Aplikacija za praćenje stanja pacijenata

Hajduković, Hrvoje

SVEUČILIŠTE JOSIPA JURJA STROSSMAYERA U OSIJEKU

FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA OSIJEK

Stručni studij

Aplikacija za praćenje stanja pacijenata

Završni rad

Hrvoje Hajduković

Osijek, 2021.

Sadržaj

1. UVOD

U današnje vrijeme internet je posvuda oko nas i vezan je uz životne navike pojedinca, pa i većine današnjih organizacija koje bez interneta ne bih mogle funkcionirati na razini na kojoj u današnjici funkcioniraju. Možemo reći da je prednost interneta beskonačna, te nam on pruža mogućnosti koje prije dvadesetak godina nismo mogli ni zamišljati. S rastom broja korisnika rasla je i potreba za funkcionalnostima koje pruža internet, te su tako kreirane aplikacije koje su pružale korisniku određene mogućnosti da se multimedijski materijal pohrani, izradi ili nekome pošalje.

U ovom radu opisati će se proces izrade i funkcioniranja internetske aplikacije koja će pratiti stanje pacijenata, te zbog sve većih potreba u sustavu zdravstva i palijativne skrbi aplikacija će omogućiti jednostavnije i sveobuhvatnije praćenje zdravlja pacijenta. *Web* oblik aplikacije je odabran zbog svoje mogućnosti primjene na različitim tipovima uređaja i većini preglednika koji podržavaju trenutne tehnologije, te će tako biti omogućeno pacijentu da s bilo kojeg uređaja pristupi aplikaciji, te unese svoje podatke o stanju (npr. krvni tlak, temperatura, broj sati sna) u bilo kojem trenutku i gdje god se nalazio, a da ima pristup internetu i pregledniku. Takav način rada aplikacije također će olakšati zdravstvenom djelatniku u vezi postavljanja dijagnoze i određivanja daljnjih koraka u liječenju pacijenta.

Aplikacija je kreirana pomoću PHP-ovog radnog okvira Laravel, te se koristiti MySQL sustav za upravljanje bazom podataka. Programski okvir Laravel koristi se kako bih podržao jednostavniji i puno efikasniji način razvoja rješenja za pojedinu aplikaciju. Zbog toga se razvoj aplikacije odvija puno brže jer osoba koja je izrađuje može posvetiti svoje vrijeme na razvoj funkcionalnosti aplikacije bez značajnog gubitka vremena na razvoj pojedinosti koje sadrže niže razine aplikacije.

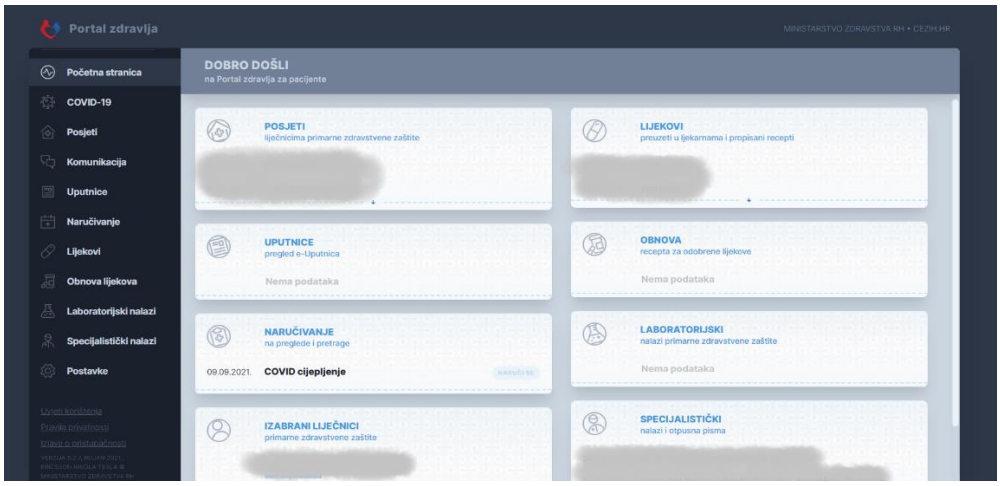
U radu će biti objašnjeno što je aplikacija, koje vrste aplikacija postoje i tehnologije koje se koriste pri izradi aplikacije ovog rada. U praktičnom dijelu biti će opisan način kreiranja Laravel projekta, baze podataka koja je korištena unutar aplikacije i dijelovi MVC arhitekture.

1.1 Zadatak završnog rada

Napraviti aplikaciju za praćenje stanja pacijenata u bolnici. Moguće je pratiti liječnike i medicinske sestre koji su odgovorni za pacijente, njihove terapije i stanje svakog pacijenta.

2. PREGLED PODRUČJA

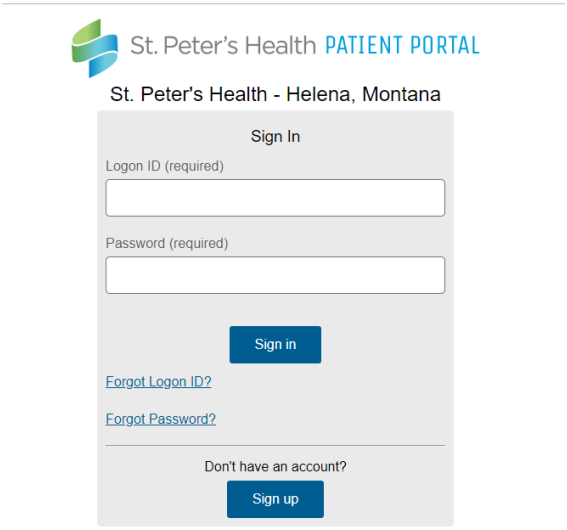
Najsličnija aplikacija koja se trenutno koristi za praćenje zdravstvenog stanja pacijenata je Portal zdravlja. Aplikacija se koristi tako da se za svakog pacijenta na jednom zaslonu prikazuju po posjetama grupirani podaci koji uključuju zdravstvene podatke i listu podataka prikupljenih s terena (mjerenja, bilješke, upitnike, itd.). Za svako mjerenje je dostupan detaljan pregled vrijednosti, sukladno pravilima i ovlastima korisnika. Komentare na pregledana mjerenja moguće je upisati u obliku bilješke u tijeku samog pregleda mjerenja. Također aplikacija sadrži početni zaslon, popis pacijenata i ostalih korisnika, pregled podataka i kontrola pacijenata, te administratorske funkcije (dodavanje korisnika, prikaz pacijenata i liječnika i izmijenja njihovih podataka). U usporedbi s aplikacijom kreiranom u ovom radu koja ima vrlo slične mogućnosti, a to su vođenje evidencije o pacijentima, unos kontrola i mogućnost kreiranja kontrola koje će se izvršiti u budućem vremenskom razdoblju, aplikacija pruža mogućnost unosa kontrole samostalno od strane pacijenta, te se na taj način smanjuje opterećenje zdravstvenog sustava i smanjuje utrošeno vrijeme potrebno liječniku za obaviti kontrolu koju je pacijent i sam u mogućnosti izvršiti iz vlastitog doma. Na taj način se ne gubi važnost liječnika u tom procesu jer nakon pacijentovog unosa kontrole, liječnik ju pregledava i piše svoje mišljenje o njoj.



Sl. 2.1. Početna stranica aplikacije Portal zdravlja

Također aplikacija koja je usporedbom vrlo slična kreiranoj u radu je *Patient portal*, a to je službena aplikacija bolnice St. Peter's Health u Heleni (Sjedinjene Američke Države). Njezine funkcionalnosti su praćenje pacijentovih dolazaka i odlazaka iz bolnice, praćenje kontrola i

lijekova koji su izdati pacijentu i praćenje općenitog zdravlja pacijenta. Takav oblik prikupljanja, pohranjivanja i mjerenja podataka pruža veliko olakšanje zdravstvenom sustavu i pruža veću dostupnost zdravstvene skrbi zbog veće količine dostupnih informacija i manji broj potrebnih posjeta specijalistu. Također olakšava praćenje popisa pacijenata i za svakog pacijenta je dostupan grupirani pregled podataka koji uključuje osobne podatke, pregled izvršenih mjerenja i bilješke koje su unijete.



Sl. 2.1. Stranica za prijavu u sustav aplikacije Patient portal

3. APLIKACIJA

Primjenjivi program, također poznat kao aplikacija ili *app* (engl. *application software*) računalni je program dizajniran za pomoć korisnicima da bi izvršavali jedan ili više određenih zadataka. Iz te definicije možemo zaključiti da osnovna svrha svake aplikacije je da izvršava nekakvu funkciju, tj. obavlja nekakvu radnju za korisnikove potrebe u ovisnosti o svrsi za koju je aplikacija dizajnirana. U osnovi aplikacija manipulira multimedijskim elementima kao što su tekst, brojevi, grafički elementi ili kombinacija svega od nabrojanog.

3.1. Vrste aplikacija

Tehnološkim razvojem računala i mobilnih uređaja proporcionalno je rasla i potreba za različitim vrstama aplikacija, pa se te vrste mogu podijeliti u određene skupine primjenjivih programa kao što su:

1. Programski paketi – predstavljaju skupinu primjenjivih programa koji u suštini imaju povezane funkcionalnosti, obilježja i korisničko sučelje, a jedna od zajedničkih točaka je mogućnost interaktivne manipulacije sa zajedničkim podacima koje sadrže neki od programa unutar paketa. Jedan od takvih, a moglo bi se reći i najpoznatiji programski paket je Microsoft Office koji povezuje programe za uređivanje i prikaz teksta, te različite oblike manipulacije tablicama.

2. Poslovni programi – predstavljaju aplikacije koje služe za organizacijske procese i razmjenu podataka unutar neke organizacije, te olakšava rad i organizaciju poslovanja unutar pojedinog odjela. Primjer poslovnog programa možemo pronaći u posebno kreiranoj aplikaciji za evidenciju radnog vremena zaposlenika i evidenciji radnih zadataka, te smo time dobili mjesto na kojem je grupirana kompletna evidencija o zaposlenicima, a time je olakšano vođenje nekoliko odijela unutar same organizacije.

3. Poslovni infrastrukturni programi - riječ je o aplikacijama kojima je osnovna djelatnost pružiti potporu za kvalitetno i sigurno poslovanje. Primjer takvih programa su: baza podataka, e-mail serveri, sustavi za upravljanje mrežama i sustavi za upravljanje sigurnosti.

4. Programi za pregled i uređivanje sadržaja - sadrži široku paletu aplikacija kojima je osnovna funkcionalnost manipulacija sadržaja u svrhe za koje je aplikacija kreirana. Primjer: tražilice, medijski izvođači.

5. Obrazovni programi – kao što i sam naziv kaže to su aplikacije koje se koriste za olakšanje edukacije i razmjenu znanja. Vrlo često su korišteni od strane nastavnika i učenika za provođenje testova, ali također služe i za praćenje gradiva i razmjenu edukacijskog sadržaja.

6. Programi za izradu medija i simulaciju – to su vrste aplikacija koje se većinom koriste u komercijalne i edukacijske svrhe. Osnovna funkcionalnost im je izrada multimedije kao što su slika, zvuk, film i brojni drugi.

7. Mobilni primjenjivi programi – predstavljaju sve aplikacije koje se izvršavaju na uređajima koji se drže u ruci ili su prenosivi, te u današnje vrijeme sve je više prijenosni uređaja kao što su pametni satovi i narukvice za koje su kreirane aplikacije koje izvršavaju specifične funkcionalnosti kao mjerenje koraka, broja otkucaja srca i sl.

3.2. *Web* aplikacija

Web aplikacijom se smatra svaki otvoreni ili zatvoreni sustav upravljanja sadržajem koje se koristi za poslovanje, edukaciju ili razmjenu multimedije putem interneta i preglednika. One mogu biti izrađene u svrhu korištenja kao internet trgovina, sustav za upravljanje sadržajem (eng. CMS – *Content management system*) ili sustav kontrole sveukupnog procesa rada i djelovanja unutar neke organizacije. Nije nužno da se web aplikacija mora nalaziti na internetskom serveru, ona može biti smještena i na serveru unutar internog okruženja poslovnog prostora neke organizacije, te njemu mogu pristupiti samo računala koja se nalaze unutar tog okruženja. Tako se dobiva na sigurnosti unutar te organizacije i umanjena je mogućnost potencijalnim napadačima da pristupe sadržaju koji im zapravo ne bih trebao biti dostupan. Zbog takvog pristupa smatra se da je *web* aplikacija samo naziv za oblik softvera koji je izrađen u nekoj od *web* tehnologija i koji se može kontinuirano izvršavati na jednom središnjem mjestu koje nazivamo server, a njemu ima mogućnost pristupiti neograničen broj korisnika koristeći se samo s internetskim preglednikom.

Mnogobrojne su prednosti *web* aplikacija i odnosu na slične računalne sustave, a neke od njih su jednostavno i lako održavanje, jer krajnji korisnik nema potrebe nadograđivati aplikaciju nego su promijene aplikacije učinjene na serveru od strane administratora aplikacije i vidljive su odmah krajnjem korisniku. Također, administrator ima mogućnosti raditi promijene u trenutku kad je aplikacija aktivna i u trenutno u uporabi. Kao što je spomenuto u uvodnom poglavlju, velika prednost *web* aplikacija je ta da je korisniku za pristup aplikaciji potreban samo internetski preglednik i veza, te ima pristup aplikaciji s bilo kojeg mjesta na svijetu. A veliku prednost takve

vrste aplikacija čine za organizacije koje sadrže veći broj korisnika, jer nije potrebno kupovati stotinjak licenciranih programa za računala nego se kupuje samo jedna aplikacija koju može koristiti svaki korisnik unutar te organizacije.

Pored svih prednosti *web* aplikacije mogu sadržavati i pojedine nedostatke, a neki od njih su nedovoljna usuglašenost internetskih standarda, te se mogu pojaviti razlike u prikazu unutar aplikacije u ovisnosti u verziji preglednika koja se nalazi na računalu korisnika. Također, brzina aplikacije može ovisiti o računalnoj mreži ili serveru i mogu se pojaviti problemi sa sigurnosti te mreže. Zbog svoje kompleksnosti izrade i funkcioniranja tvrtka koja izrađuje *web* aplikaciju ima potpunu kontrolu nad aplikacijom i podacima tvrtke koja je korisnik aplikacije te se tako gubi privatnost unutar organizacije koja je u stvarnosti krajnji korisnik te aplikacije.

4. TEHNOLOGIJE

U ovom poglavlju biti će opisane tehnologije koje su korištene u izradi aplikacije. Jedna od njih je skriptni jezik PHP i njegov razvojni okvir Laravel. Za potrebe pospremanja podatka o pacijentima korišten je MySQL zbog toga što je jedan od najkorištenijih relacijskih sustava koji se koristi za upravljanje bazom podataka.

4.1. PHP

Iza naziva PHP (eng. *Hypertext Preprocessor*) krije se vrlo moćan skriptni jezik za opću namjenu i za koji se smatra da je vrlo pogodan kod rješavanja *web* problema. Njegov kod izvršava se na strani servera pomoću interpretera, nakon korisnikovog upita server odgovara na taj upit i vraća putem preglednika svoj odgovor. Prva verzija PHP-a kreirana je 1994. godine kao skup Pearl skripti koje je razvio Rasmus Lerdorf s namjerom brojanja posjetitelja na vlastitoj web stranici. Tijekom svoje povijesti sadržavao je četiri važne stepenice za svoj razvoj. Kreator PHP-a je zbog povećane potrebe za određenim funkcionalnostima razvio novu verziju u programskom jeziku C. Prvotno je omogućio rad s bazom podataka i mogućnost kreiranja dinamičkih web stranica. Nakon toga pružio je mogućnost zaprimanja i korištenja varijabli dobivenih putem HTTP formi i njegovog protokola i pružena je mogućnost da se u HTML sintaksu uključi u PHP-ov kod.

Godine 1998. nakon objave verzije PHP 3.0 jezik doživljava nagli uspon, te sve veći broj korisnika i servera na svijetu koriste njegov alat za razvoj. Smatra se da je to prva verzija PHP-a koja je vrlo slična današnjoj verziji koja je u trenutnoj uporabi. U suradnji s još nekolicinom programera Rasmus je nastavio na razvoju skriptnog jezika PHP i tako omogućio dodatne funkcionalnosti za koje se smatralo da su mu tada bile potrebne. Neke od tih funkcionalnosti su mogućnost korištenja različitih tipova baza podataka, protokola i *API*-a, a također jedan od velikih koraka u razvoju je dodavanje objektno orijentiranog programiranja i sintakse koja je imala svoju konzistenciju. Četvrtom stepenicom u razvoju PHP-a uvedena je podrška za HTTP sesije i pružena je mogućnost ugradnje u većinu *web* poslužitelja. Od siječnja 2021. godine je u uporabi PHP 8.0 verzija, te će se na njoj temeljiti i aplikacija koja je izrađena u ovom radu.

4.1.1. Osnove i primjena

Namjena za koju je skripti jezik PHP kreiran prvotno je bio stvaranje dinamičkih web stranica. Posebno ugrađen interpreter unutar servera na kojem se nalazi projekt pokreće prolazi kroz cijeli kod koji je korisnik proslijedio, otkriva dijelove koda u kojima je označena PHP sintaksa i nakon toga ih izvršava.

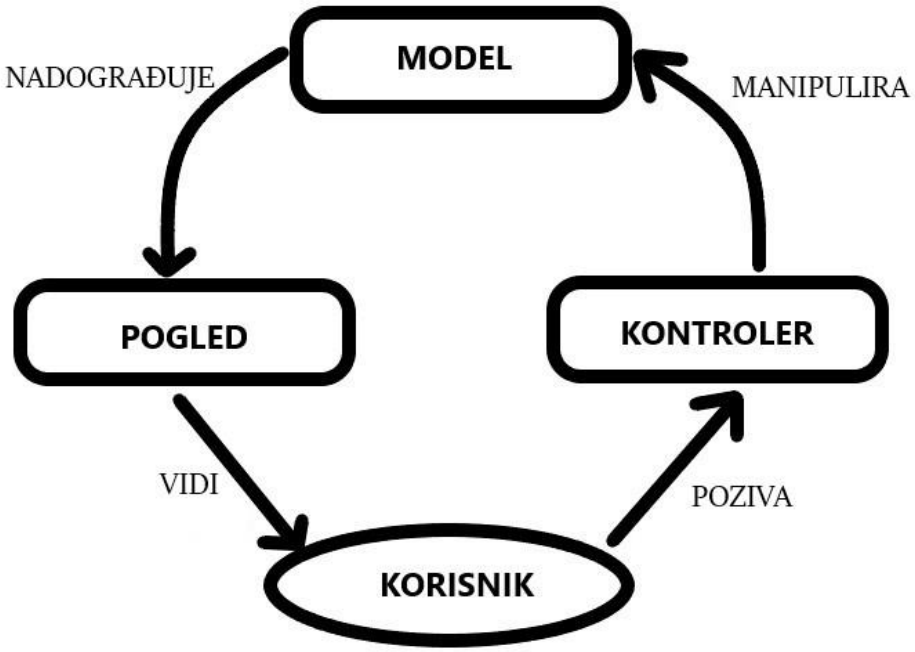
PHP stranica može biti promatrana kao i svaku drugu HTML stranicu, ali se u nju unose dodatne PHP naredbe koje su isprepletene s HTML kodom. Skripte koje su kreirane u tom jeziku imaju nastavak *.php* i naredbe napisane unutar te skripte izdvojene su s posebnim oznakama koje označavaju početak i kraj naredbe. Primjer jednostavnog PHP koda isprepletenog unutar HTML koda prikazan je na slici 3.1..

```
1  <html>
2  <head>
3  <title>Primjer PHP koda</title>
4  </head>
5  <body>
6  <?php  echo “Primjer teksta kojeg ispisuje echo PHP naredba”;  ?>
7  </body>
8  </html>
```

Sl. 3.1. Primjer PHP naredbe unutar HTML koda

4.2. Laravel i MVC

Laravel je kreirao Taylor Otwell kako bih proširio mogućnosti razvojnog okruženja kao što su do tad bili Zend, Slim i slični. On je razvojno okruženje PHP-a koje je otvorenog koda, a njegovi temelji se pronalaze na već postojećem razvojnom okruženju kao što je Simfony 2. Temelji se na MVC (*eng. Model View Controller*) arhitekturi, a nju možemo definirati s tim što razdvaja samu logiku aplikacije, pravila i upite nad bazom za koje se smatra da je odgovoran model. Dok je pogled aplikacije korišten za prikazivanje podataka korisniku koji su unutar kontrolera obrađeni i proslijeđeni putem modela.



Sl. 3.1. Prikaz MVC arhitekture, [4]

Model

Za model se smatra da je jedan od glavnih sastavnica *MVC* arhitekture nekog sustava i ima ulogu povezivanja pogleda i upravljačkog dijela, te se ne može direktno pozvati. Njegova osnovna zadaća je da upravlja i obrađuje podatke o korisniku aplikacije i njegovim segmentima, objektima iz baze i SQL upitima. Iz toga se može zaključiti da model nije u mogućnosti samostalno pokretati zahtjeve nego obrađuje zahtjeve koji su poslani od strane pogleda i upravljačkog dijela.

Pogled

Pogled se koristi za prikaz podataka, izgled korisničkog sučelja, odnosno obrasce, tablice, gumbe, slike i sl. Može se reći da je osnovna zadaća pogleda da prikazuje objekte iz modela i pruža korisniku mogućnost promijene podataka, no njegova razina odgovornosti za sigurnost aplikacije prenesena je na model. Također jedino je on vidljiv korisniku, dok model i upravljački dio nalaze se unutar pozadinskog dijela aplikacije.

Upravljački dio

Osnovna zadaća kontrolera je da uvodi i održava pravila i funkcije koje sadržavaju programsku logiku aplikacije. Upravlja zahtjevima korisnika koji se šalju pogledu i modelu, a neki od tih zahtjeva su dobro poznati (*GET, POST,* klik na element i sl.). Direktno je odgovoran za tok kontrole programa, jer kada korisnik pokrene neku od kontrola aplikacije poziva se model te tako nastaju promijene stanja unutar modela koje je korisnik zahtijevao. Također prilikom pokretanja *web* aplikacije upravljački sloj je prvi sloj koji se tom prilikom pokreće.

4.2.1 Prednosti i mane

MVC način strukturiranja aplikacije je među popularnijim obrascima izrade aplikacije zbog toga što olakšava testiranje i naknadno održavanje za razliku od ostalih načina strukturiranja, jer kroz razvoj aplikacije događa se puno promjena koda i tehničke dokumentacije. Ali također zbog svoje kompleksnosti nije ju preporučeno koristiti pri izradi manjih i jednostavnijih aplikacija.

Mnoge programere je kod Laravela najviše privukla jednostavnost, jer način na koji je pisan je vrlo intuitivan i njegova struktura pruža vrlo jednostavno snalaženje unutar dijelova aplikacije. Međutim i ako se dogodi da se zapne negdje prilikom izrade ili razumijevanja određenog dijela koda, tu je Laravel-ova dokumentacija koja pokriva većinu pitanja na kojima programer može pronaći rješenje svog problema.

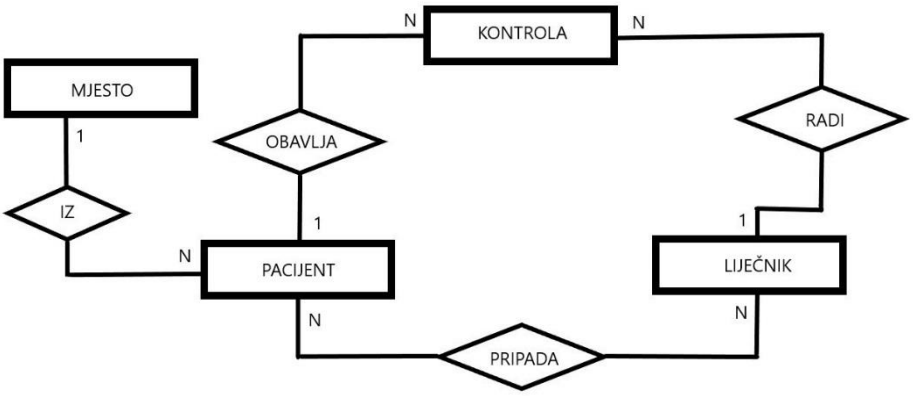
5. IZRADA APLIKACIJE

Za izradu aplikacije prvotno je potrebno na računalo instalirati Composer. To je konzolni alat koji se koristi za upravljanje paketima unutar PHP projekta i omogućava kreatoru aplikacije da definira pakete koji će biti korišteni unutar njegovog projekta. Postoji nekoliko njegovih načina instalacije na računalo te svaki od tih načina zahtijeva da je PHP već instaliran na računalo. Odabrani način koji je korišten u izradi aplikacije za potrebe ovog rada je da putem naredbenog retka i naredbom *php -r "readfile('https://getcomposer.org/installer');"*. Za daljnje korištenje Composera potrebno je u naredbeni redak unijeti naredbe na slijedeći način: *composer [naredba][opcija]*. Također on zahtijeva način na koji se može saznati koje točno pakete projekt sadržava i očekuje, te nazivi tih paketa definiraju se unutar composer.json datoteke. To je tekstualna datoteka koja sadržava nazive paketa koji se koristite unutar projekta, a također može sadržavati i pojedine metapodatke o projektu (baza podataka, verzija PHP-a i sl.). Nakon unosa popisa svih potrebnih paketa potrebno je iz naredbenog retka doći do mape koja sadržava composer.json te pozvati Composer pomoću naredbe *composer install*.

Nakon uspješne instalacije Composer-a potrebno je na računalo instalirati aplikacijski okvir Laravel i kreirati novi projekat. Kroz naredbeni redak potrebno je doći do mape u kojoj je instaliran XAMPP te unutar datoteke *htdocs* unošenjem slijedeće naredbe *composer create-project laravel/laravel aplikacija* alat će preuzeti Laravel i sve komponente o kojima je ovisan te će kreirati novu mapu. Funkcionalnost projekta može se provjeriti pomoću internetskog preglednika i razvojnog okruženja XAMPP tako da unutar tražilice unesemo lokalnu adresu servera *localhost/aplikacija/public*, te će on nakon odgovora servera vratiti početnu stranicu prethodno kreirane aplikacije.

5.1. Model baze podataka

Za potrebe razumijevanja aplikacije kreiran je ER model koji je prikazan Chenovim dijagramom. Iz njega se može zaključiti sama logika aplikacije, ali i izgled baze podataka te veze koje se pojavljuju unutar nje između pojedinih atributa.



Sl. 4.1. ER model baze podataka

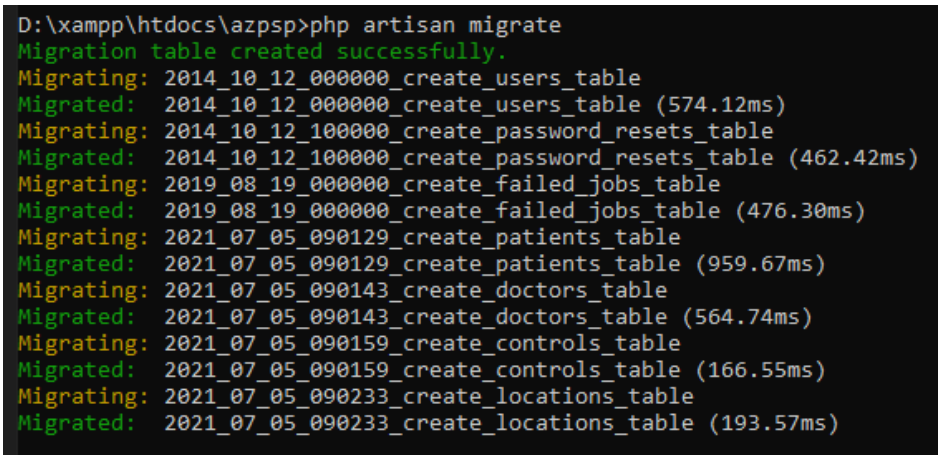
Na osnovu ER dijagrama sa slike 4.1. primjećuje se da jednom pacijentu može pripadati nekoliko liječnika (liječnik opće prakse, dermatolog, urolog i sl.), ali i da svaki liječnik zasebno može imati više pacijenata te je ta relacija označena s N:N. Također jedan pacijent može obavljati više kontrola, ali svaka kontrola se zasebno registrira za pojedinog pacijenta i ta veza je označena s 1:N. Svaku pojedinu kontrolu može obaviti samo jedan liječnik te se ta relacija vodi kao 1:1. Za svakog pacijenta je uvedena evidencija iz kojeg je mjesta, dok iz istog mjesta može biti više pacijenata, pa se da zaključiti da je relacija pacijent – mjesto jednaka N:1.

5.2. Kreiranje i povezivanje baze podataka

Koristeći phpMyAdmin besplatan softver za upravljanje bazom podataka kreirana je baza s nazivom *azpsp*. Potrebno je kreiranu bazu povezati s aplikacijom na način da se unutar *.env* datoteke promijeni naziv baze podataka koju će aplikacija koristiti.

5.2.1. Fizičko oblikovanje

Nakon oblikovanja ER modela unutar aplikacijskog oblika Laravel definiraju se tablice i atributi koje tablice sadržavaju. Laravel je olakšao kreatoru aplikacije te je pomoću migracija omogućeno kompletno kreiranje baze podataka koja je korištena u aplikaciji. Pomoću naredbenog retka nakon popunjavanja migracija pomoću naredbe *php artisan migrate* artisan kreira tablice koje se pospremaju u bazu podataka koja je navedena i kreirana u prethodnom poglavlju.



Sl. 4.2. Prikaz naredbenog retka nakon pokrenutih i izvršenih migracija

Iz isječka 4.3. može se zaključiti jednostavnost kreiranja tablica i popunjavanja modela pomoću aplikacijskog okvira Laravel i njegovih migracija u kojima su unijeti nazivi i tipovi atributa pojedine tablice ili konkretno u ovom primjeru tablice pacijenata.

```php
public function up()
{
    Schema::create('patients', function (Blueprint $table) {
        $table->id();
        $table->string('name',30);
        $table->string('surname',30);
        $table->date('birth_date');
        $table->string('address',60);
        $table->unsignedBigInteger('location_id')->index();
        $table->unsignedBigInteger('doctor_id')->index();
        $table->string('gender',15);
        $table->string('email')->unique();
        $table->string('phone')->unique();
        $table->timestamps();
    });
}
```

Sl. 4.3. Isječak koda dokumenta 2021_07_05_090129_create_patients_table.php

5.3. Model

Model je odgovoran za održavanje i provjeru stanja podataka koji se izmjenjuju između korisnika aplikacije i sustava u kojem je pohranjena baza podataka. Također primaju zahtjeve od upravitelja iz kojih se može protumačiti koji podaci će biti prikazani korisniku, pa se može zaključiti da kompletna logika koja se odvija s bazom podataka odvija se pomoću modela. Za potrebe aplikacije kreira se pet modela: *Patient, Doctor, Control, Location* i Laravelov model za autentifikaciju korisnika *User*.

```
D:\xampp\htdocs\azpsp>php artisan make:model Patient -m
Model created successfully.
Created Migration: 2021_07_05_090129_create_patients_table

D:\xampp\htdocs\azpsp>php artisan make:model Doctor -m
Model created successfully.
Created Migration: 2021_07_05_090143_create_doctors_table

D:\xampp\htdocs\azpsp>php artisan make:model Control -m
Model created successfully.
Created Migration: 2021_07_05_090159_create_controls_table

D:\xampp\htdocs\azpsp>php artisan make:model Location -m
Model created successfully.
Created Migration: 2021_07_05_090233_create_locations_table
```

Sl. 4.4. prikaz kreiranih modela pomoću naredbenog retka

5.3.1. Patient

Model koji je prvotno kreiran je model *Patient* koji će upravljati podacima koji su vezani za pacijente. Unutar modela su pospremljene varijable koje su korištene prilikom dodavanja novih pacijenata te je tako pružena sigurnost od unosa neželjenih atributa unutar tablice. Također unutar modela su sadržane metode modela koje definiraju veze modela *Patient* s ostalim modelima. Varijabla *fillable* sadrži atribute koje model može proslijediti bazi prilikom upita za pospremanje novog unosa u bazu podataka, a funkcije *doctor* i *location* povezuju strane ključeve koji se koriste između modela *Patient* i modela *Doctor* i *Location*.

```php
class Patient extends Model
{
    use HasFactory;
    protected $fillable = ['name', 'surname', 'birth_date', 'address', 'location_id', 'doctor_id', 'gender', 'email', 'phone'];

    public function location()
    {
        return $this->belongsTo('App\Models\Location', 'location_id');
    }

    public function doctor()
    {
        return $this->belongsTo('App\Models\Doctor', 'doctor_id');
    }
}
```

Sl. 4.5. Isječak koda iz dokumenta ./app/Models/Patient.php

5.4. Kontroler

Kontroleri mogu biti predstavljeni kao sučelje između pogleda i modela i odgovorni su za obradu svih zahtijeva korisnika. Zaprima zahtjeve korisnika, obrađuje ih i vraća odgovor putem pogleda. Tako korisnik klikom na link ili gumb šalje zahtjev kontroleru koji pomoću ruta pronalazi odgovarajući kontroler i aktivira funkciju koja je zadana unutar tog kontrolera. Za potrebe aplikacije kreirana su kontroleri za autorizaciju, te *PatientsController*, *DoctorsController*, *ControlsController* i *LocationsController*.

```
D:\xampp\htdocs\azpsp>php artisan make:controller --resource DoctorsController
Controller created successfully.

D:\xampp\htdocs\azpsp>php artisan make:controller --resource PatientsController
Controller created successfully.

D:\xampp\htdocs\azpsp>php artisan make:controller --resource LocationsController
Controller created successfully.

D:\xampp\htdocs\azpsp>php artisan make:controller --resource ControlsController
Controller created successfully.
```

Sl. 4.6. Kreiranje kontrolera pomoću naredbenog retka

5.4.1. PatientsController

Kreiranjem kontrolera *PatientsController* koji je korišten za kontrolu upita vezanih za pacijente *artisan* automatski kreira i sedam funkcija: *index, create, store, show, edit, update i destroy.* Njihova funkcionalnost je zamišljena da one prikazuju, pospremaju, promijene i obrišu resurse koji su vezani za taj kontroler. Konkretno, za prikaz forme koja služi za dodavanje novog

pacijenta korištena je funkcija *create,* te su pomoću nje pogledu se vraćaju podaci o svim mjestima stanovanja koje liječnik može izabrati prilikom dodavanja novog pacijenta i podaci o liječniku koji dodaje pacijenta. Nakon popunjavanja forme naredbom *store* i unosom dodatnih podataka kao što su ime, prezime, datum rođenja, adresa, spol i email, podaci se zajednički pospremaju u bazu podataka.

```
33      public function create()
34      {
35          $locations = DB::select('select * from locations');
36          $doctors = Doctor::get();
37          return view('addpatient', ['locations' => $locations, 'doctors' => $doctors]);
38      }
39
40      public function store(Request $request)
41      {
42          Patient::create($request->only(
43              'name',
44              'surname',
45              'birth_date',
46              'address',
47              'location_id',
48              'doctor_id',
49              'gender',
50              'email',
51              'phone'
52          ));
53          return redirect('addpatient');
54      }
```

Sl. 4.7. Isječak koda iz dokumenta ./app/Http/Controllers/PatientsController.php

Iz isječka 4.7. prikazan je kod funkcije *create* koja je prethodno objašnjena, ali i funkcija *store* koja ima zadaću iz zahtijeva koji je poslao korisnik, tj. u ovom slučaju liječnik posprema podatke o novom pacijentu koji se uvodi u sustav. Pomoću ugrađene funkcije *only* korisnik nije u mogućnosti dodati niti jedan drugi atribut unutar baze podataka osim onih koji su navedeni unutar te funkcije.

5.4.2. ControlController

Jedan je od najvažnijih kontrolera unutar aplikacije i njegova zadaća je da upravlja zahtjevima od korisnika vezanim za podatke o kontrolama koje liječnik ili pacijent unose i šalju zahtjev za prikaz. Dvije funkcije koje imaju veliku važnost su *index* i *create*. *Index* služi za prikaz

kontrola ovisno o korisniku koji je prijavljen, ako je prijavljen pacijent funkcija *index* će prikazati samo kontrole koje su vezane samo za njegov profil, dok ako je prijavljen liječnik prikazat će se kontrole svih pacijenata s kojima je on vezan. Takav način provjere autorizacije omogućen je tako da se unutar tablice modela *User* uključi novi atribut koji je naziva *role* te tako unutar koda aplikacije *role=”0”* služi za označavanje liječničkih prava pristupa, role=”1” smatra se da je pacijent, a *role=”2”* administrator sustava koji ima zadaću dodavanja i prikaza svih liječnika, dodavanje novih korisnika, prikaz i izmjena svih pacijenata.

```php
public function index()
{
    if (Auth::user()) {

        $user = Auth::user();
        $user_email = $user->email;
        if (Auth::user()->role == 1) {
            $id = Patient::where('email', 'LIKE', $user_email)->first()->id;
            $controls = Control::where('patient_id', 'LIKE', $id)->whereDate('control_date', '<=', Carbon::now())->sortable()->paginate(2550);
            $newcontrols = Control::where('patient_id', 'LIKE', $id)->whereDate('control_date', '>', Carbon::now())->sortable()->paginate(2550);
        } else {
            $id = Doctor::where('email', 'LIKE', $user_email)->first()->id;
            $controls = Control::where('doctor_id', 'LIKE', $id)->whereDate('control_date', '<=', Carbon::now())->sortable()->paginate(2550);
            $newcontrols = Control::where('doctor_id', 'LIKE', $id)->whereDate('control_date', '>', Carbon::now())->sortable()->paginate(2550);
        };
        return view('controls', compact(['controls', 'newcontrols']));
    } else return view('login');
}

public function create()
{
    $user = Auth::user();
    $user_email = $user->email;
    if (Auth::user()->role == 1) {
        $id = Patient::where('email', 'LIKE', $user_email)->first()->id;
        $patients = Patient::where('id', 'LIKE',  $id)->get();
        $doctor_id = Patient::where('id', 'LIKE',  $id)->first()->doctor_id;
        $doctors = Doctor::where('id', 'LIKE',  $doctor_id)->get();
    } else {
        $id = Doctor::where('email', 'LIKE',  $user_email)->first()->id;
        $patients = Patient::where('doctor_id', 'LIKE', $id)->get();
        $doctors = Doctor::where('id', 'LIKE', $id)->get();
    }
    return view('addcontrol', ['patients' => $patients, 'doctors' => $doctors]);
}
```

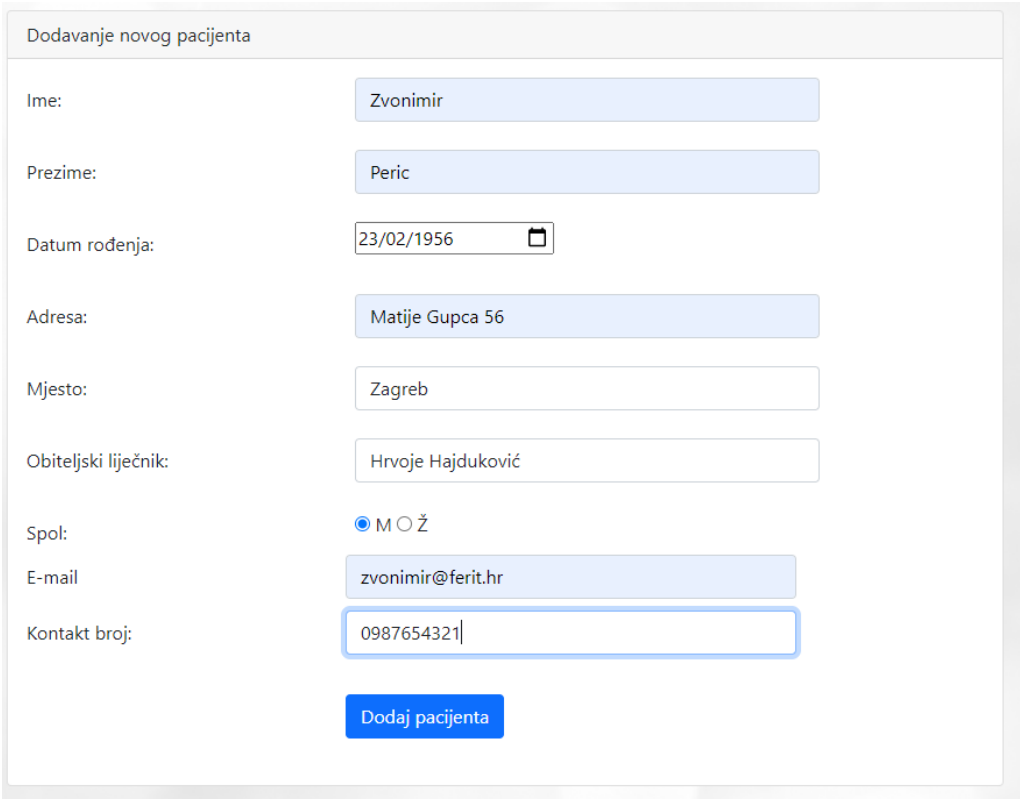
Sl. 4.8. Isječak koda iz dokumenta ./app/Http/Controllers/ConotrolsController.php

5.5. Pogled

Svi pogledi mogu se pronaći u direktoriju *./resource/views* i osnovna im je zadaća prikazivanje multimedijskih podataka korisniku. Njegova velika prednost koju pruža aplikacijski okvir Laravel je ta da mogu biti nasljeđivani. Tako pogled *app.blade.php* je nasljeđivan u svim ostalim pogledima koji se koriste u aplikaciji. Njegova osnovna zadaća je uključiti sva potrebna zaglavlja u naslijeđene poglede. Svi pogledi se sastoje od HTML elemenata koji su kreirani kao forme za unos podataka ili tablice za prikaz pojedinih podataka kao npr. sve kontrole koje je korisnik unosio u aplikaciju.

5.5.1. Pogledi za dodavanje i prijavu korisnika

Postoje tri pogleda za dodavanje novog korisnika. Jedan od osnovnih pogleda koji služi za dodavanje pacijenta i obavlja ju liječnik, a prikazan je na slici 4.9.. Forma se sastoji od osobnih podataka pacijenata i gumb koji služi za dodavanje pacijenta. Također postoji pogled za dodavanje novog liječnika i obavlja ju administrator sustava, a pogled je vrlo sličan pogledu za pacijenta. Treći tip pogleda je pogled za dodavanje novog korisnika u sustav i on se sastoji od imena i prezimena korisnika, e-mail adrese, lozinke i tipa prava pristupa unutar aplikacije (pacijent ili liječnik). Tek nakon što je administrator sustava registrirao korisnika on se može prijaviti u aplikaciju i koristiti ju za kreiranje i praćenje kontrola.

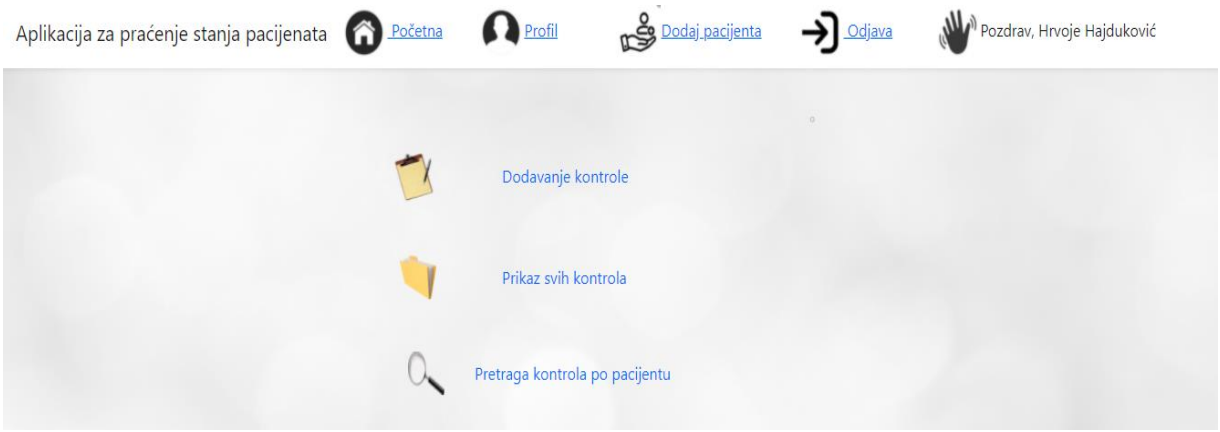


Sl.4.9. Forma za dodavanje pacijenta

5.5.2. Pogledi početne stranice

Zbog toga što aplikacija sadrži tri razine pristupa unutar nje, također sadrži i tri različita pogleda na početnu stranicu. Pogled nakon prijave pacijenta sadrži jednostavnu naslovnu traku s vezom na početnu stranicu aplikacije, profil pacijenta, odjavu iz sustava i pozdravnu poruku sa

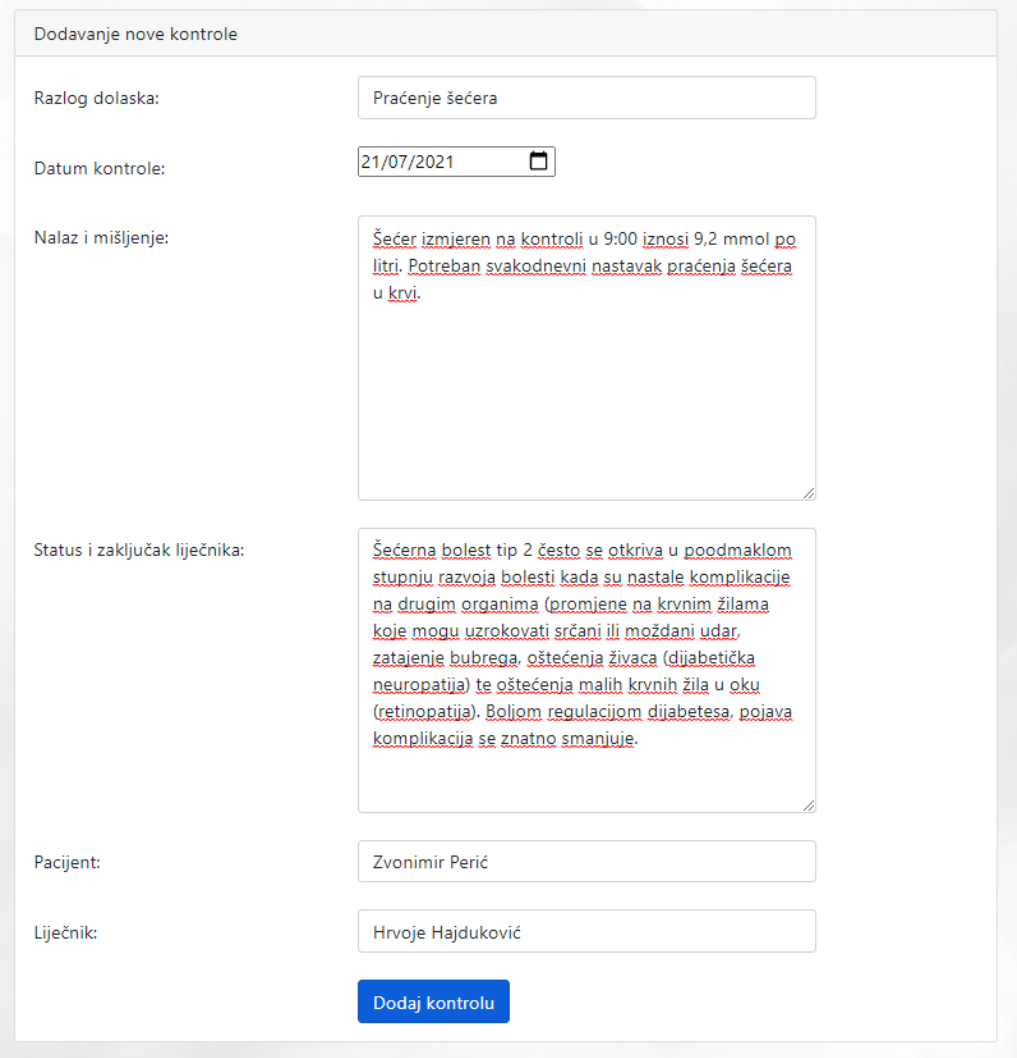
svojim imenom, a tijelo same aplikacije sadrži gumb koji vodi do forme za dodavanje nove kontrole i gumba za tablice s prikazom svih svojih kontrola. Naslovna traka kod liječnika prikazana je na slici 4.10. i za razliku od naslovne trake pacijenta ona sadržava i vezu na formu koja služi za dodavanje novog pacijenta, a tijelo aplikacije sadrži dodatni gumb *"Pretraga kontrola po pacijentu"* koji upućuje liječnika na formu koja služi za prikaz kontrola za određenog pacijenta u odabranom vremenskom intervalu. Dok se nakon prijave administratora prikazuje naslovna traka s vezom na početnu stranicu i profil administratora, vezom na formu za dodavanje liječnika i formu za dodavanje novog korisnika u sustav aplikacije i vezom na formu za odjavu iz aplikacije. Tijelo početne stranice administratora sadrži gumbe na tablice s podacima o pacijentima i liječnicima koji su registrirani unutar sustava i pružena je funkcionalnost izmijene podataka o pacijentu.



Sl. 4.10. Početni zaslon nakon prijave liječnika

5.5.3. Forma za dodavanje nove kontrole

Forma za dodavanje nove kontrole mogu ispunjavati pacijent i liječnik. Jedina razlika je ta što samo liječnik ima mogućnost unosa podataka vezanih za stupac *Status i zaključak liječnika*, jer nakon što pacijent unese svoju kontrolu, liječnik ju pregleda i unese svoj zaključak, te se nakon toga se smatra da je kontrola izvršena u potpunosti što možemo vidjeti u primjeru slike 4.11..

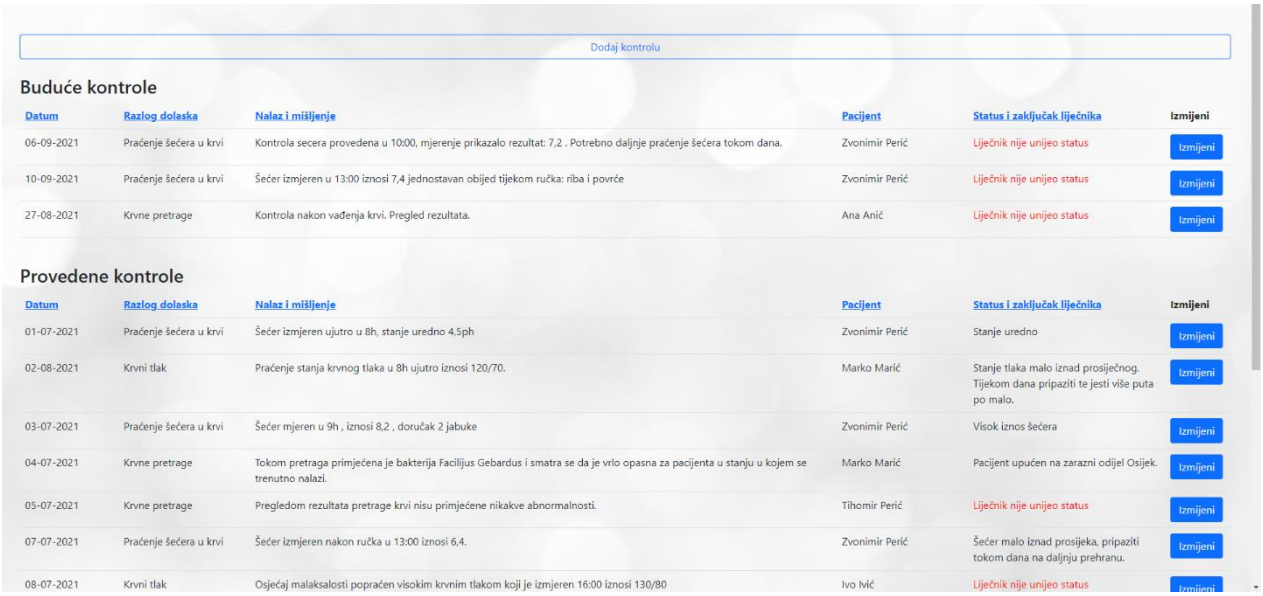


Sl. 4.11. Forma za dodavanje nove kontrole

5.5.4. Pregled kontrola

Pregled svih svojih kontrola pacijent ima mogućnost izvršiti samostalno prijavom u sustav i klikom na gumb *Pregled svih kontrola* dobiva povratnu informaciju o kontrolama koje je proveo i koje su zabilježene u sustav. Tako je i prikaz podijeljen na *Buduće kontrole*, tj. kontrole koje će se u budućnosti tek obaviti i *Provedene kontrole*, a to su sve potvrđene i nepotvrđene kontrole koje su unesene do dana pregleda. Pogled se sastoji od tablice s podacima o datumu, razlogu dolaska, nalazu i mišljenju i na kraju statusu i zaključku liječnika. Pregled kontrola u slučaju liječnika je vrlo sličan prethodno objašnjenom pogledu kod pacijenta, ali s tim da liječnik ima prikaz kontrola svih pacijenata s kojima je on vezan. Također, liječnik za svaku unesenu kontrolu ima gumb *Izmjeni* koji je veza na formu zaslužnu za izmjenu podataka o

kontroli, te tako liječnik unosi *Status i zaključak* i nakon toga se smatra da je kontrola u potpunosti provedena. Primjer prikaza svih kontrola u slučaju liječnika može se vidjet i na slici 4.12..



Sl. 4.12. Prikaz svih kontrola za liječnika

6. ZAKLJUČAK

U radu je prikazan način izrade jednostavne web aplikacije koja može na vrlo lagan način biti proširena na dodatne funkcije ukoliko korisniku budu potrebne. Također temelji se na serverskom dijelu, odnosno *backend*-u, jer je unutar rada prikazan način izrade i pohranjivanja u bazu podataka, te komuniciranje modela i baze s podacima koji su poslani putem kontrolera. U današnje vrijeme web aplikacije su u velikom uspon zbog svoje jednostavnosti izrade i implementiranja dodatnih servisa i funkcionalnosti iako je PHP već pomalo zastario kao skriptni jezik, Laravel mu na taj način implementacije pojedinih elemenata dodaje sasvim novu svrhu.

LITERATURA

Internet stranice

[1] Službena stranica Portala zdravlja, https://portal.zdravlje.hr/portalzdravlja (09.09.2021.)

[2] Službena stranica bolnice St. Peter's Health u Heleni (Sjedinjene Američke Države), https://www.sphealth.org (09.09.2021.)

[3] Xampp stranica, https://www.apachefriends.org/index.html (22.5.2021.)

[4] Službena stranica Composer programa, https://getcomposer.org/doc (22.5.2021.)

[5] Službena stranica Laravel, https://laravel.com (8.6.2021.)

[6] Stranica Laravel dokumentacije, https://laravel.com/docs/8.x (8.6.2021.)

[7] Službena stranica PHP skriptnog jezika, https://www.php.net/docs.php (11.6.2021.)

[8] Stranice Stackoverflow-a, https://stackoverflow.com (3.7.2021.)

[9] Stranice W3Schools-a, https://www.w3schools.com (19.7.2021.)

SAŽETAK

Tema završnog rada je izrada aplikacije za praćenje stanja pacijenata. Osnovna zadaća aplikacije je olakšati liječniku provođenje kontrole na taj način da aplikacija pruža pacijentu mogućnost samostalnog unosa kontrole (mjerenje šećera, krvnog tlaka i sl.), te nakon toga da liječnik potvrdi kontrolu i unese svoje zaključak. Aplikacija je kreirana u PHP-ovom aplikacijskom okviru Laravel zbog svoje jednostavnosti izrade, ali i lakšeg unosa promjena ukoliko budu potrebne. Prvotno je kreirana baza podataka sa svim potrebnim tablicama koje se koriste unutar aplikacije. Nakon toga su kreirani modeli poštujući veze iz baze podataka zbog toga što su oni ti koji su odgovorni za komunikaciju s bazom podataka. Kao sučelje između modela i pogleda koriste se kontroleri i njihova zadaća je da zaprimaju korisnikove zahtjeve, obrade ih i proslijede potrebne podatke pogledu koji te podatke prikazuje krajnjem korisniku.

Ključne riječi: Composer, HTML, Laravel, MVC, PHP

ABSTRACT

PATIENT MONITORING APPLICATION

The topic of the final work is the development of an application for monitoring the condition of patients. The basic tasks of the application facilitate the medical control in such a way that the application provides the patient with the possibility of self-control (measurement of sugar, blood pressure, etc.), after which the doctor confirms the control and enters his conclusion. The application was created in PHP's Laravel application framework because of its ease of creation, but also easier entry changes if needed. Initially, a database was created with all the necessary tables used within the application. Models were created respecting the connections from the database because they are the ones responsible for communicating with the database. Controllers are used as the interface between the model and a view. Their tasks are to receive user requests, process them and pass on the necessary data in terms of which this data is displayed to the end user.

Keywords: Composer, HTML, Laravel, MVC, PHP

ŽIVOTOPIS

Hrvoje Hajduković rođen je 12.1.1996. u Našicama. Osnovnu školu pohađao je u Viljevu, a srednju školu smjera ekonomist u Valpovu. Za to vrijeme aktivno igrao nogomet i bavio se sviranjem glazbenih instrumenata. Upisuje Elektrotehnički fakultet u Osijeku i prikazuje visoko zanimanje za izradu aplikacija u web tehnologijama. 2021. godine završava „PHP i SQL“ tečaj na visokom učilištu Algebra.

U Osijeku, rujan 2021. Hrvoje Hajduković

Potpis:

_____

PRILOZI

[1] P. 1. Programski kod aplikacije: https://github.com/Hajdukovic/aplikacija