Web stranica za naručivanje pacijenata

Matijašević, Dino

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

Sveučilišni studij računarstva

WEB APLIKACIJA ZA REZERVACIJU PACIJENATA

Diplomski rad

Dino Matijašević

Osijek, 2016.

SADRŽAJ

1. UVOD

Kako u zadnje vrijeme ljudi imaju sve manje vremena radi raznih obveza, tako je sve veća potreba za dobrom organizacijom dana. Svi su se barem jednom našli u situaciji da trebaju nešto brzo dogovoriti, obaviti ili nešto pronaći. Ništa drugačije nije ni zakazivanje termina kod doktora.

Pokušamo kontaktirati doktora telefonom da bi ustanovili da je nedostupan, potom pokušamo osobno dogovoriti s doktorom termin za neku vrstu liječenja da bi u čekaonici proveli pola dana.

Jednostavnije bi bilo problem riješiti WEB aplikacijom gdje doktori sami unesu svoje slobodne termine, a pacijent samo rezervira slobodan termin koji mu odgovara.

1.1. Zadatak diplomskog rada

Napraviti WEB aplikaciju u kojoj doktor unosi svoje slobodne termine u kalendar na koje se pacijent tada može upisati.

2. PROGRAMSKI JEZICI

Kako bi lakše razumjeli problematiku zadatka i kako ga riješiti, potrebno je i upoznati se sa alatima koji će se koristiti, konkretno, programski jezici pomoću kojih će se napisati (isprogramirati) aplikacija. Programskih jezika je mnogo i nema potrebe ih sve nabrajati, no oni koji su izabrani za ovaj rad su navedeni i pojedinačno ukratko objašnjeni u nastavku ovog poglavlja.

Svaka web stranica i/ili aplikacija se nalazi na serveru, koji obrađuje podatke i šalje klijentu (krajnjem korisniku). Kako to izvršava, ovisi o njegovom operativnom sustavu i kako je konfiguriran, no najpopularniji model je komponiran od operativnog sustava Linux, Apache HTTP servera, MySQL sustava za obradu baza podataka i PHP programskog jezika, a sve skupa čine akronim LAMP.

2.1. HTML

Sve internetske stranice koje otvaramo u svojim preglednicima pisane su opisnim jezikom za opis strukture i sadržaja Internet dokumenta, a naziv mu je *HyperText Markup Language*, poznatiji pod akronimom HTML.

Prvo, HTML se razvijao usporedno s razvojem interneta. Njegovim ocem smatra se Tim Berners Lee, kada je krajem osamdesetih za vrijeme svoga rada u CERN-u (*Conseil Européen pour la Recherche Nucléaire*) u Švicarskoj kreirao prvu WEB stranicu info.cern.ch. [1]

Stranica je bila prototip tamošnjim znanstvenicima za dijeljenje informacija i raznih dokumenata. Karakterističnost tih dokumenata je ta da se jedan pozivao na drugi, u čemu je Berners prepoznao potrebu za boljom organizacijom nad tim dokumentima na što efikasniji način. Klasični način korišten u to vrijeme (*Standardized General Markup Language - SGML*) je bio neadekvatan za rješavanje takvog problema, jednostavnog povezivanja dokumenata preko mreže. Iz tog razloga razvio je HTML, koji nam danas služi za kodiranje dokumenata i *HyperText Transport Protocol* (HTTP), za prijenos istih.

Prvi prototip preglednika pokrenut je 1991. godine na NeXT računalu koje je ujedno i služilo kao server za prvu WEB stranicu.

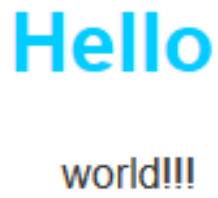
HyperText [2] se sastoji od međusobno povezanih dijelova objekata tako da posjetitelj interaktivno određuje redoslijed čitanja, dok *Markup Language* [3] označava način unosa informacija unutar dokumenta.

[1] Wikipedia – HTML, History - Development <https://en.wikipedia.org/wiki/HTML>, (16.6.2016.)

Kada se kaže da je HTML opisni jezik, pod time se podrazumijeva njegov zadatak opisivanja određenih dijelova teksta. Kako će ga preglednik obraditi, koja je njegova funkcija i na kraju kako prikazati na ekranu.

Na primjer jedan od osnovnih HTML elemenata, *paragraf*, koji se označava sa <p>. On pregledniku govori da se radi o tekstualnom elementu te da njegov sadržaj na ekranu prikaže kao tekst.

```
<!DOCTYPE HTML>
<html>
  <body>
    <h1 style="color: #0cf">Hello</h1>
    <p style="margin-left:20px">world!!!</p>
    <!--Ovo je komentar i neće se prikazati na stranici-->
  </body>
</html>
```



Sl. 2.1. Lijevo je HTML kod, a desno rezultat koji će se prikazati na ekranu.

Sa svakom novijom iteracijom HTML-a dolaze novi elementi sa dodatnim mogućnostima, no najveći napredak pokazao se prelaskom sa HTML4 na HTML5 gdje je uveden veliki broj specifičnih elemenata koji su programerima pružili puno više opcija u dinamičkom uređivanju stranica. Neki od primjera su <*video*> i <*audio*> koji dolaze s vlastitim API funkcijama za olakšano postavljanje multimedije na stranice. Još jedan dobar primjer je <*input*> kojem je dodijeljen atribut *type* za veću fleksibilnost, poput *tel*, *number*, *search*, *color* itd.

```
<form method="GET">
  Odaberi omiljenu boju:
  <input type="color" name="favcolor" value="#ff0000"/>
  <input type="number" name="quantity" min="1" max="5"/>
  <input type="submit" value="Potvrdi"/>
</form>
```

Sl. 2.2. Kod forme i rezultat istog na ekranu

Gornji kod stvara formu za unos boje, broja i potvrdno dugme, a rezultat je donja slika.

[2] Wikipedia – Hypertext <https://en.wikipedia.org/wiki/Hypertext>, (16.6.2016.)
[3] Wikipedia – Markup language <https://en.wikipedia.org/wiki/Markup_language>, (16.6.2016.)

2.2. CSS

Pošto HTML nije dovoljan sam po sebi za stiliziranje sadržaja stranice, tu ulogu preuzima stilski jezik CSS (eng. *Cascading Style Sheet*) koji deklarira izgled elemenata pomoću selektora unutar HTML element *<style>* ili odvojenog dokumenta ekstenzije.*css*, a taj dokument onda povezujemo sa HTML kodom pomoću elementa *<link rel=”stylesheet” gref=”style.css” type=”text/css”>*.

Selektori mogu biti definirani samim elementom:

```
h1 {
  color: #00f;
}
```

ili pomoću klasa/identifikatora:

```
<p id=”identifikator” class=”klasa”>Ovo je tekst!</p>
<p class=”klasa”>Ovo je drugi tekst!</p>

#identifikator {
  margin-left: 20px;
}
.klasa{
  color: #ff0000;
}
```

Sl. 2.3. Jednostavan prikaz označavanja CSS selektora

Iz primjera se vidi da se identifikator odnosi isključivo na jedan element. U bilo kojem trenutku može biti samo jedan element sa određenim identifikatorom, dok se klase mogu dodijeliti više elemenata radi lakšeg i dinamičnijeg uređivanja.

Selektori se mogu deklarirati i hijerarhijski, odnosno deklariranjem prvo roditelja elementa, te zatim idućeg elementa u stablu kojeg želimo urediti:

```
<div class=”klasa”>
  <h1>Ovo je naslov!</h1>
  <p>Ovo je tekst!</p>
  <p>Ovo je drugi tekst!</p>
</div>


.klasa h1{ color: blue; }
```

Sl. 2.4. Naslijedni selektor

Gornji primjer će odabrati DIV element sa klasom *klasa* te gledati isključivo nasljednika *h1* i dodijeliti mu plavu boju.

Kao što je bila riječ o HTML-u, tako je i CSS kroz godine dobivao sve više funkcionalnosti za olakšan rad dizajnera WEB stranica. [4] CSS1 je imao najosnovnije manipulacije poput: *margin, padding, border, float,* manipulacije lista i fontova. [5] Sa CSS2 dobilo se malo više slobode u pozicioniranju elemenata pomoću *position* atributa i u današnje vrijeme vrlo bitnog *@media*[6] pravila s kojim odredimo kako će se stranica ponašati u raznim prikazima, odnosno kako će izgledati ako se treba ispisati na pisaču ili jednostavno prikazati na ekranu:

```
@media print {
  body { font-size: 10px }
}
@media screen {
  body { font-size: 13px }
}
```

Sl. 2.5. Selektori upita (eng. query)

Gornji primjer će na ispisanom papiru prikazati tekst veličine 10px, dok će na ekranu text prikazivati veličine 13px.

CSS3, koji je trenutni standard, je donio mogućnosti modernijeg dizajniranja sa zakrivljenim rubovima pomoću atributa *border-radius*, elegantnijeg bojanja elemenata sa *linear-gradient*, izrade animacija bez korištenja *JavaScript* programskog jezika, te nadogradnja sve važnijeg *@media* za definiranje ponašanja elemenata u responzivnom dizajnu.[7]

[4] Wikipedia - Cascading Style Sheets, History <https://en.wikipedia.org/wiki/Cascading_Style_Sheets#History>, (16.6.2016.)
[5] W3C - Cascading Style Sheets, level 1 <https://www.w3.org/TR/CSS2/media.html#q7.0>, (16.6.2016.)
[6] W3C - Cascading Style Sheets Level 2 Revision 1 (CSS 2.1) Specification, 7 Media types <https://www.w3.org/TR/CSS2/media.html#q7.0>, (16.6.2016.)
[7] Wikipedia - Cascading Style Sheets, CSS 3 <https://en.wikipedia.org/wiki/Cascading_Style_Sheets#CSS_3>, (16.6.2016.)

2.3. PHP

Originalni, puni naziv programskog jezika PHP je, u doslovnom smislu riječi, "Osobna Kućna Stranica" (eng. *Personal Home Page*), što i nije neobično s obzirom da je njegova prvobitna svrha bila privatni projekt kanadskog programera Rasmus Lerdorfa za održavanje njegove privatne web stranice.[8]

Prve iteracije su bile samo obične skripte za uređivanje web formi te komunikacije s bazama podataka. Nije mu ni bila namjera napraviti programski jezik, ali kako je podijelio svoj projekt sa drugima, jezik se jednostavno razvio sam od sebe nakon nekoliko nadogradnji originalnog koda čija je sintaksa ličila na Perl.

Nekoliko izmjena i par iteracija kasnije, uz pomoć Zeev Suraskia i Andi Gutmansa (koji su napisali originalni kod za primanje ulaznih varijabli te pretvarajući ih u konkretne strukture podataka.[9] Time je nastao prvi istinski skriptni programski jezik za razvoj web aplikacija pod nazivom PHP: *Hypertext Preprocessor*. Važno je napomenuti, kako je PHP besplatan programski jezik te kako mu je to jedan od razloga zašto se toliko raširio i razvijao kroz godine.

Kao *server-side* (izvršava se na serveru) programski jezik, njime se može obraditi razne informacije prije nego ih se pošalje krajnjem korisniku, odnosno, PHP skripte generiraju HTML kod te korisniku šalju stranicu spremnu za prikaz u web pregledniku. Što znači da web stranice koje klijent vidi na ekranu nigdje ne postoje konkretno u HTML obliku, već se nalaze kao PHP kod na serveru. Pošto se HTML generira na samom kraju izvršavanja PHP skripte, moguće je generiranje dinamičkih sadržaja koji će klijentu biti prikazane ovisno o samom klijentu i njegovoj komunikaciji sa serverom.

```
<?php echo '<p>Pozdrav!</p>'; ?>
```

Sl. 2.6. Najjednostavniji zapis PHP koda

Najosnovnija PHP operacija, ispisivanje HTML paragrafa, gdje se vidi nekoliko bitnih stavki glede PHP sintakse. Prije svega vidi kod se nalazi između oznaka <? i ?> uz dodatnu napomenu da nakon prvog znaka mora doći i kratica *php* kako bi server znao da se radi o PHP kodu, a ne XML-u. Također je bitno postavljanje točke-zarez na kraju svake naredbe, kao u C-u. Te oznake su bitne jer u .php datotekama se može pisati i HTML i PHP sintaksa.

[8] Wikipedia – PHP, History – Early history <https://en.wikipedia.org/wiki/PHP#History>, (17.6.2016.)
[9] Wikipedia – PHP, History – PHP 3 and 4 <https://en.wikipedia.org/wiki/PHP#History>, (17.6.2016.)

```
<html>
  <head>
    <title> Primjer kombinacije HTML-a i PHP-a. </title>
  </head>
  <body>
    <?php
      echo "Ovaj dio koda će se prikazati na ekranu";
    ?>
  </body>
</html>
```

Sl. 2.7. Paralelno pisanje PHP-a i HTML-a

Imamo i oznake za komentare, // za komentar u produžetku reda ili /* */ za obuhvaćanje komentara kroz više redova.

```
<?php
  Ovaj dio koda će se izvršiti //no sve nakon ove dvije kose crte, neće
  /* Sve što
  se nalazi
  unutar ovih oznaka je komentar*/
?>
```

Sl. 2.8. Zapisivanje komentara u PHP-u

Funkcije se mogu deklarirati vrlo jednostavno kao: *function funk*(){*return true*;} i na njih se pozivati dalje u kodu. No da bi se pozivalo na iste kroz više datoteka, mora se pozvati jednu datoteku u drugu pomoću nativne funkcije *include()* ili *require()*, što se ne pokazuje baš kao najpraktičniji način, a i pokazuje se kao relativno nesigurna metoda. Za malo bolju fleksibilnost i sigurnost, PHP (točnije PHP3 [10]) podržava i objektno orijentirano programiranje, odnosno deklariranje klasa. U njima se onda deklariraju funkcije na koje se poziva dalje u aplikaciji.

```
<?php
  $a = 2;
  $b = 3;
  function zbroj($a, $b){
     $c = $a + $b;
     echo "Rezultat je:".$c;
     return true;
  }
?>
```

Sl. 2.9. Jednostavan primjer računskih operacija u PHP-u

[10] Wikipedia – PHP, Object-oriented programming <https://en.wikipedia.org/wiki/PHP#Object oriented_programming>, (18.6.2016.)

Funkcija uzima dvije varijable, zbroji ih te ispiše na ekran, za kraj vraća da je operacija uspješno izvršena.

Što se tiče funkcionalnosti, PHP pruža dosta veliku knjižnicu već pred definiranih naredbi kojima se može postići željeni rezultat bez raspisivanja kompletnog koda. Neke naredbe su nativne PHP-u, a neke su samo nadogradnje, odnosno poveznice sa već postojećim rješenjima kako bi se dodatno olakšao rad, jedni od glavnih primjera su:

- Tidy (aplikacija za čišćenje i ispravljanje pogrešnog HTML koda),
- JSON (jezično neovisan oblik podataka, koji se najčešće koristi kod asinkrone komunikacije između servera i preglednika),
- PDFlib (PHP klasa koja služi za jednostavno, generiranje .pdf dokumenata),
- MySQL (konektor s bazama podataka).

2.3.1. MySQL

Inicijalno kreiran od strane Švedske kompanije MySQL AB 1995 godine za njihove projekte jer im se tadašnji sustav mSQL (ili Mini SQL) činio kao prespor[11], a kao besplatan sustav za upravljanje bazama podataka, MySQL je zadobio veliku popularnost, što ga je dovelo da bude jedan od najrasprostranjenijih sustava takve namjene.

Kako je ranije navedeno, PHP ima pristup bazama podataka preko svog konektora MySQL. Najjednostavniji primjer poveznice je:

[11] Wikipedia – MySql, History <https://en.wikipedia.org/wiki/MySQL#History>, (20.6.2016.)

```php
<?php
//Spajanje na bazu
$link = mysql_connect('mysql_server', 'mysql_korisnik', 'mysql_lozinka') or die('Ne mogu se spojiti, razlog: ' . mysql_error());
echo 'Uspješno spojen na bazu!';
mysql_select_db('moja_baza') or die('Došlo je do greške, nemoguće odabrati bazu!');
//Izvršavanje naredbe odabiranja tablice u bazu
$poziv = 'SELECT * FROM moja_tablica';
$rezultat = mysql_query($poziv) or die('Poziv neuspješan, razlog: ' . mysql_error());
//Čitanje informacija
while ($linija = mysql_fetch_array($rezultat, MYSQL_ASSOC)) {
  foreach ($linija as $vrijednost_stupaca) {
    echo "$vrijednost_stupca<br/>";
  }
}

//Oslobađanje baze
mysql_free_result($rezultat);
mysql_close($link);
?>
```

Sl. 2.10. Primjer osnovnog koda za spajanje na bazu uz pomoć mysql_cennect

Ovdje se vidi da nakon uspješnog spajanja na bazu, naredbe za rad u samoj bazi se pišu kao obični *string*-ovi u naredbu *mysql_query()* što će reći konektoru da izvrši naredbu na samoj bazi. Rezultat kojeg vraća je oblika *resurs*, posebne namjene za baze, slike i dokumente, jer u njemu se nalaze dodatne informacije o vrsti podataka spremljenih u njemu, te se kao takav mora obraditi uz pomoć *foreach* naredbe gdje se, u slučaju baza, podatak izvlači liniju po liniju i odabire adekvatan stupac.

id	ime	prezime	godine
1	Dino	Matiašević	29
2	Ivan	Horvat	28

Tablica 2.11. Vizualni prikaz zapisa u bazi podataka

Gornji kod bi iz tablice na ekranu prikazao:

Uspješno spojen na bazu!

1
Dino
Matijašević
29
2
Ivan
Horvat
28

Sl. 2.12. Prikaz rezultata upita iz slike 2.13.

Nadogradnja na postojeći konektor bazi je *mysqli* koji je definiran kao klasa koja vraća objekte, što je daleko praktičnije od klasičnog *mysql* konektora koji vraća resurs. Pošto je vraćena vrijednost objekt, lako možemo manipulirati bazom.

```php
<?php
//Spajanje na bazu
$link = new mysqli('mysql_server', 'mysql_korisnik', 'mysql_lozinka', 'mysql_baza');

//provjera da li je veza sa bazom uspostavljena i ako nije, ispiši greške
if ($link ->connect_errno) {
   echo "Errno: " . $spoj->connect_errno . "\n";
   echo "Error: " . $$spoj->connect_error . "\n";
}
else echo "<out>Uspješno spojen na bazu</out>";


//Izvršavanje naredbe odabiranja tablice u bazu
$poziv = 'SELECT * FROM moja_tablica';
$upit = $spoj->query($upit);
$rezultat = $upit->fetch_array();

//Čitanje informacija
foreach ($rezultat as $vrijednost_stupaca) {
   echo "$vrijednost_stupca<br/>";
}

//Oslobađanje baze
$upit->free();
$link->close();
?>
```

Sl. 2.13. Prikaz osnovnog upita uz pomoć klase mysqli

2.4. JavaScript

U samim začecima internet ranih 90-tih godina, WEB stranice su bile potpuno stacionarne. Što nije bilo slučajno, serveri su tada bila znatno slabiji nego danas, kao što je i brzina veze sa internetom bila ograničena na sporu komunikaciju preko telefonske linije. No 1995. američka tvrtka Natscape Communications Corporation je zadala zadatak Brendan Eich-u da osmisli novi programski jezik za njihov WEB preglednik Natscape Navigator.[12] Novi jezik je bio namijenjen da se umjesto sva informacija šalje od server, obradu sadržaja odradi od strane krajnjeg korisnika (eng. Client), odnosno samog WEB preglednika. Eich je odlučio razviti programski jezik koji će komplimentirati već dobro poznatu Javu, no cilj mu je bio napraviti ga lakšim za savladati kako bi neprofesionalni programeri mogli što brže naučiti sintaksu i početi pridonositi razvoju WEB-a.

Projekt je započeo pod nazivom Mocha, te se preimenovao u LiveScript, da bi na posljetku odlučili programski jezik nazvati JavaScript, što je zapravo zavaravalo ljude (pa i do dan danas) da je JS proizašao iz Jave, no zapravo sintaksa je derivirana iz dobro utemeljenog programskog jezika C.

Pomoću ove tehnologije se drastično rasteretilo opterećenje servera od raznih operacija kako bi korisniku prikazali jednostavan i brz rezultat pomoću podijele tereta, npr. umjesto da server obrađuje sve informacije WEB stranice, dio posla prebaci na klijenta te prepusti da on obradi podatke i prikaže ih u pregledniku:

Umjesto da server obradi operaciju (PHP kod vidljiv u sl. 2.4.), istu operaciju može napraviti i korisnik pomoću JavaScript-a:

```
var a = 2;
var b = 3;
var c = a + b;
alert(“Rezultat je: ”+c);
```

Sl. 2.14. Zapis varijable u JavaScriptu i prikaz rezultata kao 'pop-up'

Pošto se željelo postići jednostavnost korištenja ovog programskog jezika, svaka funkcija se tretirala kao objekt, odnosno JavaScript je u svojoj suštini objektno orijentirani programski jezik.

[12] Wikipedia – JavaScript, History - Beginnings at Netscape <https://en.wikipedia.org/wiki/JavaScript#History>, (21.6.2016.)

Što znači da je vrlo jednostavno nadovezati jednu funkciju na drugu:

```
.parent().find();
```

Sl. 2.15. Nadovezivanje jedne funkcije na drugu

Kako je sintaksa proizašla iz C-a, može se vidjeti mnogo sličnosti u samom kodu, poput '*točka-zarez*' na kraju svakog reda, zakomentiranjem koda pomoću // za jedan red ili /* */ za više redova. JS je također naslijedio i strukturno programiranje kao što su *if* operacija, te petlje *while, do shile, switch* i *for.*

```
var links = document.getElementByClassName('poveznica').elements;
for(var i = 0; i < links.length; i++){
  links[i].style.backgroundColor = "#A1B2C3";
}
```

SL. 2.16. Pisanje jednostavne funkcije u JavaScript-u

Gornji primjer će pregledati sve elemente koji imaju klasu '*poveznica*' te će svakom pojedinačno obojati pozadinu u boju '*#A1B2C3*'.

2.4.1. jQuery

Razvojem interneta razvijala se sve veća potražnja za dinamičnim sadržajem stranica, te sve veća potreba za korištenje JavaScripte kako bi se postigli šareni rezultati, odnosno sve veća potreba ta interaktivnosti stranice preko raznih '*onclick*' događaja.

Da bi se rasteretili programeri, napisana je JS knjižnica pod nazivom jQuery. Njen glavni zadatak je da pomogne u navigaciji kroz dokument, a vrlo lako se poziva unutar JavaScript koda uz pomoć predznaka $ na početku naredbe. Uz pomoć nje, vrlo lako se kreiraju razne animacije, obrađuju događaji, razvijaju Ajax aplikacije i prije svega jednostavno odabiru selektori.

Ako se želi odabrati element sa identifikatorom '*identifikator*' i klasom '*klasa*', Javascriptom bi se to postiglo sa:

```
document.getElementById('identifikator');
document.getElementByClassName('klasa');
```

Sl. 2.17. Odabiranje elementa po klasi i identifikatoru uz pomoć čistog JavaScript-a

Isti zadatak se znatno lakše postiže uz pomoć jQuery-a:

```
$('#identifikator');
$('.klasa');
```

Sl. 2.18. Isti rezultat kao iz slike 2.17. samo rađeno uz pomoć jQuery-a

Kako bi se dodatno olakšalo programiranje, JQ funkcije obično vraćaju objekt što znači da se naredbe vrlo lagano mogu lančano nizati za postizanje željenog rezultata:

```
$('div.klasa').add('p.quote').addClass('blue').slideDown('slow');
```

Sl. 2.19. Lančano programiranje u jQuery-u

Osim jednostavnosti, omogućena je i fleksibilnost sa mogućnošću dodavanja novih DOM elemenata u već postojeće elemente i novim naredbama opće primjene poput naredbe *each*.

```
function JednakeVisine(grupa) {
  var najvislji = 0;
  grupa.each(function() {
    var sadasnjaVisina = $(this).height();
    if(sadasnjaVisina > najvislji) {
      najvislji = sadasnjaVisina;
    }
  });
  grupa.height(najvislji);
}
$(document).ready(function() {
  JednakeVisine($(".visina")); //svi elementi s klasom .visina se uspoređuju
});
```

Sl. 2.20. Pisanje jednostavne funkcije za kontrolu visine elemenata s istom klasom

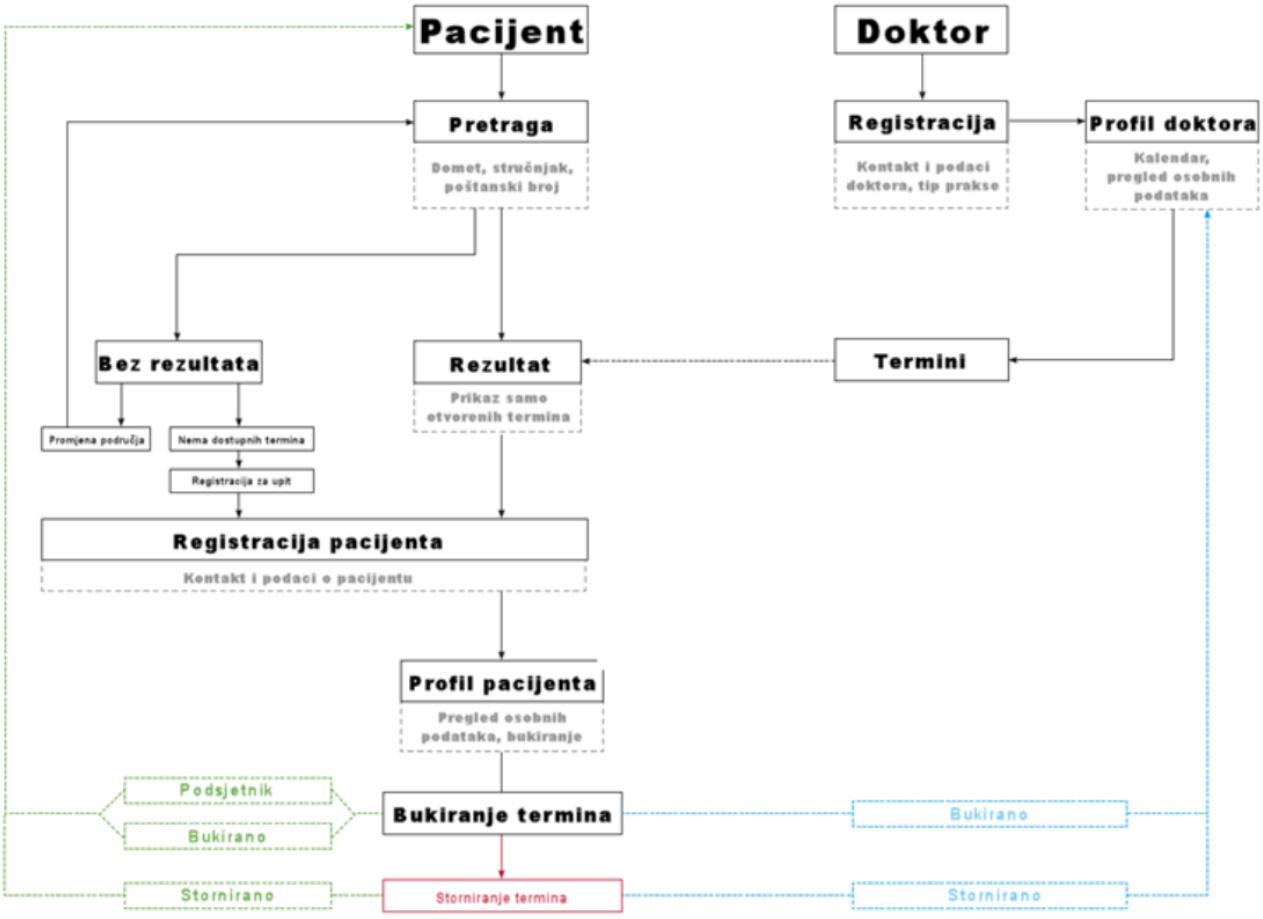
3. WEB STRANICA

U ovome segmentu objašnjeno je sve vezano za samu aplikaciju, od analize, kodiranja do opisa rada same aplikacije.

3.1. Analiza problematike

Zadatak je vrlo jednostavan, trebalo je napraviti web aplikaciju koja će pacijentima omogućiti lako pronalaženje doktora kojeg trebaju u što jednostavnijem okruženju kako bi se postigla maksimalna efikasnost.

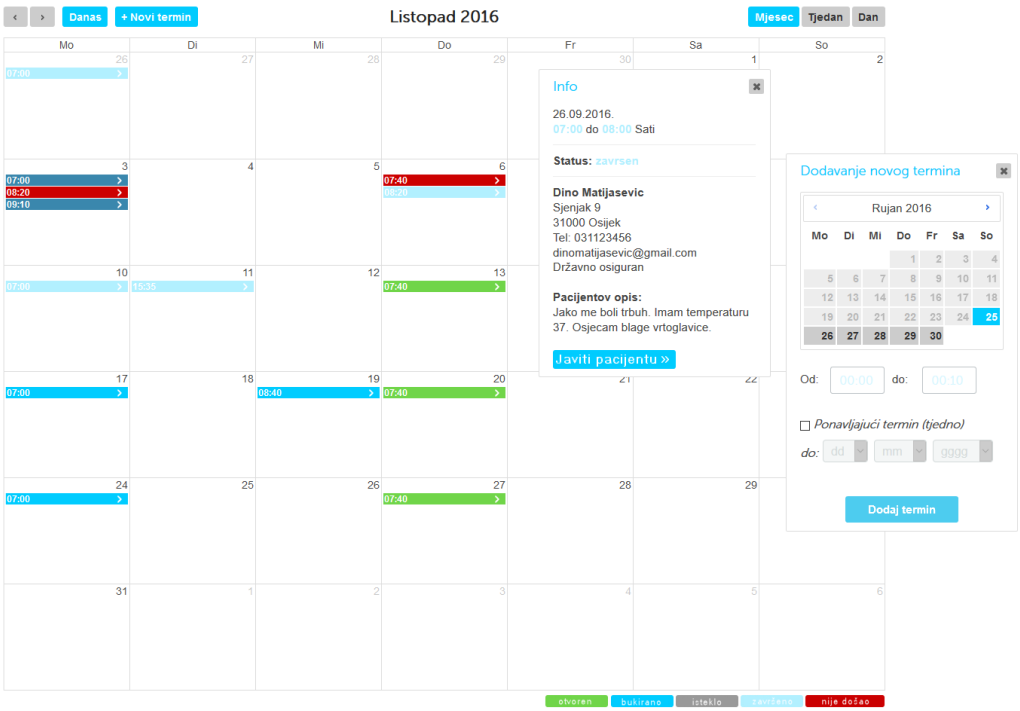
S druge strane trebalo je doktorima omogućiti podjednaku jednostavnost, da imaju grafičko sučelje preko kojeg mogu unositi sve svoje termine, koji će potom biti vidljivi pacijentima.



Sl. 3.1. Konceptualni prikaz logike rada aplikacije

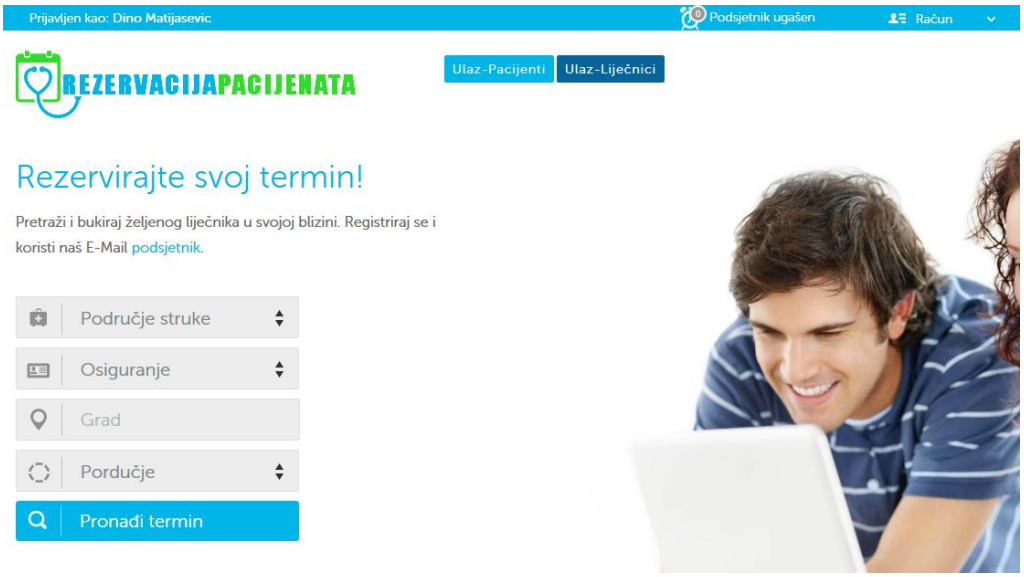
Mora se prije svega računati na činjenicu da krajnji korisnici računala i interneta nisu tako rečeno tehnološki osviješteni, što znači da aplikacija mora biti jasna, pregledna i

jednostavna. S tim mentalitetom, zamišljena je aplikacija u obliku kalendara u kojega doktor unosi termine po danu i satu. Sve termine može pregledati, mijenjati i brisati. Osim toga moraju biti odmah jasni koji su slobodni, a koji su zauzeti, što je nabolje postići vizualno, odnosno sa bojama termina.



Sl. 3.2. Prikaz kalendara kojem imaju pristup samo doktori.

S druge strane, pacijenti ne trebaju kalendar i tražiti koji doktor je kada slobodan, njima više znači sam doktor, npr. zanima ih gdje im je najbliži dermatolog ili ih zanimaju slobodni termini dr. Kovač-a. Zbog toga aplikacija pacijentima prikazuje drugačije sučelje koja ima prije svega svrhu tražilice. Pacijent treba moći pronaći željenog doktora, zakazati željeni termin i ukratko opisati doktoru o čemu se radi.



Sl. 3.3. Prikaz početne stranice

3.2. Izrada stranice

Prije svega trebalo je podesiti razvojno okruženje u kojem se može raditi na aplikaciji, a kako je već spomenuto u ovome radu za to je potrebno LAMP okruženje na lokalnom računalu, točnije aplikacija XAMPP za Windows operativni sustav, koja se može skinuti sa njihove službene stranice (https://www.apachefriends.org/index.html). Instalacija je vrlo jednostavna, dovoljno je ostaviti sve predefinirane postavke kako bi sve radilo bez ikakvih poteškoća.

3.2.1 Kreiranje baze podataka

Sve informacije su spremljene u bazi podataka, a njena struktura je definirana tako da obuhvate sve funkcionalnosti koje su navedene u analizi problematike. Baza je koncipirana od 3 tablice:

- appointment
- doctor
- patient

Tablice *doctor* i *patient* sadrže sve informacije korisničkih računa doktora i pacijenata, a stupci koji se u njima nalaze su:

id (intiger) – redni broj unosa u tablici, automatski se povećava za 1 svakim novim unosom
title (varchar) – titula kao npr dr. ili ing.
gender (varchar) – spol osobe
name (varchar) – ime osobe
surname (varchar) – prezime osobe
email (varchar) – e-Mail kontakt osobe, koji ujedno služi kao korisničko ime prilikom ulaska u aplikaciju
password (varchar) – lozinka korisničkog računa
tel (varchar) – telefonski broj
mob (varchar) – broj mobilnog uređaja
zip (intiger)– poštanski broj osobe
city (varchar) – grad stanovanja osobe
address (varchar) – adresa stanovanja osobe, ulica, naselje
active (binary) – binarni unos, da li je korisnički račun aktivan
newsletter – da li je pacijent prijavljen na newsletter

Dalje se tablice razlikuju u tome što kod pacijenata ima još:

born (varchar) – datum rođenja pacijenta, potrebna doktoru radi lakše dijagnoze
block – da li je pacijent blokiran

Kod doktora imamo još:

spec (varchar) – specijalizacija doktora
subspec (varchar) – podspecijalizacija doktora
pactis (varchar) – ime privatne prakse ili bolnice

Tablica *appointement* sadrži sve informacije samih termina koje doktor unosi poput početnog i krajnjeg vremena termina (datum i vrijeme), status termina kao i eventualnu komunikaciju između pacijenta i doktora.

id (intiger) – ista svrha kao i kod prethodne dvije tablice
start (datetime) – početno vrijeme termina
end (datetime) – završno vrijeme termina
title (varchar) – pacijentov opis poteškoće

diagnose (varchar) – diagnoza doktora

doctor (intiger) – id broj doktora (iz tablice doctor)

patient (intiger) – id broj pacijenta (iz tablice patient)

status (varchar) – trenutni status termina, da li je otvoren, bukiran, završen ili istekao

booked (datetime) – vrijeme kada je termin bukiran

repeats (datetime) – do kada se termin tjedno ponavlja

reminder (int) – da li na ovom terminu ima podešen podsjetnik

last_modified (datetime) – vrijeme zadnje promjene

3.2.2. Kodiranje stranice

Nakon postavljene baze, napravljen je kod same aplikacije. Prije svega potrebno je bilo napraviti poveznicu između baze i aplikacije, što je izvedeno na način spomenut u teorijskom dijelu rada, pomoću MySQLi konektora:

```php
<?php
//Spajanje na bazu
$web = 'localhost';
$user = 'korisnik';
$pass = '0123456';
$baza = 'rezervacija';
$DB = new mysqli($web, $user, $pass, $baza);

echo 'Uspješno spojen na bazu!';
?>
```

Sl. 3.4. Početak koda aplikacije, spajanje na bazu

Kako bi se kasnije lakše povezivalo na bazu, kod spajanja je sačuvan u posebnoj datoteci *db.php* koja se onda samo poziva u ostalim datotekama na vrhu koda pomoću:

```php
require("db.php");
```

Sl. 3.5. Poziv na spajanje baze unutar ostalih datoteka

Na taj način, varijabla $DB sa pozivom na bazu, je uvijek prisutna te nema potrebe u svakoj idućoj datoteci dodavati korisničko ime, lozinku itd.

Aplikacija je rađena od više datoteka koje obavljaju zasebnu funkciju, a međusobna komunikacija između njih je ostvarena uz pomoć jednostavnih POST i GET formi.

Najjednostavniji primjer iz koda je obična prijava na sustav što se može vidjeti na slici 3.6. gdje pacijent treba unijeti e-Mail i lozinku s kojom se registrirao ili može zatražiti novu lozinku.

```
<form method="post" action="login.php">
 <input class="span4" name="email" type="text" placeholder="E-Mail">
 <input class="span4" name="pass" type="password" placeholder="Lozinka">
 <p class="bottom-p"><a href="lozinka-zaboravljena.php">Zaboravljena
lozinka?</a></p>
 <input type="hidden" id="rezo" name="rezo" value="1">
 <input type="submit" class="btn btn-block btn-info submit-mob"
value="Prijava">
</form>
```

Sl. 3.6. Forma prijave pacijenta na sustav

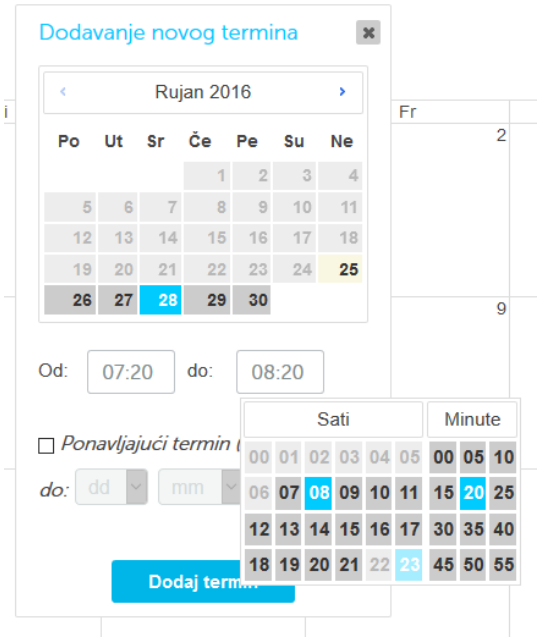
Nakon slanja upita iz navedenog koda, vrijednosti iz <*input*> elemenata sa imenima *email* i *pass* se šalju POST metodom na datoteku **login.php** te se upitom na bazu, uspoređuju podaci. Ako zapis traženog računa postoji i podudara se lozinka, pristup je omogućen i korisnika se dalje šalje ili na kalendar[13] (ako je doktor) ili na podatke o korisniku (ako je pacijent).

Kompletan kod se može naći u prilogu A na CD-u.

3.3. Opis rada stranice

Ako se ulogira kao doktor, prvo što se vidi je kalendar (kojeg možemo vidjeti u slici 3.2.) no valja napomenuti što se tu sve točno može. Prije svega, termini se mogu dodavati jednostavnim klikom na "Novi termin" (dugme u gornjem lijevom uglu). Nakon čega se otvara mali prozor za dodavanje termina (slika 3.7.)

[13] Kalendar je rađen uz pomoć knjižnice *fullcalendar.js* na kojoj su rađene modifikacije kako bi se prilagodilo potrebama aplikacije - FullCalendar <https://fullcalendar.io>, (10.5.2016.)

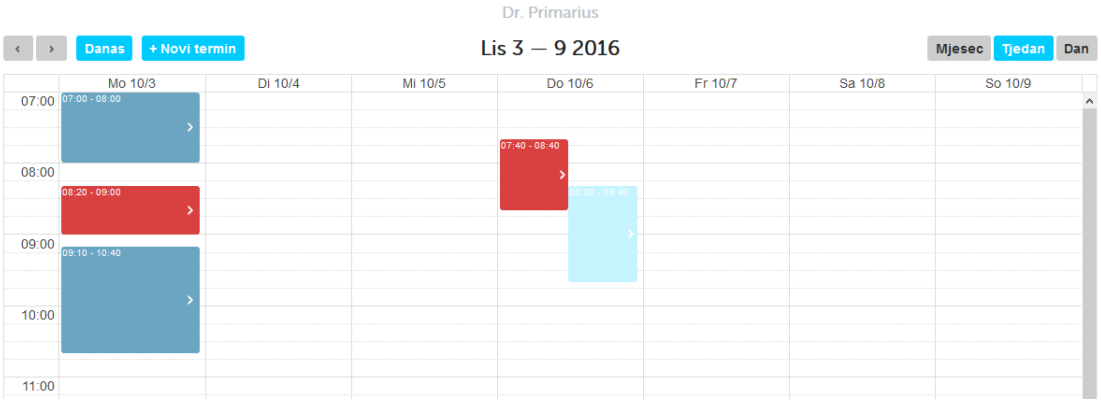


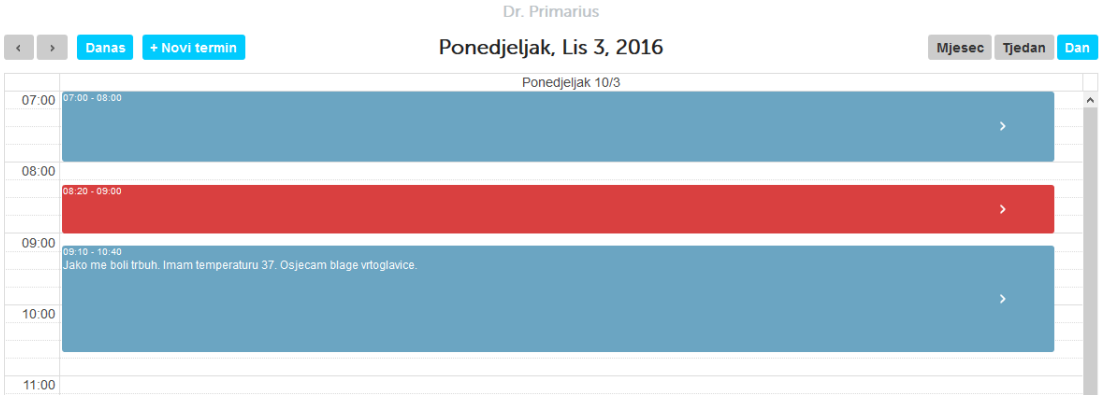
Sl 3.7. Sučelje za dodavanje novog termina

Na slici pri, vrhu može se odabrati mjesec ako se želi dodati termin kasnije, pošto aplikacija uvijek otvara tekući dan. Klikom na datum, može se dalje odabrati termin, kojeg se odabire sa klikovima na željene satnice i minute, te će se same vrijednosti unijeti u polja.

Uz navedeno, postoji i opcija tjednog termina, ako se želi da se termin ponavlja jednom tjedno do zakazanog datuma, npr. ako se na 28.9.2016. postavi termin i odredi da se ponavlja do 1.11.2016., ponavljat će se svakog četvrtka cijeli deseti mjesec.

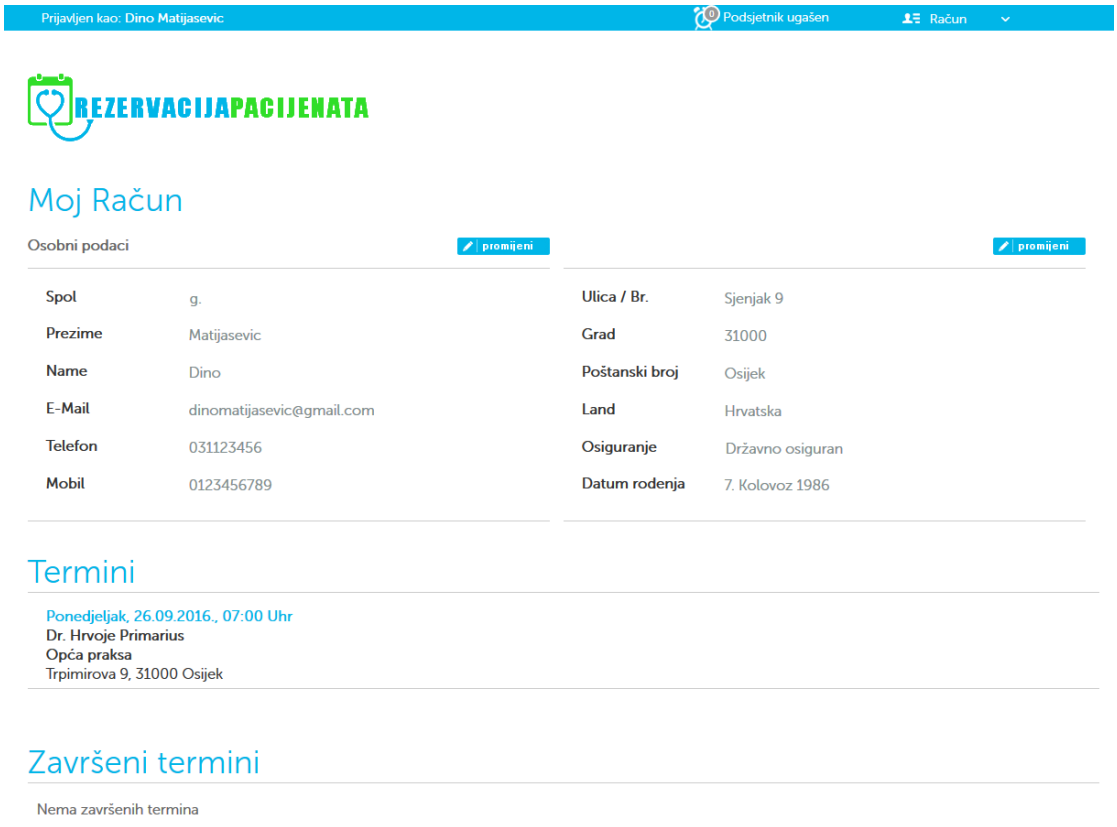
Također se u slici 3.2. vidi i prozor sa otvorenim terminom, u kojemu pacijent može zapisati kratak opis u čemu je problem. Oba prozora se mogu micati radi lakše preglednosti, kako bi doktor lakše mogao vidjeti ostatak svoga mjeseca, no ako želi može i odabrati pregled po tjednima ili po danima.





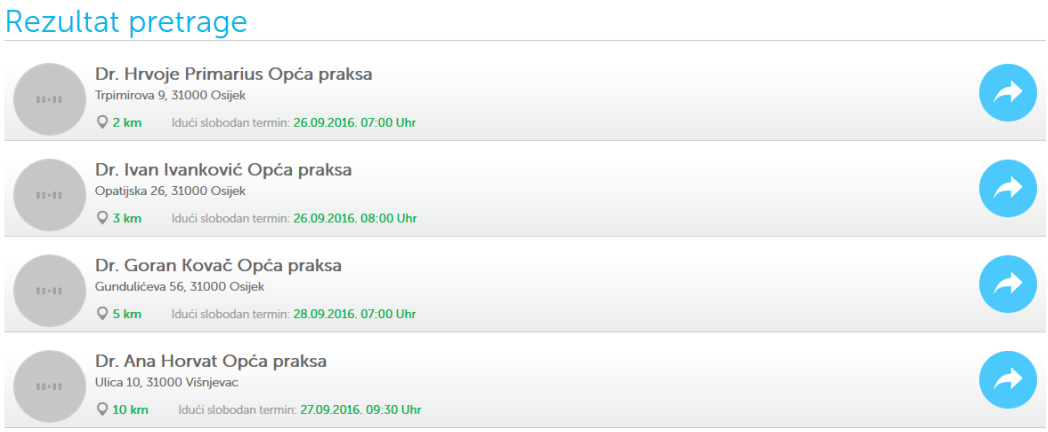
Sl 3.8. Prikaz termina po tjednima (gore) i po danima (dolje)

Od strane pacijenta postoji malo drugačija logika. Njima se prije svega prikaže korisnički račun kako bi imali pregled svih termina na koje su se zakazali. Uz to imaju odmah uvid u svoje podatke, da li su ih dobro unijeli te da iz izmjene po potrebi. (slika 3.9.)



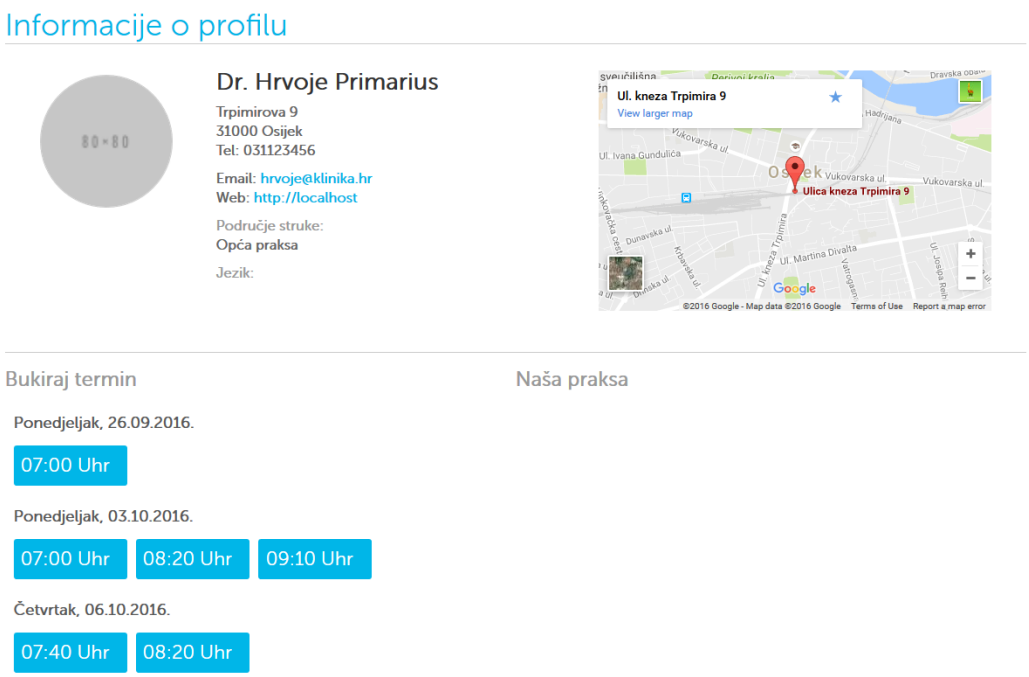
Sl 3.9. Grafičko sučelje korisničkog računa pacijenta

No ako žele pronaći doktora, nakon što su unijeli kojeg doktora trebaju (npr. doktor opće prakse), unose svoju lokaciju i domet područja po kojemu žele tražiti. Rezultat će im prikazati koji doktori su u njihovoj blizini[14] te koji je njihov prvi slobodan termin (slika 3.10.).



Sl. 3.10. Rezultat pretrage po tipu doktora

Odabirom na željenog doktora, dolaze na idući korak gdje odabiru točno koji termin žele. Svi slobodni termini će im biti izlistani, kao i lokacija ordinacije na karti, te neke osnovne informacije o doktoru. (slika 3.11.)



Sl 3.11. Pacijentov odabir slobodnog termina

[14] Udaljenosti su kalkulirane pomoću Google Maps API-a - Distance Matrix API <https://developers.google.com/maps/documentation/distance-matrix/start>, (13.5.2016.)

Kada odabere termin, otvorit će mu se polje za unos opisa svoje poteškoće. Što će doktor vidjeti u svom sučelju kalendara.

4. ZAKLJUČAK

Kao što je već spomenuto u samom zadatku rada, glavni cilj je bio napraviti jednostavnu web aplikaciju u kojoj će se korisnici moći lako snaći. Rezultat je točno to, doktori imaju jednostavno i pregledno grafičko sučelje s čim mogu voditi svoj plan i program svakog radnog dana. Odnos sa pacijentom je sveden na direktnu komunikaciju preko aplikacije kako bi jednostavne probleme mogli riješiti kada stignu, za razliku od telefonskih razgovora koji zahtijevaju momentalni odgovor.

Pacijentima je s druge strane omogućeno jednostavno pronalaženje doktora kojeg trebaju, bilo to opće prakse ili neke vrste specijalizacije i to u njihovoj blizini. Kako ne bi morali listati kroz kalendar kojeg doktori imaju, omogućeno im je pretraga baze preko jednostavne tražilice. Nakon toga, umjesto da pitaju sestru ili samog doktora, kada imaju slobodne termine i zajednički pronaći najbolji termin, prezentiran im je popis nadolazećih slobodnih termina dotičnog doktora i mogu si sami odabrati najbolje vrijeme kada im odgovara, što sve skupa smanjuje administrativne poslove na koje djelatnici troše vrijeme, kao i eliminaciju frustracije pacijenata stupanja u kontakt sa doktorom.

Relativno brz odziv aplikacije je potpomogla njena jednostavnost koda i činjenica da dio odrađuje JavaScript (npr. sučelje kalendara). Jedina prava prepreka bi bila zauzeće baze podataka, tj. nakon određenog vremena će baza porasti sa sve više korisnika i samim time eksponencionalnim rastom broja termina. Brzina čitanja po bazi je svu sreću poprilično velika, no svakako bi ovisila o jačini servera na kojoj se aplikacija nalazi. U slučaju usporavanja aplikacije iz navedenih razloga, moglo bi se rastaviti tablice na još više komada, npr. posebne tablice za korisnike koje bi sadržavale isključivo osobne podatke, a u drugoj tablici korisničke podatke za pristup aplikaciji, a u trećoj adrese i specijalizacije.

Aplikacija ima još mnogo prostora za napredak. Jedan od nedostataka je podsjetnik na nadolazeći termin sa mobilnom aplikacijom koja bi bila povezana sa ovom web aplikacijom jer na podsjetnike mobitela ipak malo više reagiraju (npr. SMS). Još jedna od opcija bi bila sinkronizacija kalendara što se može postići sa protokolom kojeg koristi većina kalendara u web i mobilnim aplikacijama, tzv. CalDAV protokolom koji sadrži sve informacije kalendara kao događaje (eng. event). Preko takvog načina bi se moglo postići praćenje i rezerviranje termina preko bilo kojeg kalendara (na mobitelu, web aplikaciji, osobnom računalu…) koji radi na toj tehnologiji.

5. LITERATURA

1. Wikipedia – HTML <https://en.wikipedia.org/wiki/HTML>, (16.6.2016.)
2. W3C – HTML5 Differences from HTML4<https://www.w3.org/TR/html5-diff>, (16.6.2016.)
3. W3C – Cascading Style Sheets Level 1 <https://www.w3.org/TR/CSS1>, (16.6.2016.)
4. W3C – Cascading Style Sheets Level 2 <https://www.w3.org/TR/CSS2>, (16.6.2016.)
5. W3C – Cascading Style Sheets <https://www.w3.org/Style/CSS>, (16.6.2016.)
6. Wikipedia – PHP <https://en.wikipedia.org/wiki/PHP>, (17.6.2016.)
7. PHP – PHP <http://php.net/manual/en/introduction.php>, (17.6.2016.)
8. Wikipedia – MySQL <https://en.wikipedia.org/wiki/MySQL>, (20.6.2016.)
9. MySQL <http://dev.mysql.com/doc>, (20.6.2016.)
10. PHP – mysqli <http://php.net/manual/en/mysqli.overview.php>, (20.6.2016.)
11. Wikipedia – JavaScript <https://en.wikipedia.org/wiki/JavaScript>, (21.6.2016.)
12. Wikipedia – jQuery <https://en.wikipedia.org/wiki/JQuery>, (21.6.2016.)
13. jQuery <https://jquery.com>, (21.6.2016.)
14. Distance Matrix API <https://developers.google.com/maps/documentation/distance-matrix>, (13.5.2016.)
15. FullCalendar <http://fullcalendar.io>, (10.5.2016.)

6. KRATICE

HTTP – kratica za protokol prijenosa podataka preko interneta(engl. HyperText Transfer Protocol)

SQL – kratica za računalni jezik namijenjen izradi, pretraživanju, obrađivanju podataka u relacijskim bazama podataka (engl. Structured Query Language)

HTML – prezentacijski (opisni) jezik za izradu web stranica (engl. HyperText Markup Language)

HyperText – tekstualna struktura načinjena od više međusobno povezanih informacijskih jedinica

Markup Language – sustav označavanja teksta u dokumenta pomoću tagova, kako bi stroj znao kako ga treba prezentirati na ekran

CSS – kratica za stilski jezik koji služi za uređivanje HTML-a (engl. Cascading Style Sheets)

PHP – programski jezik za izradu web stranica(engl. Personal Home Page, kasnije *Hypertext Preprocessor*)

XML – kratica za programski jezik koji služi za označavanje podataka (engl. EXtensible Markup Language)

mSQL – kratica za Mini SQL, minimalistički sustav za obradu baza podataka

MySQL – sustav za obradu baza podataka koji je nastao kao zamjena za mSQL

SQLite – sustav za obradu baza podataka korišten u mobilnim uređajima koji rade na Android i iOS operativnim sustavima

JS – kratica za JavaScript, programski jezik koji se izvršava u Internet pregledniku korisnika

JQ – kratica za JavaScript ekstenziju jQuery

Ajax – kratica za tehniku korištenja više različitih tehnologija za razvoj web-a na strani klijenta, kako bi se postiglo asinkrone web aplikacije (engl. asynchronous JavaScript and XML)

7. SAŽETAK

Web stranica za naručivanje pacijenata

Web aplikacija čija je svrha olakšati čin odlaska kod doktora sa jednostavnim grafičkim sučeljem u obliku kalendara u kojem doktor unosi svoje nadolazeće slobodne termine. Doktor također unosi svoje informacije kao što su specijalizacija i adresa privatne prakse ili bolnice kako bi pacijent mogao pronaći doktora specijalista koji mu treba u njegovoj blizini. Da bi to postigao, pacijentovo sučelje je koncipirano kao tražilica pomoću koje mu aplikacija izbaci rezultat najbližeg specijalistu i sve njegove slobodne termine. Pacijent zatim rezervira termin i kratko opiše svoj problem na što mu doktor može dati svoj komentar ili konačnu dijagnozu. Za rješenje zadatka, korišteno je okruženje LAMP (Linux operativni sustav, Apache web server, MySQL baza podataka, PHP programski jezik).

Ključne riječi: web aplikacija, doktor, pacijent, kalendar, rezervacija, termin

ABSTRACT

Web page for patient appointments

A web application who's purpose is to make the act of going to the doctor easier with a simple graphic interface in a style of a calendar, where the doctor can input all his upcoming free appointments. The doctor also writes in his information like his expertise and address of his private practice or hospital so the patient can find the specialist he needs closest to his location. For this reason, the patients user interface is like a search engine with witch the application returns a specialist near him and all his free appointments. Then he can book the appointment and also mention the type of his issue on witch the doctor can leave a comment or give his final diagnosis. The application was created using a LAMP environment (Linix operating system, Apache web server, MySQL database, PHP programming language).

Key words: web application, doctor, patient, calendar, rezervation, appointemnt

8. ŽIVOTOPIS

Student je rođen 7. kolovoza 1986. u Osijeku gdje je odrastao i dan danas živi. Pohađao osnovnu školu Mladost u Osijeku za vrijeme koje je išao na natjecanja iz tehničkog i informatike gdje se zaljubio u tehniku i sve vezano za nju.

U srednjoj Elektrotehničkoj i Prometnoj školi Osijek je nastavio tu tradiciju sa odlascima na smotre mladih mehatroničara, a za cijelo to vrijeme kod kuće tražio nove izazove i tako načuo razne vještine, od elektroinstalacije, vodo instalacije, građevine, varenja i slično, no uvijek ga je najviše zanimalo računalo i kako ono "diše". Svakim novim malim projektom u Basic-u ili C-u mu je samo još više rastao interes, a taj interes na kraju doveo u FERIT i upis u smjer računarstva.

Tijekom studiranja je kontinuirano radio razne poslove, pokoji posao u informatičkom sektoru, dosta fizičkih, no najviše anketarskih poslova gdje je radio kao anketar, a kasnije i kao koordinator pojedinih projekata. Uz još nekoliko kolega iz vaterpolo kluba, položio je tečaj za spasioca na vodi, te radio kao spasioc nekoliko sezona, a danas kao koordinator spasioca i tajnik Udruge Spasilaca na Vodi – Osijek.

U rujnu 2014. počeo je volontirati u informatičkoj tvrtki Ofir d.o.o. i u veljači 2015. započeo sa stručnim osposobljavanjem, a završetkom osposobljavanja potpisao ugovor o zapošljavanju gdje radi kao PHP programer na razvoju WEB aplikacija u platformama poput Wordpress, PrestaShop, Plentymarkets i Moodle.

9. PRILOZI

Svi prilozi se nalaze na CD-u pod datotekom PRILOZI