstdFlags)

store, err := data.NewHealthCareRepo(timeoutContext, storeLogger)
if err != nil {
    logger.Fatal(err)
}
defer store.DisconnectMongo(timeoutContext)
store.Ping()
```

Listing 1 - Inicijalizacija i konfiguracija loggera i baze

# Implementacija Backenda

**2. Rutiranje zahteva:** Zahtevi korisnika obrađuju se kroz definisane rute koristeći Gorilla Mux biblioteku za rutiranje. Svaka ruta je povezana sa odgovarajućim handlerom koji izvršava poslovnu logiku za svaki specifičan zahtev ao što je prikazano na listingu broj 2.

```
// Inicijalizacija rutera i dodavanje middleware-a za sve zahteve
router := mux.NewRouter()
router.Use(MiddlewareContentTypeSet)

getStudents := router.Methods(http.MethodGet).Subrouter()
getStudents.HandleFunc(path: "/students", healthCareHandler.GetAllStudents)

insertStudent := router.Methods(http.MethodPost).Subrouter()
insertStudent.HandleFunc(path: "/students", healthCareHandler.InsertStudent)
insertStudent.Use(healthCareHandler.MiddlewareStudentDeserialization)

router.HandleFunc(path: "/appointments", healthCareHandler.GetAllAppointments).Methods(http.MethodGet)
router.HandleFunc(path: "/therapies", healthCareHandler.GetAllTherapies).Methods(http.MethodGet)
router.HandleFunc(path: "/doneTherapies", healthCareHandler.GetDoneTherapiesFromFoodService).Methods(http.MethodGet)

scheduleAppointment := router.Methods(http.MethodPost).Subrouter()
scheduleAppointment.HandleFunc(path: "/appointments/schedule", healthCareHandler.ScheduleAppointment)
//scheduleAppointment.Use(healthCareHandler.MiddlewareAppointmentDeserialization)
```

Listing 2 - Rutiranje zahteva

# Implementacija Backenda

**3. Handlerski sloj:** Svaki handler je odgovoran za obradu specifičnih zahteva, kao što su upravljanje studentima, zakazivanje pregleda, manipulacija terapijama i zdravstvenim kartonima. Svaki handler pristupa podacima putem odgovarajuće baze podataka ili eksternih servisa kao što je prikazano na listingu broj 3.

```
func (h *HealthCareHandler) CreateAppointment(rw http.ResponseWriter, r *http.Request) { 2 usages
    appointmentData := r.Context().Value(KeyProduct{}).(*data.AppointmentData)

    err := h.healthCareRepo.CreateAppointment(appointmentData)
    if err != nil {
        h.logger.Print(v...: "Database exception: ", err)
        http.Error(rw, error: "Error creating appointment.", http.StatusInternalServerError)
        return
    }

    rw.WriteHeader(http.StatusCreated)
}
```

Listing 3 - Primer handlera za kreiranje pregleda

# Implementacija Backenda

**4. Middleware:** Middleware funkcionalnost je implementirana kako bi se osigurala sigurnost i konzistentnost aplikacije. Na primer, MiddlewareContentTypeSet postavlja odgovarajući Content-Type i sigurnosne zagrade za sve zahteve kao što je prikazano na listingu broj 4.

```
func MiddlewareContentTypeSet(next http.Handler) http.Handler { 1 usage  👤 Andjela Lozanov
    return http.HandlerFunc(func(rw http.ResponseWriter, h *http.Request) {
        //s.logger.Println("Method [" + h.Method + "] - Hit path :"+ h.URL.Path)

        rw.Header().Add(key: "Content-Type", value: "application/json")
        rw.Header().Set(key: "X-Content-Type-Options", value: "nosniff")
        rw.Header().Set(key: "X-Frame-Options", value: "DENY")
        rw.Header().Set(key: "Content-Security-Policy", value: "script-src 'self' https://code.jquery.com")

        next.ServeHTTP(rw, h)
    })
}
```

Listing 4 - Primer handlera za kreiranje pregleda



# Implementacija Backenda

5. **Repozitorijum** - U "repo" paketu se obavlja glavna komunikacija sa bazom podataka. Ovaj sloj upravlja svim operacijama koje se tiču čuvanja, ažuriranja i preuzimanja podataka kao što je prikazano na listingu broj 5.

```
func (rr *HealthCareRepo) CreateAppointment(appointmentData *AppointmentData) error { 2 usages Andje
    ctx, cancel := context.WithTimeout(context.Background(), 30*time.Second)
    defer cancel()

    examinationsCollection := rr.getCollection( collectionName: "examinations")

    appointmentData.Reserved = false

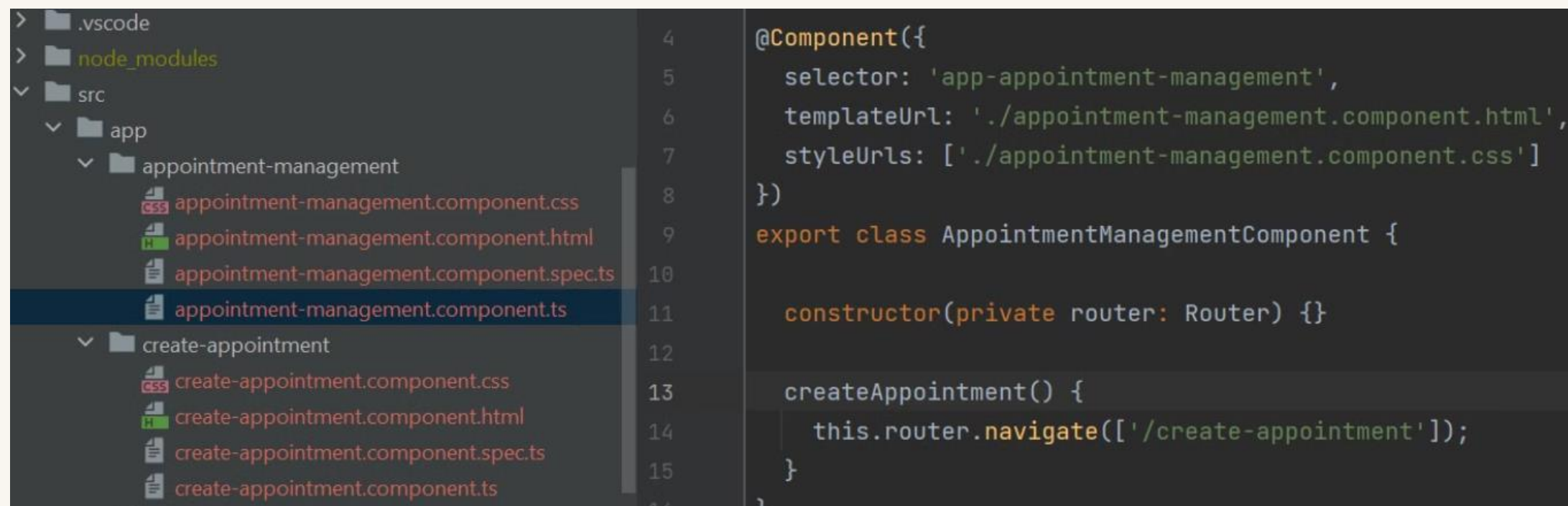
    _, err := examinationsCollection.InsertOne(ctx, appointmentData)
    if err != nil : err

    return nil
}
```

Listing 5 - Primer funkcije za kreiranje tj. čuvanje pregleda

# Implementacija Frontenda

**1. UI Komponente:** Aplikacija je podeljena na modularne UI komponente koje omogućavaju lako održavanje i ponovnu upotrebu. Svaka komponenta predstavlja specifičan deo korisničkog interfejsa, kao što su forme, liste ili kartice za pregled informacija. Evo primer jedne od komponenti na listingu broj 6.



The screenshot shows a VS Code editor with a file explorer on the left and a code editor on the right. The file explorer shows a project structure with a folder named 'src' containing an 'app' folder. Inside 'app', there is a folder 'appointment-management' which contains four files: 'appointment-management.component.css', 'appointment-management.component.html', 'appointment-management.component.spec.ts', and 'appointment-management.component.ts'. The 'appointment-management.component.ts' file is selected and its code is displayed in the editor. The code defines an Angular component named 'AppointmentManagementComponent' with a selector 'app-appointment-management', a templateUrl 'appointment-management.component.html', and a styleUrls array containing 'appointment-management.component.css'. The component has a constructor that takes a 'Router' as a parameter and a 'createAppointment()' method that navigates to the '/create-appointment' route.

```
4 @Component({
5   selector: 'app-appointment-management',
6   templateUrl: './appointment-management.component.html',
7   styleUrls: ['./appointment-management.component.css']
8 })
9 export class AppointmentManagementComponent {
10
11   constructor(private router: Router) {}
12
13   createAppointment() {
14     this.router.navigate(['/create-appointment']);
15   }
16 }
```

Listing 6 - Prikaz komponente AppointmentManagement u Angularu

# Implementacija Frontenda

**2. Navigacija:** Navigacija je jednostavna i intuitivna, omogućavajući korisnicima brzo prebacivanje između različitih delova aplikacije. Korišćenjem Angular Router-a, omogućeno je glatko kretanje kroz rute i dinamičko učitavanje sadržaja, kao što je prikazano na listingu broj 7 (linije 2, 11 i 14).

```
2      import { Router } from '@angular/router';
3
4      @Component({
5          selector: 'app-appointment-management',
6          templateUrl: './appointment-management.component.html',
7          styleUrls: ['./appointment-management.component.css']
8      })
9      export class AppointmentManagementComponent {
10
11          constructor(private router: Router) {}
12
13          createAppointment() {
14              this.router.navigate(['/create-appointment']);
15          }
```

Listing 7 - Primer rutiranja sa glavne na stranicu za kreiranje pregleda

# Implementacija Frontenda

**3. Upravljanje zahtevima:** Komunikacija sa backend-om je organizovana kroz servise koji se brinu o slanju HTTP zahteva i obradi odgovora, obezbeđujući sigurnost i konzistentnost podataka kao što je prikazano na listingu broj 8.

```
@Injectable({
  providedIn: 'root'
})
export class AppointmentService {

  private url = "examinations";
  constructor(private http: HttpClient) { }

  createAppointment(appointment: Appointment): Observable<any> {
    return this.http.post<any>(`${environment.baseApiUrl}/${this.url}/appointments`, appointment);
  }
}
```

Listing 8 - Primer slanja zahteva za kreiranje pregleda na backend



# Demostracija

Naš sistem je implementiran kako bi olakšao administraciju i vođenje zdravstvenih pregleda. Ovo poglavlje demonstrira tipičan način korišćenja aplikacije za upravljanje zdravstvenim pregledima, prateći sve potrebne korake i osiguravajući tačnost i ažurnost podataka.

## Demonstracija korišćenja aplikacije za kreiranje zdravstvenih pregleda

Ovo poglavlje demonstrira korake za kreiranje zdravstvenog pregleda od strane doktora u našoj aplikaciji.

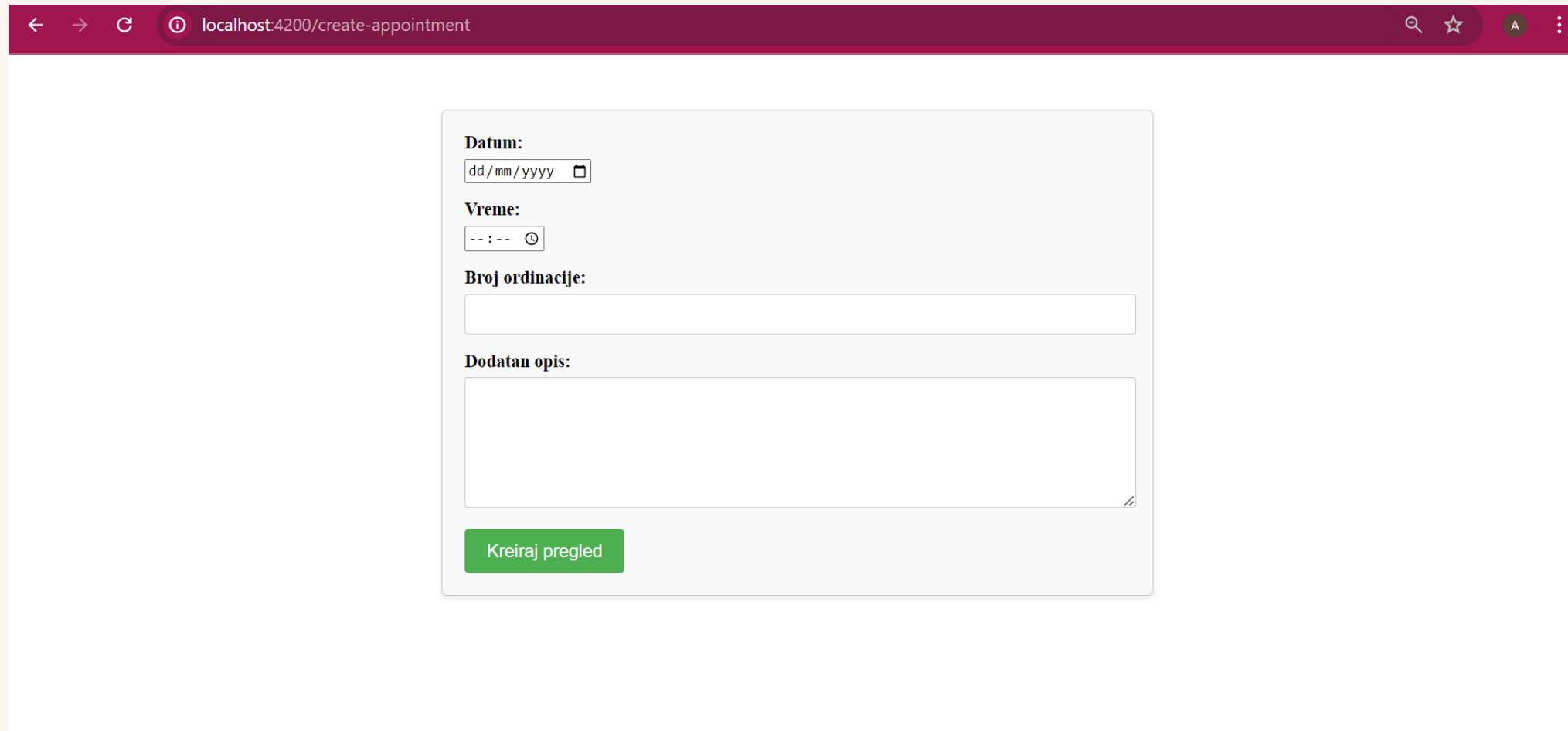
- 1. Prijavljivanje doktora** Doktor se prijavljuje u aplikaciju koristeći svoje doktorske pristupne podatke.
- 2. Pregled dostupnih opcija** Nakon uspešne prijave, doktor ima pristup glavnom meniju gde može izabrati opciju za kreiranje novog zdravstvenog pregleda kao što je prikazano na slici broj 4.
- 3. Unos informacija** Doktor unosi podatke o pregledu tj. bira željeni datum i vreme pregleda, unosi broj prostorije i dodatan opis, ako je potreban, kao što je prikazano na slici broj 5.
- 4. Potvrda kreiranja pregleda** Nakon unosa svih potrebnih informacija, doktor potvrđuje kreiranje pregleda, kao što je prikazano na slici broj 5.

# Demonstracija



Slika broj 4 - Prikaz dostupnih opcija, odabir opcije "Kreiranje pregleda"

# Demonstracija



The screenshot shows a web browser window with a dark red header bar. The address bar displays "localhost:4200/create-appointment". The page content is a light gray form with the following fields:

- Datum:** A date input field with the placeholder "dd/mm/yyyy" and a calendar icon.
- Vreme:** A time input field with the placeholder "--:--" and a clock icon.
- Broj ordinacije:** A text input field.
- Dodatan opis:** A large text area for additional notes.
- Kreiraj pregled:** A green button at the bottom of the form.

Slika broj 5 - Prikaz forme kreiranje termina za zdravstveni pregled

# Zaključak

U ovom radu je predstavljeno softversko rešenje za korišćenje zdravstvenih usluga koje je uspešno odgovorilo na potrebe efikasnog zakazivanja i praćenja termina. Prikazano rešenje olakšava korisnicima da putem web aplikacije zakazuju zdravstvene preglede, omogućava zdravstvenim radnicima da objavljuju i ažuriraju svoju listu slobodnih termina, dok studenti rezervisati termine.

Dobrim stranama ovog rešenja može se smatrati intuitivan korisnički interfejs koji olakšava korisnicima navigaciju i brzo zakazivanje pregleda, uz efikasnu komunikaciju sa serverskom bazom podataka putem REST API-ja. Sa druge strane, nedostaci kao što su potreba za dodatnim unapređenjem UI/UX dizajna ili optimizacija performansi mogu se prevazići daljim iteracijama i upotrebom savremenih tehnologija kao što su Angular i Node.js.

Sistem prikazan u ovom radu, u poređenju sa srodnim rešenjima, pruža jednostavnost pri korišćenju, a istovremeno obezbeđuje potrebne funkcionalnosti. S obzirom na to da u razvoju analiziranog rešenja učestvuju timovi inženjera, očekivano je da ona poseduju izvesne prednosti u odnosu na rešenje predstavljeno u ovom radu.

Za dalji razvoj ovog sistema preporučuje se implementacija dodatnih funkcionalnosti poput integracije sa elektronskim zdravstvenim kartonima ili unapređenje podrške za različite platforme, kao što su mobilne aplikacije ili pristup putem veb pregledača. To bi omogućilo širenje dostupnosti sistema i poboljšanje korisničkog iskustva, što bi rezultiralo većim prihvatanjem od strane krajnjih korisnika.

# Literatura

[1] Detalji o eUpravi. Preuzeto sa <https://euprava.gov.rs/?>.

[2] Detalji o eZdravlju. Preuzeto sa <https://euprava.gov.rs/zivotna-oblast/17?active-tab=0>.

[3] Izvor citata. Preuzeto sa <https://www.ite.gov.rs/>.

[4] Malo više o tome šta je Java [https://www.java.com/en/download/help/whatis\\_java.html](https://www.java.com/en/download/help/whatis_java.html).

[5] Malo više o tome kako se koristi C# <https://www.w3schools.com/cs/index.php>.

[6] Malo više o JavaScriptu <https://www.hostinger.com/tutorials/what-is-javascript>.

[7] Malo više o Reactu [https://www.w3schools.com/whatis/whatis\\_react.asp](https://www.w3schools.com/whatis/whatis_react.asp).

[8] Malo o Oracle bazi podataka <https://docs.oracle.com/en/database/oracle/oracle-database/12.2/cncpt/introduction-to-oracle-database.html>.