

Seminarski rad za predmet Tehnologije i sistemi eUprave

eZdravstvo

Milan Miljuš

Fakultet tehničkih nauka, Univerzitet u Novom Sadu

Trg Dositeja Obradovića 6 21000 Novi Sad

miljus.sr57.2020@uns.ac.rs

Sažetak

U današnjem digitalnom dobu, tehnološki napredak je transformisao mnoge sektore, uključujući i zdravstvo. eZdravstvo, koje predstavlja primenu digitalnih tehnologija u zdravstvenom sektoru, ima potencijal da unapredi dijagnostiku, lečenje, praćenje i kvalitet pružene pomoći građanima.

U ovom radu je opisana veb aplikacija eUprava, konkretno sistem eZdravstva.

Ključne reči:

eUprava, veb aplikacija, komunikacija između sistema, zdravstvo

1. Uvod

Sa napretkom informacionih tehnologija i elektronskih sistema, pojavio se sistem namenjen poboljšanju zdravstvene nege, poznat kao eZdravstvo. Ovaj inovativni pristup, uz pomoć različitih alata omogućava značajan napredak u načinu pružanja pomoći građanima.

Cilj ove veb aplikacije jeste da građanima omogući jednostavnu, brzu i efikasnu komunikaciju sa sistemom eZdravstva. Kreiranje i čuvanje podataka u elektronskom obliku smanjuje šansu ljudske greške, ukida čekanje za šalterom i eliminiše potrebu čuvanja papirnih dokumenata.

2. Srodna istraživanja

U ovom odeljku je dat pregled postojećih rešenja za sistem eZdravstva.

Republika Srbija već ima nacionalni veb portal koji predstavlja pristupnu tačku elektronskoj upravi na kojoj se realizuju usluge elektronske uprave. Taj portal se naziva eUprava. Kao jedan od sistema cele eUprave je sistem eZdravlje[1].

Pristup sistemu je moguće izvršiti u nekoliko koraka:

1. Pristup internetu i veb sajtu na <https://euprava.gov.rs>
2. Zahtev za registraciju
3. Logovanje na sistem putem kreiranog korisničkog imena i šifre ili pomoću kartice zdravstvenog osiguranja

The image shows a web registration form for the eUprava system. The form is divided into several sections:

- Personal Information:** Fields for Name (Име), Surname (Презиме), JMBG (ЈМБГ), and Place of Residence (Пребивалиште). A dropdown menu is provided for selecting the place of residence.
- Identification Document:** A section for uploading a personal ID card or passport. It includes a dropdown for document type (Изаберите тип документа) and a field for document number (Број документа). A note states: "Максимално два документа, ЗМБ сваки. Дозвољени формати: .pdf, .png, .jpg." A red icon indicates that the document must be submitted electronically to verify identity.
- User Name and Password:** Fields for creating a username (Корисничко име) and a password (Лозинка). The password field includes a strength indicator (Јачина лозинке) and a checkbox for agreeing to the terms of use.
- Registration:** A "Региструј ме" button with a right arrow.

On the right side, there is a sidebar with instructions:

- 1 Региструјте налог корисничким именом и лозинком**
 - Попуните форму за регистрацију. Потребно је приложити очитан, скениран или фотографисан лични документ (личну карту или пасош).
 - Послаћемо Вам мејл са линком за потврду адресе електронске поште. Потврдите адресу електронске поште у року од 24 сата.
 - Прегледаћемо достављене податке и у року од највише 48 сати одобрити Ваш захтев за регистрацију, о чему ћемо Вас обавестити путем мејла.
- 2 Активирајте мобилну апликацију ConsentID**

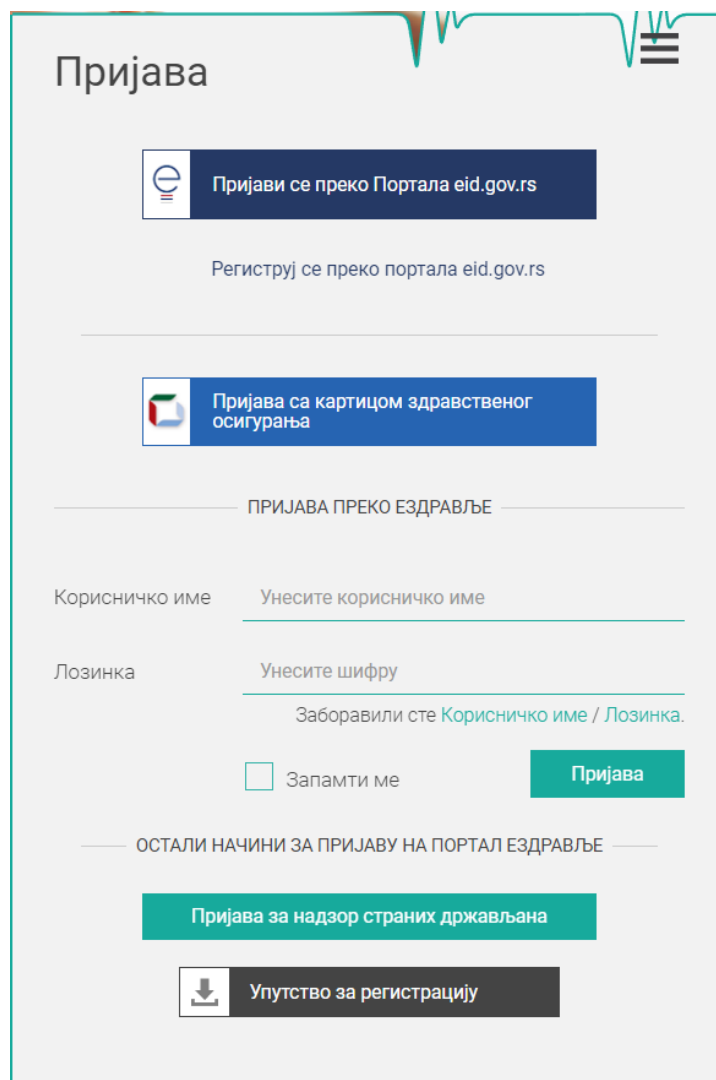
Пријава мобилном апликацијом ConsentID представља пријаву високог нивоа поузданости и омогућава Вам приступ свим услугама и функционалностима електронске управе.
- 3 Активирајте квалификовани електронски сертификат у клауду**

Квалификовани електронски сертификат у клауду (КЕС у клауду) Вам омогућава да користите свој квалификовани електронски потпис на даљину. Нисте везани за рачунар и потписивање можете да обавите у време и на месту које Вам одговара помоћу апликације ConsentID на мобилном уређају.

Below the instructions, there is a section titled "Постаните еГрађанин" (Become eCitizen) with a brief description of the registration process and a "Више детаља" (More details) button.

At the bottom right, there is a link: "↩ На почетну страницу" (Back to the home page).

Слика 1 – Регистрација на систем еУправе



Пријава

Пријави се преко Портала eid.gov.rs

Региструј се преко портала eid.gov.rs

Пријава са картицом здравственог осигурања

ПРИЈАВА ПРЕКО ЕЗДРАВЉЕ

Корисничко име Унесите корисничко име

Лозинка Унесите шифру

Заборавили сте [Корисничко име / Лозинка.](#)

☐ Запамти ме **Пријава**

ОСТАЛИ НАЧИНИ ЗА ПРИЈАВУ НА ПОРТАЛ ЕЗДРАВЉЕ

Пријава за надзор страних држављана

Упутство за регистрацију

Slika 2 – Stranica za prijavu na sistem eZdravlje

Nakon uspešnog logovanja korisnik ima mogućnost da koristi sistem eZdravlja.

4. Korišćene tehnologije

U ovom odeljku su ukratko predstavljene tehnologije koje su korišćene za razvijanje veb aplikacije.

Aplikacija se može podeliti na tri celine:

- Poslovna logika aplikacija (Backend)
- Korisnički interfejs aplikacije (Frontend)
- Baza podataka

Za izradu poslovne logike aplikacije korišćen je programski jezik Golang[2].

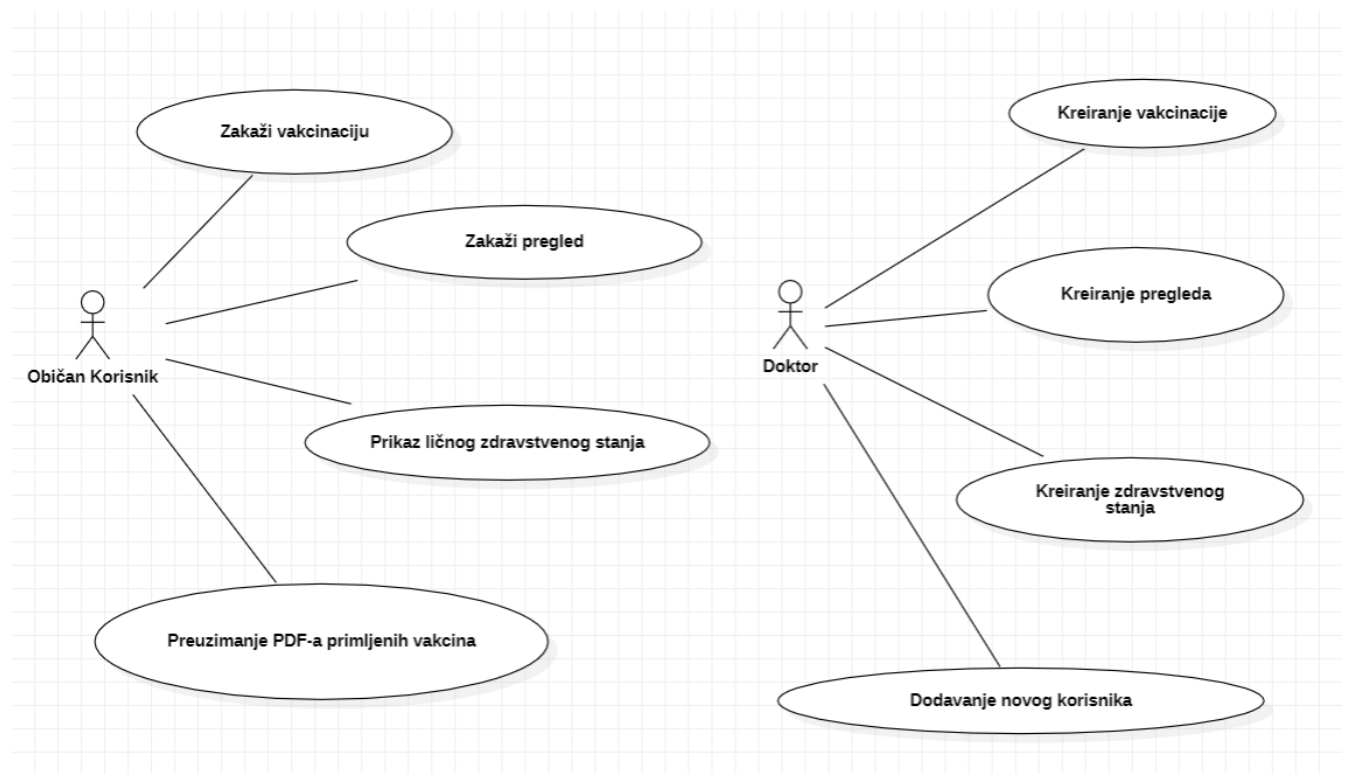
Za izradu korisničkog interfejsa aplikacije korišćen je Angular[3] sa Typescript[4] programskim jezikom.

Za čuvanje podataka korišćena je NoSQL[5] baza podataka MongoDB[6].

5. Specifikacija zahteva

U ovom odeljku su opisani funkcionalni zahtevi koje je potrebno da ispunjava softversko rešenje za eZdravstvo.

Na slici 3 je prikazan UML[7] dijagram slučajeva korišćenja sa funkcionalnim zahtevima ovog softverskog rešenja.



Slika 3 – Dijagram slučajeva korišćenja

Tabela 1 prikazuje opis slučaja korišćenja “Kreiranje vakcinacije”.

Naziv	Kreiranje vakcinacije
Učesnici	Doktor
Preduslovi	1. Korisnik koji je logovan mora biti Doktor
Koraci	1. Doktor bira opciju za dodavanje novog termina vakcinacije 2. Doktor unosi vreme i tip vakcine 3. Doktor potvrđuje unos
Rezultati	Kreiran je novi termin vakcinacije
Izuzeci	Već postoji zakazana vakcinacija u unetom vremenu

Tabela 1 – Opis slučaja korišćenja “Kreiranje vakcinacije”

Tabela 2 prikazuje opis slučaja korišćenja “Dodavanje novorođene osobe”.

Naziv	Dodavanje novorođene osobe
Učesnici	Doktor
Preduslovi	1. Korisnik koji je logovan mora biti Doktor
Koraci	1. Doktor bira opciju za dodavanje novorođene osobe 2. Doktor unosi ime, prezime, JMBG, datum rođenja, mesto rođenja, ime oca, JMBG oca, ime majke, JMBG majke, pol 3. Doktor potvrđuje unos
Rezultati	Korisnik je poslat na sistem Matičara i sačuvan
Izuzeci	JMBG već postoji

Tabela 2 – Opis slučaja korišćenja “Dodavanje novorođene osobe”

Tabela 3 prikazuje opis slučaja korišćenja “Zakaži vakcinaciju”.

Naziv	Zakaži vakcinaciju
Učesnici	Običan korisnik
Preduslovi	1. Korisnik mora biti prijavljen na sistem
Koraci	1. Korisnik bira jedan od slobodnih termina vakcinacije 2. Korisnik potvrđuje odabir
Rezultati	Korisnik je zakazao termin vakcinacije
Izuzeci	Termin vakcinacije ne sme biti zauzet

Tabela 3 – Opis slučaja korišćenja “Zakaži vakcinaciju”

Tabela 4 prikazuje opis slučaja korišćenja “Preuzimanje PDF-a primljenih vakcina”.

Naziv	Preuzimanje PDF-a primljenih vakcina
Učesnici	Običan korisnik
Preduslovi	1. Korisnik mora biti prijavljen na sistem
Koraci	1. Korisnik odlazi na stranicu sa primljenim vakcinama 2. Korisnik preuzima PDF klikom na dugme
Rezultati	Korisnik je preuzeo PDF na svoj računar
Izuzeci	-

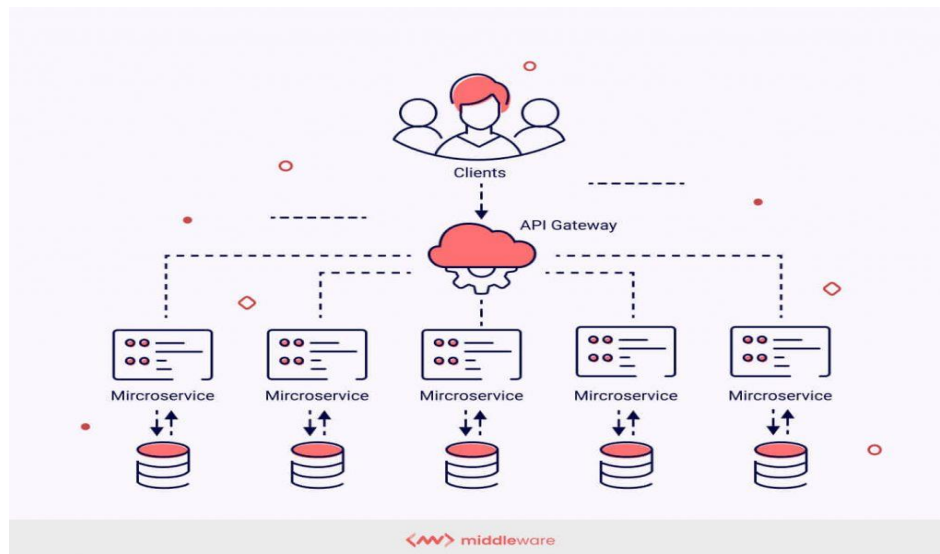
Tabela 4 – Opis slučaja korišćenja “Preuzimanje PDF-a primljenih vakcina”

6. Specifikacija dizajna

U ovom odeljku se objašnjava dizajn softverskog rešenja sistema eZdravstva.

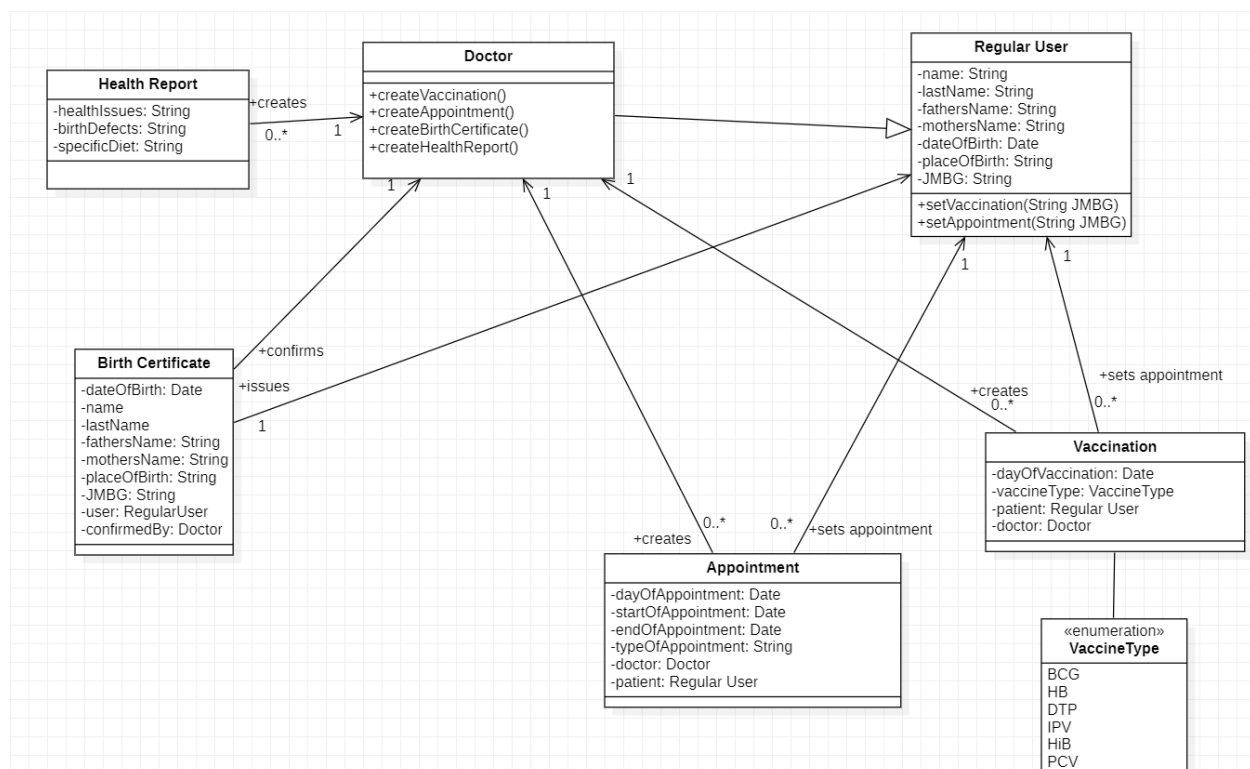
Sistem je realizovan kao mikroservisna arhitektura gde različiti servisi komuniciraju jedni sa drugima dok u isto vreme ostaju nezavisni jedni od drugih u slučaju da jedan otkáže a da aplikacija nastavi da radi.

Na slici 4 je prikazan primer rada mikroservisne arhitekture.



Slika 4 – Primer mikroservisne arhitekture

Na slici 5 je prikazan klasni dijagram sa entitetima sistema eZdravstva.



Slika 5 – Klasni dijagram

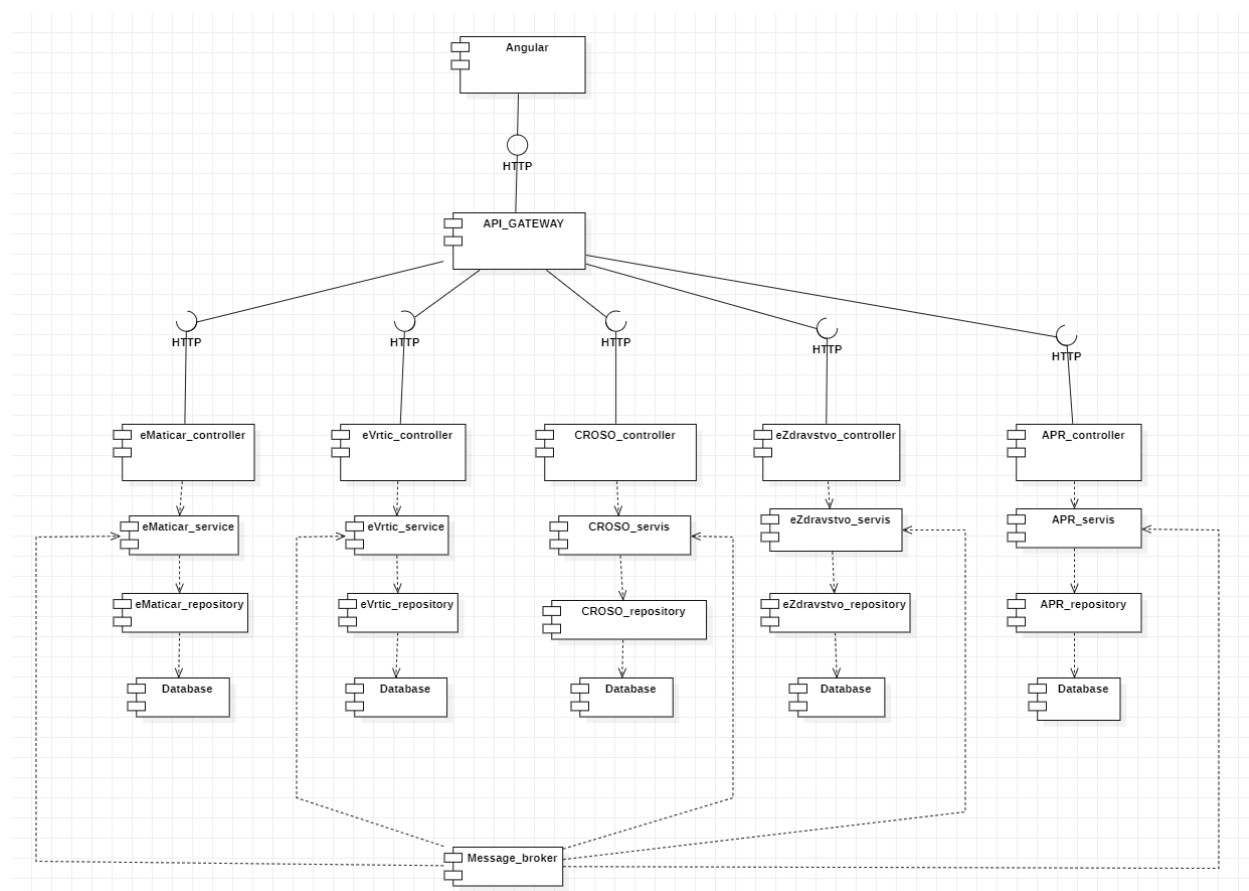
Klasom doktor i običan korisnik predstavljena su dva tipa korisnika u sistemu eZdravstva.

Klase vakcinacija i pregled predstavljaju entitete koje doktor može dodavati a običan korisnik kasnije zakazati.

Klasa zdravstveno stanje predstavlja entitet koji doktor može izdati za nekog korisnika ili dete, koji će kasnije komunicirati sa sistemom eVrtića tokom bodovanja dece za konkurs.

Klasa novorođeni predstavlja entitet koji doktor popunjava pa komunicira sa sistemom eMatičara kako bi novorođena osoba bila dodata u bazu matičara.

Na slici 6 je prikazan dijagram komponenti za celu strukturu veb aplikacije.



Slika 6 – Dijagram komponenti

Komponenta Angular šalje HTTP zahtev na komponentu API Gateway[8] koja onda preusmerava svaki zahtev na određeni servis u zavisnosti od URL na koji je zahtev poslat.

7. Implementacija

U ovom odeljku se prikazuje način na koji su implementirani klijentski(Frontend) i serverski(Backend) deo aplikacije.

Klijent

Klijentski(Frontend) deo aplikacije je implementiran pomoću Typescript programskog jezika koristeći Angular radno okruženje(framework).

Na slici 7 je prikazan HTML kod koji se koristi za unošenje nove vakcinacije u sistem. [formGroup] prikuplja vrednosti unete u input polja.

```

1 <div>
2   <div>
3     <mat-card class="add-vaccine-card">
4       <mat-card-title class="card-title-main">Kreiranje Nove Vakcinacije</mat-card-title>
5       <div class="form-group">
6         <form [formGroup]="vaccinationFormGroup">
7           <mat-form-field class="date-pick" appearance="outline">
8             <mat-label>Početak i Kraj</mat-label>
9             <mat-date-range-input [rangePicker]="picker">
10              <input matStartDate formControlName="startOfVaccination" placeholder="Početak"
11                [ngClass]="{ 'is-invalid': submitted && vaccinationFormGroup['startOfVaccination'].errors }">
12              <div *ngIf="submitted && vaccinationFormGroup['startOfVaccination'].errors" class="invalid-feedback">
13                <div *ngIf="vaccinationFormGroup['startOfVaccination'].errors['required']">Početak je obavezan.</div>
14              </div>
15              <input matEndDate formControlName="endOfVaccination" placeholder="Kraj"
16                [ngClass]="{ 'is-invalid': submitted && vaccinationFormGroup['endOfVaccination'].errors }">
17              <div *ngIf="submitted && vaccinationFormGroup['endOfVaccination'].errors" class="invalid-feedback">
18                <div *ngIf="vaccinationFormGroup['endOfVaccination'].errors['required']">Kraj je obavezan.</div>
19              </div>
20            </mat-date-range-input>
21            <mat-datepicker-toggle matSuffix [for]="picker"></mat-datepicker-toggle>
22            <mat-date-range-picker #picker></mat-date-range-picker>
23          </mat-form-field>
24          <mat-form-field class="vaccine-pick" appearance="outline">
25            <mat-label>Tip Vakcine</mat-label>
26            <mat-select formControlName="vaccineType">
27              <mat-option style="font-size: 15px; text-align: center" *ngFor="let vaccineType of vaccineTypes" [value]="vaccineType">
28                {{vaccineType}}
29              </mat-option>
30            </mat-select>
31          </mat-form-field>
32          <h2 class="already-exists" *ngIf="alreadyExists">Vakcinacija već postoji u datom vremenu
33            <button mat-button (click)="removeError()">
34              <mat-icon>remove_circle</mat-icon>
35            </button>
36          </h2>
37          <h2 class="time-display">Početak: {{vaccinationFormGroup.value.startOfVaccination | date}}</h2>
38          <h2 class="time-display">Kraj: {{vaccinationFormGroup.value.endOfVaccination | date}}</h2>
39          <h2 class="vaccine-display">Tip Vakcine: {{vaccinationFormGroup.value.vaccineType}}</h2>
40          <div class="form-group form-check">
41            <button type="submit" class="btn btn-primary" (click)="onSubmit()">Dodaj</button>
42          </div>
43        </form>
44      </div>
45    </mat-card>
46  </div>
47 </div>

```

Slika 7 – HTML kod za dodavanje vakcinacije

Na slici 8 je prikazan Typescript kod koji unutar funkcije “onSubmit()” preko [formGroup] prikuplja vrednosti unete unutar input polja na HTML stranici. Nakon toga se vakcinacija šalje na odgovarajući servis.

```

46  onSubmit() {
47      this.submitted = true
48
49      if (this.vaccinationFormGroup.invalid) {
50          return;
51      }
52
53      let addVaccination: AddVaccination = new AddVaccination()
54
55      var StartOfVaccination: Date = new Date(this.vaccinationFormGroup.get('startOfVaccination')?.value)
56      var EndOfVaccination: Date = new Date(this.vaccinationFormGroup.get('endOfVaccination')?.value)
57      var VaccineType = this.vaccinationFormGroup.get("vaccineType")?.value
58
59      addVaccination.startOfVaccination = Number(StartOfVaccination.getTime()) / 1000
60      addVaccination.endOfVaccination = Number(EndOfVaccination.getTime()) / 1000
61      addVaccination.vaccineType = VaccineType
62
63      this.healthcareService.AddVaccination(addVaccination)
64          .subscribe({
65              next: (data) => {
66                  this.router.navigate(['/Vaccinations-Doctor'])
67              },
68              error: (error) => {
69                  console.log(error)
70                  if (error.status = 406) {
71                      this.alreadyExists = true
72                  }
73              }
74          })
75      }
76

```

Slika 8 – Typescript kod za slanje vakcinacije na servis

Na slici 9 je prikazan zdravstvo servis koji sadrži HTTP Post zahtev koji prima “RequestBody” tipa “Vakcinacija”. Slanjem HTTP metode Post na odgovarajuću adresu ostvaruje se konekcija sa backend delom i vakcinacija se šalje na dalju obradu.

```

public AddVaccination(addVaccination: AddVaccination): Observable<AddVaccination> {
    return this.http.post<AddVaccination>(`${environment.baseApiUrl}/${this.url}/newVaccination`, addVaccination);
}

```

Slika 9 – Slanje vakcinacije na backend putem servisa

Server

Serverski(Backend) deo aplikacije je implementiran pomoću Go programskog jezika pisanim u IntelliJ GoLand[6] IDE razvojnom okruženju.

Na slici 10 je prikazan kontroler koji prihvata pristigle zahteve sa klijentske strane.

Nakon što kontroler primi pristigli zahtev, on u zavisnosti od putanje na koju je zahtev poslat poziva odgovarajuću funkciju i obrađuje podatke ili izvršava neku drugu logiku.

```
func (controller *HealthcareController) Init(router *mux.Router) {
    authEnforcer, err := casbin.NewEnforcerSafe(
        params...: "./auth_model.conf",
        policyPath: "./policy.csv")
    if err != nil {
        log.Fatal(err)
    }

    router.HandleFunc(path: "/allAppointments", controller.GetAllAppointments).Methods(methods...: "GET")
    router.HandleFunc(path: "/myAppointmentsDoctor", controller.GetAllMyAppointmentsDoctor).Methods(methods...: "GET")
    router.HandleFunc(path: "/myAvailableAppointmentsDoctor", controller.GetMyAvailableAppointmentsDoctor).Methods(methods...: "GET")
    router.HandleFunc(path: "/myTakenAppointmentsDoctor", controller.GetMyTakenAppointmentsDoctor).Methods(methods...: "GET")
    router.HandleFunc(path: "/allAvailableAppointments", controller.GetAllAvailableAppointments).Methods(methods...: "GET")
    router.HandleFunc(path: "/getAppointmentByID/{id}", controller.GetAppointmentByID).Methods(methods...: "GET")
    router.HandleFunc(path: "/newAppointment", controller.CreateNewAppointment).Methods(methods...: "POST")
    router.HandleFunc(path: "/setAppointment/{id}", controller.SetAppointment).Methods(methods...: "PUT")
    router.HandleFunc(path: "/deleteAppointmentByID/{id}", controller.DeleteAppointmentByID).Methods(methods...: "DELETE")
    router.HandleFunc(path: "/allVaccinations", controller.GetAllVaccinations).Methods(methods...: "GET")
    router.HandleFunc(path: "/myVaccinationsDoctor", controller.GetAllMyVaccinationsDoctor).Methods(methods...: "GET")
    router.HandleFunc(path: "/myAvailableVaccinationsDoctor", controller.GetMyAvailableVaccinationsDoctor).Methods(methods...: "GET")
    router.HandleFunc(path: "/myTakenVaccinationsDoctor", controller.GetMyTakenVaccinationsDoctor).Methods(methods...: "GET")
    router.HandleFunc(path: "/allAvailableVaccinations", controller.GetAllAvailableVaccinations).Methods(methods...: "GET")
    router.HandleFunc(path: "/getVaccinationByID/{id}", controller.GetVaccinationByID).Methods(methods...: "GET")
    router.HandleFunc(path: "/newVaccination", controller.CreateNewVaccination).Methods(methods...: "POST")
    router.HandleFunc(path: "/setVaccination/{id}", controller.SetVaccination).Methods(methods...: "PUT")
    router.HandleFunc(path: "/deleteVaccinationByID/{id}", controller.DeleteVaccinationByID).Methods(methods...: "DELETE")
    router.HandleFunc(path: "/allZdravstvenoStanje", controller.GetAllZdravstvenoStanje).Methods(methods...: "GET")
    router.HandleFunc(path: "/getZdravstvenoStanjeByID/{id}", controller.GetZdravstvenoStanjeByID).Methods(methods...: "GET")
    router.HandleFunc(path: "/getZdravstvenoStanjeByJMBG/{jmbg}", controller.GetZdravstvenoStanjeByJMBG).Methods(methods...: "GET")
    router.HandleFunc(path: "/newZdravstvenoStanje", controller.CreateNewZdravstvenoStanje).Methods(methods...: "POST")
    router.HandleFunc(path: "/deleteZdravstvenoStanjeByJMBG/{jmbg}", controller.DeleteZdravstvenoStanjeByJMBG).Methods(methods...: "DELETE")
    router.HandleFunc(path: "/addPersonToRegistry", controller.AddPersonToRegistry).Methods(methods...: "POST")
    router.HandleFunc(path: "/getMe", controller.GetMe).Methods(methods...: "GET")
    http.Handle("/", router)
    log.Fatal(http.ListenAndServe(addr: ":8005", authorization.Authorizer(authEnforcer)(router)))
}
```

Slika 10 – Kontroler zdravstva sa dostupnim putanjama i funkcijama koje svaka poziva

Na slici 11 je prikazana funkcija koja se poziva nakon slanja zahteva na putanju “/newVaccination”. Funkcija obrađuje pristigle podatke i zatim je šalje na servisni sloj.

```
func (controller *HealthcareController) CreateNewVaccination(writer http.ResponseWriter, req *http.Request) { 1 usage MilanM2001
    var vaccination model.Vaccination
    err := json.NewDecoder(req.Body).Decode(&vaccination)

    if err != nil {
        writer.WriteHeader(http.StatusInternalServerError)
        writer.Write([]byte("There is a problem in decoding JSON"))
        return
    }

    jmbg, err := extractJMBGFromClaims(writer, req)

    value, err := controller.service.CreateNewVaccination(&vaccination, jmbg)
    if value == 1 {
        writer.WriteHeader(http.StatusNotAcceptable)
        writer.Write([]byte("Vaccination already exists in that time"))
        return
    }
    if err != nil {
        writer.WriteHeader(http.StatusInternalServerError)
        return
    }

    jsonResponse(vaccination, writer)
    writer.WriteHeader(http.StatusOK)
}
```

Slika 11 – Obrada podataka unutar funkcije za dodavanje vakcinacije

Slika 12 prikazuje dalju obradu podataka u servisnom sloju nakon što stignu iz kontrolera. Unutar funkcije se proverava vreme nove vakcinacije i poredi sa već postojećim vakcinacijama iz baze kako se termini vremenski ne bi poklapali. Ukoliko nova vakcinacija prođe validaciju šalje se repozitorijumu.

```

func (service *HealthcareService) CreateNewVaccination(vaccination *model.Vaccination, jmbg string) (int, error) {
    dataToSend, err := json.Marshal(jmbg)
    if err != nil {
        log.Println("Error Marshaling JMBG")
    }
    existingVaccinations, err := service.repository.GetAllVaccinations()
    for _, existingVaccination := range existingVaccinations {
        if (existingVaccination.StartOfVaccination >= vaccination.StartOfVaccination && existingVaccination.StartOfVaccination <= vaccination.EndOfVaccination) ||
            (existingVaccination.EndOfVaccination >= vaccination.StartOfVaccination && existingVaccination.EndOfVaccination <= vaccination.EndOfVaccination) ||
            (existingVaccination.StartOfVaccination >= vaccination.StartOfVaccination && existingVaccination.EndOfVaccination <= vaccination.EndOfVaccination) {
            return 1, nil
        }
    }
    if err != nil {
        log.Println("Error getting All Vaccinations", err)
        return 0, err
    }

    response, err := service.natsConnection.Request(os.Getenv("key: GET_USER_BY_JMBG"), dataToSend, 5*time.Second)

    var doctor model.User
    err = json.Unmarshal(response.Data, &doctor)
    if err != nil {
        log.Println("Error in Unmarshalling json")
        return 0, err
    }

    vaccination.ID = primitive.NewObjectID()
    vaccination.Doctor = &doctor
    vaccination.User = nil

    err = service.repository.CreateNewVaccination(vaccination)
    if err != nil {
        log.Println("Error in trying to save Vaccination")
        return 0, err
    }
    return 0, nil
}

```

Slika 12 – Obrada i validacija podataka unutar servisnog sloja

Slika 13 prikazuje funkciju u repozitorijumu koja prima vakcinaciju iz servisnog sloja i čuva je u bazi zdravstva.

```

func (repository *HealthcareRepositoryImpl) CreateNewVaccination(vaccination *model.Vaccination) error {
    _, err := repository.vaccination.InsertOne(context.Background(), vaccination)
    if err != nil {
        return err
    }
    return nil
}

```

Slika 13 – Čuvanje vakcinacije unutar baze

8. Demonstracija

U ovom odeljku je predstavljena demonstracija korišćenja veb aplikacije eZdravstvo.

Nakon prijave na sistem unošenjem JMBG-a i šifre, korisnik je preusmeren na odgovarajuću stranicu u zavisnosti od dodeljene uloge.

Doktor na stranici sa vakcinacijama može da vidi sve termine koje je on zakazao kao i da doda novi termin.

PREGLEDI VAKCINACIJE ZDRAVSTVENO STANJE DODAVANJE OSOBE SERVISI LOGOUT

Moje Vakcinacije

Tip Vakcinacije

Slobodan Termin
Doktor: Doktor Doktorski
Tip Vakcine: BCG
Početak: June 1, 2023
Kraj: June 2, 2023

Dodavanje Vakcinacije

Slika 14 – Stranica sa svim vakcinacijama ulogovanog doktora

Klikom na dugme za dodavanje vakcinacija, doktor je preusmeren na stranicu za dodavanje novog termina vakcinacije.

PREGLEDI VAKCINACIJE ZDRAVSTVENO STANJE DODAVANJE OSOBE SERVISI LOGOUT

Kreiranje Nove Vakcinacije

Početak i Kraj
6/15/2023 – 6/16/2023

Tip Vakcine *
IPV

Početak: Jun 15, 2023
Kraj: Jun 16, 2023
Tip Vakcine: IPV

Dodaj

Slika 15 – Doktor kreira novi termin vakcinacije

Nakon klika na dugme za dodavanje, novi termin vakcinacije je zakazan i dodat u bazu.

Moje Vakcinacije

Tip Vakcinacije

Dodavanje Vakcinacije

Slobodan Termin
Doktor: Doktor Doktorski
Tip Vakcine: BCG
Početak: June 1, 2023
Kraj: June 2, 2023

Slobodan Termin
Doktor: Doktor Doktorski
Tip Vakcine: IPV
Početak: June 15, 2023
Kraj: June 16, 2023

Slika 16 – Prikaz svih vakcinacija nakon dodavanja novog termina

9. Zaključak

U seminarskom radu je opisan sistem eZdravlja, njegova implementacija i njegove funkcionalnosti. Kroz jedan sistem se na brz i jednostavan način zakazuju pregledi, vakcinacije i izdaju zdravstvena stanja i time ne trošimo vreme koje bismo inače potrošili.

10. Literatura

- [1] eZdravlje. Preuzeto 29.06.2023 sa <https://www.e-zdravlje.gov.rs/>
- [2] The Go Programming Language. Preuzeto 29.06.2023 sa <https://go.dev/>
- [3] Angular. Preuzeto 29.06.2023 sa <https://angular.io/>
- [4] Typescript: Javascript with syntax for types. Preuzeto 29.06.2023 sa <https://www.typescriptlang.org/>
- [5] What Is NoSQL? NoSQL Databases Explained. Preuzeto 29.06.2023 sa <https://www.mongodb.com/nosql-explained>

- [6] MongoDB: The Developer Data Platform. Preuzeto 29.06.2023 sa <https://www.mongodb.com/>
- [7] UML – Unified Modeling Language. Preuzeto 04.07.2023 sa <https://www.uml.org/>
- [8] What is an API Gateway? A Quick Learn Guide. Preuzeto 01.07.2023 sa <https://tinyurl.com/yzrnsnx>