Web sustav za potporu biomedicinskoj dijagnostici zasnovan na klasificiranju

Dumančić, Andrija

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

Diplomski studij

WEB SUSTAV ZA POTPORU BIOMEDICINSKOJ DIJAGNOSTICI ZASNOVAN NA KLASIFICIRANJU

DIPLOMSKI RAD

Andrija Dumančić

Osijek, 2018.

Sadržaj

1. UVOD

Ovaj rad se bavi problemom dijagnosticiranja bolesti na temelju unosa simptoma koji je zasnovan na klasificiranju. Cilj je izraditi web aplikaciju s korisničkim sučeljem koja daje približnu točnost bolesti nakon što korisnik unese simptome. Potrebno je omogućiti da se korisnik prijavi na sustav i pristupi dijagnosticiranju. Web aplikacija koristi podatke koji se nalaze u bazi podataka, dok podatke poput simptoma unosi korisnik nakon čega slijedi dijagnosticiranje. Na temelju rezultata dijagnosticiranja, liječnik daje konačnu dijagnozu i šalje povratnu informaciju korisniku.

Problem kojim se bavi rad je kako ostvariti pomoć, odnosno na koji način doći do dijagnoze iz svog prostora bez odlaska liječniku što se u pravilu ne savjetuje. Liječnici su upućeniji u stanje pacijenta nego što je to web aplikacija.

Drugo poglavlje govori općenito o biomedicini te načinima na koje strojno učenje pomaže u biomedicini. Također, navode se računalne tehnologije u dijagnostici te način na koji se donose odluke. Prikazan je i uvid u slične aplikacije koje također služe u svrhu dijagnosticiranja na temelju simptoma. Treće poglavlje prikazuje idejno rješenje sustava kao i zahtjeve koje mora zadovoljiti aplikacija kako bi korisnik bio zadovoljan s njom. Četvrto poglavlje opisuje programske jezike i okvire koji su korišteni prilikom izrade web aplikacije. Ukratko će se objasniti poslužitelj XAMPP, PHP okvir Laravel, HTML te CSS. Peto poglavlje prikazuje kreiranje i definiranje projekta te izradu baze podataka u kojoj se nalaze bolesti i simptomi. Također će se vidjeti i struktura baze podataka. Šesto poglavlje prikazuje korištenje web aplikacije s opisanim koracima i mogućoj pomoći pri dijagnosticiranju. Prikazani su i rezultati dijagnosticiranja na temelju unesenih simptoma što je bio i cilj ovog diplomskog rada.

2. POSTUPCI DIJAGNOSTIKE U BIOMEDICINI

Biomedicina ili medicinska biologija [1] predstavlja granu medicine u kojoj se primjenjuju biološka, odnosno druga prirodna i znanstvena načela u kliničkoj praksi. Biomedicina obuhvaća također i grane poput bio informatike, mikrobiologije, laboratorijsku medicinu i tome slično.

Biomedicina je temelj laboratorijske dijagnostike kao i modernog zdravstva te se zasniva na molekularnoj biologiji.

2.1 Računalne tehnologije kao pomoć u dijagnostici

Prema [7], računalne tehnologije imaju veliku važnost u dijagnostici te je nezamislivo u današnje vrijeme uopće raditi, poslovati ili se brinuti o podacima na drugi način. Uz pomoć informacijskih i komunikacijskih tehnologija lakše je ostvariva veza za razmjenu informacija kao i upravljanje njima samima, te se može u bilo kojem trenutku doći do informacija o pacijentu neovisno u kojem dijelu zemlje se nalazili što uvelike olakšava rad liječnicima, bolnicama te svim pružiteljima medicinskih usluga. Veliku važnost informacijske i komunikacijske tehnologije ima u farmaceutskoj industriji jer pomažu pri procijeni te istraživanju lijekova. Informacijska i komunikacijska tehnologija je u velikoj mjeri pomogla u poboljšanju kvalitete zdravstvene zaštite diljem svijeta.

Slijede primjeri računalnih tehnologija u dijagnostici:

Praćenje pacijenta (Patient monitoring) – računala se koriste za praćenje kritično bolesnih pacijenata koji se nalaze na intenzivnom odjelu. Pacijent ima na sebi priključene senzore koji prate otkucaj srca, pulsa, krvnog tlaka, disanja te aktivnosti koje se događaju u mozgu. Ukoliko se stanje pacijenta pogorša alarm se oglašava te to doziva medicinsko osoblje da je pacijent u lošem stanju.

Zapisi o pacijentima (Patient records) – Sve bolnice imaju organiziranu bazu podataka u koje se pohranjuju sve informacije o pacijentima. Uz pomoć baza podataka lakše se prebacuju informacije iz odjela u odjel kao što se prebacuju od bolnice do bolnice.

Svaki liječnik na osnovu baze podataka provjerava prošle nalaze, zakazane termine, obavljene preglede itd.

U ovoj web aplikaciji se kao i u prethodnim primjerima nalazi organizirana baza podataka u koju se pohranjuju informacije. Također, liječnik ima uvid u prošle nalaze te obavljene preglede. Za razliku od praćenja pacijenta pomoću senzora, ova web aplikacija prati pacijenta na temelju njegovih prethodnih dijagnoza.

2.2 Postupci dijagnostike temeljeni na strojnom učenju

Strojno učenje [9] bavi se načinom na koji strojevi uče iz iskustva. Mnogi znanstvenici pojam strojnog učenja uspoređuju s umjetnom inteligencijom. Umjetna inteligencija je dio računalne znanosti u kojoj je cilj računala učiniti pametnijima. Osnovni je zahtjev učenje kao inteligentno ponašanje. Algoritmi strojnog učenja su od samog početka bili korišteni za analizu u medicini. Danas strojno učenje predstavlja jeftin i dostupan način prikupljanja, odnosno pohrane podataka. Sve bolnice su opremljene uređajima koji prikupljaju podatke i dijele drugim sustavima. Prikupljeni podaci o ispravnim dijagnozama su dostupni u obliku medicinskih zapisa u svakom odjelu kao i u svakoj bolnici svijeta. Posao liječnika je unos pacijentovih zapisa s poznatom ispravnom dijagnozom u računalni program koji tada pokreće algoritam učenja što predstavlja veliko pojednostavljenje te se zaključak može donijeti iz opisa slučajeva koji su rješavani u prošlosti.

Kao jedan od primjera strojnog učenja je Bayesov algoritam koji daje odlične rezultate u analizi podataka u medicini. Bayes algoritam je jednostavan, ali može pružiti velika objašnjenja koja su potvrdili i sami liječnici. Jedan je od najučinkovitijih algoritama koji može nadmašiti i one najnaprednije algoritme u medicinskim kao i nemedicinskim problemima. Bayesov algoritam pomogao je 1993. godine kada je Spiegehalter sa suradnicima tijekom nekoliko mjeseci razvio ekspertni sustav na Bayesovim mrežama vezanih uz dijagnosticiranje bolesti srca kod novorođenih beba. Također, još jedan algoritam je korišten u medicini, a to je stablo odluke. Stablo odluke je korišteno 1984. godine (Kononenko) te 1987. godine (Cestnik i suradnici) gdje se primjenilo na probleme u onkologiji (lokalizacija primarnog tumora, karcinoma dojke, limfografija),

urologiji (smetnje donjeg urinarnog traka). Godine 1985. se stablo odluke koristi za dijagnozu bolesti štitnjače koja se razvijala do 1987. godine.

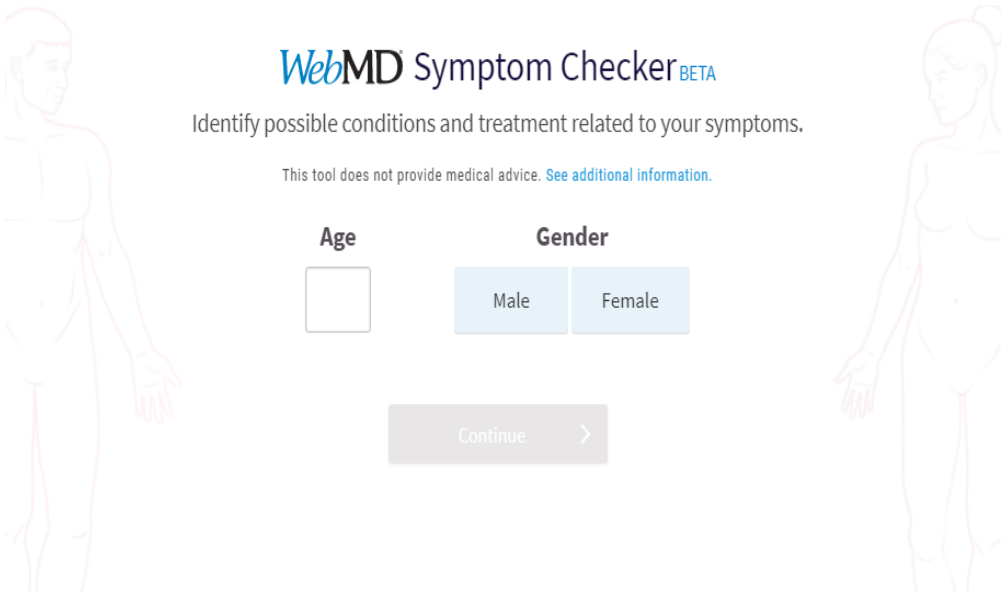
Google Inc. je izradio sustav [8] za borbu protiv raka gdje je dokazano da je strojno učenje bilo uspješnije nego dijagnoze patologa. Primjenjeno je strojno učenje, točnije prediktivna analiza i prepoznavanje uzoraka gdje je točnost bila nevjerojatnih 89%.

2.3. Prikaz postojećih rješenja

Postoji niz web aplikacija otkrivaju dijagnozu na temelju simptoma bez odlaska liječniku. Dijagnoza na temelju tih simptoma nema visoku točnost jer svaki pojedini simptom ne nosi istu težinu kod raznih bolesti. Iako se napravi dijagnoza pomoću web aplikacije, predlaže se odlazak liječniku.

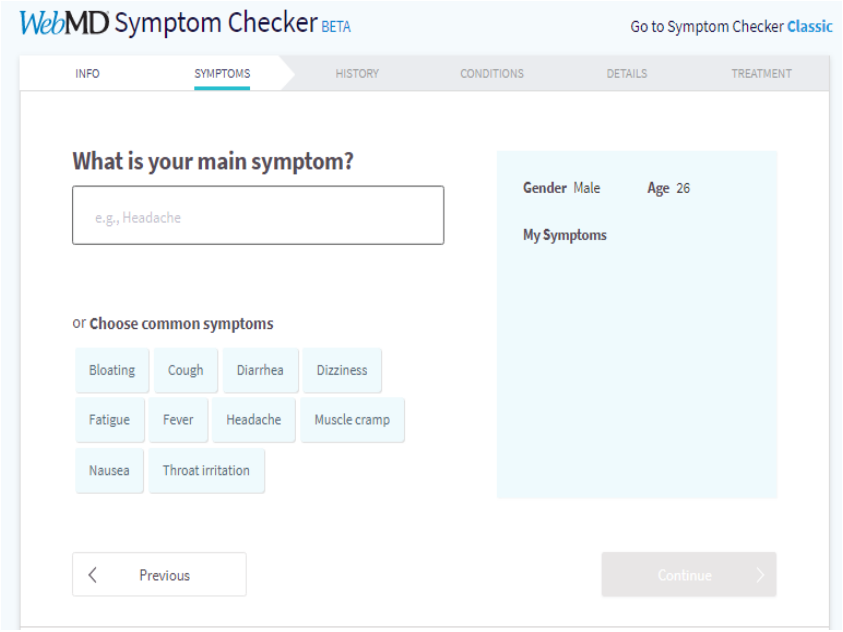
Na internetu postoji nekolicina web aplikacija kao i aplikacija koje ne zahtijevaju internetsku vezu. Većina web aplikacija zahtjeva registraciju korisnika, ali to nije uvjet.

Na slici 2.1 je vidljivo početno sučelje internetskog portala WebMDSymptomChecker [27] koji pruža korisnicima informacije o njihovoj potencijalnoj bolesti. Aplikacija se sastoji od niz procedura za unos podataka kako bi dijagnoza bila što točnija. Internetski portal je u radu od 2005. godine uz stalna poboljšanja.



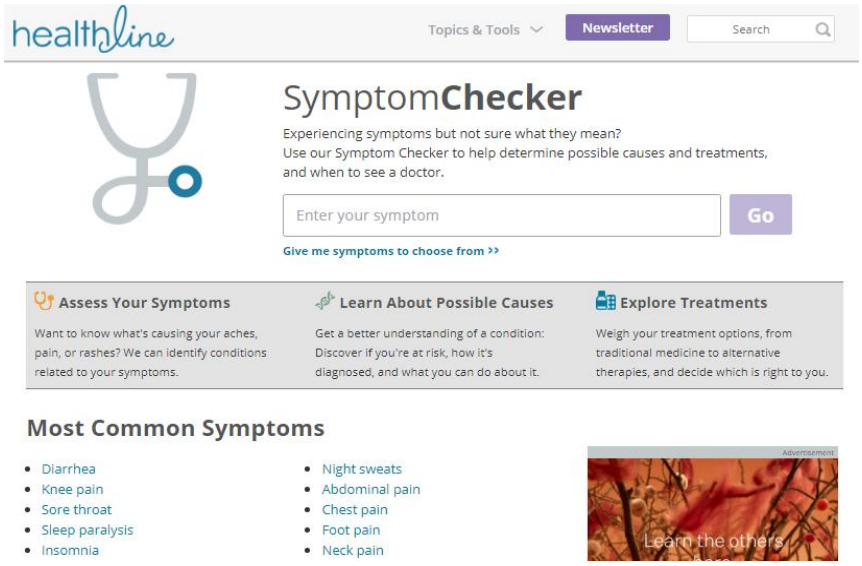
Sl.2.1. Početni sučelje internetskog portala WebMDSymptomChecker

Nakon unosa godina i određivanja spola, aplikacija nastavlja na drugi korak koji je prikazan na slici 2.2. U drugom koraku se unose simptomi koje korisnik osjeća, a na temelju unosa većeg broja simptoma lakše se može dobiti dijagnoza.



Sl. 2.2. Unos simptoma na internetskom portalu WebMDSymptomChecker

Kao još jedan primjer dijagnosticiranja bolesti online je i aplikacija Healthline [28] koja je prikazana na slici 2.3.



Sl.2.3. Stranica internetskog portala Healthline

Podaci se unose u tražilicu te ona na temelju unosa prikazuje koji se simptomi nalaze u bazi podataka. Nakon unosa simptoma, dobije se prijedlog simptoma.

Dva navedena internetska portala su jedni od najpopularnijih te nude niz drugih mogućnost uz samo dijagnosticiranje. Oba internetska portala su navela javljanje liječniku za detaljnije informacije, kao i da sve proizvode i postupke koje posjetitelj radi, radi na svoj vlastiti rizik.

Za razliku od navedenih aplikacija, aplikacija izrađena u sklopu ovog rada omogućuje izbor simptoma na padajućem izborniku te je ograničena na izbor od minimalno četiri simptoma koji pomažu u sužavanju moguće dijagnoze. U odnosu na prethodno spomenute aplikacije, postoji i razlika u bazi podataka koja je u ovom slučaju manja te obuhvaća mali broj bolesti, kao i simptome koji su povezani s njima.

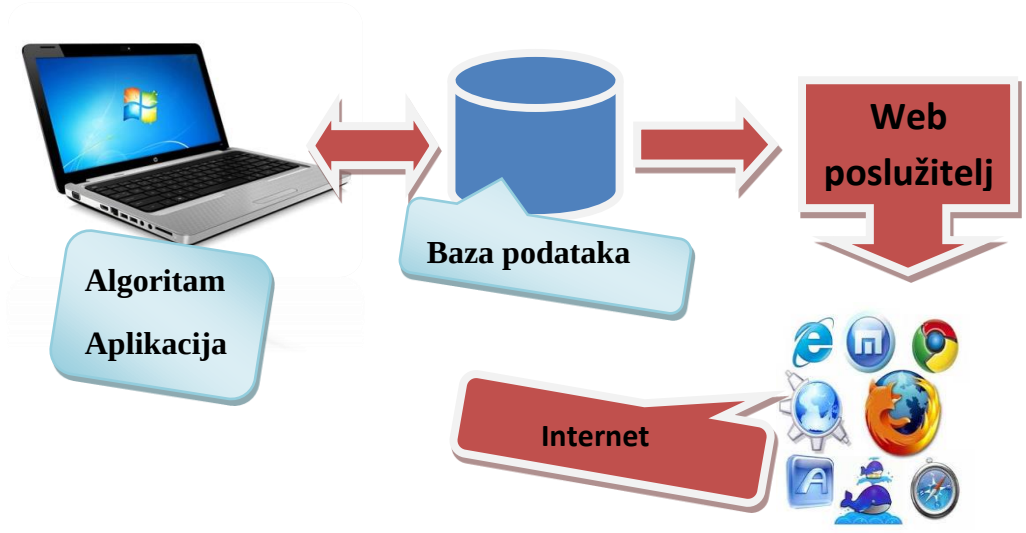
3. MODEL WEB APLIKACIJE ZA DIJAGNOSTICIRANJE BOLESTI

U ovom poglavlju je prikazana ideja rješenja koja prikazuje zamisao kako bi aplikacija trebala funkcionirati, model odlučivanja, popis simptoma i bolesti.

3.1 Ideja rješenja

Cilj ovog diplomskog rada je izrada web aplikacije koja na temelju unosa simptoma određuje dijagnozu, odnosno određuje potencijalnu bolest zbog koje se korisnik loše osjeća. Aplikacija ne daje točnu dijagnozu i svakako se predlaže odlazak k liječniku te se ne preporučuje uzimanje bilo kakvih lijekova na svoju ruku nakon dijagnosticiranja.

Na temelju unesenih simptoma u bazi podataka, zadaća koju aplikacija mora izvršiti je ponuditi potencijalnu bolest koja je povezana s prethodno unesenim simptomima. Ukoliko su simptomi isti kod pojedinih bolesti, prikazat će više mogućih bolesti u postotcima. Postotak bolesti će biti prikazan u grafičkom obliku. Grafički prikaz je u obliku kružnog dijagrama koji ima vrijednost maksimalno 100% te se unutar njega dobije dijagnoza. Da bi aplikacija bila što točnija, potrebno je unijeti minimalno četiri simptoma koja korisnik osjeća. Slika 3.1 prikazuje kako su aplikacija i baza podataka međusobno povezani, a konačni prikaz web aplikacije se ostvaruje putem web poslužitelja.



Sl. 3.1. Idejno rješenje sustava

3.2 Zahtjevi na web aplikaciju

Kako bi web aplikacija bila što više korištena, potrebno je zadovoljiti standarde korisnika koji ju koristi. Neke od stvari koje većina korisnika zahtjeva su jednostavno korištenje, preglednost prilikom unosa i dijagnoze. Uz svu preglednost i jednostavnost, potreban je dizajn koji je prilagođen uređaju na kojem se vrši dijagnosticiranje.

U tablici 3.1 su prikazani zahtjevi koje korisnik želi da aplikacija zadovoljava. Svaki od tih zahtjeva ima prioritet te je potreban što brži odziv kako bi korisnik bio zadovoljan.

Prije pokretanja aplikacije potrebno je dohvatiti podatke iz baze podataka te se nakon toga popunjava sučelje web aplikacije s podacima poput bolesti i simptoma. Korisnik ima pristup podacima te nakon registracije ili prijave može pristupiti dijagnosticiranju ili arhivi prethodnih dijagnoza.

Tablica 3.1 Zahtjevi koje aplikacija mora imati da bi korisnik bio zadovoljan

Redni broj	Prioritet	Zahtjevi korisnika
1	1	Prilikom registracije korisnika potrebno je da se svi podaci nalaze u bazi podataka te da su prikazani u web aplikaciji.
2	1	Potrebno je da se simptomi zapamte u odabiru. Odabir je izvršen simptomima iz baze podataka.
3	1	Korisnik je izvršio odabir i potvrđuje svoj odabir simptoma.
4	1	Web aplikacija prikazuje koja je moguća dijagnoza na grafu. Graf ima zapisane podatke u postotcima za bolesti koje su povučene iz baze na temelju odabranih simptoma.
5	1	Korisnik završava dijagnosticiranje te se dijagnoza šalje na pregled liječniku.

4. KORIŠTENE PROGRAMSKE TEHNOLOGIJE I ARHITEKTURE

Prilikom izrade ovog diplomskog rada korišten je PHP programski jezik, te okvir Laravel. Također, korišten je HTML jezik koji definira strukturu izgleda aplikacije te CSS koji definira izgled stila aplikacije. Sva podaci, od simptoma, korisnika do bolesti se nalaze na phpmyadmin. Za podizanje aplikacije korišten je program XAMPP.

4.1 Korišteni programski jezici i programi

4.1.1 PHP

Prema [11], PHP (*Personal Home Page ili po današnjem nazivu HypertextPreprocessor*) je skriptni programski jezik koji je namijenjen programiranju dinamičkih web stranica. PHP je također program otvorenog koda što znači da je izvorni kod ili dizajn dostupan javnosti na uvid, izmjene te uporabu. Pogodan je za rad, odnosno razvoj mrežnih aplikacija. Točnije, može se kreirati HTML stranica na serveru prije slanja klijentu, stranicu s dinamičkim sadržajem. Nije potrebno pisati niz naredbi koje kao rezultat daju HTML, nego je moguće ugraditi PHP kod u HTML kod sa dvije procesorske naredbe *(naredbu<?php za početak, a za kraj ?>*). Izrada stranice se svodi na rad s predlošcima.

4.1.2 XAMPP

Prema [15], XAMPP je poslužitelj koji ima funkciju web poslužitelja na lokalnom računalu. Instalacijom XAMPP-a u paketu dobijmo Apache 2, MySQL, PHP 4 i 5, PHP My Admin te druge protokole, moguće ga je pokrenuti na svim većim platformama. XAMPP je jedan od najjednostavnijih načina te najpopularnijih programa za pokretanje PHP programa iako postoji i niz drugih poslužitelja,

Besplatna inačica XAMPP-a se može pronaći na službenoj internet stranici klikom na poveznici https://www.apachefriends.org/download.html [14].

XAMPP se sastoji od četiri osnovne komponente:

Apache – web poslužiteljska aplikacija za obrađivanje i isporučivanje web sadržaja na računalo. Najpopularniji je internetski poslužitelj na mreži preko kojeg se pokreće 54% web stranica.

MySQL – Najpopularniji sustav upravljanja bazom podataka na svijetu.

PHP – Skriptni jezik na stranici poslužitelja koji se koristi na najpopularnijim web stranicama uključujući Facebook i WordPress. Open source koji je relativno lagan za naučiti i savršeno radi sa MySQL-om što ga čini najpopularnijim izborom za web programiranje.

Perl – dinamički jezik visoke razine jezik koji se koristi u mrežnom programiranju te administraciji sustava. Manje je popularan u svrhu razvoja web stranica.

Pojedine inačice XAMPP-a mogu imati posebne komponente popust phpMyadmina, OpenSSL i nekolicinu drugih da bi se stvorio punopravni web poslužitelj.

4.1.3 Laravel

Prema [26], Laravel je razvojno okruženje otvorenog koda koji je baziran na PHP-u koji je nastao od strane Taylora Otwella kako bi se osigurano napredniji okvir. Prva inačica Laravel 1 se pojavila u 2011. godini. Velika promjena nakon niza izdanja je došla na Laravel 5.1 verziji gdje je objavljeno da će inačica 5.1 imati dugoročnu podršku. Laravel ima svoj alat koji umanjuje posao oko izrade dizajna i samim time je funkcionalnije. Alat za predloške se naziva Blade. Instalacija Laravela [16] nije toliko zahtjevna, a postupci su navedeni u tekstu ispod. Prije instalacije Laravela potrebno je preuzeti XAMPP (nije uvjet jer postoje i drugi programi za imitiranje WEBHOST servera) zbog MySQL baze podataka. Nakon instaliranja XAMPP-a potrebno je s https://getcomposer.org/download preuzeti composer koji je paket za PHP programski jezik te daje pojedine biblioteke za rad.

Postupak instalacije composera, te kreiranje novog laravel projekta:

- Instalacija composera naredbom – composer globalrequire"laravel/installer"
- Kreiranje projekta – composer create-project--prefer-distlaravel/laravel
- Pokretanje projekta – php artisan serve

Nakon što se pokrene projekt, on se podiže na http://localhost:8000

4.1.4 HTML

Prema [19], HTML (HyperTextMarkupLanguage) prezentacijski je jezik za izradu web stranica. HTML oblikuje sadržaj te se pomoću njega stvaraju hiperveze hipertekst dokumenata. Jednostavan je za korištenje i vrlo lako ga je naučiti te je zbog toga toliko prihvaćen i popularan. HTML nije programski jezik i programeri ga ne koriste često jer se njime ne može odraditi ni najjednostavnija zadaća iz razloga što HTML prikazuje samo tekstualne datoteke s .html ili .htm ekstenzijom. HTML se prvi puta pojavio javno 1991. godine, a zaslužan za to je Tim Barnes-Lee.

4.1.5 CSS

Prema [20], CSS *(CascadingStyleSheets)* stilski je jezik koji se koristi prilikom opisa prezentacije dokumenta napisanog u nekom prezentacijskom jeziku tipa HTML-a. CSS se koristi za vizualni izgled web stranice te izgled korisničkih sučelja koji su prethodno zapisani u HTML-u.

Sintaksa CSS-a se svodi na sljedeće što je prikazano na slici 4.1.

```
H1 {boja:plava; veličina fonta:12px;}
```

Slika 4.1 Sintaksa CSS-a

Selektor prikazuje koji HTML element se želi urediti dok su u vitičastim zagradama prikazane deklaracije za koje postavljamo parametre. Deklaracije su odvojene točkama te postoji jedna ili više deklaracija. U slučaju na slici postoji deklaracija za boju i deklaracija za veličinu fonta. U CSS obliku zapis je prikazan ispod.

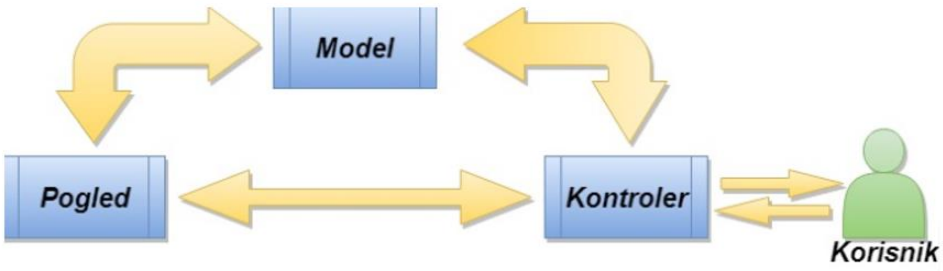
4.1.6 MySQL

MySQL je okruženje pomoću kojeg se upravlja bazama podataka. MySQL baza je stabilna i za nju postoji niz dokumentiranih modula te ekstenzija kao i podršku od niza programskih jezika poput PHP-a, Jave, Pythona itd. Baza je relacijskog tipa te ima najbolji način skladištenja te pretrage velikih količina podataka. Postoje osnovni elementi koji se pohranjuju u bazu poput entiteta koji predstavlja npr. osobu, objekt, događaj i sl. Druga važna stavka su relacije koje predstavljaju relacije između raznih entiteta. Relacije se dijele u četiri skupine, a to su jedan prema jedan, jedan prema više, više prema jedan te više prema više. MySQL je kreirala švedska tvrtka 1995. godine. MySQL je napisan u programskim jezicima C++ i C.

4.2 Korištene programske arhitekture

4.2.1 MVC

Prema [29], MVC je model koji je nastao 1970. godine. Model i domena su odvojeni od korisničkog sučelja. Upravo zbog toga je održavanje i testiranje aplikacije jednostavno. MVC dijeli aplikaciju na tri različita dijela, a to su modeli, pogledi i kontroleri kao što je prikazano na slici 4.2.



Slika 4.2 MVC model

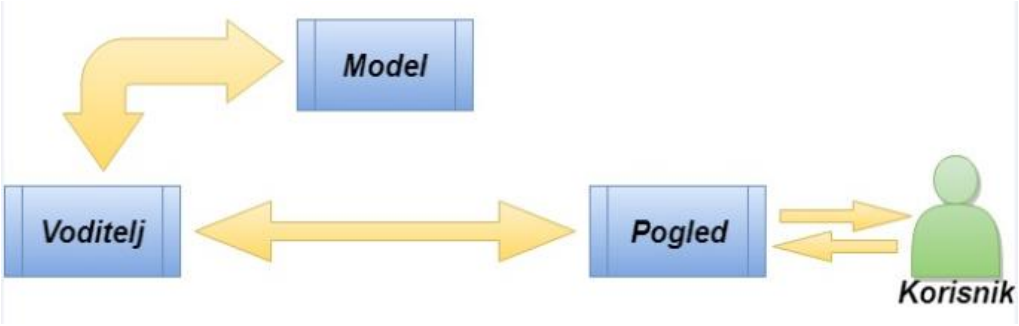
Modeli (Model) – predstavlja kolekciju klasa koje objašnjavaju poslovni model i model podataka gdje se nalaze operacije pristupa datotekama. Također, definira pravila za podatke i način na koji se podaci mogu mijenjati i manipulirati.

Pogledi (View) – predstavljaju komponente korisničkog sučelja, a u tu skupinu pripadaju CSS, jQuery, HTML, itd. Na izlazu se prikazuju podaci koji su primljeni od kontrolera.

Kontroleri (Controller) - obrađuje dolazne zahtjeve. Korisnik dobije informacije putem pogleda (View), a nakon toga uređuje podatke putem modela (Model) i prenosi rezultate u poglede (View). Kontroler je posrednik između pogleda i modela.

4.2.2 MVP

Prema [29], MVP je model koji je sličan MVC modelu, a kontroler je zamijenjen voditeljem (Presenter). Ovaj model dijeli aplikaciju na modele, poglede i voditelje kao što je prikazano na slici 4.3.



Slika 4.3 MVP model

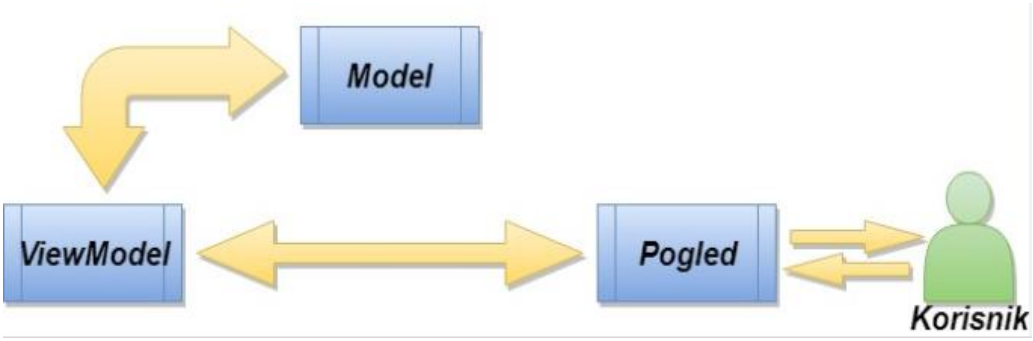
Modeli i pogledi imaju istu svrhu kao što je to opisano prethodno kod MVC modela.

Voditelj (Presenter) – odgovoran je za rješavanje svih događaja korisničkog sučelja koji su vezani za poglede. Prima podatke preko pogleda, a nakon toga obradi podatke putem modela i vrati ih u poglede. Pogledi i voditelj su potpuno odvojeni za razliku od pogleda i kontrolera, a komunikacija se odvija putem sučelja. Voditelj ne obrađuje promet zahtjeva kao što to radi kontroler.

4.2.3 MVVM

Prema [29], MVVM je model koji podržava dvostrano povezivanje podataka između View i ViewModel. Omogućuje širenje promjene unutar stanja od ViewModela do

Viewa. ViewModel koristi promatrača kako bi obavijestio promjene modela viewModela. Struktura MVVM-a je prikazana na slici 4.4.



Slika 4.4 MVVM model

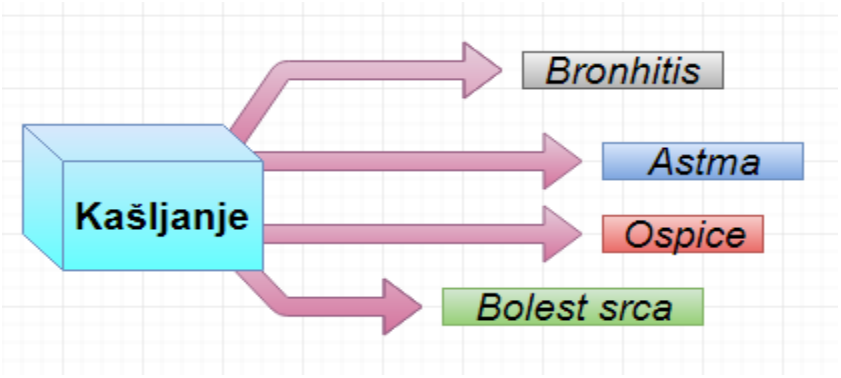
Model i pogledi imaju iste funkcije kao i kod MVC-a te MVP-a

ViewModel – odgovoran je za prikazivanje metoda, naredbi i drugih funkcija koje pomažu u održavanju pogleda, manipuliranju Modela (Model) kao rezultata akcija na Pogledu (View) i pokretanje događaja u samom pogledu.

U ovom slučaju korišten je MVC model. Modelom je odrađena povezanost podataka u tablicama, točnije način komunikacije između tablica te unos novih podataka u tablicu. U ovom radu se model koristio prilikom komunikacije kod tablica za bolesti, simptoma te korisnika općenito.
U prvom slučaju se uspostavlja komunikacija između simptoma i bolesti pomoću identifikacijskog broja za pripadajuću bolest, odnosno simptom. Određena bolest ima svoje pripadajuće simptome, odnosno simptom ima pripadajuću bolest. Kod korisnika je postavljenja povezanost podataka između tablica korisnika i simptoma također pomoću pripadajućeg identifikacijskog broja. Pogledi su iskorišteni kako bi odvojili logiku aplikacije od logike prikazivanja podataka aplikacije, a koriste Blade, alat za stvaranje predložaka. Na kraju, kontroleri su zaslužni za logiku. Pomoću kontrolera su izvedene opcije za spremanje, uređivanje, dodavanje i brisanje bolesti ili simptoma, kao i uređivanje podataka koji su vezani za korisnika.

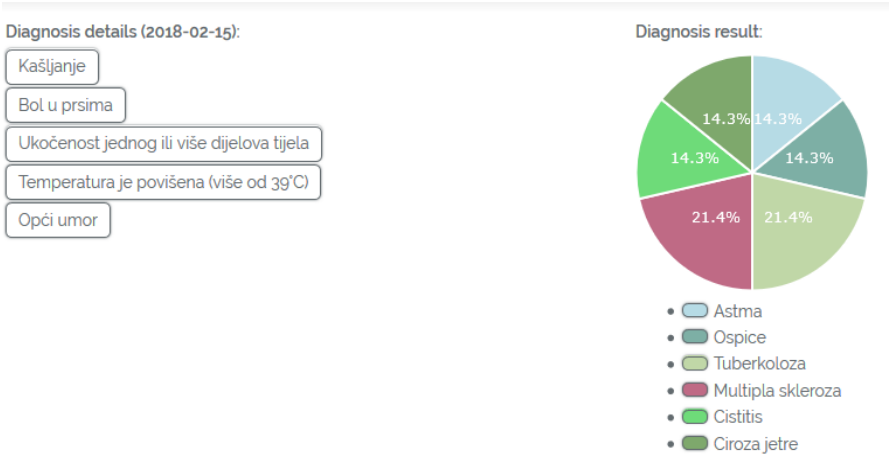
4.3 Postupak klasifikacije

Klasifikacija je ovdje potrebna da bi se simptomi mogli jednostavnije dodijeliti određenoj bolesti. Simptomi se razvrstavaju u skupine te se povezuju s bolestima, dodjeljuje se od pet do osam simptoma koji su povezani s nekom bolešću. Potreban je odabir minimalno četiri simptoma kako bi razvrstavanje bilo moguće, odnosno kako bi aplikacija grupirala simptome vezane uz bolest. Izborom simptoma npr. kašljanja, stvara se grupa bolesti uz koje je simptom kašlja povezan (sl. 4.5), te se stvara postotak dijagnoze unutar kružnog dijagrama.



Slika 4.5 Povezane bolesti sa simptomom kašljanja

Nakon što su grupirani podaci, provjerava se težina pojedinog simptoma, jer nije težina kašljanja ista za bronhitis i za astmu. Težine su određene prije samog početka dijagnosticiranja unutar programskog koda. Od četiri bolesti koje su povezane sa simptomom kašljanja, jedna ima posebnu važnost u ovom slučaju. U ovom slučaju astma ima bitnu važnost, kao infarkt miokarda i moždani udar jer su opasne po život i neovisno o postotku koji joj je izračunat ispisat će se u kružnom dijagramu. Bolesti koje nisu zadovoljile minimalno 20% dijagnosticiranja, neće se prikazati u kružnom dijagramu te će ih zanemariti, a bolesti koje imaju minimalno 20% rasporedi unutar vrijednosti od 100% koje su postavljene u kružni dijagram kao što je prikazan po na slici 4.6.

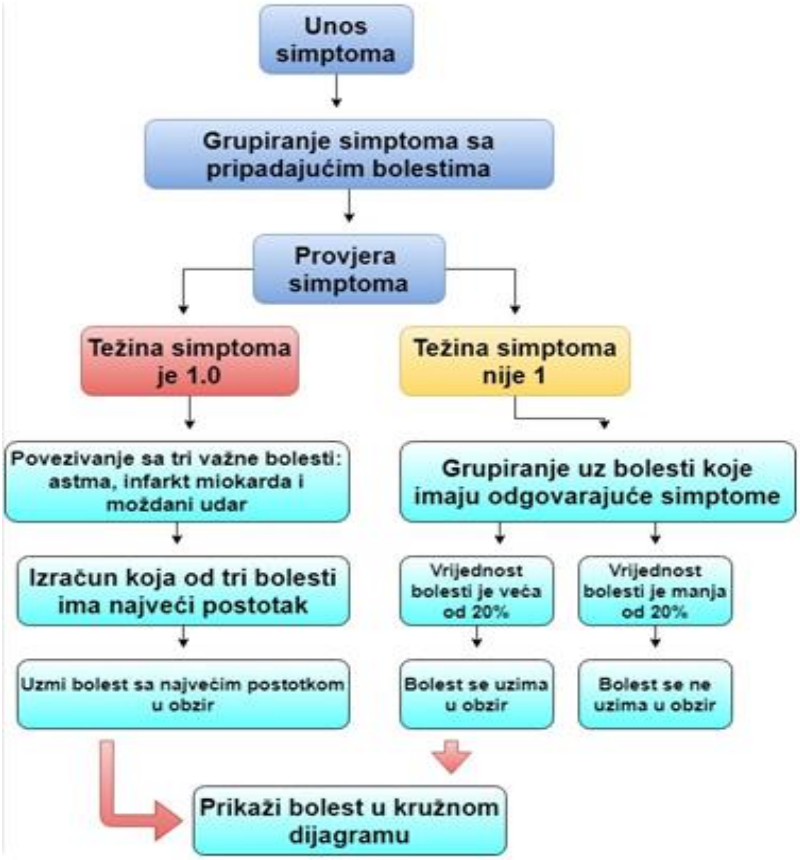


Slika 4.6 Prikaz raspoređivanja bolesti unutar kružnog dijagrama

Ukoliko se prilogom dijagnosticiranja pojave dvije ili sve tri važne bolesti (astma, infarkt miokarda i moždani udar), tada se prvo vrši provjera naziva bolesti, a nakon toga koja od tri bolesti ima najveći postotak te se nju ispisuje u kružni dijagram. Postupak provjere je prikazan u programskom kodu 1. Cijeli postupak izvršavanja aplikacije je prikazan na slici 4.7.

```php
$temp_array = []; //pomocni array za 3 bitne bolesti
foreach($new_array as $key => $value) {
    $total = ($value/$symptom_num)*100;
    // var_dump($total);
    if($total>20){
        $new_array[$key] = ($value/$symptom_num)*100;
        if($key == 'Infarkt miokarda' || $key == 'Astma' || $key == 'Moždani udar'){ //Ovaj dio
        gleda naziv bolesti pa ubacuje u temp array
            $temp_array[] = [$key => ($value/$symptom_num)*100];
        }
    }else{
        if($key == 'Infarkt miokarda' || $key == 'Astma' || $key == 'Moždani udar'){ //Ovaj dio
        gleda naziv bolesti pa ubacuje u temp array bez obzira da li je manje od 20% pojavljivanje
            $new_array[$key] = ($value/$symptom_num)*100;
            $temp_array[] = [$key => ($value/$symptom_num)*100];
        }else{
            unset($new_array[$key]);
        }
    }
}
```

Programski kod 4.1. Provjera naziva bolesti i njihov ispis neovisno o postotku



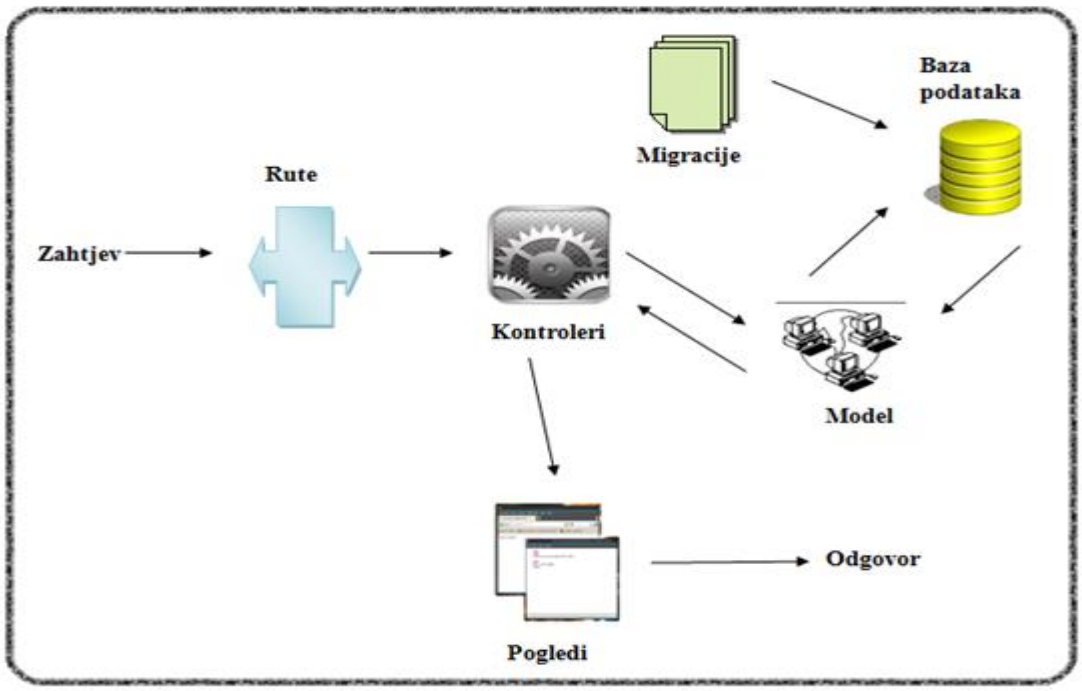
Slika 4.7 Postupak izvršavanja aplikacije

Stablo odluke je postupak klasificiranja koji je sličan rješenju u ovom diplomskom radu. Jedna odluka vodi do druge, a nakon toga se odluka grana na više opcija i tako do konačne odluke nakon koje se prikaže rezultat u kružnom dijagramu.

5. PROGRAMSKO RJEŠENJE SUSTAVA

U prethodnim poglavljima su objašnjeni predlošci i aplikacije koji su se koristile u izradi ove web aplikacije te dio o biomedicini koji daje bolji uvid u rad ove web aplikacije. Ovaj dio poglavlja, kao i sljedeća potpoglavlja, prikazuju funkcionalnost web aplikacije te strukturu programskog rješenja koje je prikazano na slici 5.1.

Diplomski rad je izrađen u programu Sublime koji se koristi kao uređivač teksta za niz programskih jezika. Prije korištenja programa potrebno je preko komandne liniji (CMD u Windows operacijskom sustavu) izraditi projekt, odnosno u ovom slučaju, diplomski rad.



Slika 5.1. Struktura programskog rješenja

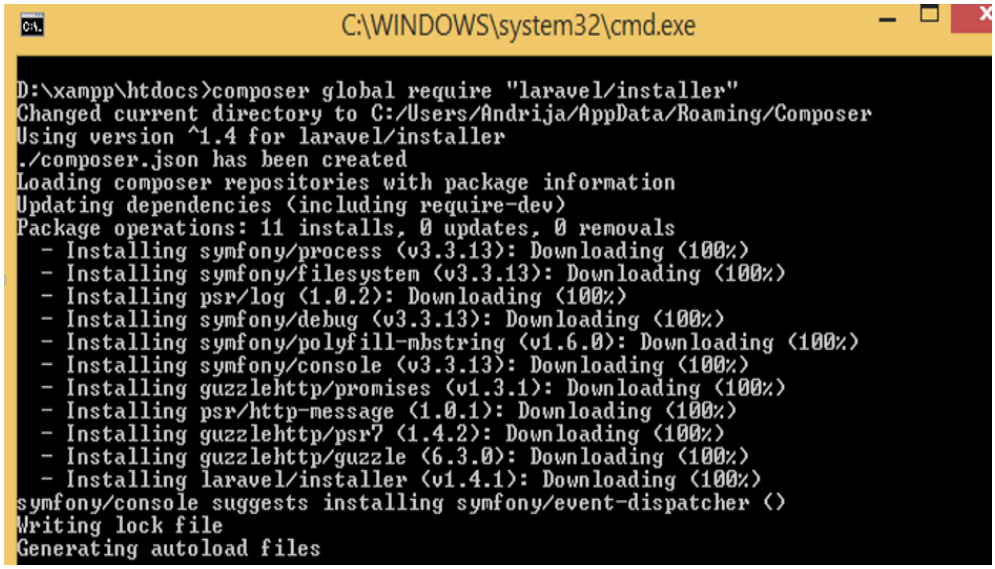
5.1 Kreiranje projekta

Na početku kreiranja projekt potrebno je zadovoljiti nekoliko uvjeta, odnosno potrebno je da poslužitelj zadovolji nekoliko uvjeta kako bi instalacija Laravela bila moguća.

Uvjeti koje poslužitelj mora zadovoljavati:

- OpenSSL PHP ekstenziju
- PDO PHP ekstenziju
- Mbstring PHP ekstenziju
- Tokenizer PHP ekstenziju
- XML PHP ekstenziju

Nakon što su uvjeti zadovoljeni, može se instalirati Laravel pomoću komposera koji mora dohvatiti instalaciju na način da pokrenemo CMD te naredbom composer globalrequire"laravel/installer"pokrenemo te provedemo njegovu instalaciju (Sl. 5.2).



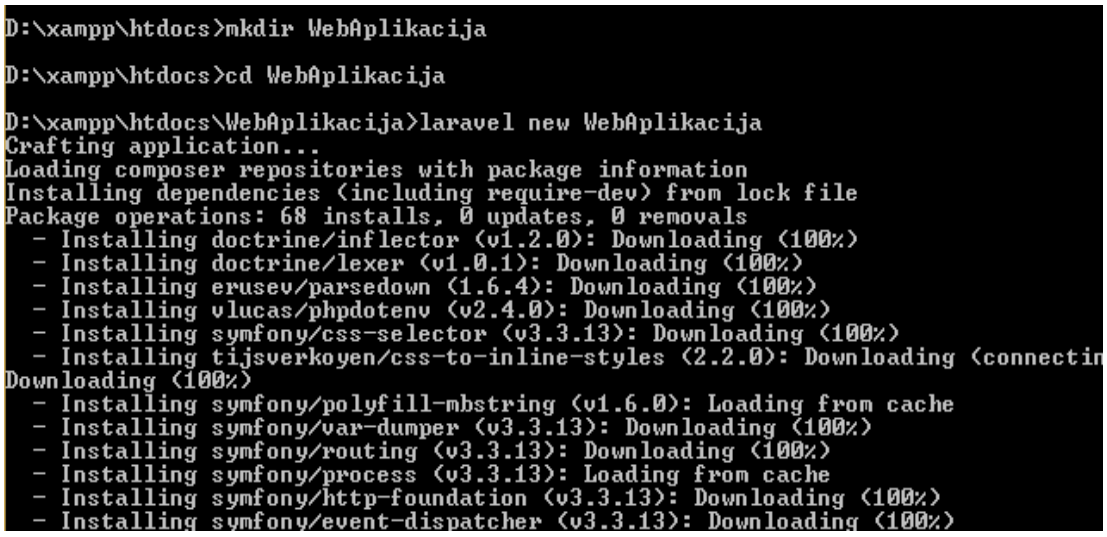
Sl.5.2. Instalacija Laravela pomoću composera

Ukoliko instalacija prođe bez ikakvih problema, potrebno je stvoriti novi projekt. Jedna od bitnijih stavki, odnosno ključni dio je da se točno odredi direktorij u koji se novi projekt treba smjestiti.

Koriste se tri naredbe prikazane u tablici 5.1 koje su bitne za stvaranje projekta, a započinje se stvaranjem direktorija u xampp-ovom dijelu u direktoriju */htdocs*. Prva naredba je mkdir koja služi za stvaranje direktorija s time da mu dodijelimo ime. Sljedeća bitna naredba je da se laravel projekt stvori unutar te mape te je potrebno odrediti putanju naredbom cd uz dodatak imena projekta koji je prethodno zadan. Na kraju se može instalirati laravel naredbom laravel new uz dodatak naziva projekta.

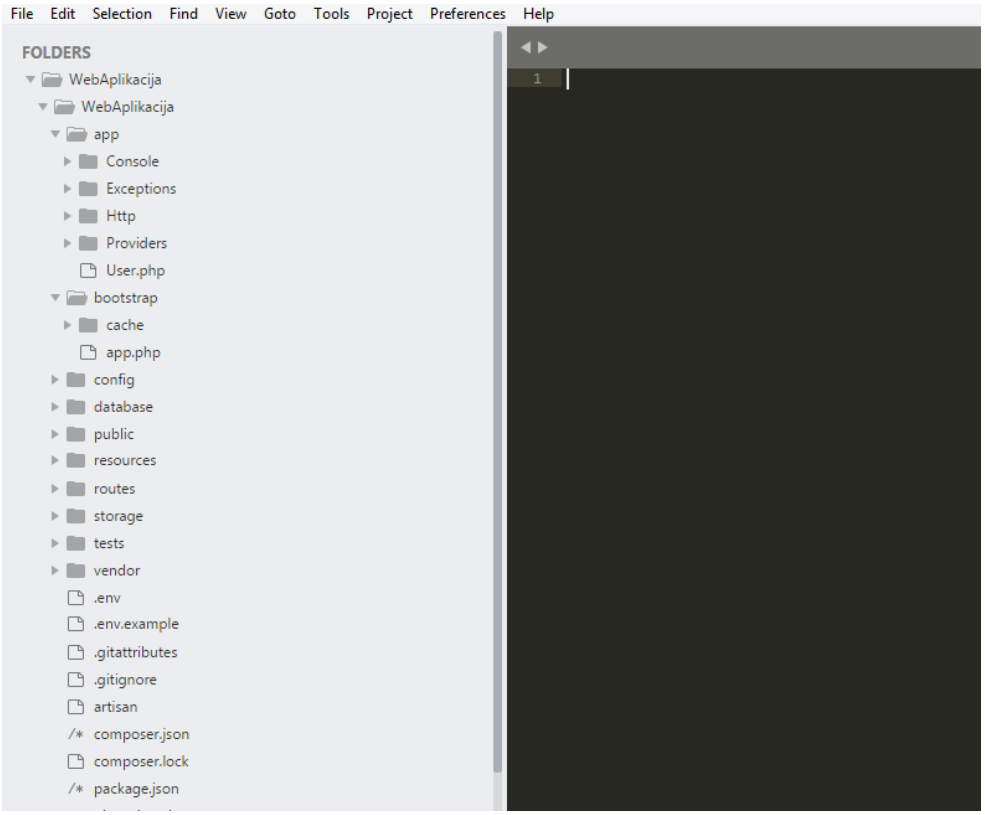
Tablica 5.1 Stvaranje novog projekta

mkdirWebAplikacija	Stvaranje direktorija po imenu WebAplikacija
cd WebAplikacija	Određivanje putanje, odnosno ulazak u direktorij WebAplikacija
laravelnewWebAplikacija	Instalacija novog laravel projekta pod nazivom WebAplikacija



Sl. 5.3. Procedura instalacije novog Laravel projekta

Nakon što se obavi instalacija svih osnovnih dijelova, stvoren je novi projekt. Novi stvoreni projekt je vidljiv u našem editoru *Sublime* na sl. 5.3 .



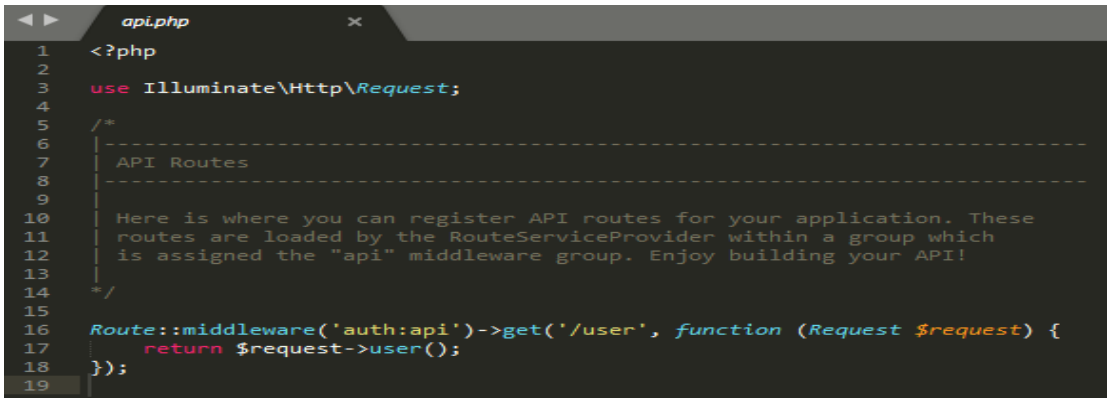
Sl.5.3. Prikaz stvorenog projekta

Kako bi projekt bio vidljiv, potrebno je pokrenuti poslužitelj naredbom *php artisan serve (sl. 5.4)* uz uvjet da se putanja postavi unutar novostvorenog projekta. Nakon što se aplikacija pokrene, bit će vidljiva na internet lokaciji "http://127.0.0.1:8000/". Treba dodati da je uvjet također i pokretanje Apache i MySQL-a u XAMPP-u.

Sl.5.4. Pokretanje poslužitelja

5.2 Definiranje projekta

Prva bitna stavka rada su rute koje služe za prosljeđivanje kontrole. U klasi Route se nalaze i četiri vrlo bitne skripte koje su postavile sve rute Laravela, a to su *api.php, channels.php, web.php i console.php.* Rute se postavljaju u grupe što omogućuje lakše dijeljenje atributa između zadanih ruta kako bi došlo do smanjenja ponavljanja koda. Grupa ruta se definira unutar skripte api.php (Programski kod 5.1), te se podaci zapisuju u array formatu.

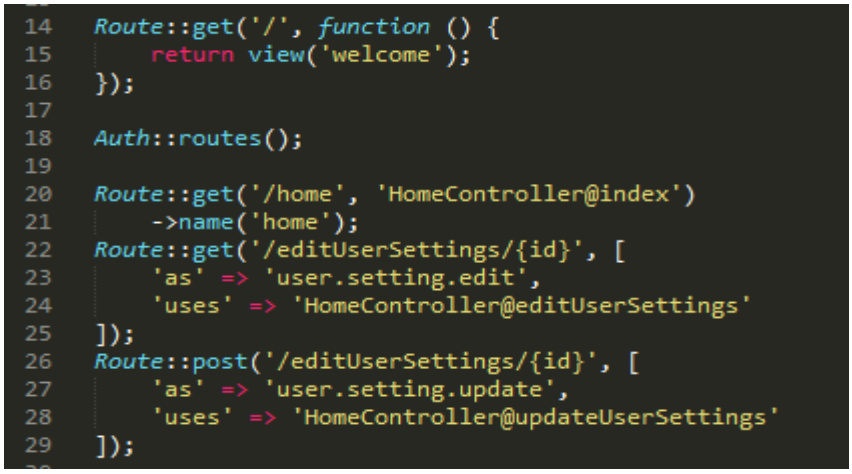


Programski kod 5.1. Definiranje rute unutar skripte api.php

middleware – pruža mehanizam za filtriranje HTTP zahtjeva. Ako korisnik aplikacije postoji onda ga middleware potvrđuje i pušta dalje u aplikaciju, a ukoliko ne postoji zadržava mu pristup.

auth:api – predstavlja pristupni token i uključuje autentifikaciju za nastavak putanje.

U programskom kodu 5.1 vidljivo je da će korisnik pristupiti nastavku aplikacije ukoliko postoji u bazi. U slučaju da korisnik ne postoji, zahtijeva se ponovni unos podataka. Sljedeći bitan dio rada je skripta web.php u kojoj se nalazi popis kontrolera vezanih uz korisnika, bolesti, dijagnoze te simptome. U programskom kodu 5.2 je vidljiva ruta koja nam daje povratnu informaciju za početni zaslon Laravel projekta, kao i *HomeController* o kojem će kasnije biti više riječi. Unutar tablice 5.2 opisane su linije koda, odnosno njihove funkcije.



```
14  Route::get('/', function () {
15      return view('welcome');
16  });
17
18  Auth::routes();
19
20  Route::get('/home', 'HomeController@index')
21      ->name('home');
22  Route::get('/editUserSettings/{id}', [
23      'as' => 'user.setting.edit',
24      'uses' => 'HomeController@editUserSettings'
25  ]);
26  Route::post('/editUserSettings/{id}', [
27      'as' => 'user.setting.update',
28      'uses' => 'HomeController@updateUserSettings'
29  ]);
```

Programski kod 5.2. Poziv korisnika sa pripadajućim id-om i HomeController-om

Tablica 5.2 Popis funkcija koje se obavljaju u programskom kodu 5.2.

Linija 14 i 15	Ruta preusmjerava na početnu stranicu
Linija 18	Klasa koja pomaže generirati sve puteve koji su potrebni za autentifikaciju korisnika
Linija 20 i 21	Poziva HomeController te prikazuje početnu stranicu
Linije 22, 23 i 24	Poziva se korisnik (user) sa pripadajućim id-om u dio aplikacije za izmjenu podataka

Sljedeći popis ruta su putanje za bolesti, simptome i za samu dijagnozu. Na programskom kodu 5.3 su prikazane rute koje su bitne za upravljanje bolestima od strane administratora. Administrator ima opcije brisati bolesti, dodavati nove te ih uređivati.

```
32  /*  Sickness */
33  Route::get('/sickness', [
34      'as' => 'sickness.index',
35      'uses' => 'SicknessController@index'
36  ]);
37  Route::post('/sickness', [
38      'as' => 'sickness.store',
39      'uses' => 'SicknessController@store'
40  ]);
41  Route::get('/sickness/{id}', [
42      'as' => 'sickness.edit',
43      'uses' => 'SicknessController@edit'
44  ]);
45  Route::post('/sickness/{id}', [
46      'as' => 'sickness.update',
47      'uses' => 'SicknessController@update'
48  ]);
49  Route::get('/sickness/{id}/delete', [
50      'as' => 'sickness.delete',
51      'uses' => 'SicknessController@destroy'
52  ]);
```

Programski kod 5.3. Uređivanje i spremanje novih bolesti

Tablica 5.3 Popis funkcija koje se obavljaju u programskom kodu 5.3.

Linija 33, 34 i 35	Prikaz stranice bolesti u administratorskom sučelju
Linija 37, 38 i 39	Spremanje nove bolesti u bazu
Linija 41, 42 i 43	Pristup uređivanju bolesti koja je vezana uz pripadajući id-om
Linija 45, 46 i 47	Spremanje uređene bolesti koja je vezana uz pripadajući id
Linija 49, 50 i 51	Brisanje bolesti koja je vezana uz pripadajući id

Programski kod 5.4 prikazuje opcije koje administrator može izvršavati vezano u simptome. Administrator može uređivati, brisati i dodavati nove simptome.

```
55  /*  Symptoms */
56  Route::get('/symptoms', [
57      'as' => 'symptoms.index',
58      'uses' => 'SymptomsController@index'
59  ]);
60  Route::post('/symptoms', [
61      'as' => 'symptoms.store',
62      'uses' => 'SymptomsController@store'
63  ]);
64  Route::get('/symptoms/{id}', [
65      'as' => 'symptoms.edit',
66      'uses' => 'SymptomsController@edit'
67  ]);
68  Route::post('/symptoms/{id}', [
69      'as' => 'symptoms.update',
70      'uses' => 'SymptomsController@update'
71  ]);
72  Route::get('/symptoms/{id}/delete', [
73      'as' => 'symptoms.delete',
74      'uses' => 'SymptomsController@destroy'
75  ]);
```

Programski kod 5.4. Opcije kojima administrator raspolaže vezano uz simptome

Tablica 5.4 Popis funkcija koje se obavljaju u programskom kodu 5.4.

Linija 56, 57 i 58	Prikaz stranice simptoma u administratorskom sučelju
Linija 60, 61 i 62	Spremanje simptoma u bazu
Linija 64, 65 i 66	Pristup uređivanju simptoma koja je vezana uz pripadajući id-om
Linija 68, 69 i 70	Spremanje uređenog simptoma koji je vezan uz pripadajući id
Linija 72, 73 i 74	Brisanje simptoma koji je vezan uz pripadajući id

U programskom kodu 5.5 prikazane su mogućnosti koje postoje za rad s dijagnozama kao što su odgovaranje na izvršenu dijagnozu korisnika, uvid u dijagnozu korisnika, pregled obavljenih dijagnoza od nekolicine korisnika, brisanje dijagnoza i slično.

```
78  /*  Diagnosis */
79  Route::get('/diagnosis', [
80      'as' => 'diagnosis.user.symptom',
81      'uses' => 'SymptomsController@userSymptom'
82  ]);
83  Route::post('/diagnosis', [
84      'as' => 'diagnosis.user.symptom.insert',
85      'uses' => 'SymptomsController@userSymptomInsert'
86  ]);
87  Route::get('/diagnosis/{id}', [
88      'as' => 'diagnosis.symptom.details',
89      'uses' => 'SymptomsController@userSymptomDetails'
90  ]);
91  Route::get('/adminDiagnosis', [
92      'as' => 'diagnosis.admin',
93      'uses' => 'SymptomsController@adminDiagnosisIndex'
94  ]);
95  Route::post('/diagnosis/{id}', [
96      'as' => 'diagnosis.symptom.details.reply',
97      'uses' => 'SymptomsController@adminDiagnosisReply'
98  ]);
99  Route::post('/diagnosis/{id}/deleteReply', [
100     'as' => 'diagnosis.admin.reply.delete',
101     'uses' => 'SymptomsController@adminDiagnosisDeleteReply'
102 ]);
103 Route::post('/diagnosis/{id}/delete', [
104     'as' => 'diagnosis.user.delete',
105     'uses' => 'SymptomsController@userDiagnosisDelete'
106 ]);
```

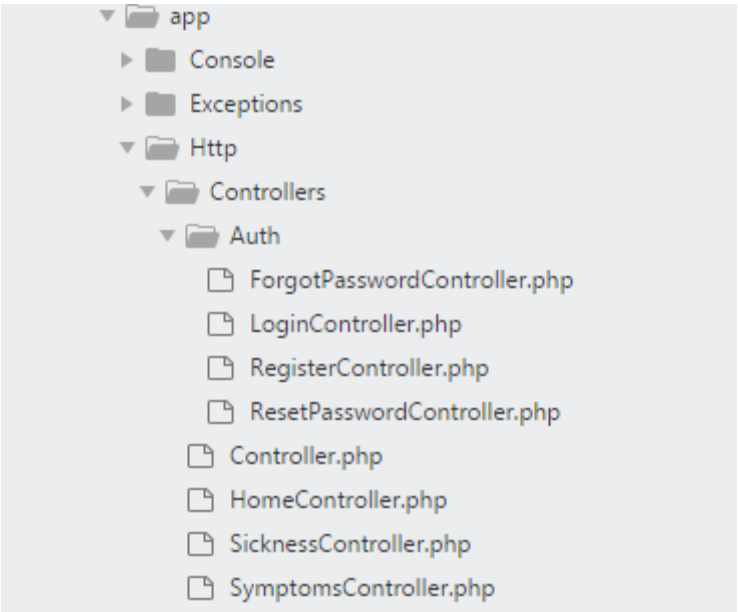
Programski kod 5.5. Rad s dijagnozama

Tablica 5.5 Popis funkcija koje se obavljaju u programskom kodu 5.5.

Linije 79, 80 i 81	Prikazuje se sučelje za dijagnozu.
Linije 83,84 i 85	Pokretanje novog dijagnosticiranja.
Linije 87, 88 i 89	Prikazuje se izvršena dijagnoza povezana uz id odrađene dijagnoze.
Linije 91, 92 i 93	Prikazuje se sučelje u kojem je popis korisnika koji su odradili dijagnosticiranje, Administratorsko sučelje.
Linije 95, 96 i 97	Detalji dijagnoze. Slanje odgovora korisniku.
Linije 99, 100 i 101	Brisanje odgovora od strane administratora.
Linije 103, 104 i 105	Brisanje dijagnoze u administratorskom sučelju.

Sljedeća važna stavka rada su kontroleri koji si zaduženi za upravljanje i prihvaćanje podataka koje unosi ili zahtjeva korisnik. Kao što je bilo vidljivo u rutama, definirano je da rute pozivaju pojedine klase kontrolera koje obrađuju pojedine zahtjeve.

Kontroleri su vidljivi u mapi App/Http/Controllers (sl. 5.6). Stvaranje kontrolera se izvršava naredbom *phpartisanmake:controller*uz naziv kontrolera.



Sl.5.6.Prikaz svih kontrolera koji se nalaze u projektu

Tablica 5.6 Popis kontrolera koji se nalaze unutar App/Http/Controllers

ForgotPasswordController.php	Kontroler za zaboravljenu lozinku u web aplikaciji.
LoginController.php	Kontroler za prijavu u web aplikaciju.
RegisterController.php	Kontroler za registraciju korisnika..
ResetPasswordController.php	Kontroler za resetiranje lozike.
Controller.php	Kontroler koji potvrđuje zahtjeve i odobrava ulazak u web sučelje.
HomeController.php	Kontroler koji služi za rad korisnika, njegov unos podataka
SicknessController.php	Kontroler za bolesti. Mogućnost uređivanja, brisanja i dodavanja novih bolesti.
SymptomsController.php	Kontroler za simptome. Mogućnost uređivanja, brisanja i dodavanja novih simptoma.

Na programskom kodu 5.6 vidljive su skripte vezane uz prijavu korisnika, registraciju korisnika, ponovno postavljanje lozinke kao i važne stavke poput kontrolera za simptome, bolesti te HomeController u kojem se nalaze stavke za uređivanje rola i ažuriranje korisničkih podataka.

```
32      public function editUserSettings($id){
33          if(Auth::user()->isRole('admin')){
34              $user = User::find($id);
35          }else{
36              $user = Auth::user();
37          }
38
39          return view('user.userSettings')->with('user', $user);
40      }
41
```

Programski kod 5.6. Skripta za prijavu korisnika

Uređivanje korisničkih podataka gdje se korisnik pronađe na temelju ID-a koji mu pripada te se stvara prikaz opcije promjena korisničkih podataka.

U programskom kodu 5.7 su vidljive opcije koje administrator može uređivati korisnicima koji su registrirani u bazi.

```
43      public function updateUserSettings($id, Request $request){
44          if(Auth::user()->isRole('admin')){
45              $user = User::find($id);
46          }else{
47              $user = Auth::user();
48          }
49
50          $format_date = explode('-', $request->date);
51
52          if($request->password != null){
53              $user->update([
54                  'name' => $request->name,
55                  'email' => $request->email,
56                  'password' => bcrypt($request->password),
57                  'oib' => $request->oib,
58                  'address' => $request->address,
59                  'city' => $request->city,
60                  'date' => $format_date[2].'-'.$format_date[1].'-'.$format_date[0],
61              ]);
62          }else{
63              $user->update([
64                  'name' => $request->name,
65                  'email' => $request->email,
66                  'oib' => $request->oib,
67                  'address' => $request->address,
68                  'city' => $request->city,
69                  'date' => $format_date[2].'-'.$format_date[1].'-'.$format_date[0],
70              ]);
71          }
72
73          return redirect()->back();
74      }
```

Programski kod 5.7. Administratorske opcije za uređivanje podataka registriranih korisnika

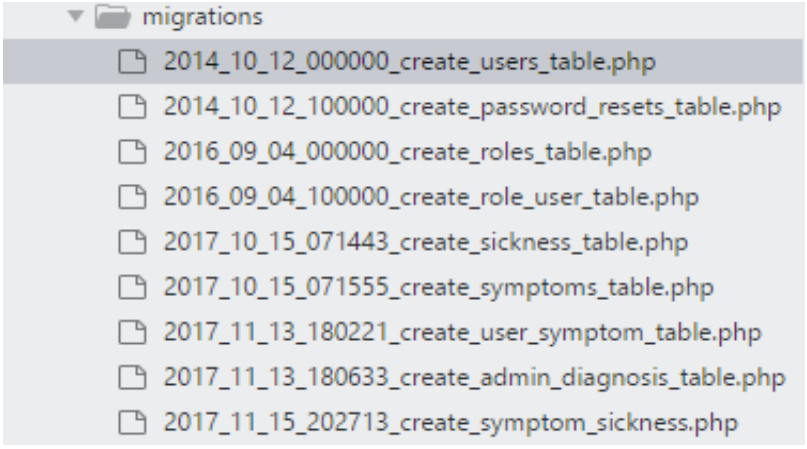
Kontroler za bolesti dopušta opcije poput spremanja, uređivanja, ažuriranja te brisanje podataka koji su uneseni. Kontroler za simptome ima opcije korisničkog unosa simptoma uz mogućnost da korisnik briše i dodaje svoje simptome ukoliko je pogriješio prilikom unosa. Također se u kontroleru za simptome nalaze i opcije koje administrator obavlja nakon što je korisnik dobio početnu dijagnozu.

5.3 Baza podataka

Baza podataka u Laravelu je jednostavna jer olakšava komunikaciju. Na ovom radu se koristila baza MySQL koja je zadana već na početku.

Sve informacije vezane uz bazu podataka je moguće pronaći pod */system/database* gdje su vidljive tri stavke, a to su factories, migrations i seeds. Važna stavka su migracije koje se šalju na phpMyAdmin besplatnu bazu. Migracije predstavljaju način, odnosno oblik kontroliranja baze podataka. Potrebno je prvo izraditi tablice, a nakon toga ih proslijediti pomoću naredbe *phpartisanmigrate*te nakon toga osvježavati svaku promjenu koja se dogodila naredbom *phpartisanmigrate:refresh --seed –vvv.*

Iz slike 5.8 su vidljive tablice koje su izrađene i vezane uz kreiranje tablica korisnika (usera), uloga (roles), bolesti, simptoma, dijagnoze i dr. Detaljnije značenje pojedinih tablica je opisano u tablici 5.7.

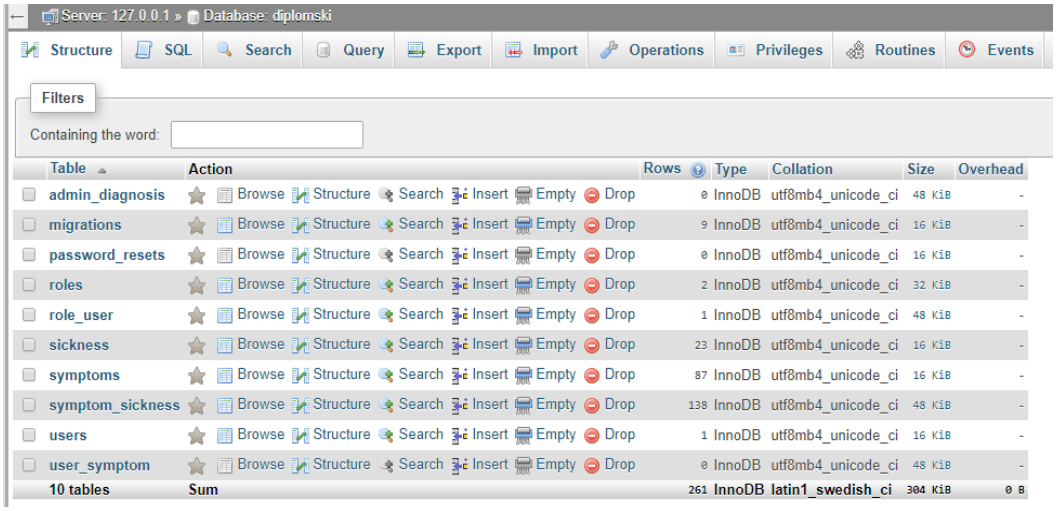


Sl.5.8. Kreirane tablice

Tablica 5.7 Popis kreiranih tablica

create_user_table.php	Kreiranje tablice korisnika.
create_password_resets_table.php	Kreiranje tablice za reset lozinki.
create_roles_table.php	Kreiranje tablice sa rolama koje korisnik ima.
create_role_user_table.php	Tablica koja povezuje role i korisnike.
create_sickness_table.php	Kreiranje tablice bolesti.
create_symptoms_table.php	Kreiranje tablice simptoma.
create_user_symptom_table.php	Tablica koja povezuje korisnike i simptome.
create_admin_diagnosis_table.php	Kreiranje tablice za uvid u dijagnoze.
create_symptom_sicness.php	Kreiranje tablice koja povezuje bolesti i simptome.

Na slici 5.9 je prikazana struktura baze podataka na lokaciji phpMyAdmin.



Sl. 5.9. Struktura baze podataka

Sljedeća važna stavka baze podataka su podaci koji se popunjavaju u gore kreirane tablice.

U programskom kodu 5.8. vidljive su dvije role koje su napravljene. Uloga administratora (admina) predstavlja nam doktora koji daje konačnu dijagnozu dok je rola pod id2 vezana uz običnog korisnika koji se prijavljuje na sustav.

```
15          //Role id 1
16          $adminRole = Role::create([
17              'name' => 'Admin role', //doctor
18              'slug' => 'admin',
19              'description' => 'Admin',
20              'group' => 'default'
21          ]);
22
23          //Role id 2
24          $userRole = Role::create([
25              'name' => 'user role', //normal user
26              'slug' => 'user',
27              'description' => 'User role',
28              'group' => 'default'
29          ]);
30      }
31  }
32
```

Programski kod 5.8. Uloge administratora i korisnika

Sljedeća vrlo važna stavka je popis bolesti koje se nalaze u *SicknessTableSeeder* za koje je moguće odrediti dijagnozu. Također, vrlo je važno naglasiti da se i unutar same aplikacije mogu dodavati nove bolesti (programski kod 5.9) i simptomi. Popis simptoma je vidljiv na programskom kodu 5.10, a nalaze se unutar *SymptomTableSeeder*. Prilikom kreiranja bolesti, svaka bolest dobije svoj identifikacijski broj.

```
13      public function run()
14      {
15          Sickness::create([ 'name' => 'Astma']);
16          Sickness::create([ 'name' => 'Konjuktivitis']);
17          Sickness::create([ 'name' => 'Multipla skleroza']);
18          Sickness::create([ 'name' => 'Cistitis']);
19          Sickness::create([ 'name' => 'Bronhitis']);
20          Sickness::create([ 'name' => 'Prehlada i gripa']);
21          Sickness::create([ 'name' => 'Tuberkoloza']);
22          Sickness::create([ 'name' => 'Anemija']);
23          Sickness::create([ 'name' => 'Ciroza jetre']);
24          Sickness::create([ 'name' => 'Gastritis']);
25          Sickness::create([ 'name' => 'Angina']);
26          Sickness::create([ 'name' => 'Sinusitis']);
27          Sickness::create([ 'name' => 'Nesanica']);
28          Sickness::create([ 'name' => 'Ospice']);
29          Sickness::create([ 'name' => 'Vodene kozice']);
30          Sickness::create([ 'name' => 'Depresija']);
31          Sickness::create([ 'name' => 'Dijabetes tipa 1']);
32          Sickness::create([ 'name' => 'Giht']);
33          Sickness::create([ 'name' => 'Salmoneloza']);
34      }
35  }
```

Programski kod 5.9. Kreiranje bolesti

```php
//Astma
$si = Sickness::findOrFail(1); //Astma id=1 iz tablice Sickness

$sy = Symptoms::create([   //ID = 1
    'name' => 'Kašljanje'
]);
$sy->setToSickness()->attach($si);

$sy = Symptoms::create([  //ID = 2
    'name' => 'Otežano disanje'
]);
$sy->setToSickness()->attach($si);

$sy = Symptoms::create([
    'name' => 'Iritacija nosa' //3
]);
$sy->setToSickness()->attach($si);

$sy = Symptoms::create([
    'name' => 'Iritacija grla' //4
]);
$sy->setToSickness()->attach($si);

$sy = Symptoms::create([
    'name' => 'Bol u prsima' //5
]);
$sy->setToSickness()->attach($si);
```

Programski kod 5.10. Povezivanje bolesti sa simptomima

U tablici 5.8 slijedi popis svih bolesti povezanih s određenim simptomima koji upućuju o kojoj se bolesti radi. Pojedine bolesti imaju i zajedničke simptome.

Tablica 5.8 Popis bolesti i simptoma koji se nalaze u bazi podataka

Bolesti	**Simptomi**
Astma	Kašljanje, otežano disanje, iritacija nosa, iritacija grla i bol u prsima.
Konjuktivitis	Iritacija oka, svrbež, osjećaj stranog tijela u oku, peckanje, crvenilo očiju.
Multipla skleroza	Opća slabost, opći umor, ukočenost jednog ili više dijelova tijela, vrtoglavica, temperatura je povišena (više od 39°C).
Cistitis	Opći umor, temperatura umjereno povišena (38°C - 39°C), povraćanje, učestalo mokrenje, krv u mokraći.
Bronhitis	Slabost, temperatura umjereno povišena (38°C - 39°C), grlobolja, bol u leđima, kašalj.
Prehlada i gripa	Temperatura je povišena (više od 39°C), temperatura umjereno povišena (38°C - 39°C), začepljenost nosa, kihanje, glavobolja, grlobolja.
Tuberkoloza	Temperatura je povišena (više od 39°C), opći umor, krv u iskašljaju, bol u prsima, gubitak težine.
Anemija	Bljeskovi pred očima te zujanje u ušima, gubitak krvi, probavne smetnje, glavobolja, opća slabost.
Ciroza jetre	Opći umor, gubitak apetita, mučnina, nakupljanje tekućine u trbuhu, temperatura je povišena (više od 39°C).
Gastritis	Neprobavljivost, mučnina, povraćanje, gubitak apetita, povraćanje krvi ili sadržaja sličnog kavi.

Angina	Bol u grlu, glavobolja, temperatura umjereno povišena (38°C - 39°C), opći umor, crvenilo ili oticanje krajnika.
Sinusitis	Začepljenost nosa, glavobolja, temperatura umjereno povišena (38°C - 39°C), bol u području zahvaćenog sinusa, suzenje očiju, zubobolja, gubitak njuha ili bol u uhu.
Nesanica	Pospanost, razdražljivost, buđenje tijekom noći, nemogućnost spavanja noću, dnevni umor.
Ospice	Temperatura umjereno povišena (38°C - 39°C), isprekidana bol, kašalj, osip na tijelu, temperatura je povišena (više od 39°C).
Vodene kozice	Temperatura umjereno povišena (38°C - 39°C), isprekidana bol, osip na tijelu, vodenasti mjehurići, svrbež, temperatura je povišena (više od 39°C).
Depresija	Potištenost, Napadaj plača, tuga, samosažaljenje, osjetljivost, razdražljivost, psihološki umor, iscrpljenost, emocionalni umor, ranjivost.
Dijabetes tipa 1	Učestalo mokrenje, žeđ, ekstreman umor, zamućen vid, gubitak težine, opća slabost.
Giht	Bol u zglobovima, osjetljivost, crvenilo, oticanje zglobova, toplina mišića ili zglobova, otežano kretanje.
Salmoneloza	Bol u trbuhu, gubitak apetita, mučnina, proljev, dehidracija, krv u stolici.
Moždani udar	Zbunjenost, nemogućnost govora, glavobolja, nagla vrtoglavica, gubitak ravnoteže, gubitak koordinacije, oduzetost jedne strane tijela, ekstreman umor, zamućen vid, povraćanje.
Dijabetes tipa 2	Pojačana žeđ, učestalo mokrenje, suhoća usta, mutan vid, česte infekcije, impotencija, trnci u rukama i nogama, sporo zarastanje rana.
Infarkt miokarda	Bol u prsima, bol u rukama, mučnina, probavne smetnje, žgaravica, gubitak daha, znojenje, hladan znoj, tjeskoba, opći umor, vrtoglavica, pritisak i stezanje u prsima.
Bolesti srca	Nepravilan puls, neugodan osjećaj u prsima, dugotrajan kašalj, znojenje, oticanje nogu, gubitak daha, opća slabost, tjeskoba

Valja naglasiti da je potrebno prvo kreirati bazu na phpMyAdmin.net te se naziv mora poklapati sa nazivom navedenim u .env unutar projekta. Na slici 5.10. je prikazana povezanost sa MySQL bazom na lokaciji 127.0.0.1 te naziv baze koju izrađujemo na phpMyAdmin.

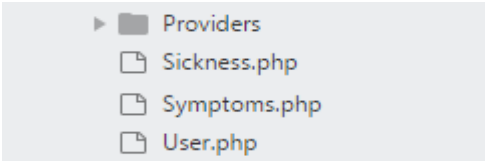
```
8   DB_CONNECTION=mysql
9   DB_HOST=127.0.0.1
10  DB_PORT=3306
11  DB_DATABASE=diplomski
12  DB_USERNAME=root
13  DB_PASSWORD=
```

Sl.5.10. Podaci vezani u bazu podataka

Još jedna važna stavka vezana uz baze podataka su relacije koje se nalaze */System/App/Providers.* Postoji više vrsta relacija ali su najčešće tri, a to su jedan prema više (1:M), jedan prema jedan (1:1) i više prema više (M:M).

U ovom diplomskom radu postavljeni su modeli za korisnika, simptome te bolesti što je prikazano slikom 5.11. Unutar skripti se nalaze modeli za svaku tablicu koji se koriste za interakciju s tom tablicom. Model symptoms i model sickness se koriste za kreiranje simptoma i bolesti. Oba modela imaju svoje zasebne tablice koje su definirane unutar samih modela, a zapisani su kao *protected $table = 'symptoms';*za Symptom model i *protected $table = 'sickness;*za Sickness model. Unutar modela postoji i svojstvo koje se naziva $fillable i određuje koja se kolona unutar tablice može uređivati i kreirati.

Kao primjer može koristiti programski kod 5.10. gdje se kreira simptom i bolest pomoću modela. Unutar modela je na $fillable postavljena kolona 'name', a njezina vrijednost se određuje prilikom kreiranja simptoma i bolesti.



Slika 5.11.Modeli unutar usluga

Tablica 5.9 Popis modela bolesti, simptoma i korisnika

Sickness.php	Model za bolesti
Symptoms.php	Model za simptome
User.php	Model za korisnike

Unutar skripti se nalaze relacije pomoću kojih baza funkcionira i povlači korisnike, podatke o bolestima i simptomima te povezuje jedno s drugim kako bi baza podataka kao i aplikacija imala funkcionalnost.

Skripta *Sickness.php* koristi relaciju *belongsToMany* koja govori da jedan simptom u isto vrijeme ima više bolesti, a jedna bolest u isto vrijeme može imati više simptoma. *Symptom sickness* je među tablica između simptoma i bolesti (Programski kod 5.11).

```
18      public function setToSymptom(){
19          return $this->belongsToMany('App\Symptoms', 'symptom_sickness', 'sickness_id', 'symptom_id')->
                withTimestamps();
20      }
21  }
22
```

Programski kod 5.11. Tablica između simptoma i bolesti unutar skripte sickness.php

Skripta *Symptoms.php* koristi također relaciju *BelongsToMany* gdje je vidljiva ista situacija kao kod Sickness.php skripte. U ovom slučaju je razlika što govori da bolest ima više simptoma (programski kod 5.12).

```
18      public function setToSickness(){
19          return $this->belongsToMany('App\Sickness', 'symptom_sickness', 'symptom_id', 'sickness_id')->
                withTimestamps();
20      }
21  }
22
```

Programski kod 5.12. Tablica između simptoma i bolesti unutar skripte symptoms.php

Skripta *User.php* prikazuje također relaciju *BelongsToMany* (programski kod 5.13) kao i prethodne dvije skripte. Ova skripta prikazuje kako jedan simptom pripada kod više korisnika i povezuje se sa id-om korisnika kojem pojedini simptom pripada. Simptom je također definiran po id-u.

```
32      public function userSymptom()
33      {
34          return $this->belongsToMany('App\Symptoms', 'user_symptom', 'user_id', 'symptoms_id')->withTimestamps()
                ;
35      }
36  }
37
```

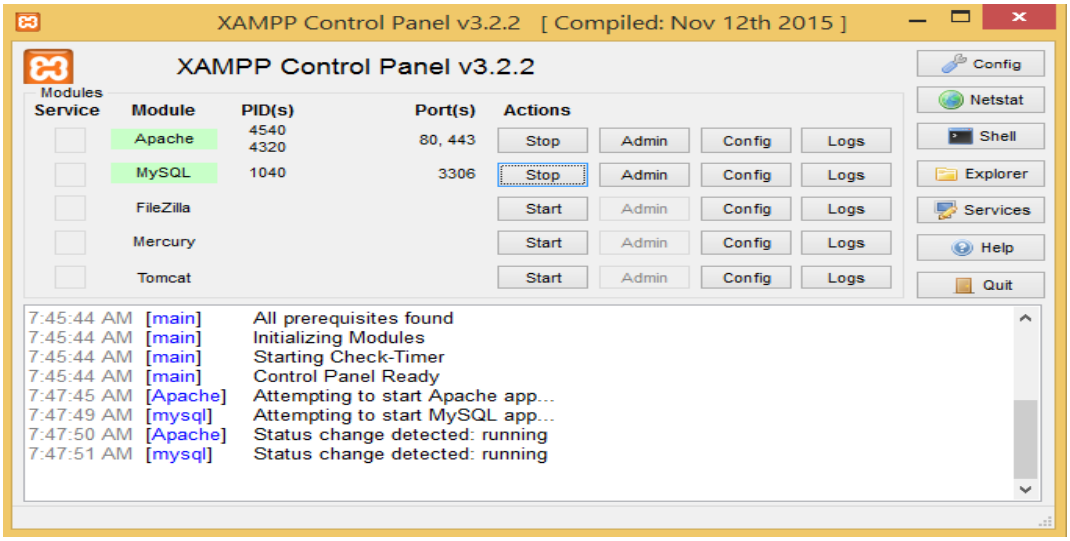
Programski kod 5.13. Tablica između simptoma i korisnika unutar skripte users.php

Iz prethodno navedenog primjer kreiranja simptoma i bolesti na programskom kodu 5.10. povezivanje bi izgledalo na način da zapišemo *$symptom ->setToSickness ()->attach ($sickness);*.

6. PRIKAZ RADA SUSTAVA S TESTIRANJEM I ANALIZOM

6.1 Pokretanje web aplikacije

Pokretanje web aplikacije započinje se pokretanjem XAMPP-a (sl. 6.1) koji stvara poslužitelja za izrađenu web aplikaciju. Potrebno je u XAMPP-u pokrenuti Apache i MySQL.



Sl. 6.1. Pokretanje poslužitelja klikom na Apache i MySQL

Nakon što se pokrenu dviju navedene akcije potrebno je otići na web lokaciju https://localhost/phpmyadmin/ u pregledniku. Na navedenoj lokaciji potrebno je napraviti novu bazu podataka pod nazivom kao što je definirano u programu, točnije u .env dijelu programa.

Nakon izrađene baze iz lokacije */xampp/htdocs/Diplomski rad/diplomski/system* s lokalnog diska potrebno je u datoteci pokrenuti cmd (Command prompt). Unutar cmd-a potrebno je izvršiti sljedeće naredbe vezane uz pokretanje web aplikacije.

Naredbe koje je potrebno izvršiti (sl. 6.2):

1. Php artisan config:clear
2. Php artisan cache:clear
3. Php artisan optimize

```
D:\xampp\htdocs\Diplomski rad\diplomski\system>php artisan config:clear
Configuration cache cleared!

D:\xampp\htdocs\Diplomski rad\diplomski\system>php artisan cache:clear
Cache cleared successfully.

D:\xampp\htdocs\Diplomski rad\diplomski\system>php artisan optimize
```

Sl.6.2. Prikaz izvršenih naredbi

Nakon što su datoteke optimirane potrebno je izvršiti migracije na myPhpAdmin.net sljedećom naredbom. Naredba radi migracije tablica i svih podataka koji su navedeni u programu poput popisa bolesti, simptoma te poveznica između svih tih skripti (sl. 6.3).

Php artisan migrate:refresh --seed –www

```
D:\xampp\htdocs\Diplomski rad\diplomski\system>php artisan migrate:refresh --see
d -vvv
Migration table not found.
Migration table created successfully.
Migrating: 2014_10_12_000000_create_users_table
Migrated:  2014_10_12_000000_create_users_table
Migrating: 2014_10_12_100000_create_password_resets_table
Migrated:  2014_10_12_100000_create_password_resets_table
Migrating: 2016_09_04_000000_create_roles_table
Migrated:  2016_09_04_000000_create_roles_table
Migrating: 2016_09_04_100000_create_role_user_table
Migrated:  2016_09_04_100000_create_role_user_table
Migrating: 2017_10_15_071443_create_sickness_table
Migrated:  2017_10_15_071443_create_sickness_table
Migrating: 2017_10_15_071555_create_symptoms_table
Migrated:  2017_10_15_071555_create_symptoms_table
Migrating: 2017_11_13_180221_create_user_symptom_table
Migrated:  2017_11_13_180221_create_user_symptom_table
Migrating: 2017_11_13_180633_create_admin_diagnosis_table
Migrated:  2017_11_13_180633_create_admin_diagnosis_table
Migrating: 2017_11_15_202713_create_symptom_sickness
Migrated:  2017_11_15_202713_create_symptom_sickness
Seeding: RoleTableSeeder
Seeding: UserTableSeeder
Seeding: SicknessTableSeeder
Seeding: SymptomTableSeeder
```

Sl.6.3. Izvršavanje migracija

Zadnja konačna naredba je php artisan serve (sl. 6.4) koja podiže aplikaciju te joj možemo pristupiti na pregledniku.

```
D:\xampp\htdocs\Diplomski rad\diplomski\system>php artisan serve
Laravel development server started: <http://127.0.0.1:8000>
```

Sl.6.4. Pokretanje aplikacije

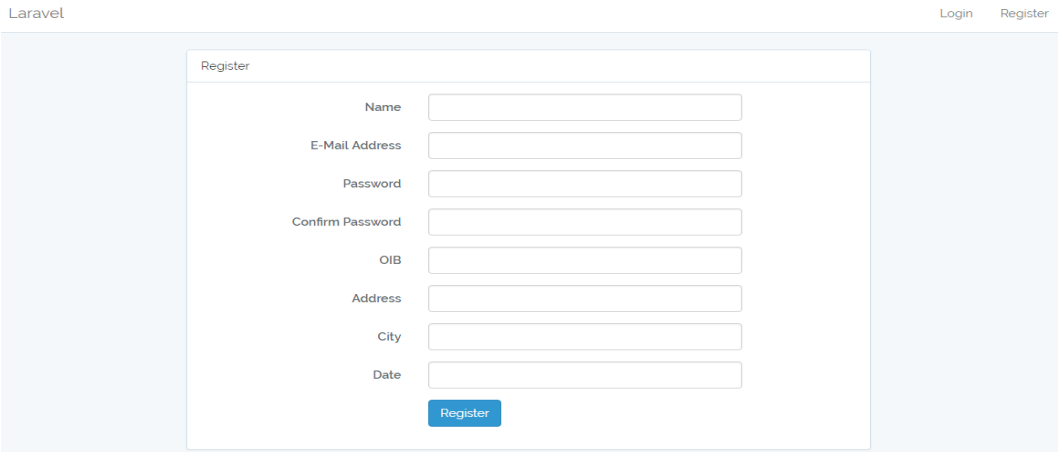
6.2 Izgled web aplikacije

Nakon što je obavljen postupak u prethodnom potpoglavlju na lokaciji http://127.0.0.1:8000/ vidljiva je početna web stranica (home page). Na početnoj web aplikaciji se nude opcija prijave te opcija registracije (sl. 6.5).

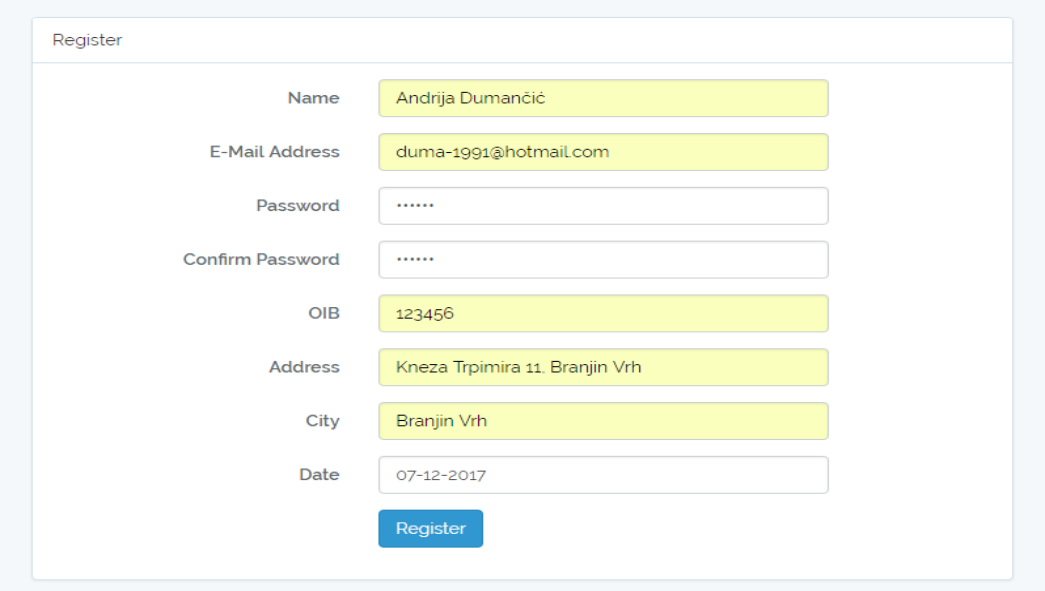
Sl.6.5. Izgled početnog zaslona web aplikacije

6.2.1 Korisnički pregled

U ovom poglavlju korisnik započinje svoju registraciju klikom na *"REGISTER"* opciju u gornjem desnom kutu. Klikom na opciju registracije, korisnik dobije sljedeći prikaz koji je prikazan na slici 6.6.



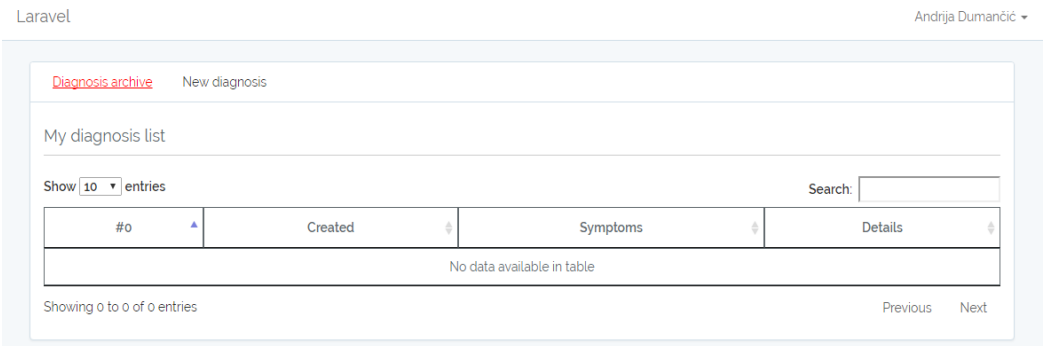
Sl.6.6. Registracija korisnika



Sl 6.7. Sučelje za registraciju s popunjenim podacima

Kako bi se korisnik registrirao, potrebno je da svi podaci budu uneseni. Nakon izvršavanja popune traženih stavki, korisnik završava svoju registraciju pritiskom opcije *Register*.

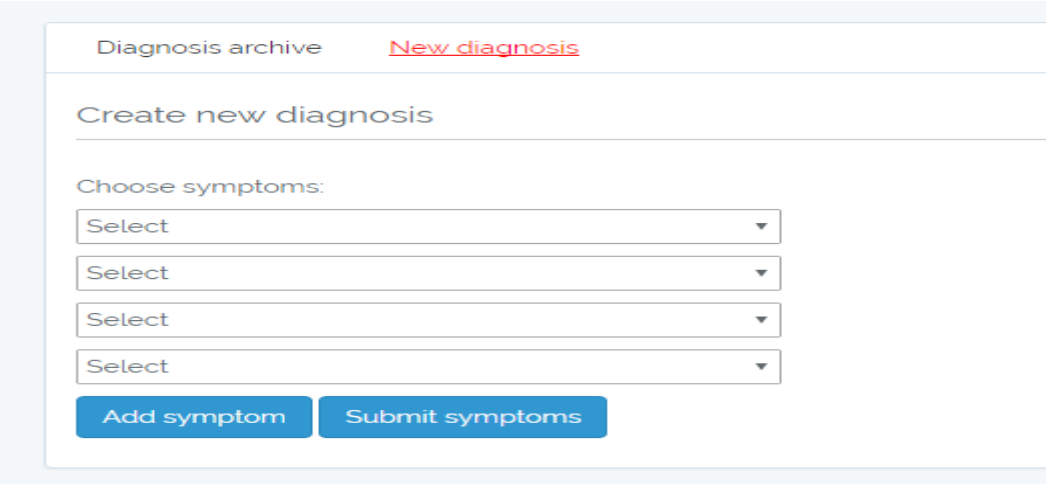
Početni zaslon korisničkog portala izravno prikazuje sve odrađene dijagnoze te pretragu odrađenih dijagnoza. U gornjem lijevom kutu ponuđena je opcija *Diagnosis archive* koja daje početni zaslon vidljiv na slici 6.8. Druga opcija je odlazak na odabir simptoma te traženje dijagnoze na temelju simptoma koji budu uneseni.



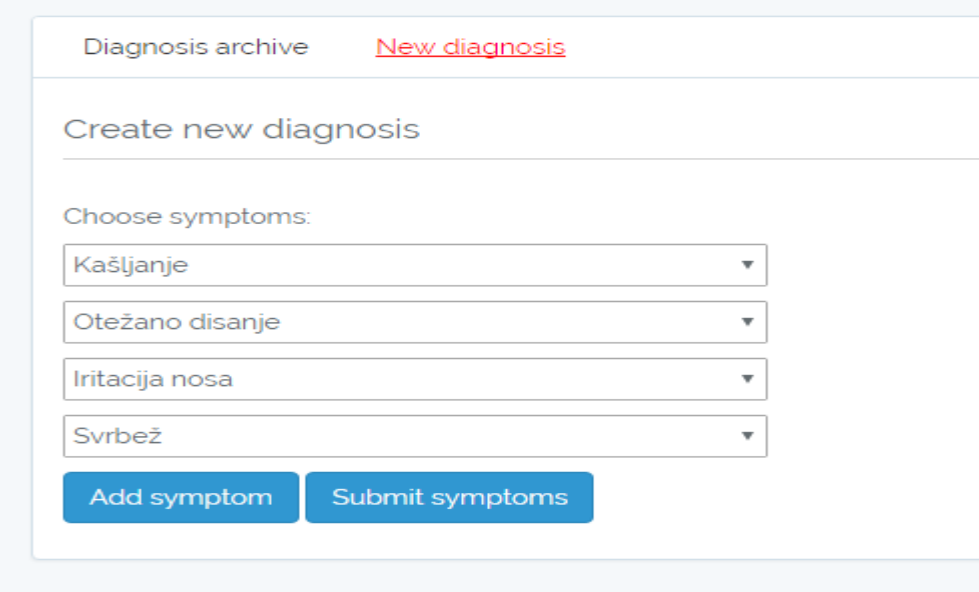
Sl.6.8. Prikaz početnog zaslona / Popis odrađenih dijagnoza

Na slici gore vidljivo je da nema dijagnoza te se pristupa opciji *New diagnosis* koja daje opciju izbora simptoma što je prikazano na slici 6.9. Kako bi rezultat bio što

realniji i točniji, postavljen je minimalni odabir na četiri simptoma uz mogućnost dodavanja dodatnih simptoma.

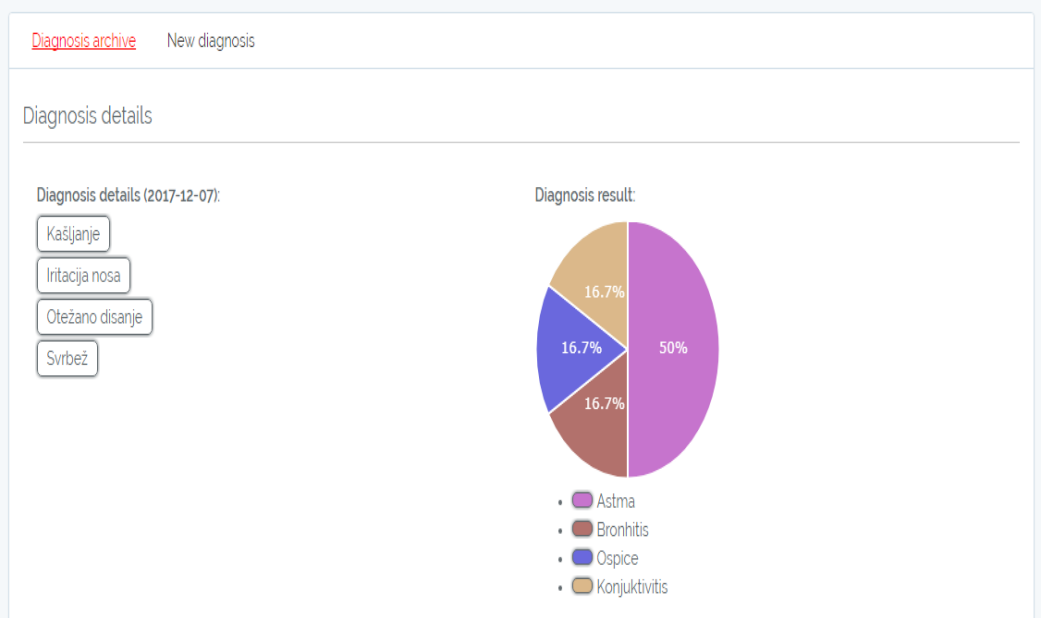


Sl. 6.9. Odabir simptoma



Sl. 6.10. Uneseni simptomi

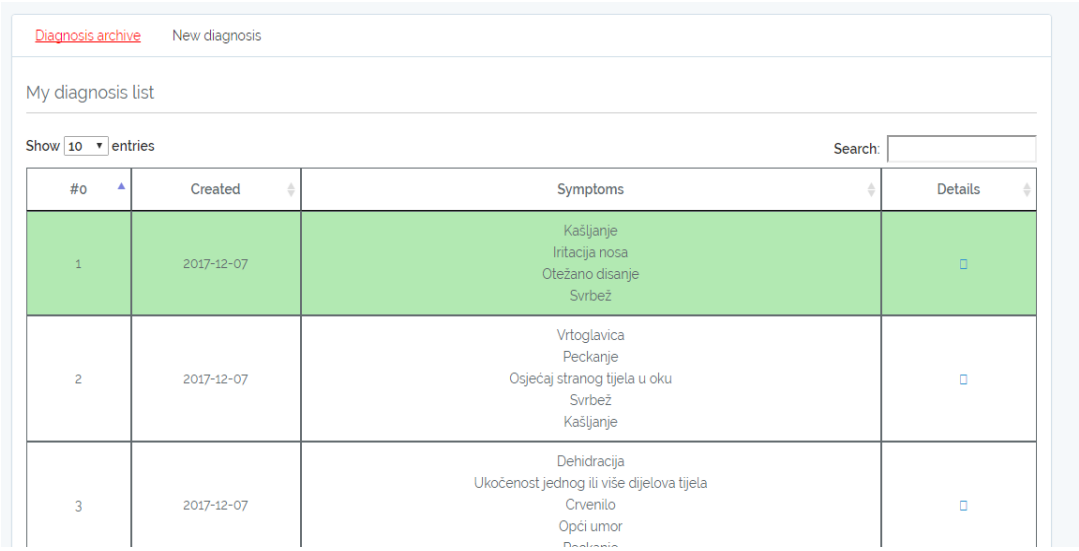
Nakon što se simptomi odaberu, izvršava se dijagnoza pritiskom na opciju *Submit symptoms.* Na temelju unesenih simptoma dobije se dijagnoza (sl. 6.11).



Sl.6.11. Dijagnoza bolesti na temelju simptoma

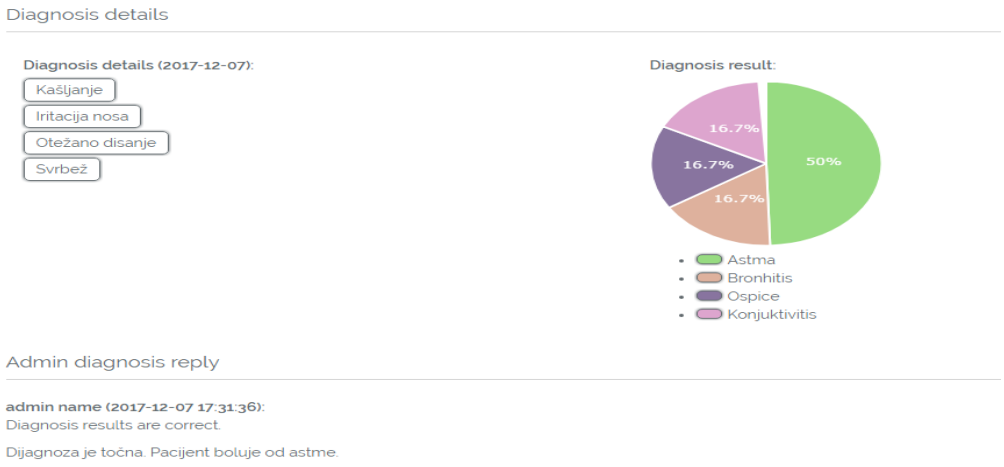
Na slici 6.11 je vidljiva dijagnoza koja se dobije na temelju unesenih simptoma s lijeve strane (kašljanje, otežano disanje, iritacija nosa i svrbež) na dan 07.12.2017. po detaljima dijagnostike predviđa da je rezultat dijagnosticiranja astma najvjerojatnija dijagnoza s 50% podudarnosti na temelju unesenih simptoma. Kao preostale mogućnosti sa 16.7% navode se bronhitis, ospice i konjuktivitis. Sljedeći postupak koji je potreban potvrda je administratora, koji je u ovom slučaju liječnik, da potvrdi točnost ili netočnost dijagnoze. Također uz potvrdu točnosti ili netočnosti, liječnik može dodati u napomenu svoje mišljenje te poslati povratnu informaciju, no o tome više u poglavlju liječničkog pregleda. Nakon što je korisnik dobio dijagnozu, moguće je pogledati arhivu svih bolesti klikom u gornjem lijevom kutu na *Diagnosis archive.*

Nakon što administrator odradi svoj posao, potvrdi točnost ili netočnost dijagnoze, korisnik ima uvid na svojoj početnoj stranici je li administrator dao svoje mišljenje. Dijagnoza bude označena zelenom bojom što pokazuje da je dijagnoza obrađena od strane administratora što je vidljivo na slici 6.12.



Sl. 6.12. Obrađena dijagnoza

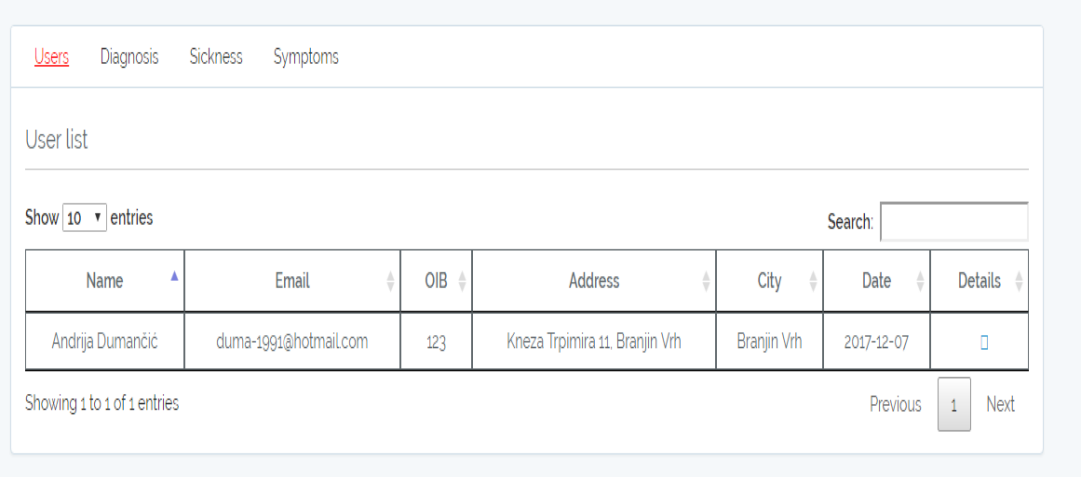
Nakon što je dijagnoza obrađena, provjera se može napraviti ulaskom u detalje gdje je vidljiv komentar koji se dobije od strane administratora (sl. 6.13).



Sl.6.13. Korisnikov uvid u komentar administratora

6.2.2 Administrativni (liječnički) pregled

Za razliku od korisnika, administrator se ne mora registrirati nego samo prijaviti s podacima koji su navedeni u pozadinskom dijelu aplikacije. Kod korisnika je vidljiva arhiva odrađenih dijagnoza (sl. 6.14) dok je kod administratora prikazana lista korisnika koji su registrirani u sustav sa svim podacima koji su bili uvjet registracije korisnika.

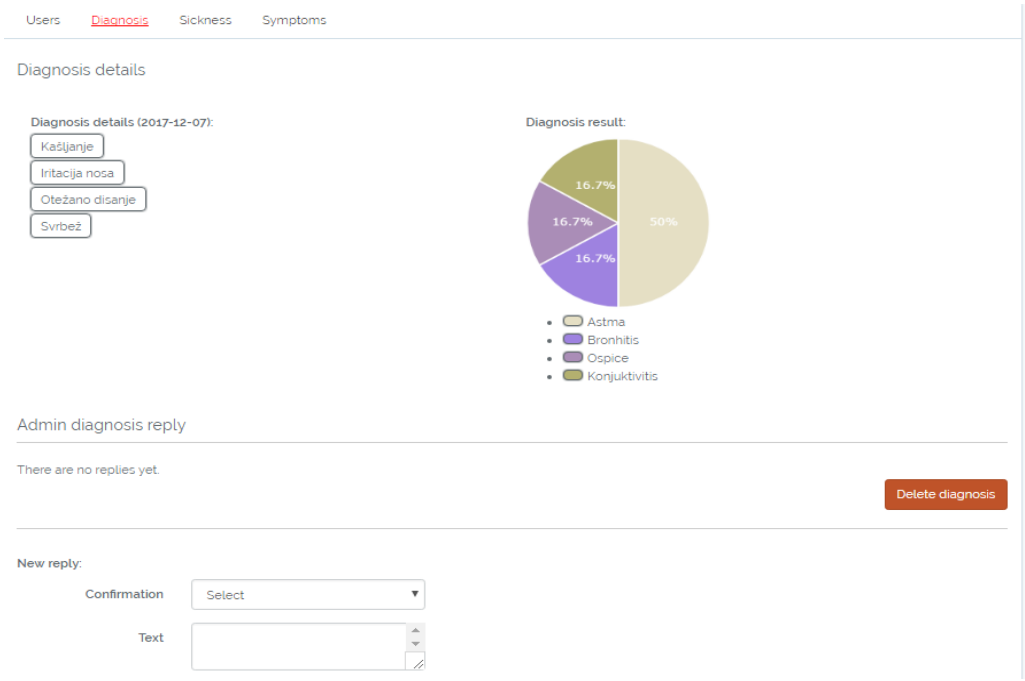


Sl.6.14. Početni zaslon administratora

Kao što je i očekivano, administrator ima veće ovlasti. U lijevom kutu su četiri opcije s kojima administrator raspolaže, a to su *Users* što predstavlja početni zaslon sa slike 6.8, zatim *Diagnosis* opcija koja predstavlja sve dijagnoze kojima su korisnici pristupili, *Sickness* gdje se nalaze bolesti te *Symptoms* gdje su smješteni simptomi.

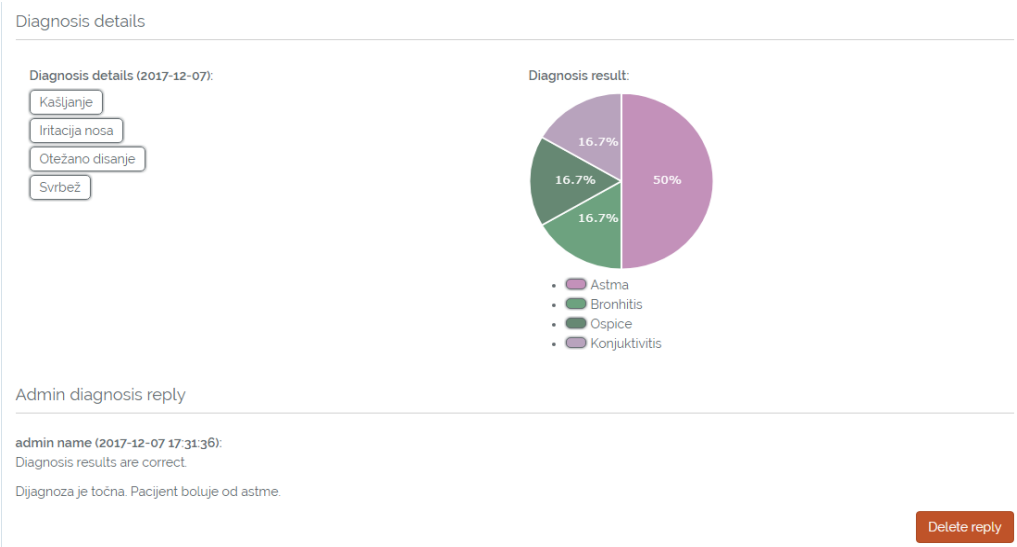
Opcija *Diagnosis* daje uvid u ono što su korisnici upisivali, točnije određene simptome koje su odabrali te na kraju i konačnu dijagnozu.

U donjem dijelu slike vidljiv je jedan pristup dijagnosticiranju na dan 29.11.2017 sa svim navedenim simptomima koje je korisnik osjetio. U ovom dijelu nastaje posao administratoru da na temelju vidljivih rezultata odredi točnost dijagnoze. Administrator nastavlja postupak klikom na detalje dijagnoze u donjem desnom kutu te mu se otvara novi zaslon prikazan na sl. 6.5.



Sl. 6.5. Administrativna potvrda dijagnosticiranog korisnika

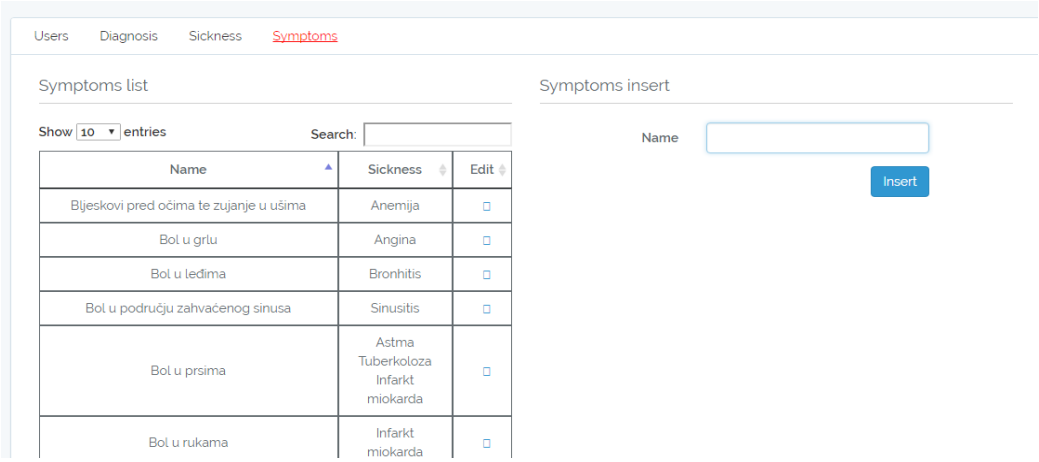
Administrator je pristupio pregledu dijagnosticiranog korisnika te potvrdio njegovu dijagnozu uz komentar koji mu daje razumnije objašnjenje rezultata dijagnoze. Pod dijelom *Admin diagnosis reply* vidljiva je potvrda dijagnoze koja je povezana s datumom i vremenom kad je administrator dao povratnu informaciju (sl. 6.6).



Sl.6.6. Potvrda da je administrator obradio dijagnozu

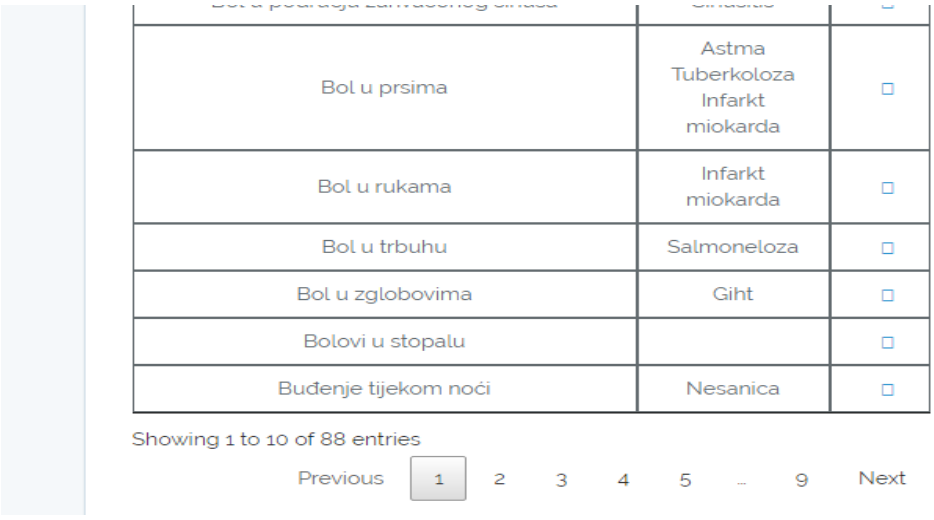
Sljedeće dvije opcije s kojima administrator raspolaže su dodavanje bolesti i simptoma u samoj aplikaciji. Dodavanje bolesti je jednostavnije u samoj aplikaciji nego unutar programskog koda jer automatski dodjeljuje identifikacijski broj novoj bolesti i simptomima.

Za početak prije stvaranja nove bolesti potrebno je unijeti simptome koji su povezani s njom. Unosi se nekolicina simptoma i potvrđuje na insert opciji (sl. 6.7).



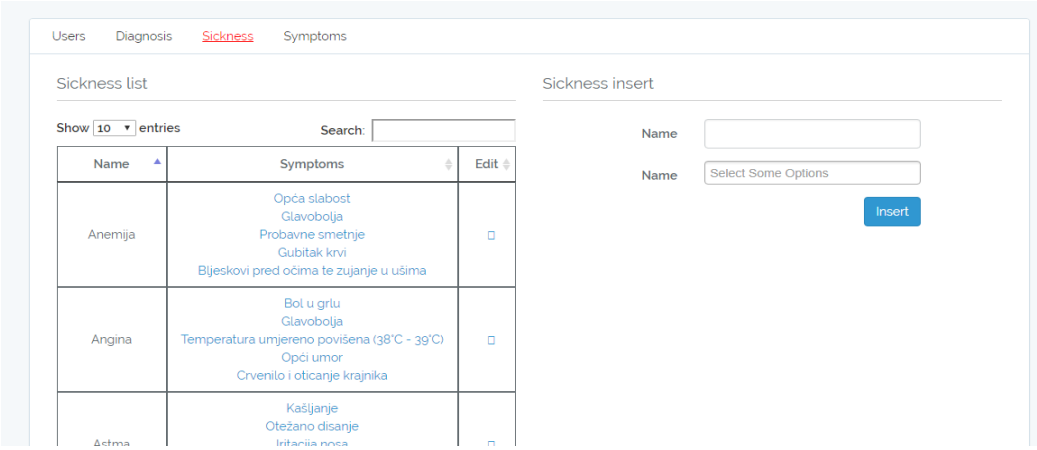
Sl. 6.7. Dodavanje simptoma

Kao primjer dodan je simptom bolova u stopalu koji se potvrdom na opciju insert pojavio u listi simptoma s lijeve strane kao što je vidljivo na slici 6.8.



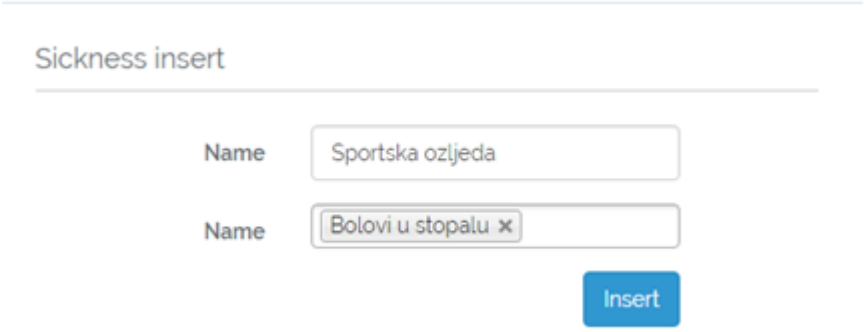
Sl. 6.8. Dodavanje novog simptoma na listu simptoma

Nakon što je dodan simptom, potrebno ga je povezati s nekom novom bolesti što se odrađuje na izborniku *Sickness* kao što je prikazano na slici 6.9.



Sl. 6.9. Izbornik za dodavanje nove bolesti

Kao što je prikazano na slici 6.10. izvršava se pridruživanje simptoma novostvorenoj bolesti koja sadrži taj simptom.



Sl.6.10. Pridruživanje simptoma određenoj bolesti

Nakon što se potvrdilo dodavanje opcijom *Insert*, bolest se pojavila na popisu bolesti što je vidljivo na slici 6.11.

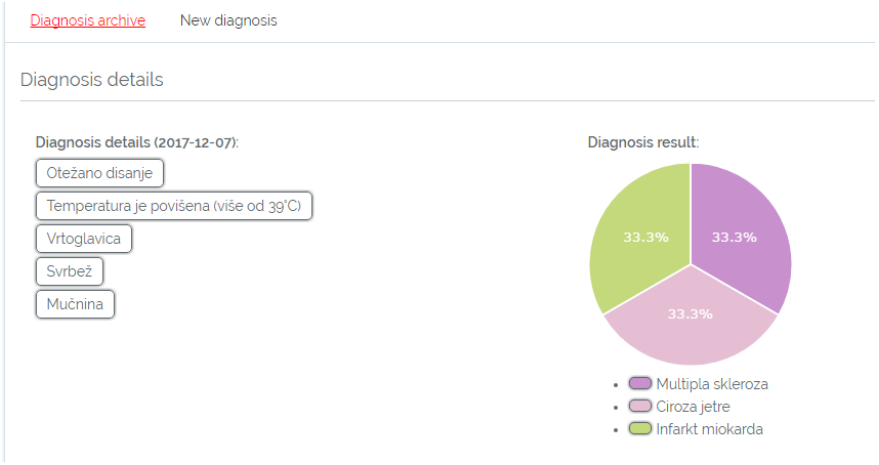
Infarkt miokarda	Probavne smetnje Žgaravica Gubitak daha Hladan znoj Tjeskoba Vrtoglavica Pritisak i stezanje u prsima	☐
Bolesti srca	Nepravilan puls Neugodan osjećaj u prsima Dugotrajan kašalj Tjeskoba Opća slabost Gubitak daha Hladan znoj Oticanje nogu	☐
Sportska ozljeda	Bolovi u stopalu	☐

Sl. 6.11. Bolest je unesena u tablicu bolesti

6.3. Primjeri dijagnosticiranja

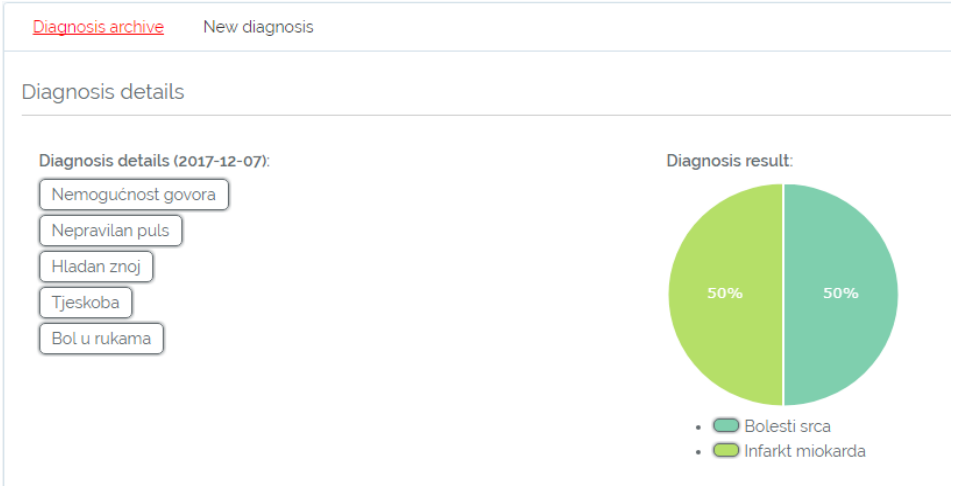
U ovom potpoglavlju prikazano je nekoliko dijagnosticiranja, točnije vrijednosti koje se dobiju na temelju unesenih simptoma.

Na temelju unesenih simptoma prikazanih na slici 6.12 prikazani su dijagnosticirani rezultati. Na temelju simptoma koji su uneseni web aplikacija je ponudila tri potencijalne bolesti sa istim omjerom od 33.3%. Potencijalne dijagnoze su multipla skleroza, ciroza jetre i infarkt miokarda. Dijagnoza koju je web aplikacija odradila šalje se liječniku koji daje konačnu potvrdu te upućuje pacijenta na točnu dijagnozu.



Sl. 6.12. Rezultati dijagnosticiranja 1

Na slici 6.13 prikazano je drugo dijagnosticiranje gdje je unosom simptoma (nemogućnost govora, nepravilan puls, hladan znoj, tjeskoba i bol u rukama) dijagnosticiraju se dvije bolesti. Bolesti su prilično slične te sadrže slične simptome, a i na temelju bolesti srca dolazi do vjerojatnost da je pacijent pretrpio srčani udar.

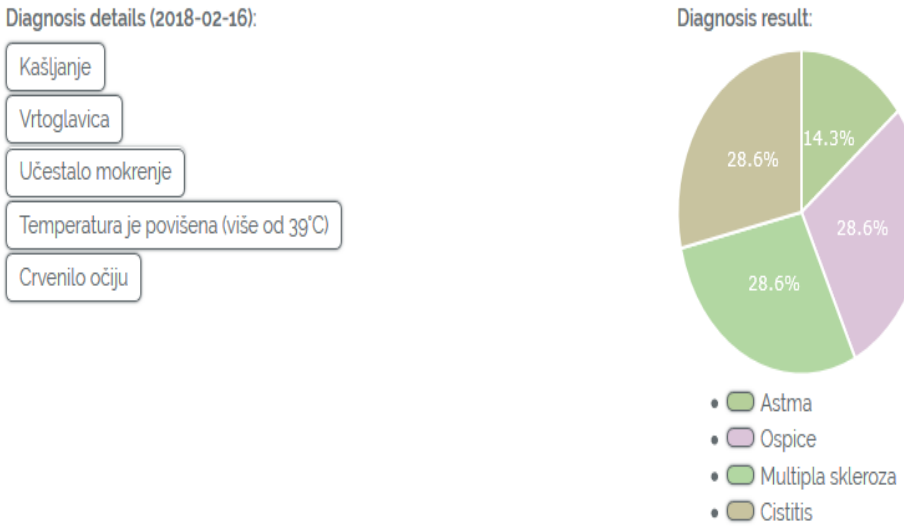


Sl. 6.13. Rezultati dijagnosticiranja 2

U trećem dijagnosticiranju na slici 6.14 je vidljivo da na temelju unesenih simptoma (oduzetost jedne strane tijela, bol u prsima, vrtoglavice, povraćanja i glavobolje) pacijent prema web aplikaciji pretrpio moždani udar sa 60% mogućnosti. Kao druga mogućnost je da je došlo do infarkta miokarda. Kao i u prethodnim slučajevima konačnu dijagnozu daje liječnik koji je upoznat detaljnije sa stanjem pacijenta.

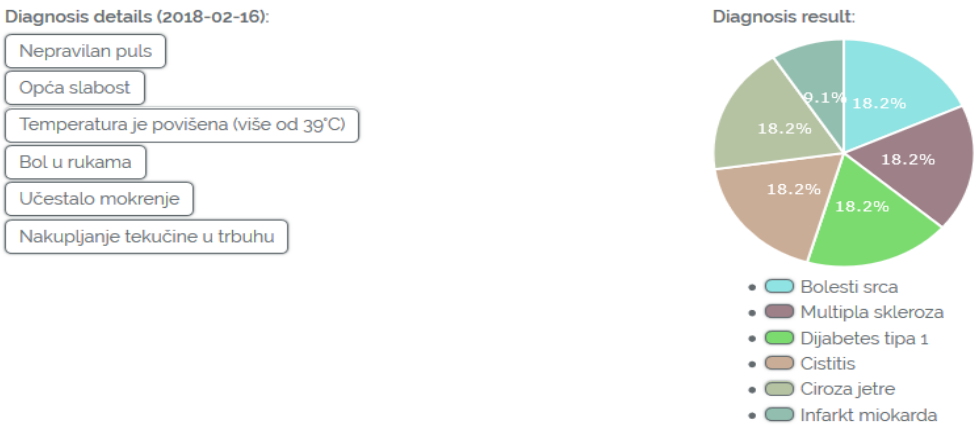
Sl. 6.14. Rezultati dijagnosticiranja 3

Na slici 6.15 je vidljivo dijagnosticiranje u kojem se uzima važna bolest u obzir, neovisno o svojem postotku. Astma se uzela u obzir jer je navedena kao bolest koju se mora uzeti u obzir dok su ostale bolesti postavljene u kružni dijagram jer su ostvarili minimalnih 20% dijagnoze.



Sl. 6.15 Dijagnosticirana je astma sa postotkom manjim od 20%

Na slici 6.16 vidljivo je da infarkt miokarda ima najmanji postotak u odnosu na ostale bolesti, ali je infarkt miokarda važna bolest i potrebno ju je uzeti u obzir neovisno o njenom rezultatu.



Sl. 6.16 Dijagnosticiran je infarkt miokarda sa postotkom manjim od 20%

7. ZAKLJUČAK

U ovom diplomskom radu prikazan je rad izrađen u PHP besplatnom web okviru Laravel. Cilj je bio izraditi web aplikaciju koja na temelju unosa simptoma dolazi do dijagnoze i prikazuje je korisniku. Ugrađeni algoritam je odlučivao na temelju podataka koji su u bazi i na temelju tih podataka prikazao dijagnozu u grafičkom obliku. Kao što je rečeno, web aplikacija ne može dati potpuno pouzdane rezultate te je nužno da se korisnik obrati svojem liječniku. Aplikacija je izrađena u svrhu olakšavanja korisnikove brige za svoje zdravlje, barem jednim dijelom dok liječnik ne donese konačnu dijagnozu. Prilikom testiranja aplikacije, pokazalo se da aplikacija radi ispravno. Aplikacija se može poboljšati dodavanjem veće baze podataka simptoma i bolesti, a moguće je i dodatno grafičko uljepšavanje web aplikacije.

LITERATURA

[1] Biomedicina, https://bs.wikipedia.org/wiki/Biomedicina, (pristup 20.11.2017)

[2] H. Alder, M. A. Beat, C. Marx, et al., Computer-Based Diagnostic Expert Systems in Rheumatology: Where Do We Stand in 2014?, International Journal of Rheumatology, Vol. 2014 (2014), Article ID 672714, 10 pages, https://www.hindawi.com/journals/ijr/2014/672714/ (pristup 20.11.2017)

[3] J. Anju, Machine Learning Techniques for Medical Diagnosis: A Review. In: DU, Conference Center New Delhi. 2015. pp. 2449-2459. http://data.conferenceworld.in/ICSTM2/P2449-2459.pdf (pristup 20.11.2017)

[4] Data science, http://www.datascienceontology.com/papers/med.pdf (20.11.2017)

[5] B.M. Gayathri, C.P. Sumathi, T. Santhanam, Breast Cancer Diagnosis using Machine Learning Algorithms - a Survey. International Journal of Distributed and Parallel Systems (IJDPS), Vol.4, No.3, May 2013, http://airccse.org/journal/ijdps/papers/4313ijdps09.pdf (pristup 20.11.2017)

[6] I. Kavakiotis, Ioannis, et al., Machine Learning and Data Mining Methods in Diabetes Research, Computational and Structural Biotechnology Journal, Vol. 15, 2017, pp. 104-116, https://www.sciencedirect.com/science/article/pii/S2001037016300733 (pristup 20.11.2017)

[7] ICT medicina, https://revisionworld.com/gcse-revision/ict/applications-ict/ict-medicine (pristup 20.11.2017)

[8] R. Duncan, Google uses machine learning to detect breast cancer better than pathologists, 5.Ožujak 2017, https://siliconangle.com/blog/2017/03/05/google-uses-machine-learning-better-detect-breast-cancer-pathologists/ (pristup 20.11.2017)

[9] D. Faggella, 7 Applications of Machine Learning in Pharma and Medicine, Rujan 2017, https://www.techemergence.com/applications-machine-learning-in-pharma-medicine/ (20.11.2017)

[10] Uvod u PHP, http://php.com.hr/77 (pristup 23.11.2017)

[11] PHP, https://hr.wikipedia.org/wiki/PHP (pristup 23.11.2017)

[12] Povijest PHP-a, http://autopoiesis.foi.hr/wiki.php?name=PHP+za+po%C4%8Detnike&parent=NULL&page=Povijest%20PHP-a (pristup 23.11.2017)

[13] History of PHP, http://php.net/manual/en/history.php.php (pristup 23.11.2017)

[14] What is XAMPP, https://www.apachefriends.org/index.html (pristup 23.11.2017)

[15] XAMPP, https://en.wikipedia.org/wiki/XAMPP (pristup 23.11.2017)

[16] Configuration, https://laravel.com/docs/5.5#configuration (pristup 23.11.2017)

[17] MySQL, https://hr.wikipedia.org/wiki/MySQL (pristup 23.11.2017)

[18] MySQL, https://en.wikipedia.org/wiki/MySQL (pristup 23.11.2017)

[19] HTML, https://hr.wikipedia.org/wiki/HTML (pristup 23.11.2017)

[20] CSS, https://hr.wikipedia.org/wiki/CSS (pristup 23.11.2017)

[21] Database Seeding, https://laravel.com/docs/4.2/migrations#database-seeding (pristup 24.11.2017)

[22] Eloquent relationships, https://laravel.com/docs/5.5/eloquent-relationships (pristup 24.11.2017)

[23] Many to many, https://laravel.com/docs/5.5/eloquent-relationships#many-to-many (pristup 24.11.2017)

[24] One to many, https://laravel.com/docs/5.5/eloquent-relationships#one-to-many (pristup 24.11.2017)

[25] One to one, https://laravel.com/docs/5.5/eloquent-relationships#one-to-one (pristup 24.11.2017)

[26] Laravel, https://hr.wikipedia.org/wiki/Laravel (pristup 24.11.2017)

[27] WebMD Symptom Checker, https://symptomsbeta.webmd.com/#/info (pristup 05.12.2017)

[28] Health line symptom checker, https://www.healthline.com/symptom-checker (05.12.2017)

[29] Understanding The Difference Between MVC, MVP and MVVM Design Patterns, https://www.linkedin.com/pulse/understanding-difference-between-mvc-mvp-mvvm-design-rishabh-software/ (pristup 25.01.2017)

SAŽETAK

U ovom diplomskom radu osmišljena je i razvijena web aplikacija koja omogućuje prikaz moguće dijagnoze bolesti na temelju unesenih simptoma. Cilj je bio napraviti web aplikaciju koja što točnije prikazuje mogućnost nastanka bolesti. Glavni dio aplikacije je izrađen pomoću PHP-a u programskom okviru Laravel. Kao dodatak PHP-u, koristio se CSS za vizualno oblikovanje, te HTML. Nakon što je aplikacija testirana, utvrđeno je da aplikacija radi ispravno i da algoritam klasificiranja dobro povezuje simptome s mogućim bolestima. Korisnicima je omogućena provjera sumnje na bolest tako da unose simptome koje osjećaju, ali konačnu dijagnozu postavlja liječnik.

Ključne riječi: dijagnosticiranje, klasificiranje, MVC, web aplikacija.

ABSTRACT

WEB SYSTEM FOR BIOMEDICAL DIAGNOSTICS BASED ON CLASSIFICATION

In this graduate thesis, a developed web application enables the presentation of possible diagnoses of the disease based on the entered symptoms. The goal was to create a web application that more accurately describes the possibility of developing the disease. The main part of the application was created using PHP in Laravel framework. In addition to PHP, they used CSS for visual formatting and HTML. Once the app has been tested, it has been determined that the application is working properly and that the classification algorithm is well linked to symptoms with possible disease. Users are enabled to check suspected illness by entering the symptoms they feel, but the final diagnosis is set by the doctor.

Keywords: classification, diagnostics, MVC, web applications.

ŽIVOTOPIS

Andrija Dumančić rođen je 12.studenog 1991. godine u Bambergu (Njemačka). Školovanje započinje u Osnovnoj školi Branjin Vrh do četvrtog razreda, a nakon toga od petog do osmog razreda završava u Osnovnoj školi Šećerana. Nakon osnovne škole upisuje Prvu srednju školu u Belom Manastiru, smjer tehničar za računalstvo koju završava 2010. godine. Iste godine upisuje stručni studij Informatike na Elektrotehničkom fakultetu u Osijeku čime stječe titulu prvostupnika informatike. Stručni studij završava 2013. godine te iste godine upisuje razlikovnu godinu koju završava 2015. godine. Odmah nakon razlikovne godine upisuje diplomski sveučilišni studij računarstva, smjer procesno računarstvo.

PRILOZI

Prilog 1. Docx i pdf diplomskog rada

Prilog 2. Programsko rješenje diplomskog rada