Web aplikacija za automatsko zauzeće termina

Kiralj, Erik

Master's thesis / Diplomski rad

2019

Degree Grantor / Ustanova koja je dodijelila akademski / stručni stupanj: **Josip Juraj Strossmayer University of Osijek, Faculty of Electrical Engineering, Computer Science and Information Technology Osijek / Sveučilište Josipa Jurja Strossmayera u Osijeku, Fakultet elektrotehnike, računarstva i informacijskih tehnologija Osijek**

Permanent link / Trajna poveznica: https://urn.nsk.hr/urn:nbn:hr:200:874190

Rights / Prava: In copyright / Zaštićeno autorskim pravom.

Download date / Datum preuzimanja: **2025-03-21**

Repository / Repozitorij:

Faculty of Electrical Engineering, Computer Science and Information Technology Osijek

SVEUČILIŠTE JOSIPA JURJA STROSSMAYERA U OSIJEKU
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA OSIJEK

Sveučilišni studij

WEB APLIKACIJA ZA AUTOMATSKO ZAUZEĆE TERMINA

Diplomski rad

Erik Kiralj

Osijek, 2019.

SADRŽAJ:

1. UVOD 1
2. PROGRAMSKI JEZICI I OKRUŽENJA 2
2.1. PHP 2
2.2. MVC arhitektura 2
2.3. Laravel 2
2.4. HTML 3
2.5. CSS 3
2.6. XAMPP 3
2.7. HeidiSQL 4
2.8. Visual Studio Code 5
3. PROGRAMSKO RJEŠENJE 6
4. REALIZACIJA APLIKACIJE 10
4.1. Baza podataka 10
4.2. Modeli 12
4.3. Rute 12
4.4. Upravitelji 13
4.5. Pogledi 14
4.6. Opis rada aplikacije 16
5. ZAKLJUČAK 26
LITERATURA 27
SAŽETAK 28
ABSTRACT 29
ŽIVOTOPIS 30

1. UVOD

U današnje doba vrijeme je sve dragocjenije te je važno što bolje organizirati isto. Kako bi se moderniziralo i ljudima olakšalo prijavljivanje na preglede u različitim zdravstvenim granama, jedno od rješenja predstavlja izrada web aplikacije.

Putem web aplikacije ljudi bi imali uvid u slobodne termine liječnika te bi mogli bolje iskoristiti svoje vrijeme i prijaviti se na termin pregleda koji im najviše odgovara. Iz tog zaključka nastala je ideja za ovaj diplomski rad.

Zadatak diplomskog rada je napraviti web aplikaciju koja će omogućiti dodavanje pružatelja različitih zdravstvenih usluga. Svaka ponuđena usluga ima svoje trajanje te pružatelji usluga unose usluge koje pružaju. Korisnici koji se prijavljuju, u određenom trenutku žele koristiti te prijaviti se za različite usluge (preglede). Ključni dio izrade aplikacije jest osmisliti i napraviti algoritam koji će što učinkovitije raspodijeliti korisnike za korištenje usluga u slobodnim terminima.

2. PROGRAMSKI JEZICI I OKRUŽENJA

2.1. PHP

PHP (engl. Hypertext Preprocessor) je skriptni jezik otvorenog tipa koji se temelji na klijent-poslužitelj arhitekturi. Razlikuje se od klijentskih skriptnih jezika koji se izvršavaju u pregledniku po tome što se izvršava na poslužitelju. PHP se koristi za kreiranje dinamičkih internet stranica. Prilikom posjećivanja PHP web stranice, poslužitelj automatski obrađuje kod, te na osnovu njega određuje što će prikazati korisniku. Sve operacije s datotekama, varijablama te matematičke operacije ne prikazuju se korisniku, već se u internet pretraživaču vidi samo generirana HTML stranica bez PHP koda. Za izradu ovog diplomskog rada korišten je Laravel PHP okvir (engl. Framework). Kako bi bilo moguće koristit Laravel okruženje potrebno je imati instalirano PHP verziju 7.1.3. ili noviju. Podaci su navedeni prema literaturi [1].

2.2. MVC arhitektura

MVC (engl. Model View Controller) je softverska arhitektura koja se koristi za odvajanje pojedinih dijelova aplikacije u komponente ovisno o njihovoj namjeni. Model predstavlja podatkovnu logiku aplikacije, pogled (engl. View) predstavlja prethodno modelirane korisničke podatke dok upravitelj (engl. Controller), kao što i samo ime govori, upravlja korisničkim zahtjevima te prihvaća ulazne podatke i pretvara ih u naloge modelu ili pogledu. Ovakva arhitektura omogućava nezavisan razvoj, testiranje i održavanje web aplikacije. Podaci su navedeni prema literaturi [2].

2.3. Laravel

Laravel je PHP radni okvir otvorenog tipa (engl. Open-source web framework) koji služi za izradu internetskih aplikacija baziranih na MVC arhitekturi. Olakšava učenje i veoma brzo kreiranje web aplikacija. Razvijen je 2011. godine te je ubrzo postao jedan je od najpopularnijih PHP okvira koji se koristi širom svijeta čiji se cijeli izvorni kod nalazi na GitHub-u. Da bi korištenje Laravel okruženja bilo omogućeno, potrebno je instalirati Composer koji omogućuje upravljanje komponentama od kojih se Laravel sastoji. Njime se u aplikaciju uključuju biblioteke koje nisu zadane i definirane. Sučelje naredbenog retka koje se koristi u Laravelu naziva se Artisan. Artisan sadrži skup naredbi koje pomažu u izgradnji web aplikacije. Glavne značajke ovog okruženja su modularnost, mogućnost testiranja, usmjeravanje, čist, jednostavan te

pregledan programski kod, kod kojeg se svakom novom nadogradnjom uvode nove funkcije koje korisnicima omogućuju sve učinkovitije mogućnosti. Laravel je dizajniran za modularnu primjenu, te je sam kolekcija raznih komponenti. Logiku aplikacije je moguće odvojiti u različite module koji rade zajedno kako bi internetska aplikacija ispunila svoju funkcionalnost. Podaci su navedeni prema literaturi [3].

2.4. HTML

HTML (engl. HyperText Markup Language) je prezentacijski jezik za izradu web stranica. Dokument se stvara pomoću istoimenog jezika te se njime oblikuje napisani sadržaj i stvaraju hiperveze HTML dokumenata. Besplatan je, jednostavan za uporabu i lako se uči što je jedan od glavnih razloga popularnosti. Prikazuje se u internet pregledniku, a glavna zadaća je uputiti internet preglednik kako da prikaže određeni hipertekst dokument. HTML nije programski jezik, pa se tako njime ne mogu izvršiti nikakve zadaće ni operacije, već samo prikaz hipertekstualnih dokumenata s .html (.htm) nastavkom. Elementi HTML dokumenata kategorizirani su u oznake (engl. Tags). Svaka oznaka započinje s parom znakova „<“ i „>“ unutar koji se nalazi oznaka, dok je na završetku „/<“ i „>“ (npr. <html> </html>). Podaci su navedeni prema literaturi [4].

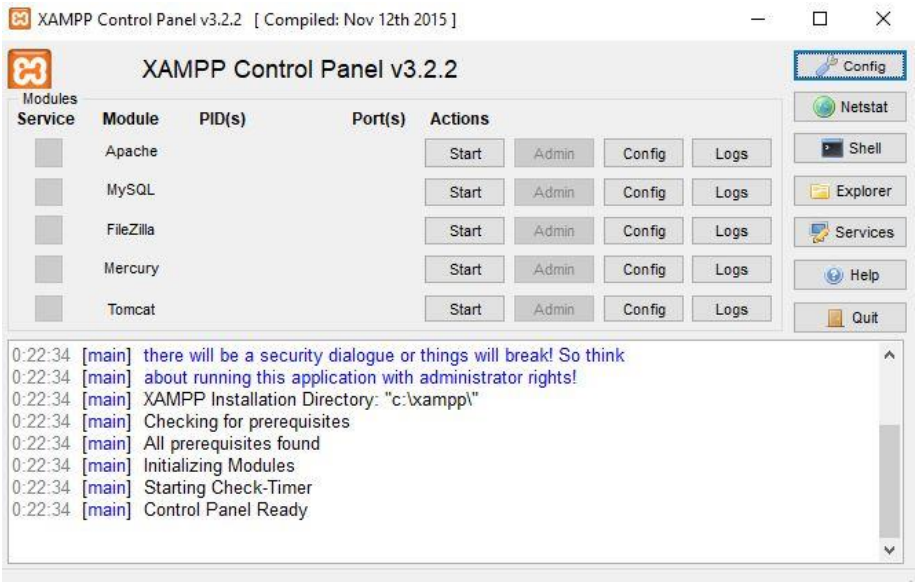
2.5. CSS

CSS (engl. Cascading Style Sheets) je stilski jezik, koji se koristi za oblikovanje internetskih stranica tj. prezentiranje dokumenata napisanog pomoću HTML prezentacijskog jezika. CSS pravila primjenjuju se kaskadno. Pravila je moguće napisati tako da budu primjenjiva na sve elemente ili samo na neke elemente ili tako da vrijedi samo za točno odabrani element. Pravilima je moguće mijenjati svojstva elemenata kao što su boja, font, veličina, izgled itd., a pišu se izravno u HTML dokumentu ili u odvojenom dokumentu s nastavkom .css. Podaci su navedeni prema literaturi [5].

2.6. XAMPP

XAMPP (Apache + MariaDB + PHP + Perl) je besplatan serverski paket otvorenog tipa koji služi za instalaciju Apache servera. XAMPP je namijenjen za upotrebu u lokalnoj mreži, a ne kao web server i prije svega služi za kreiranje servera u svrhu testiranja. Kada se radi na izradi internetske stranice ili aplikacije u programskom jeziku kao što je PHP, neophodno je imati

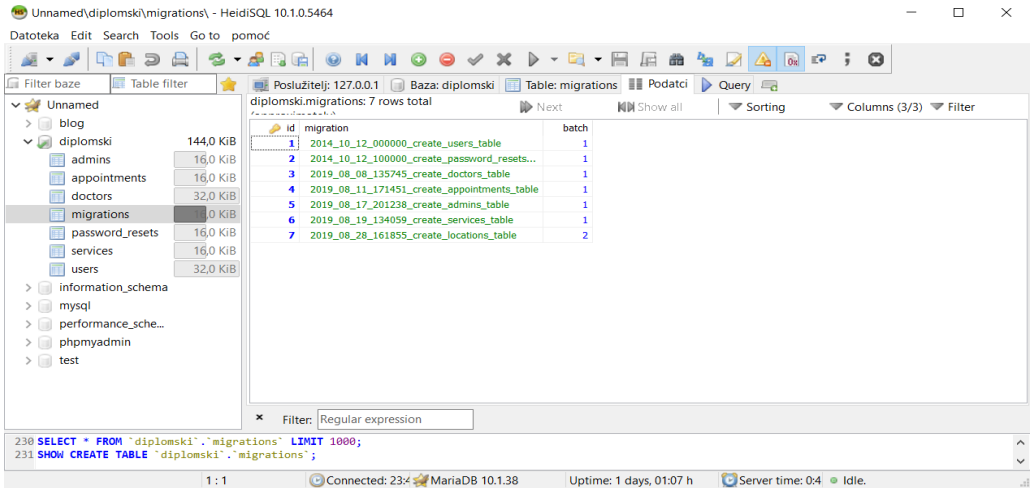
podršku za pokretanje iste. Na slici 2.1. prikazano je kontrolno sučelje XAMPP programa. Podaci su navedeni prema literaturi [6].



Sl. 2.1. XAMPP programsko okruženje.

2.7. HeidiSQL

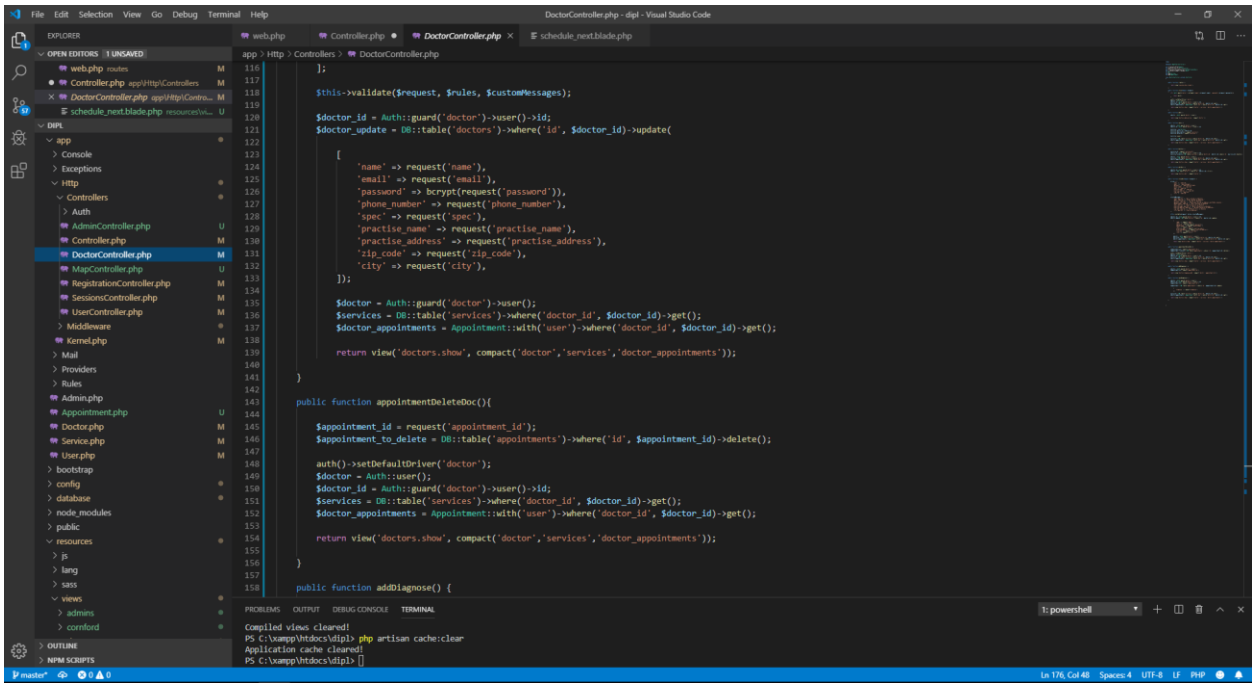
HeidiSQL je besplatno programsko okruženje za upravljanje bazama podataka koje su potrebne za izradu internetskih aplikacija. Omogućuje pregledavanje i uređivanje podataka, stvaranje i uređivanje tablica i struktura s računala koje pokreće jednu od baza podataka (MariaDB, MySQL, Microsoft SQL ili PostgreSQL). Kreiranu strukturu baze moguće je izvesti u SQL datoteku, međuspremnik ili na druge poslužitelje. Na slici 2.2. prikazana je baza podataka u HeidiSQL. Podaci su navedeni prema literaturi [7].



Sl. 2.2. HeidiSQL baza podataka.

2.8. Visual Studio Code

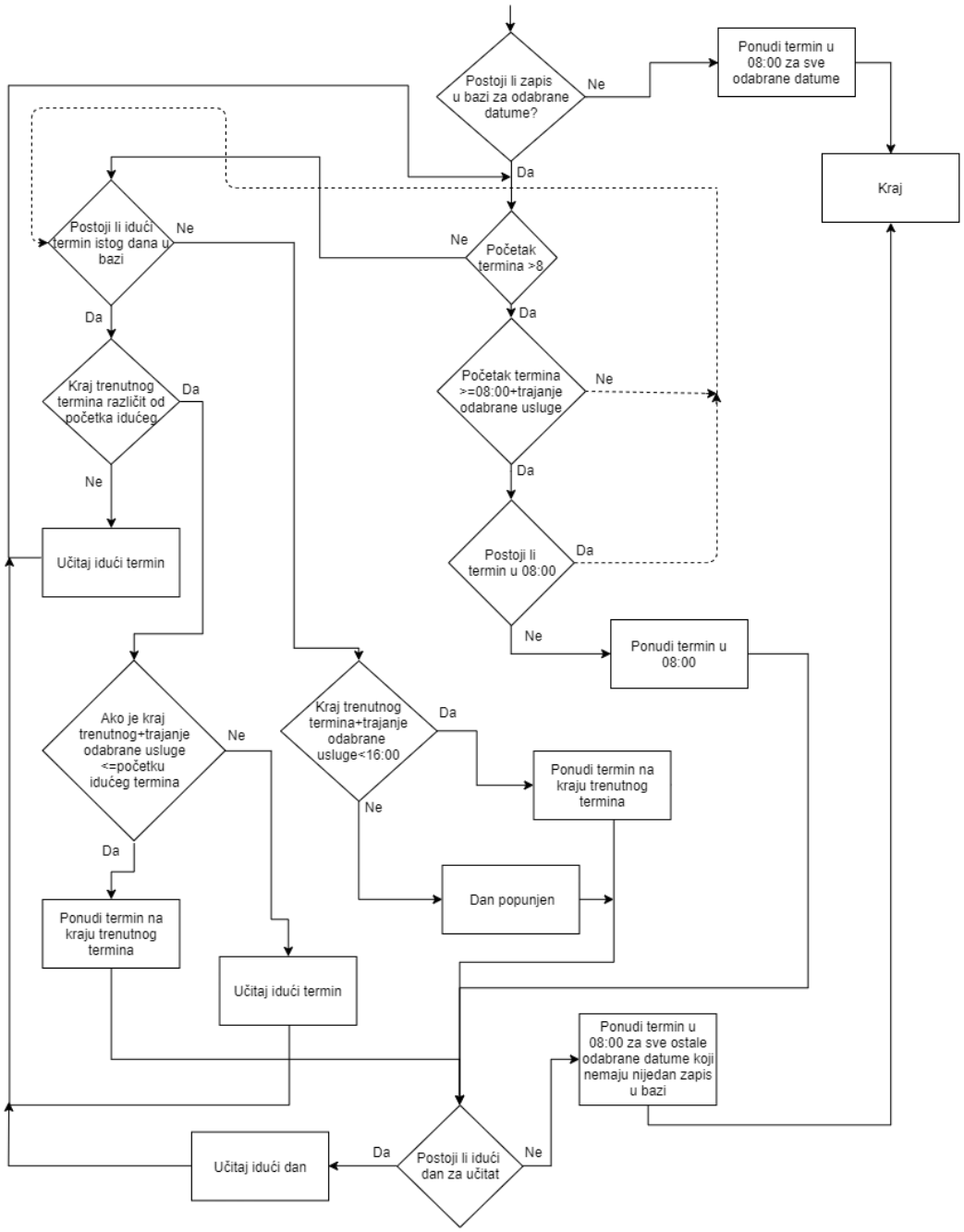
Visual Studio Code je uređivač izvornog koda razvijen od strane Microsoft-a dostupan za Windows, macOS i Linux. Posjeduje ugrađenu podršku za JavaScript, TypeScript i Node.js te ima opširan ekosustav proširenja za druge jezike (C ++, C #, Java, Python, PHP). Sadrži podršku za uklanjanje pogrešaka, ugrađenu Git kontrolu i GitHub, isticanje sintakse, inteligentno dovršavanje i preuređivanje koda. Vrlo je lako prilagodljiv, omogućuje korisnicima da promijene temu, prečace na tipkovnici, postavke i dodaju potrebna proširenja koja omogućuju dodatnu funkcionalnost. Izvorni kod je besplatan i otvorenog tipa. Na slici 2.3. prikazan je Visual Studio Code uređivač koda. Podaci su navedeni prema literaturi [8].



Sl. 2.3. Visual Studio Code.

3. PROGRAMSKO RJEŠENJE

Kako bi dodali automatsko zauzeće termina u aplikaciju, potrebno je osmisliti i napraviti algoritam koji će što učinkovitije raspodijeliti korisnike za korištenje usluga u slobodnim terminima. Na slici 2.4. prikazan je dijagram toka algoritma za raspoređivanje slobodnih termina.



Sl. 3.1. Blok dijagram algoritma za raspoređivanje slobodnih termina.

Dijagramom toka simbolički je predstavljen algoritam. Sastoji se od niza simbola povezanih strelicama koji definiraju tok i smjer realizacije programa. Ovakav prikaz algoritma je jednostavan, pregledan, te omogućuje lako pronalaženje grešaka. Problemi se mogu jednostavno analizirati, usporediti s nekim drugim problemom, što skraćuje vrijeme pronalaska rješenja.

Da bi se slobodni termini mogli rasporediti, potrebno je spremanje i pregledavanje u/iz baze podataka. Algoritam započinje pregledavanjem tablice u koju se spremaju svi zauzeti termini. Kada korisnik odabere raspon datuma u kojima bi htio rezervirati termin za odabrani pregled, prvenstveno slijedi provjera jesu li odabrani datumi valjani. Korisnik za početni datum može postaviti najranije sutrašnji dan, dok krajnji datum nije ograničen. Na slici 3.2. prikazan je dio koda zadužen za provjeru valjanosti odabranog raspona.

```
1.  if($this->validateDatesAndReturnWeekDays($from, $to) != false){
2.              $possibleDatesUserSelected = $this-
    >validateDatesAndReturnWeekDays($from, $to);
3.          }
4.          else {
5.              return Redirect::back()-
    >withErrors(['Odabran raspon datuma nije valjan, molimo unesite ponovno!']);
6.          }
```

Sl. 3.2. Programski kod provjere valjanosti unesenog raspona.

Nadalje slijedi provjera postoje li već zapisi u tablici za odabrane dane. Ako ne postoji niti jedan zapis za odabrani raspon dana, korisniku se nude termini na početku dana u 08:00 sati za svaki odabrani radni dan. Na slici 3.3. prikazan je dio koda zadužen za ponudu termina u slučaju prazne baze podataka.

```
1.
2.          if($takenAppointmentsForSelectedDates->isEmpty()){
3.              foreach($possibleDatesUserSelected as $possibleDate){
4.                  $possibleDates[] = $possibleDate->addHour(8);
5.              }
```

Sl. 3.3. Programski kod ponude termina u slučaju prazne baze podataka.

U slučaju da postoji zapis u bazi za odabrane datume, slijedi prva provjera, gdje se provjerava počinje li prvi termin u bazi za taj dan u 08:00 sati. Ako je početak prvog termina veći od 08:00, to znači da postoji praznih termina prije prvog termina u danu i slijedi provjera može li se ponuditi tada termin za željenu uslugu. Kako bi se mogao ponuditi termin, potrebno je provjeriti stane li usluga u prazan prostor do prvog termina, tj. je li zbroj 08:00 sati i trajanje odabrane usluge manji ili jednak početku prve usluge u bazi. Ako je uvjet ispunjen, slijedi još jedna provjera kako bi se utvrdilo postoji li već termin u 08:00, budući da se može dogoditi da je

prazan prostor negdje između prvog termina i trenutnog termina. Ako ne postoji termin u 08:00, korisniku se nudi prvi termin u danu, tj. 08:00 sati za trenutni dan koji se provjerava. Na slici 3.4. prikazan je opisani dio koda.

```
1.  foreach($takenAppointmentsForSelectedDates as $eachDay) {
2.              for($i=0; $i <= count($eachDay); $i++) {
3.                  if($this->start8($eachDay[$i]->date, $eachDay[$i]->start)){
4.                      if($this->start8LtePatientDuration($eachDay[$i]>date,
5.  $eachDay[$i]->start, $requestedDuration)){
6.                          if($this->appIn8($eachDay)){
7.                              goto COMPARE;
8.                          }
9.                          else {
10.                           $possibleDates[] = Carbon::parse($eachDay[$i]->date)-
    >addHour("8");
11.                             goto END;
12.                         }
13.                     }
14.                     else {
15.                         goto COMPARE;
16.                     }
```

Sl. 3.4. Programski kod provjere praznog prostora za termin na početku dana.

U slučaju da ovi uvjeti nisu ispunjeni, a ne postoji praznog prostora između trenutnog termina koji se provjerava i prethodnih u bazi, slijedi provjera postoji li idući termin za uspoređivanje u bazi istog tog dana. Provjera u kodu se odvija kod naznake „COMPARE“. Ako ne postoji idući termin za uspoređivanje, provjerava se stane li odabrana usluga na kraj trenutne, odnosno je li zbroj kraja zadnjeg termina i trajanje usluge manji ili jednak 16:00 sati, budući da je tada kraj radnog vremena. Ako je uvjet ispunjen, korisniku se nudi sljedeći termin, a ako nije, znači da je dan popunjen do kraja te se prelazi na novi datum. U slučaju da postoji idući termin u danu za usporedbu, provjerava se postoji li slobodnog prostora između trenutnog termina te sljedećeg koji se nalazi u tablici. To se izvršava uspoređivanjem kraja trenutnog termina te početka idućeg. Ako vrijednosti nisu jednake, znači da termini ne slijede jedan iza drugoga te postoji praznog prostora. No to nije dovoljno kako bi mogli ponuditi termin između njih. Također uz prethodni uvjet, potrebno je provjeriti stane li odabrana usluga u taj prazan prostor, odnosno je li zbroj kraja trenutnog termina te zatražene usluge manji ili jednak početku sljedećeg termina. Ako je uvjet ispunjen, znači da usluga stane u taj prazan prostor te se korisniku nudi taj termin. Ne ispuni li se ovaj uvjet, znači da nema dovoljno slobodnog vremena između termina. Na slici 3.5. prikazan je opisani dio koda.

```
1.  else{
2.                      COMPARE:
3.                      if(isset($eachDay[$i+1])) {
4.                          if($this->currentEndNeNextStartDifferent($eachDay[$i]-
    >date, $eachDay[$i]->end, $eachDay[$i+1]->date, $eachDay[$i+1]->start)){
5.                              if($this->currentEndPlusDurationLteNextStart($eachDay[$i]-
    >date, $eachDay[$i]->end, $eachDay[$i+1]->date, $eachDay[$i+1]-
    >start, $requestedDuration)){
6.                               $possibleDates[] = Carbon::parse($eachDay[$i]>date.
7.  $eachDay[$i]->end);
8.                                  goto END;
9.                              }
10.                             else {
11.                                 goto END;
12.                             }
13.
14.                         }
15.                         else{
16.                             goto NEXT;
17.                         }
```

Sl. 3.5. Programski kod provjere praznog prostora između trenutnog i sljedećeg termina.

```
18.                     else {
19.                         if($this->isExceedingClosingHours($eachDay[$i]-
    >date, $eachDay[$i]->end, $requestedDuration)){
20.                             goto END;
21.                         }
22.                         else{
23.                             $possibleDates[] = Carbon::parse($eachDay[$i]-
    >date . $eachDay[$i]->end);
24.                             goto END;
25.                         }
26.
27.                     }
28.                 }
29.                 NEXT:
30.             }
31.             END:
32.         }
```

Sl. 3.6. Programski kod provjere popunjenosti dana do 16:00 sati.

Budući da su sve mogućnosti provjerene, učitava se sljedeći dan iz tablice. Učitavanjem sljedećeg dana, slijedi ponovna provjera svih ovih uvjeta za sve postojeće termine toga dana. Kada se ponude termini za one dane koji postoje u tablici, na kraju svega također nudimo korisniku i termine za ostale dane gdje nema niti jednog zapisa u tablici. Za one dane za koje ne postoji niti jedan zapis nude se termini u 08:00 sati. Na slici 3.7. prikazan je programski kod zadužen za tu zadaću.

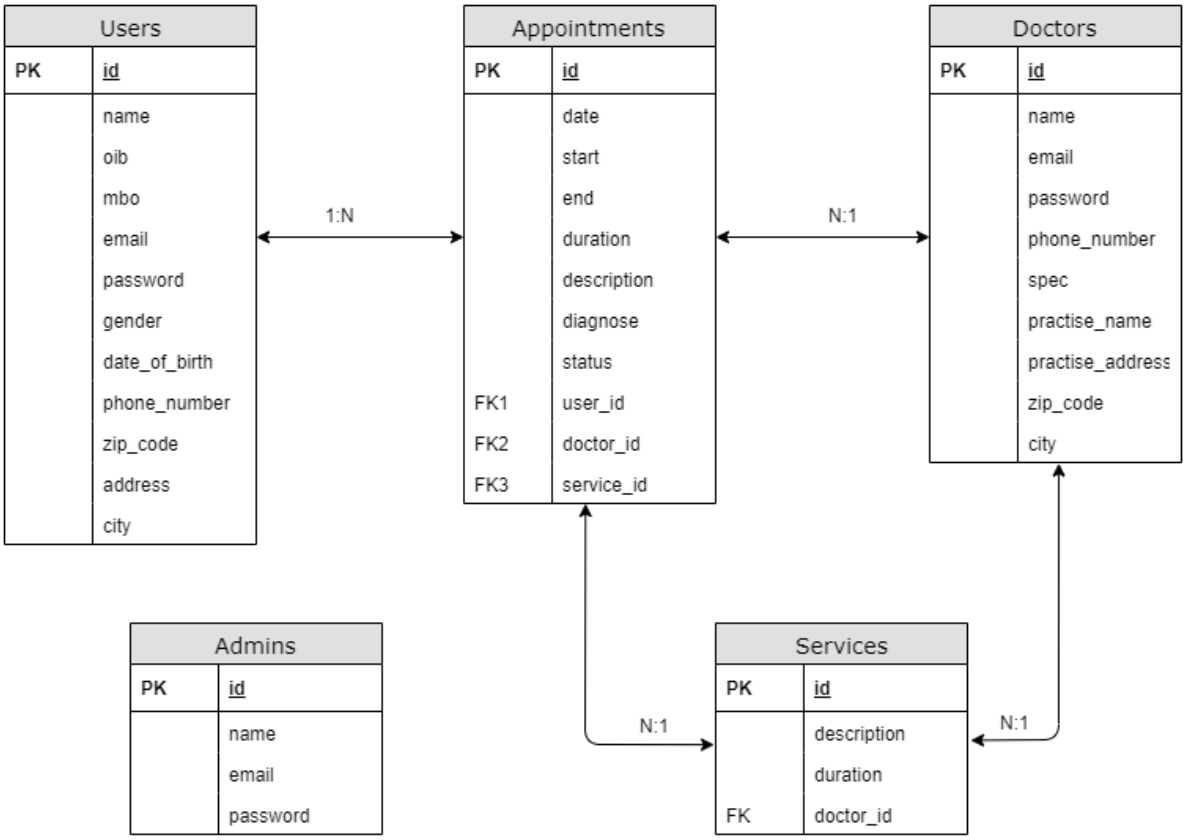
```
1.          foreach($this-
    >allDatesWithoutAppointments($possibleDatesUserSelected, $takenAppointmentsForSelectedD
    ates) as $datesWithoutAppointments) {
2.              $possibleDates[] = $datesWithoutAppointments->addHour("8");
3.          }
```

Sl. 3.7. Programski kod ponude termina za dane bez zapisa u bazi.

4. REALIZACIJA APLIKACIJE

4.1. Baza podataka

Budući da se radi o rezerviranju termina, te da ti termini negdje moraju biti spremljeni, potrebno je kreirati odgovarajuću bazu podataka koja će zadovoljiti uvjete internetske aplikacije. Baza će omogućiti pristup, upravljanje te uređivanje svim podacima koji su neophodni za uspješan rad aplikacije. Sastoji se od tablica s određenim atributima, koje se međusobno povezuju te stvaraju jednu cjelinu. Takve baze podataka nazivaju se relacijskim bazama, a sastoje se od tri vrste veza. To su 1:1 (jedan prema jedan - jedan zapis iz tablice vezan samo uz jedan zapis u drugoj tablici), 1:N (jedan prema više, jedan zapis u jednoj tablici može se odnositi na jedan ili više zapisa u drugoj tablici) te N:N (više prema više, jedan ili više entiteta iz jedne tablice može biti povezano s jednim ili više entiteta u drugoj tablici). Na slici 4.1. prikazan je izgled baze podataka koja se koristi pri izradi ove internetske aplikacije.



Sl. 4.1. Shema baze podataka korištene za izradu aplikacije.

U navedenoj bazi nalaze se sljedeće tablice: Users koja predstavlja korisnike (pacijente) aplikacije, Doctors koja predstavlja liječnike, Appointments u kojoj se nalaze zakazani termini korisnika kod pojedinog liječnika, Admins u kojoj se nalaze podaci o administratoru aplikacije te Services koja predstavlja sve moguće usluge koje pojedini liječnici nude. User i Appointments tablice povezane su vezom 1:N, što znači da jedan korisnik može zabilježiti više termina, dok se jedan termin odnosi samo na jednog korisnika (pacijenta). Nadalje tablice Appointments i Doctor su povezane vezom N:1, što označava da se jedan termin može odnositi samo na jednog liječnika, dok jedan liječnik može imati više zakazanih termina. Tablica Appointments i Services povezane su vezom N:1, budući da jedan termin može imati samo jednu uslugu, dok jedna ista usluga može pripadati više različitih termina. Uz sve to, tablice Doctors i Services povezane su N:1 vezom, što označava da jedan liječnik može nuditi više usluga, dok jedna usluga pripada samo jednom liječniku. Tablica Admins nije povezana s ostalim tablicama, već ima zasebnu ulogu.

Da bi uspješno kreirali bazu podataka, u Laravelu se koriste migracije. One predstavljaju sheme za jednostavno stvaranje i uređivanje tablica u bazi podataka. Upisivanjem naredbe „php artisan make:migration naziv_tablice“ u projektnim datotekama se pojavljuje migracija koja odgovara našim zadanim zahtjevima. Na slici 4.2. prikazana je migracija za tablicu gdje će se spremati informacije o zakazanom terminu. Migraciju je potrebno migrirati upisivanjem naredbe „php artisan migrate“ kako bi se tablica kreirala u bazi podataka.

```
1.  public function up()
2.      {
3.          Schema::create('appointments', function (Blueprint $table) {
4.              $table->bigIncrements('id');
5.              $table->integer('user_id');
6.              $table->integer('doctor_id');
7.              $table->integer('service_id');
8.              $table->date('date');
9.              $table->char('start');
10.             $table->char('end');
11.             $table->integer('duration');
12.             $table->string('description');
13.             $table->string('diagnose')->nullable();
14.         });
```

Sl. 4.2. Programski kod migracije tablice „Appointments“.

Za sve ostale tablice, migracije se izvode identično, nakon što su popunjene sve informacije o podacima koji su nam potrebni za pojedinu tablicu.

4.2. Modeli

Svaka pojedina tablica u bazi podataka ima svoj odgovarajući model koji se koristi za interakciju s tom tablicom. Modelima se omogućuje jednostavnije pretraživanje, brisanje, unošenje te uređivanje podataka u tablicama. Pojednostavljuju postupak sinkronizacije s više baza podataka koji se izvode na različitim sustavima. U ovoj internetskoj aplikaciji postoji pet različitih modela. To su „User“ – predstavlja model korisnika, „Doctor“ – predstavlja model liječnika, „Service“ – predstavlja model usluge, „Admin“ – predstavlja model administratora i „Appointment“ – predstavlja model termina. Novi model kreira se unošenjem naredbe „php artisan make:model naziv_modela“ u terminal. Nakon što su svi modeli kreirani, potrebno je odrediti relacije u njima kako bi se aplikacija ponašala onako kako je zamišljeno i omogućila nam pristup svim potrebnim podacima iz baze podataka. Na slici 4.3. prikazan je Appointment model koji se odnosi na zauzeti termin u bazi podataka.

```
1.  class Appointment extends Model
2.  {
3.      public function doctor() {
4.          return $this->belongsTo('App\Doctor');
5.      }
6.
7.      public function user() {
8.          return $this->belongsTo('App\User');
9.      }
10.
11.     public function service() {
12.         return $this->belongsTo('App\Service);
13.     }
14. }
```

Sl. 4.3. Programski kod modela „Appointment“.

4.3. Rute

Kako bi pristupili određenoj stranici aplikacije (tj. pogledu), potrebno je definirati rute (engl. Routes). Rute su definirane u datoteci „routes/web.php“, a pristupa im se unosom definiranog URL-a rute u internetskom pregledniku. Npr. prijavi korisnika moguće je pristupiti unosom „http://dipl.local/login“. Rute služe kao poveznice između upravitelja i definiranog URL-a. Na slici 4.4. prikazan je izgled programskog koda za rutu prijave.

```
1.  Route::get('/login', 'SessionsController@create')->name('login');
```

Sl. 4.4. Programski kod rute za prijavu korisnika.

4.4. Upravitelji

Kada se određeni pogled učitava, programu nije dovoljno samo znati URL adresu koja je definirana u rutama. Kao posrednik tome djeluje upravitelj (engl. Controller). Posjećivanjem određenog URL-a, program pokreće definiranu metodu u zadanom upravitelju, koji sadržava svu aplikacijsku logiku te ju odvaja od prezentacijske. U slučaju posjećivanja URL-a za prijavu korisnika, pokreće se „create()“ metoda koja je definirana u „SessionController“-u. Upravitelj se kreira unosom naredbe „php artisan make:controller naziv_upravitelja“ u terminal. Za potrebe ove aplikacije kreirani su upravitelji „Controller“, „AdminController“, „DoctorController“, „RegistrationController“, „SessionController“ te „UserController“. Logiku aplikacije moguće je smjestiti u samo jedan upravitelj, no da bi stvar bila modularnija i preglednija, korišteno je više upravitelja, svaki za svoju svrhu. Svi upravitelji pohranjeni su u „app/Http/Controllers“ datoteci projekta. Slika 4.5. prikazuje metodu „store()“ u „SessionController“-u koja služi za prijavu korisnika u sustav, dok je na slici 4.6. prikazana metoda „scheduledAppStore()“ u „Controlleru“-u koja služi za spremanje termina u bazu podataka.

```
1.  public function store() {
2.          if (! auth()->attempt(request(['mbo','password']))){
3.              return back();
4.          }
5.
6.          $user = Auth::user();
7.          return view('welcome', compact('user'));
8.      }
```

Sl. 4.5. Programski kod metode „store()“ za prijavu korisnika u „SessionController“-u.

```
1.  public function scheduledAppStore(){
2.          $user = Auth::user();
3.          $appointment = new Appointment;
4.          $appointment->user_id = request('user_id');
5.          $appointment->doctor_id = request('doctor_id');
6.          $appointment->service_id = request('service_id');
7.
8.          $service_duration = DB::table('services')->select('duration')
    >where('id', $appointment->service_id)->first();
9.
10.         $input = Carbon::parse(request('date_and_time'));
11.         $appointment->date = Carbon::parse($input)->toDateString();
12.         $appointment->start = Carbon::parse($input)->format('H:i');
13.         if($service_duration->duration == 1){
14.             $temp = Carbon::parse(request('date_and_time'))->addMinute(30);
15.         }
16.         if($service_duration->duration == 2){
17.             $temp = Carbon::parse(request('date_and_time'))->addMinute(60);
18.         }
19.
20.         $appointment->end = Carbon::parse($temp)->format('H:i');
21.         $appointment->duration = $service_duration->duration;
22.         $appointment->description = request('description');
```

```
23.
24.         $appointment->save();
25.         session()->flash('message', 'Termin uspješno rezerviran!');
26.         return view('welcome', compact('user'));
27.     }
```

Sl. 4.6. Programski kod metode „scheduledAppStore()“ za spremanje termina u „Controller“-u.

4.5. Pogledi

Kako bi upravitelji i modeli izvršili svoju funkcionalnost, potrebno je dodati zadnji element MVC arhitekture, a to su pogledi (engl. Views). Pogledi se kreiraju dodavanjem novih datoteka s nastavkom „.blade.php“ u direktorij „resources/views“. Upravitelj posjeduje mogućnost slanja podataka pogledu putem metode compact(), a upravitelju je također moguće proslijediti podatke iz forme koja se nalazi u pogledu putem zahtjeva. Poglede je također moguće oblikovati, a u ovom projektu oblikovanje je odrađeno dodavanjem razmještaja (eng. Layout). u poglede. Na početku datoteke pogleda potrebno ih je uključiti, a to se izvodi dodavanjem naredbe „@extends('naziv_razmještaja')“, te se time na pogledu prikazuju dijelovi stranice koji će se ponavljati na više različitih mjesta. Koristi se u svrhu izbjegavanja nepotrebnog ponavljanja programskog koda. Primjer dijela razmještaja je navigacijska traka koja se pojavljuje na svakoj stranici. U pogledima je moguće koristiti PHP funkcije i petlje. To se postiže dodavanjem @ znaka ispred određene funkcije (npr. @if @foreach). Varijabla „$appointment“ šalje se pomoću metode compact() u upravitelju. Nakon toga se u pogledu pomoću „@foreach“ funkcije prolazi kroz kolekciju koja je predefinirana u upravitelju te sadrži sve podatke o rezerviranim terminima. Za ispis podataka o pojedinom terminu, moguće je pristupiti atributima pojedinog termina. To se postiže korištenjem „{{$appointment->date}}“ (pristup datumu termina). Na slici 4.7. prikazan je pogled u kojem se prikazuje profil korisnika sa svim rezerviranim terminima.

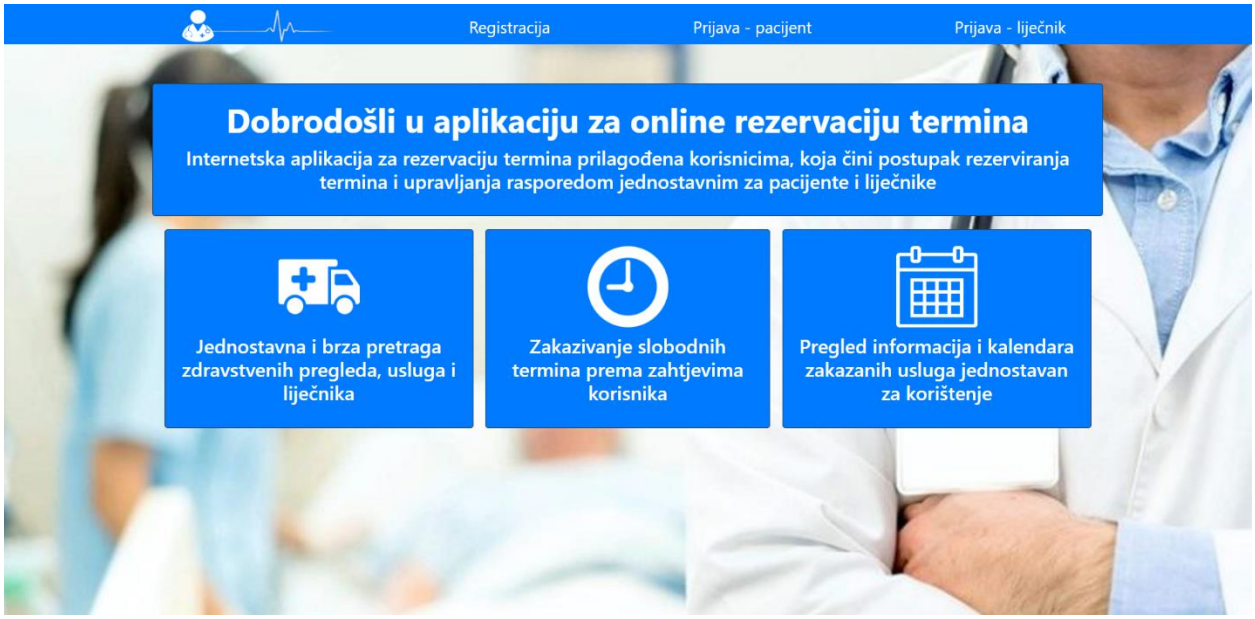
```
1.  @extends ('layouts.master')
2.
3.  @section('content')
4.
5.      <script src="//cdnjs.cloudflare.com/ajax/libs/jquery/2.1.3/jquery.min.js"></script>

6.      <script src="//cdnjs.cloudflare.com/ajax/libs/moment.js/2.9.0/moment.min.js"></scri
    pt>
7.      <script src="//cdnjs.cloudflare.com/ajax/libs/fullcalendar/2.2.7/fullcalendar.min.j
    s"></script>
8.      <link rel="stylesheet" href="//cdnjs.cloudflare.com/ajax/libs/fullcalendar/2.2.7/fu
    llcalendar.min.css"/>
9.      <script src="//cdnjs.cloudflare.com/ajax/libs/fullcalendar/2.2.7/lang-
    all.js"></script>
10.
11.     <div stlye="background-color: #A0A8A0;">
```

```
12.     {!! $calendar->calendar() !!}
13.     {!! $calendar->script() !!}
14.     </div>
15.     <br>
16.
17.     @if(count($appointment)!=0)
18. <h1>Moji termini</h1>
19. <div class="card-deck">
20.
21.     @foreach ($appointment as $termin)
22.     <form method="post" action="">
23.                 {{ csrf_field() }}
24.             <div style="max-width: 18rem;" class="{{Carbon\Carbon::now()-
    >gt(Carbon\Carbon::parse($termin->date)) ? 'border-dark card text-white bg-success mb-
    3' : 'border-dark card text-white bg-info mb-3'}}">
25.             <div class="card-header">{{$termin->service->description}}</div>
26.             <div class="card-body">
27.                 <p class="card-text">Liječnik: {{$termin->doctor->name}}</p>
28.                 <p class="card-text">Datum: {{$termin->date}}</p>
29.                 <p class="card-text">Početak usluge: {{$termin->start}}h</p>
30.                 <p class="card-text">Kraj usluge: {{$termin->end}}h</p>
31.                 <p class="card-text">Opis poteškoća: {{$termin->description}}</p>
32.                 <p class="card-text">Dijagnoza: {{$termin->diagnose}}</p>
33.                 <p class="card-text">Email: {{$termin->doctor->email}}</p>
34.                 <p class="card-text">Broj telefona: {{$termin->doctor-
    >phone_number}}</p>
35.                 <p class="card-text">Naziv ordinacije: {{$termin->doctor-
    >practise_name}}</p>
36.                 <p class="card-text">Adresa ordinacije: {{$termin->doctor-
    >practise_address}}</p>
37.             </div>
38.                     <input type="hidden" name="appointment_id" id="appointment_id" valu
    e="{{$termin->id}}">
39.                     <button type="submit" class="btn btn-
    primary" formaction="/appointment_delete">Obriši termin</button>
40.             </div>
41.         </form>
42.
43.     @endforeach
44.     </div>
45.     @else
46.         <h4>Moji termini</h4>
47.             <h6>Još nema zakazanih termina</h6>
48.     @endif
49.
50. @endsection
```

Sl. 4.7. Programski kod pogleda za prikaz profila korisnika s rezerviranim terminima.

4.6. Opis rada aplikacije

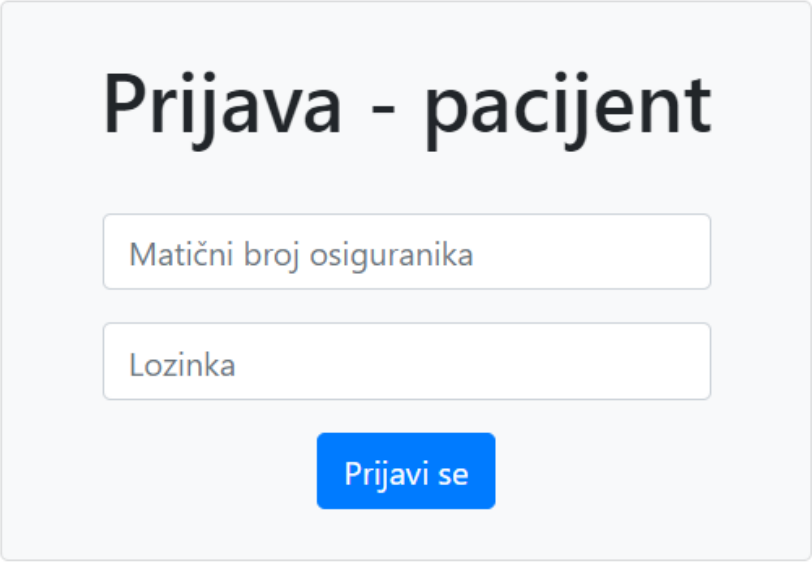
Prilikom otvaranja početne stranice, prikazuju se osnovne informacije u internetskoj aplikaciji koja služi za rezervaciju termina. Koncept stranice je zamišljen tako da korisnik odabire zavod u kojem želi zatražiti pregled te željeni grad. Nakon toga korisniku se prikažu svi dostupni liječnici, te odabirom pojedinog rezervira jedan od termina koji su slobodni a koji mu najviše odgovara. Na slici 4.8. prikazana je početna stranica internetske aplikacije.



Sl. 4.8. Početna stranica internet aplikacije.

Želi li korisnik rezervirati termin, potrebno je prijaviti se u sustav sa svojim korisničkim podacima. Na svakoj otvorenoj stranici u gornjem dijelu nalazi se navigacijska traka koja se mijenja u ovisnosti o tome koji je korisnik prijavljen.

Prijava pacijenata se izvršava odabirom „Prijava – pacijent“. Pacijent se prijavljuje unošenjem matičnog broja osiguranika (MBO) te lozinkom. Na slici 4.9. prikazan je izgled forme koja se koristi za prijavu korisnika.

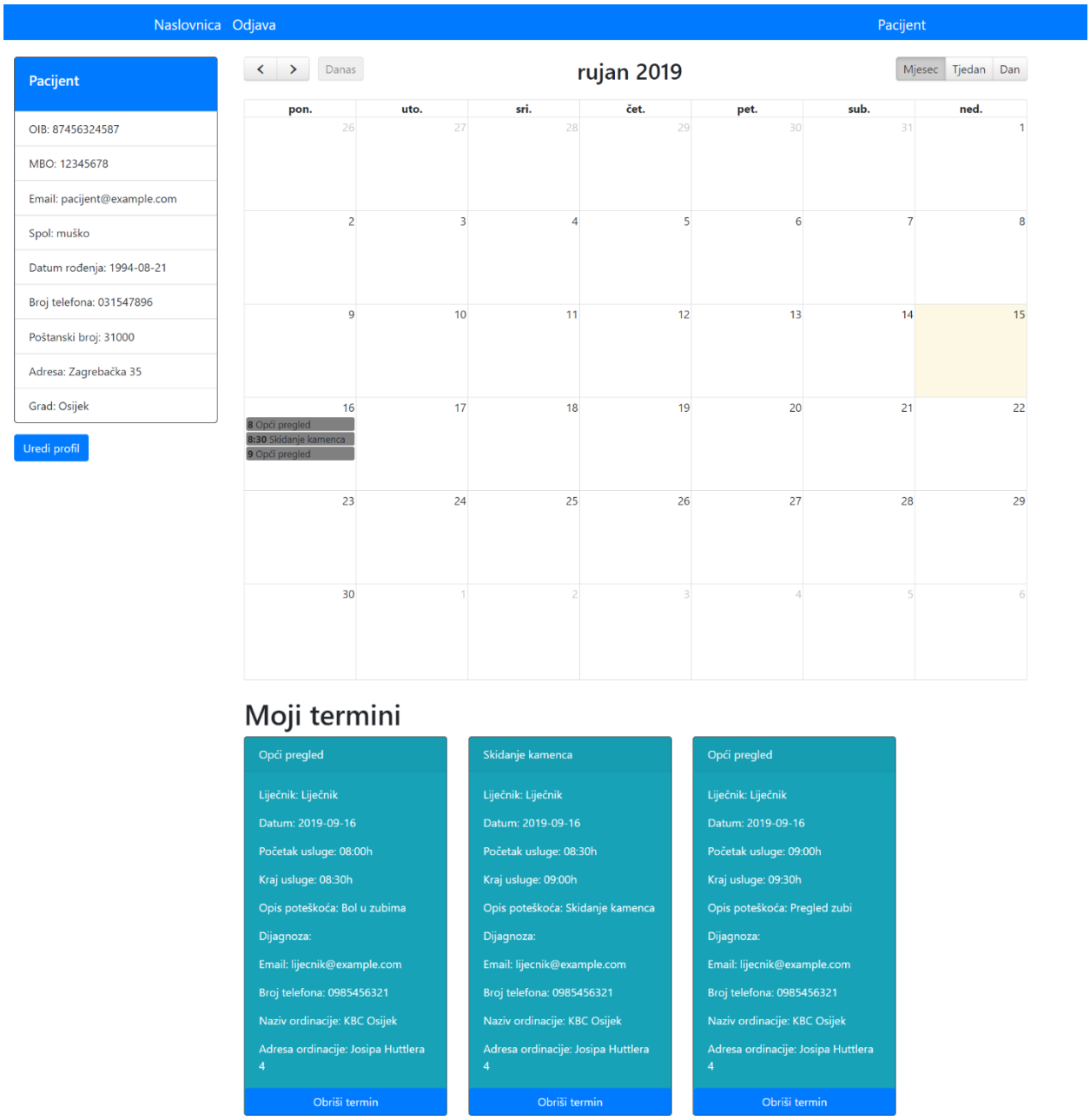


Sl. 4.9. Forma za prijavu pacijenata.

U slučaju da korisnik nema svoj račun, potrebno je izvršiti registraciju. Forma prikazana na slici 4.10. prikazuje se korisniku odabirom registracije.

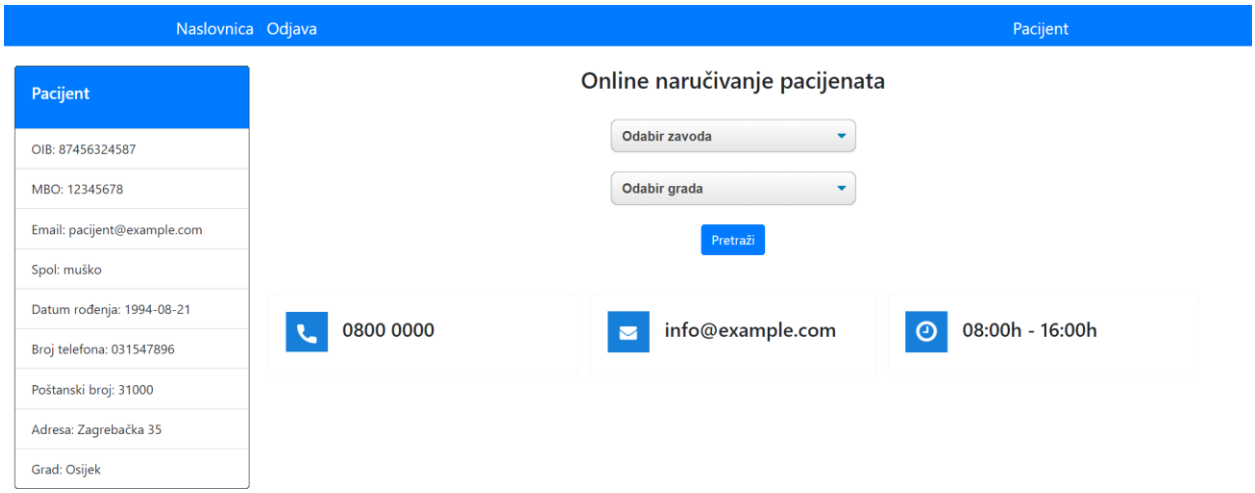
Sl. 4.10. Forma za registraciju korisnika.

Kod registracije je potrebno unijeti sve tražene podatke, kao što su ime i prezime, osobni identifikacijski broj (OIB), matični broj osiguranika (MBO), email, lozinku, potvrdu lozinke, spol, datum rođenja, broj telefona, poštanski broj, adresu te grad. Nakon uspješne prijave (ili registracije novog korisnika) slijedi preusmjeravanje na početnu stranicu koja se učitava kada je korisnik prijavljen. Na početnoj stranici s lijeve strane prikazana je bočna traka u kojem se nalaze podaci o trenutno prijavljenom korisniku. Bočnu traku s informacijama o korisniku moguće je vidjeti na svim stranicama aplikacije. U navigacijskoj traci nalaze se opcije za povratak na naslovnu stranu, odjavu korisnika, te profil korisnika. Odabirom profila korisnika učitava se stranica na kojoj su prikazani svi rezervirani termini za prijavljenog korisnika. Ovdje je moguće vidjeti kalendarski prikaz rasporeda termina. Moguć je odabir pregleda termina po mjesecu, tjednu ili danu. Kalendar je uvijek postavljen tako da se učitavaju podaci za tekući mjesec. Ispod kalendarskog prikaza su izlistani svi termini korisnika u kartičnom obliku, gdje je moguće vidjeti pojedinosti termina, kao što su datum, početak termina, završetak termina, ime liječnika kod kojeg je termin zakazan itd.. Također, na profilu korisnika moguće je odraditi izmjenu podataka odabirom na opciju „Uredi profil“, koja se pojavljuje samo na profilnoj stranici korisnika. Na slici 4.11. prikazan je izgled profila korisnika.



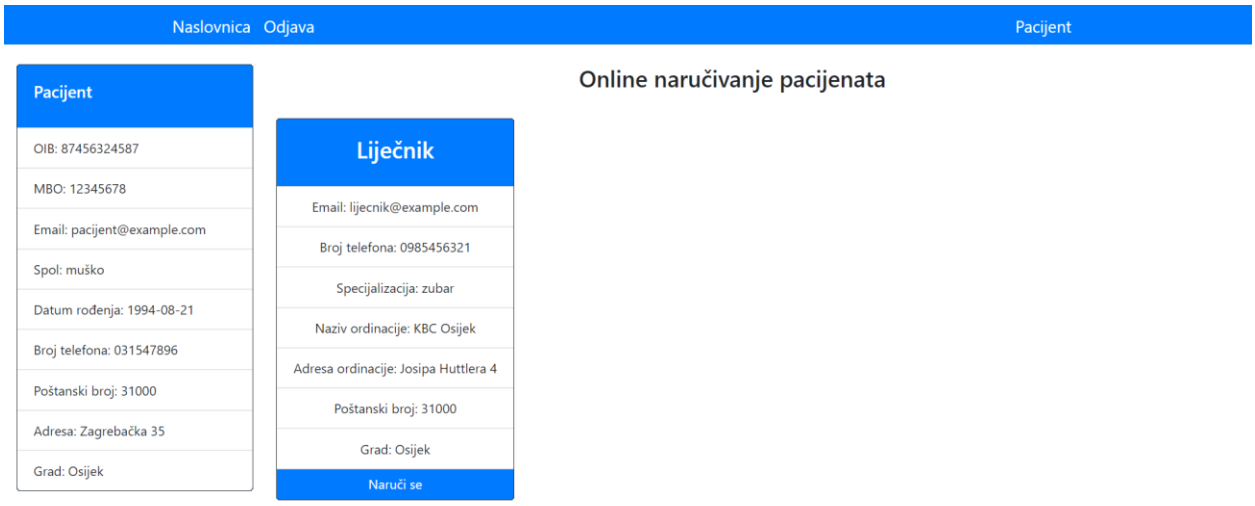
Sl. 4.11. Profil korisnika.

Na početnoj stranici kada je pacijent prijavljen nalazi se glavni dio aplikacije, a to je pretraživanje zavoda u kojem se želi zakazati pregled (usluga). Prvo što korisnik odabire su zavod i grad u kojem želi rezervirati termin. Za uspješan prelazak na sljedeći korak pri rezervaciji potrebno je popuniti oba podatka, u suprotnom korisnik se obavještava o pogrešci. U slučaju da korisnikov odabir ne pronalazi niti jedan rezultat, korisnik o tome prima povratnu informaciju, te mu se nudi povratak na početnu stranicu. Na slici 4.12. prikazan je prvi korak rezervacije



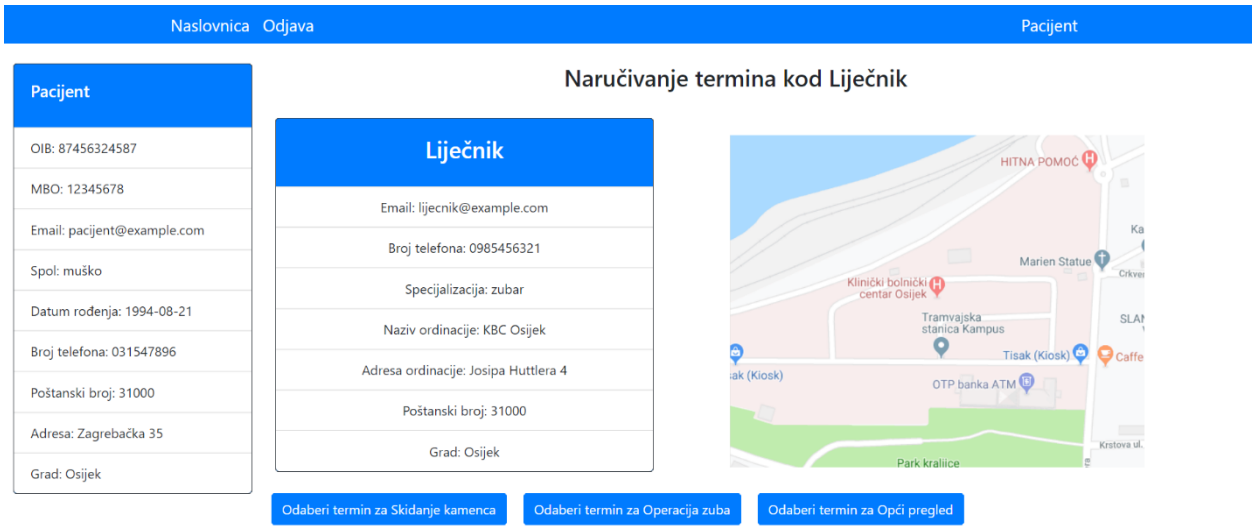
Sl. 4.12. Odabir zavoda i grada za pretragu usluge.

Kada je korisnik odabrao zavod i grad, slijedi idući korak. Na sljedećoj stranici prikazuju se svi pronađeni liječnici koji odgovaraju unesenim parametrima, te njihove osnovne informacije (ime i prezime, adresa, email, broj telefona, adresa ordinacije itd.). Korisnik odabire onog liječnika koji najviše odgovara njegovim zahtjevima. Slika 4.13. prikazuje rezultate pretrage.



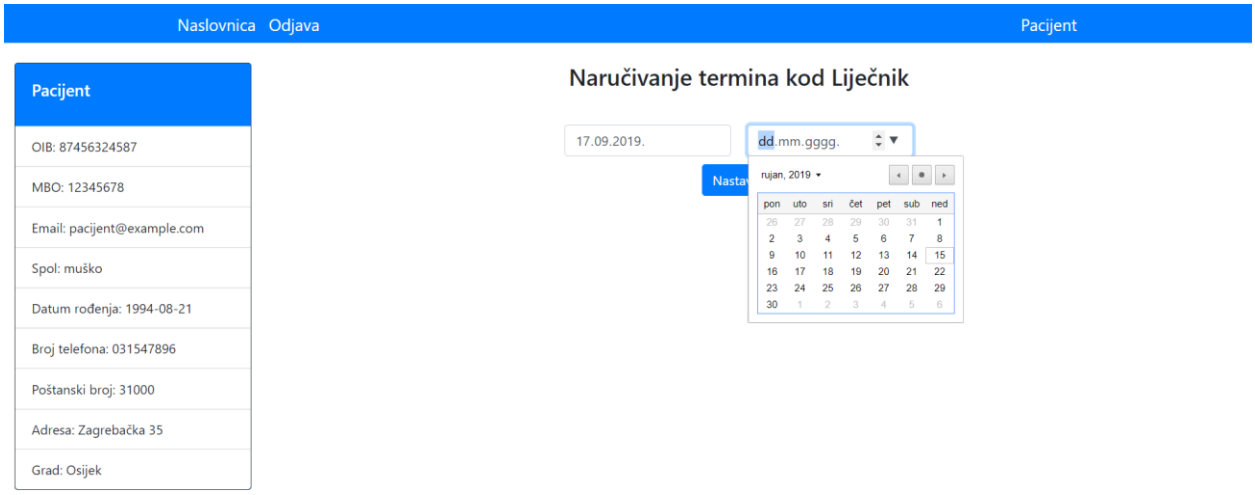
Sl. 4.13. Rezultati pretrage korisnika.

Nakon što je korisnik odabrao liječnika kod kojega želi zatražiti uslugu, slijedi odabir same usluge. Na sljedećoj stranici slijedi ispis svih usluga koje odabrani liječnik nudi. Korisnik odabire uslugu za koju želi rezervirati termin. Također, u ovom koraku rezervacije prikazana je i karta s točnom lokacijom gdje se nalazi ordinacija u kojoj se odrađuje pregled.



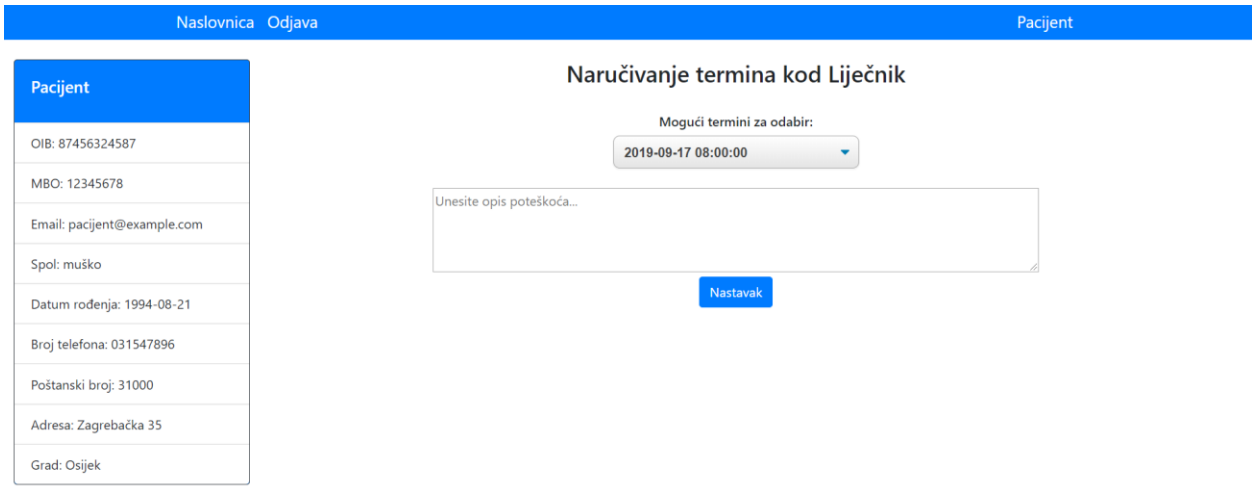
Sl. 4.14. Prikaz informacija o odabranom liječniku i ponuđenim uslugama.

U idućem koraku potrebno je odabrati raspon datuma za koje korisnik želi vidjeti slobodne termine, te po svojoj volji odabrati onaj koji mu najbolje odgovara. Početni dan raspona mora biti za jedan dan veći od tekućeg dana, budući da nije moguće rezervirati termin u tekućem danu, već najranije za idući dan koji slijedi. U slučaju da se ne ispuni taj uvjet, ponovno se učitava ista stranica te se korisnik obavještava o grešci koju je napravio.



Sl. 4.15. Odabir raspona datuma za zatraženu uslugu.

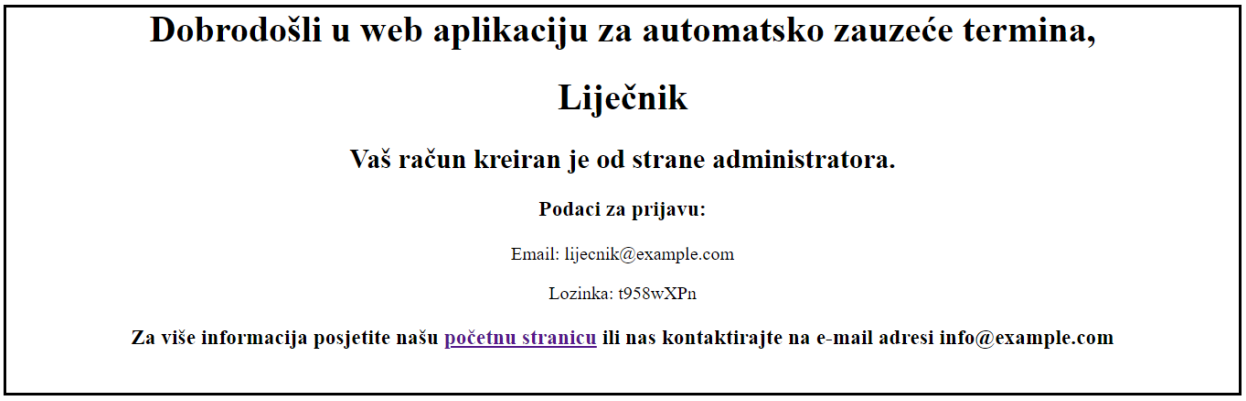
Nakon odabira valjanog raspona datuma, algoritam izvršava svoj zadatak te vraća korisniku datume i vremena za koje je moguće zabilježiti termin u obliku padajućeg izbornika. Na slici 4.16. nalazi se prikaz ponuđenih termina.



Sl. 4.16. Prikaz ponuđenih termina.

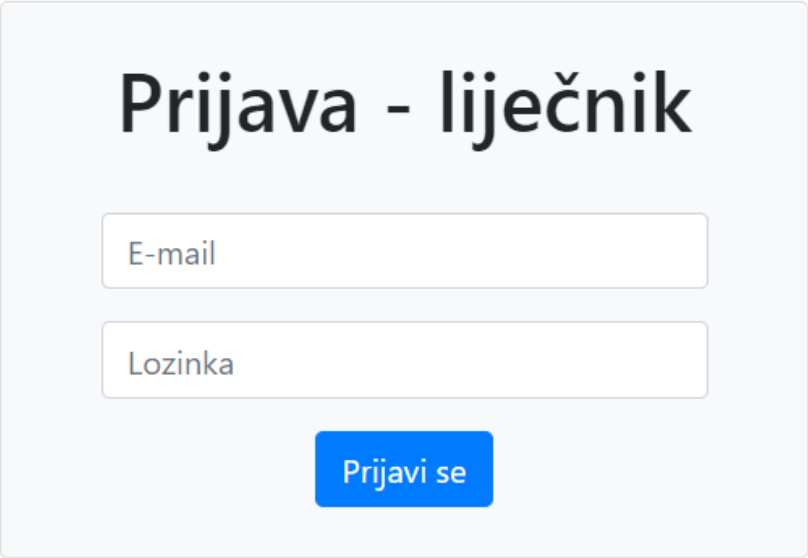
Uz odabir slobodnog termina potrebno je upisati i opis poteškoća pacijenta. Nakon što korisnik rezervira termin, prikazuje mu se informacija o uspješnoj rezervaciji, te ga se vraća na početnu stranicu aplikacije.

Osim mogućnosti prijave pacijenata, postoji mogućnost prijave i za liječnike koji nude svoje usluge. Liječnik nije u mogućnosti sam izvršiti registraciju, već liječnika u sustav dodaje administrator aplikacije. Kada administrator unese novog liječnika u sustav, liječnik na mail adresu prima poruku dobrodošlice s korisničkim imenom i lozinkom kojom se prijavljuje. Lozinka se nasumično generira pri kreiranju liječnika, te samo kreirani liječnik ima uvid u lozinku. Na slici 4.17. prikazan je izgled e-maila koji se šalje pri registraciji liječnika.



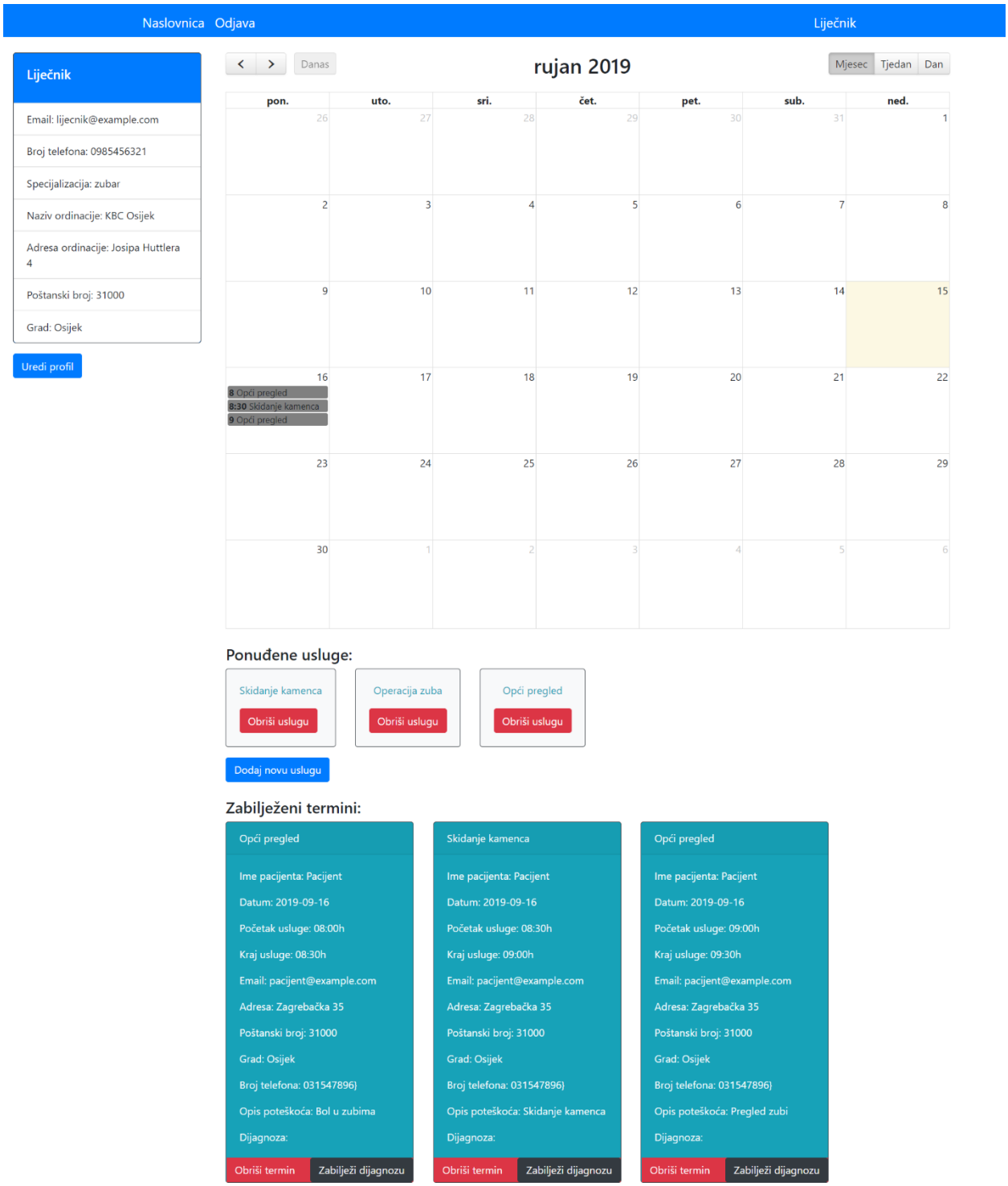
Sl. 4.17. Izgled e-maila pri registraciji liječnika.

Prijava liječnika se izvršava odabirom „Prijava – liječnik“ unošenjem email adrese liječnika te odgovarajućom lozinkom. Na slici 4.18. prikazan je izgled forme koja se koristi za prijavu korisnika.



Sl. 4.18. Forma za prijavu liječnika.

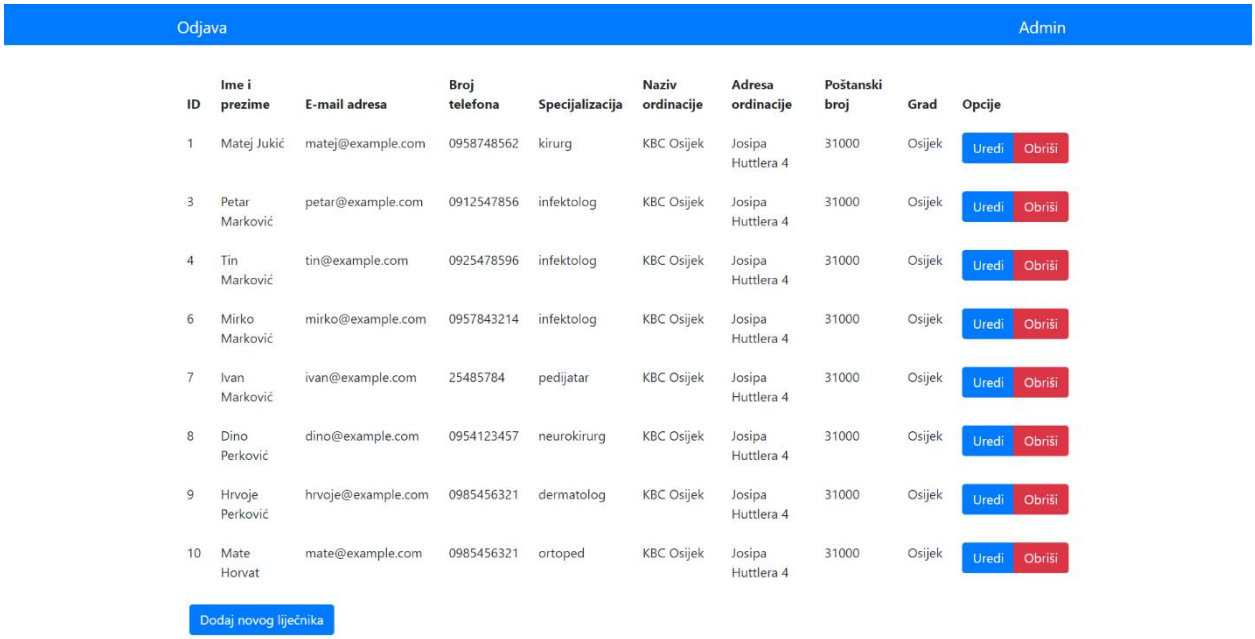
Kada je liječnik prijavljen u sustav, otvara se početna stranica gdje se prikazuju svi termini prijavljenog liječnika. Kao i kod pacijenata, postavljen je kalendarski prikaz rasporeda termina. Ispod kalendarskog prikaza su izlistani svi termini korisnika u kartičnom obliku, gdje je moguće vidjeti pojedinosti termina. Osim pojedinosti o zakazanim terminima, ispod kalendara nalazi se i forma za dodavanje usluga koje liječnik nudi. Odabirom na opciju „Dodaj novu uslugu“ otvara se stranica na kojoj liječnik popunjava opis usluge koju nudi te trajanje same usluge. Uspješnim unosom ponovno se učitava profil, gdje su izlistane ponuđene usluge. Svaku upisanu uslugu moguće je i ukloniti odabirom opcije „Obriši uslugu“ koja se nalazi kraj naziva pojedine usluge. Na slici 4.19 prikazan je izgled profila liječnika sa svim zakazanim terminima te uslugama koje su ponuđene.



Sl. 4.19. Profil liječnika.

Osim prijave pacijenata i liječnika, postoji i način prijave za administratora stranice. Administrator se prijavljuje upisivanjem URL-a „http://dipl.local/admin/login“, gdje upisuje svoju email adresu te lozinku. Nakon što je izvršena prijava, administratoru aplikacije se otvara

stranica s prikazom svih liječnika koji se nalaze u bazi podataka te njihovim informacijama. Za svakog liječnika postoje opcije s kojima administrator posjeduje mogućnost uređivanja informacija te uklanjanja liječnika iz baze podataka. Također uz postojeće liječnike, administrator ima i opciju „Dodaj novog liječnika“, budući da liječnici nemaju mogućnost registracije u sustav. Na slici 4.20. prikazan je izgled kontrolne ploče administratora.



Sl. 4.20. Kontrolna ploča administratora.

5. ZAKLJUČAK

Rezultat ovog diplomskog rada je internetska aplikacija za automatsko zauzeće termina, koja je provedena kroz koncept rezervacije termina pacijenata za određenu uslugu koju nude pojedini liječnici. Prvenstveno je izrađen plan koji opisuje kako bi aplikacija trebala izgledati, funkcionalnosti same aplikacije te način rada algoritma koji raspoređuje termine u bazu podataka. Sama aplikacija izrađena je u Laravel PHP radnom okviru koji je otvorenog tipa, a služi za izradu internetskih aplikacija baziranih na MVC arhitekturi. Osmišljena je shema izgleda baze podataka, te na temelju sheme sama baza koja sadržava sve potrebne podatke za ispravan rad aplikacije. Na temelju zahtjeva modificirani su postojeći modeli, rute, upravitelji te pogledi. Modelima su definirane relacije za dohvaćanje podataka iz baze, rutama su definirani pristupi stranicama, upraviteljima se obrađuje logika aplikacije i njene funkcije dok pogledi izvršavaju prikaz obrađenih podataka korisniku, te tako čine jednu cjelinu. Glavni dio diplomskog rada je izrada algoritma, s toga je fokus stavljen na funkcionalnost te jednostavan izgled. Algoritam je izrađen tako da se pregledava baza podataka, te vraća prazan termin prije prethodno zabilježenog termina, između dva termina, ili na kraju zadnjeg termina. U slučaju da nema zapisa za odabrane dane, nude se termini na početku radnog vremena. Aplikacija sadrži dva načina prijave korisnika, a to su pacijenti i liječnici. Svaki od njih posjeduje zasebne funkcije te različite prikaze sadržaja, budući da imaju različite uloge. Za pacijente je odrađena registracija, prijava, odabir te pregled zakazanih termina uz mogućnost uređivanja svojih podataka, dok je za liječnike onemogućena registracija te ih dodaje administrator aplikacije. Liječnici imaju mogućnost pregleda zakazanih termina te dodavanje i brisanje pojedinih usluga koje nude te mogućnost uređivanja svojih podataka. Odziv aplikacije je relativno brz, no nakon određenog vremena količina podataka u bazi bi porasla, te bi brzina ovisila o jačini servera. Problem bi se mogao riješiti pohranom podataka na nekom drugom memorijskom mjestu nakon prolaska određenog vremena od završetka termina.

LITERATURA

[1] PHP Documentation, https://www.php.net/docs.php, pristupljeno srpanj 2019.

[2] MVC Architecture, https://www.tutorialspoint.com/mvc_framework/mvc_framework_introduction.htm, pristupljeno rujan 2019.

[3] Laravel Documentation, https://laravel.cđom/docs/5.8, pristupljeno lipanj 2019.

[4] HTML, https://www.w3schools.com/html/, pristupljeno rujan 2019.

[5] CSS, https://www.w3schools.com/css/, pristupljeno rujan 2019.

[6] XAMPP, https://www.apachefriends.org/index.html, pristupljeno lipanj 2019.

[7] HeidiSQL, https://www.heidisql.com/, pristupljeno lipanj 2019.

[8] Visual Studio Code, https://code.visualstudio.com/docs, pristupljeno lipanj 2019.

[9] Bootstrap Documentation, https://getbootstrap.com/docs/4.3/getting-started/introduction/, pristupljeno kolovoz 2019.

[10] Geocoder, https://github.com/geocoder-php/GeocoderLaravel, pristupljeno rujan 2019.

[11] Googlmapper, https://github.com/bradcornford/Googlmapper, pristupljeno rujan 2019.

[12] FullCalendar, https://fullcalendar.io/docs, pristupljeno rujan 2019.

[13] Laravel Carbon, https://carbon.nesbot.com/docs/, pristupljeno lipanj 2019.

SAŽETAK

Internetska aplikacija za automatsko zauzeće termina, koja je provedena kroz koncept rezervacije termina pacijenata za određenu uslugu koju nude pojedini liječnici. Aplikacija je izrađena u Laravel PHP radnom okviru otvorenog tipa, baziranom na MVC arhitekturi. Svrha aplikacije jest olakšati i omogućiti što jednostavnije prijavljivanje na preglede. Pružatelji usluge se dodaju u sustav, unose usluge koje pružaju te svaka ponuđena usluga ima svoje trajanje. Korisnici usluge se registriraju, pretražuju usluge po parametrima koji im najbolje odgovaraju, te rezerviraju termin za zatraženu uslugu. Slobodni termini se generiraju pomoću algoritma, koji je glavni dio ovoga projekta. Algoritam pretražuje bazu podataka, te vraća prazan termin prije zabilježenog termina, između dva termina, ili na kraju zadnjeg termina.

Ključne riječi: algoritam, automatsko zauzeće termina, Laravel , MVC, usluge

ABSTRACT

Title: Web application for automatic appointment scheduling

Internet application for automatic appointment scheduling, implemented through the concept of booking appointments for patients for a specific service offered by each doctor. The application is created in Laravel open-source PHP framework based on MVC architecture. The purpose of the application is to make signing up for a specific service easier for patients. Providers are added to the system, they add the services they provide, and each service offered has its own duration. Users register, search for services matching their parameters, and schedule an appointment for the requested service. Availiable appointments are generated through the algorithm, which is the main part of this project. The algorithm searches the database and returns an availiable appointment, before the previously existing appointment, between two appointments, or at the end of the last appointment.

Keywords: algorithm, automatic appointment scheduling, Laravel, MVC, services

ŽIVOTOPIS

Erik Kiralj rođen je u Osijeku 7. Listopada 1994. godine. Pohađao je osnovnu školu Dobriše Cesarića u Osijeku, nakon koje upisuje Elektrotehničku i prometnu školu Osijek, smjer Tehničar za računarstvo. 2014. godine maturira te upisuje preddiplomski studij računarstva na tadašnjem Elektrotehničkom fakultetu Osijek (danas Fakultet elektrotehnike, računarstva i informacijskih tehnologija Osijek). Nakon završetka preddiplomskog studija računarstva 2017. godine, dobiva zvanje prvostupnik inženjer računarstva (univ. bacc. ing. comp.). Obrazovanje iste godine nastavlja na diplomskom studiju računarstva, izborni blok Računalno inženjerstvo. Trenutno je u procesu završetka studija, nakon kojega će steći zvanje magistar inženjer računarstva (mag. ing. comp.)