Aplikacija za naručivanje korisnika i vremensko raspoređivanje poslova

Gogić, Šimun

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

Sveučilišni studij

APLIKACIJA ZA NARUČIVANJE KORISNIKA I VREMENSKO RASPOREĐIVANJE POSLOVA

Diplomski rad

Šimun Gogić

Osijek, 2017.

SADRŽAJ

1. UVOD

Tema ovog diplomskog rada je izrada web aplikacije koja registriranim korisnicima omogućuje rezerviranje termina za usluge koje nudi pružatelj usluga. Diplomski rad se sastoji od dva dijela. Prvi dio je praktičan i predstavlja izradu same web aplikacije, dok drugi dio predstavlja dokumentaciju izrađene web aplikacije.

U prvom dijelu dokumentacije diplomskog rada opisane su tehnologije koje su korištene prilikom izrade web aplikacije. Nakon toga, objašnjena je realizacija same web aplikacije, zajedno sa snimkama zaslona realizirane aplikacije. I na kraju, dan je zaključak koji pojašnjava zašto je aplikacija korisna i u kojim se sve djelatnostima može koristiti.

1.1. Zadatak diplomskog rada

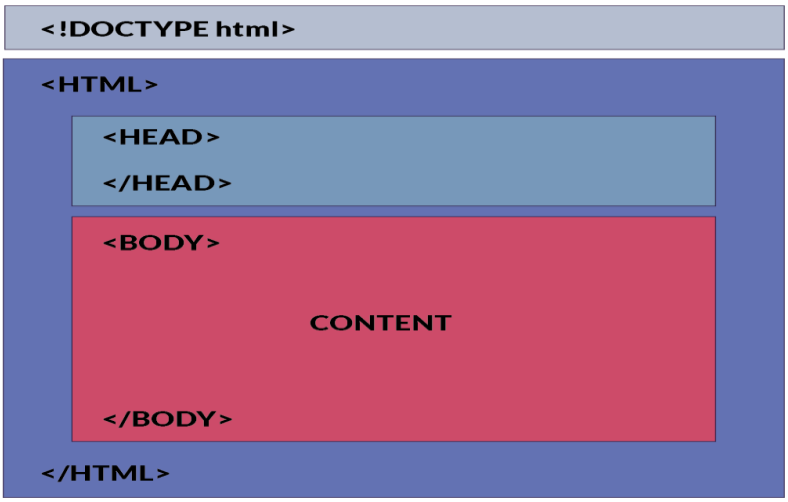
Ovaj rad je nastao iz potrebe za jednostavnijim i praktičnijim upravljanjem narudžbama u malim i srednjim obrtima te uslužnim djelatnostima. Postoje dva režima rada aplikacije, korisnički i administratorski. U korisničkom režimu, web aplikacija omogućuje registriranom korisniku prijavu za ponuđenu uslugu u određenom vremenu i trajanju. U administratorskom režimu, prikazana je tablica svih prijavljenih usluga, te administrator prihvaća ili odbija prijave. Ukoliko administrator prihvati prijavu, potvrdni e-mail s podacima o terminu usluge se šalje korisniku čiji je termin usluge prihvaćen. Također, u ovom režimu rada, administrator može dodavati i brisati nove usluge koje će biti dostupne korisnicima te određivati svoje radno vrijeme. Ovaj režim rada sadrži i kalendar sa svim prihvaćenim terminima.

2. KORIŠTENE TEHNOLOGIJE

U ovom diplomskom radu korišteni su HTML, CSS, Bootstrap i Javascript na strani korisnika, a MySQL i PHP na strani servera.

2.1. HTML

HTML ili HyperText Markup Language je označni jezik za izradu statičnih web stranica [1]. Pomoću njega se semantički opisuje struktura web stranica. HTML elementi su gradivne jedinice svakog HTML dokumenta. Pomoću HTML elemenata možemo ugraditi različite objekte na web stranicu, kao što su slike, tekst, linkove,... HTML elemente okružuju tagovi koji predstavljaju početak i kraj HTML elementa. Tagovi se sastoje od znaka < (manje od) i > (veće od) između kojih se nalazi naziv HTML elementa. Web preglednici ne ispisuju tagove, nego im oni služe za interpretiranje sadržaja web stranice. Svaki element može sadržavati atribute koji određuju njegovo ponašanje. HTML element <html> označava početak dokumenta. Prije <html> taga može se staviti <!DOCTYPE> element kojim se određuje verzija standarda koji se koristi za izradu HTML dokumenta.



Sl. 2.1. Struktura HTML dokumenta

Unutar <html> elementa se nalaze <head> i <body> elementi [2]. <head> element predstavlja zaglavlje HTML dokumenta te sadrži metadata-u, tj. podatke o HTML dokumentu, kao što su svojstva jezika, naslov dokumenta te povezivanje s vanjskim dokumentima, najčešće CSS-om ili Javascript-om. U <body> elementu se kreira sadržaj same web stranice [3].

2.2. CSS

CSS (eng. Cascading Style Sheets) je stilski jezik koji služi za prezentaciju dokumenta napisanog u jednom od opisnih jezika, kao npr. HTML [4]. Uglavnom se koristi zbog odvajanja strukture sadržaja web stranice od njenog dizajna, kao što su boje, fontovi ili položaj elemenata u HTML dokumentu. Odvajanje strukture od dizajna omogućuje jednostavniji pristup, veću fleksibilnost i kontrolu nad izgledom HTML elemenata. Više HTML dokumenata mogu koristiti jedan CSS dokument za definiranje njihovog izgleda čime izbjegavamo ponavljanje koda. CSS ima jednostavnu sintaksu koja se sastoji od engleskih riječi koje predstavljaju određeno svojstvo dizajna. CSS kod se sastoji od liste pravila.

```
2  body   {
3    background-attachment: scroll;
4    background-color: #FF8080;
5    background-position: inherit;
6    border-color: #FFFFFF;
7    border-style: dotted;
8    color: #000000;
9    float: left;
10   font-family: Arial, Helvetica, sans-serif;
11   font-size: medium;
12   font-style: normal;
13   font-variant: normal;
14   font-weight: normal;
15   height: auto;
16   letter-spacing: normal;
17   line-height: normal;
18 }
```

Sl. 2.2. Struktura CSS dokumenta

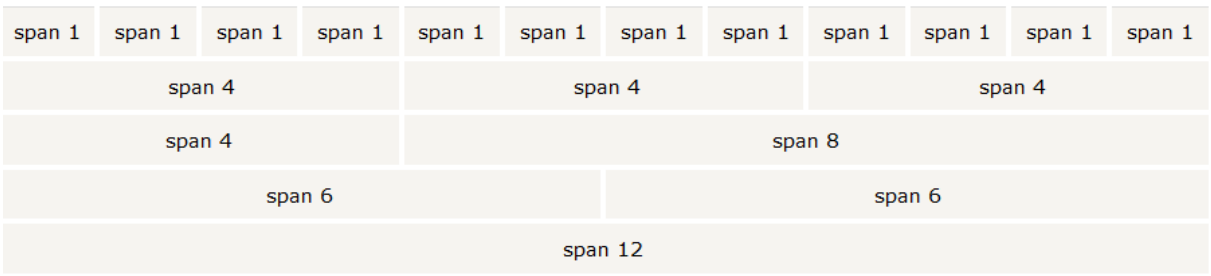
Svako pravilo se sastoji od jednog ili više selektora i deklaracijskog bloka. Selektor predstavlja određeni HTML element na koji se želi primjeniti određeni dizajn. Selektori se mogu primjeniti na sve elemente određenog tipa, npr. HTML odlomak ili <p> element. Elementima se može pristupiti i pomoću atributa kao što su id ili klasa [5]. Id je jedinstven za pojedini element, dok se klasa može odnositi na više elemenata. Također, na elemente se može primjeniti dizajn ovisno o tome gdje se nalaze u DOM-u. DOM ima strukturu stabla gdje svaki čvor prestavlja određeni dio HTML dokumenta. Deklaracijski blok se sastoji od liste svojstava koje se primjenjuju na određeni HTML element. Ukoliko postoji više svojstava u deklaracijskom bloku, onda se one razdvajaju točka-zarezom [6]. CSS kod se može nalaziti na tri mjesta. Može se nalaziti u HTML dokumentu u zaglavlju (eng. head) između tagova

<style>, u vanjskom dokumentu koji se onda povezuje sa HTML dokumentom pomoću <link> taga, ili u samom HTML elementu na koji se želi primjeniti dizajn u obliku atributa.

2.3. Bootstrap

Bootstrap je besplatan razvojni okvir otvorenog koda za izradu web stranica i aplikacija [7]. Sadrži predloške pisane u HTML-u i CSS-u za razne tipove komponenti korisničkog sučelja, kao i različite JavaScript ekstenzije. 2014. Bootstrap je bio najveći projekt na GitHub-u. Web dizajnerima omogućuje da jednostavno i brzo izrade responzivne web stranice, tj. stranice koje se automatski prilagođavaju na različite veličine zaslona, od malih kod pametnih telefona do velikih zaslona. Kompatibilan je sa svim vrstama modernih web preglednika (Chrome, Firefox, Internet Explorer, Safari, and Opera). Bootstrap dokumenti (JavaScript, CSS) se mogu uključiti u HTML dokument pomoću linka na CDN (eng. Content Delivery Network) ili linkom na lokalni bootstrap dokument na računalu [8].

Postoje dva tipa spremnika, container i container-fluid. Container omogućuje prikaz spremnika sa fiksiranom vrijednošću širine, dok container-fluid omogućuje prikaz spremnika koji zauzima cjelokupnu širinu web stranice. Prikaz web stranice u boostrap-u se zasniva na sustavu rešetki od 12 stupaca čija se širina prilagođava veličini zaslona. Sustav rešetki sadrži četiri različite klase: xs, sm, md, lg. Xs se koristi za pametne telefone, sm za tablete, md za desktop zaslone i lg za velike zaslone. Te četiri klase se najčešće kombiniraju kako bi web stranica bila prilagodljiva na većini tipova uređaja. Prednost Bootstrapa je i ta što je većina korisnika već skinulo Bootstrap prilikom posjete nekoj drugoj web stranici. Rezultat toga je da će se Bootstrap idući put učitati iz priručne memorije te će se na taj način povećati brzina učitavanja iduće web stranice. Također, sadrži i nekoliko JavaScript komponenti u obliku Jquery dodataka. Takvi dodaci omogućuju dodatne elemente korisničkog sučelja kao što su dialoški i modalni okviri, poruke upozorenja,...



Sl. 2.3. Sustav rešetki Bootstrap-a

2.4. JavaScript

JavaScript je najpopularniji skriptni jezik na Internetu kojeg podržava većina modernih web preglednika. Stvorila ga je kompanija NetScape 1996. godine kako bi se postigla interaktivnost i dinamičnost web stranica [9]. ECMA-262 standard za JavaScript odobren je od strane međunarodnog ISO (ISO/IEC 16262) standarda 1998. godine. JavaScript je interpreter, što znači da se skripta izvodi odmah naredbu po naredbu, bez prevođenja izvornog koda programa i kreiranja izvršne datoteke. Drugim riječima, ne prevodi se u strojni jezik (binarni kod).

Zbog karakteristike da se izvršava naredbu po naredbu, JavaScript se izvodi na strani korisnika. Također je i objektno orijentiran jer programer ne definira samo tip podatka, već i vrstu funkcija koje se mogu primjeniti na strukture podataka. Na taj način, struktura postaje objekt koji uključuje i podatke i funkcije. JavaScript kod se može pozvati iz HTML dokumenta tako da se nalazi unutar tagova <script> i </script> u head ili body sekciji, ili se može pozvati vanjski JavaScript dokument putem linka [11]. JavaScript omogućuje pretvaranje dinamičkog teksta u HTML stranicu (npr. ispis vrijednosti određene varijable), reagiranje na događaje (npr. klikom na određeno dugme), manipulaciju HTML elemenata (npr. promjena boje HTML bloka), validiranje podataka (npr. kod prijave, kako bi se smanjio opseg provjere na strani servera) ili kreiranje kolačića kako bi se spremile informacije o korisnikovom računalu [10].

2.4.1. JQuery

JQuery je JavaScript biblioteka koja radi na više sustava (eng. cross-platform), npr. Linux 32-bitnoj ili Windows 64-bitnoj arhitekturi [16]. Obuhvaća brojne zadatke koji zahtjevaju veliki broj naredbi JavaScripta i omotava ih u metode koje se mogu pozvati jednom linijom koda. JQuery biblioteka obuhvaća funkcionalnosti kao što su HTML/DOM manipulacija, CSS manipulacija, AJAX, efekti i animacije i metode HTML događaja. Tri najbitnije stvari koje su izdvojile JQuery u odnosu na konkurenciju su korištenje CSS selektora, nenametljiv JavaScript i jednostavnost upotrebe. JQuery koristi identične CSS selektore koje web dizajneri koriste za dizajniranje web stranica. Nenametljiv JavaScript je sintagma koja definira napredni način implementiranja JavaScript koda na web stranicama. Osnovna zadaća ovog principa je da se odvoji funkcionalnost od sadržaja i prezentacije. Na taj način se izbjegavaju standardni problemi kao što su nekonzistentnost na web preglednicima te

nedostatak skalabilnosti. JavaScript je kreiran tako da bude krajnje jednostavan za programere koji ga koriste.

2.4.2. AJAX

AJAX (asinkroni JavaScript i XML) je skupina međusobno povezanih tehnologija korištenih na strani klijenta za razvoj sinkronih i asinkronih web aplikacija. Pomoću AJAX-a, web aplikacije mogu slati i primati podatke sa servera asinkrono, tj. u pozadini bez promjene trenutnog rada i prikaza stranice. Bez obzira što se XML nalazi u imenu AJAX-a, njegovo korištenje nije obavezno te se JSON vrlo često koristi, a ni zahtjevi ne moraju biti asinkroni. Tijekom 1990-ih većina stranica je bila bazirana na isključivo HTML dokumentima. Svaka akcija je zahtjevala ponovno učitavanje sa servera te je korisniku takav pristup bio neefikasan jer bi cjelokupni sadržaj stranice nestao pa se ponovno pojavio. Isto tako, ovaj način rada je opterećivao servere jer bi se i zbog male promjene sadržaja stranica ponovno učitala te bi sav sadržaj ponovno bio poslan na server. Upravo iz tih razloga nastala je potreba za AJAX-om.

Ukoliko se koristi nestabilna internet veza, dinamičko ažuriranje web stranica može ometati interakciju sa korisnikom. Također, dinamičko ažuriranje web stranica otežava vraćanje web aplikacija i stranica na određeno stanje. Isto tako, kod starijih preglednika, stranice kreirane dinamičkim korištenjem uzastopnih AJAX zahtjeva nisu bile automatski zapisane u povijesti pretraživača tako da se, klikom na dugme „nazad“, preglednik nije uvijek vraćao na prethodno stanje stranice, već na posljednju posjećenu stranicu. Rješenje prethodno dva navedena problema je da se koristi identifikator fragmenta (dio URL-a poslije #) [12].

```
1 $.ajax({
2   method: "POST",
3   url: "some.php",
4   data: { name: "John", location: "Boston" }
5 })
6   .done(function( msg ) {
7     alert( "Data Saved: " + msg );
8   });
```

Sl. 2.4. Struktura AJAX poziva u JQuery-u

2.5. PHP

PHP (eng. Hypertext Preprocessor) je skriptni jezik namjenjen uglavnom za web razvoj [13]. Vrlo lako se može umetnuti u HTML dokument te na taj način dobiti dinamički kreirane web stranice. Izvodi se na serveru te se po tome razlikuje od drugih skriptnih jezika, poput JavaScripta, koji se izvode na klijentu [14]. PHP se može koristiti na svim poznatim operacijskim sustavima, kao što su Linux ili razne varijante UNIX-a (Solaris, OpenBSD,...), Windows te MAC OS X. Također, PHP ima podršku za većinu web poslužitelja, npr. Apache ili IIS. Standardni PHP interpreter, tj. software koji analizira i izvodi PHP program liniju po liniju, je besplatan software koji je razvila tvrtka Zend Engine. Do 2014. godine, PHP se razvijao bez formalnog standarda, pa je u stvarnosti PHP interpreter predstavljao standard.

```
<!DOCTYPE html>
<html>
<body>

<?php
$color = "red";
echo "My car is " . $color . "<br>";
echo "My house is " . $COLOR . "<br>";
echo "My boat is " . $coLOR . "<br>";
?>

</body>
</html>
```

Sl. 2.5. Umetanje PHP koda u HTML dokument

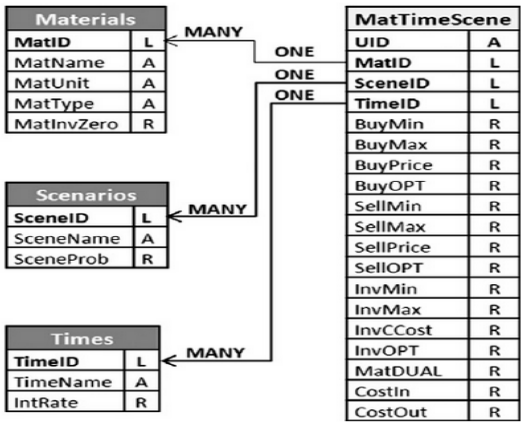
Razvoj PHP-a započeo je Rasmus Lerdorf 1995. godine kada je napisao nekoliko CGI skripti, tj. skripti koje se izvode na serveru u obliku konzolne aplikacije, kako bi održavao svoju osobnu web stranicu. Te skripte su proširene kako bi komunicirale sa bazama podataka te radile s web obrascima. 2004. godine, izašla je verzija PHP 5 koja je sadržavala nova poboljšanja, kao što su naprednija podrška za objektno-orijentirano programiranje, PDO (eng. PHP Data Objects) ekstenzija, koja predstavlja jednostavno i konzistentno sučelje za pristup bazama podataka, te brojna poboljšanja samih performansi. Zadnju stabilnu verziju predstavlja PHP 7, razvijanu tijekom 2014. i 2015. godine. PHP 7 je drastično poboljšao performanse, tj. brzina izvođenja programa je veća te je potreban manji broj servera za posluživanje istog broja korisnika. Smanjila se potrošnja memorije u odnosu na PHP 5,

dodana su poboljšanja u upravljanju iznimkama (eng. EngineError Exception) te anonimne klase, koje su već dugo godina bile prisutne u drugim objektno-orijentiranim jezicima.

2.6. MySQL

MySQL je sustav upravljanja relacijskim bazama podataka (eng. RDBMS – Relational Database Management System) [15]. Relacijske baze podataka predstavljaju temelj svakog modernog poslovnog subjekta jer su se pokazale kao najbolji način pohrane i pretraživanja velikih količina podataka. MySQL je open source projekt, tj. svatko može doprinijeti i mijenjati sadržaj izvornog koda. Podaci u ovakvim bazama se organiziraju u skup relacija između kojih se definiraju određene veze. Svaka relacija mora imati primarni ključ koji predstavlja atribut pomoću kojega se jedinstveno identificira svaka n-torka. Također, relacije proizvoljno mogu posjedovati i strani ključ, preko kojega se ostvaruje veza sa ostalim relacijama [18].

Odnosi između raznih objekata u bazi predstavljeni su vezama. Postoje tri tipa veza: jedan prema jedan (eng. one-to-one), jedan prema više (eng. one-to-many) i više prema više (eng. many-to-many). Kod jedan prema jedan veze, jednoznačna vrijednost primarnog ključa može povezivati samo jedan vanjski ključ, tj. svaki zapis u jednoj tablici može imati samo jedan odgovarajući zapis u drugoj. Kod jedan prema više veze, jednoznačna vrijednost primarnog ključa može povezivati jedan, više ili nijedan zapis u povezanoj tablici. Veza više prema više se rješava kreiranjem treće tablice, tzv. Tablice sjecišta, koja odnos više prema više rastavlja na odnos jedan prema više. U tablici sjecišta se nalaze primarni ključevi iz obje tablice.



Sl. 2.6. Shematski prikaz relacijske baze podataka

MySQL baza je vrlo stabilna i ima dobro dokumentirane module i ekstenzije te podršku za razne programske jezike kao što su PHP, Java ili Python. Velika prednost MySQL-a je što postoje verzije za sve operacijske sustave te je besplatan za kućnu upotrebu jer se izdaje pod GPL licencom. Također zbog jednostavnosti korištenja, dobrih performansi i pouzdanosti, MySQL je najzastupljenije RDBMS rješenje na internetu. Glavno ograničenje MySQL je neusklađenost sa SQL standardom te je moguće ignorirati standardnu SQL sintaksu bez prijavljivanja grešaka. Nadalje, upotreba okidača ograničena je na samo dva okidača nad jednom tablicom, prvi prije i drugi poslije unosa podataka.

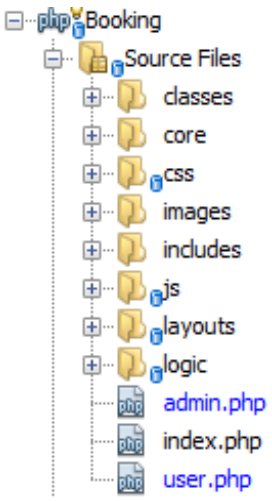
Podrazumijevani način rada s MySQL-om je komandna linija jer ne dolazi predefinirano sa grafičkim alatom. No postoje razni grafički alati razvijeni za rad s MySQL-om. Osnovna namjena tih alata je dizajn baza podataka, modeliranje podataka te administracija bazama podataka. Neki od poznatijih grafičkih alata za rad s MySQL-om su phpMyAdmin, SQLYog, DBStudio, OpenOffice, ...

3. REALIZACIJA WEB APLIKACIJE

Nakon što su u prošlom poglavlju objašnjene tehnologije pomoću kojih je ovaj diplomski rad napravljen, u ovom poglavlju je opisan način realizacije same web aplikacije. Aplikacija se sastoji od dva režima rada: administratorski i korisnički.

3.1. Struktura projekta

Struktura samog projekta je podjeljena na nekoliko mapa kako bi se programer lakše snalazio u samoj strukturi. Postoje tri stranice koje klijent, bilo to administrator ili korisnik aplikacije, može posjetiti: 'admin.php', 'user.php' i 'index.php'. Navedene tri stranice se nalaze u root-u projekta i zamišljene su kao predložak u koji se uključuju ostale datoteke u projektu koje imaju različitu svrhu, od CSS i JavaScript pa sve do PHP datoteka. Takve datoteke su uključene u predloške pomoću 'require_once' naredbe u PHP-u koja se razlikuje od 'require' naredbe po tome što provjerava je li datoteka već uključena u stranicu, te ako je, ona neće biti ponovno uključena.



Sl. 3.1. Struktura projekta

Ostali dokumenti u hijerarhiji projekta su: classes, core, css, images, includes, js, layouts i logic. U classes mapi nalaze se sve klase koje su potrebne za realizaciju projekta. Svaka klasa ima svoju datoteku koja se zove istim imenom, kako bi PHP autoloader mogao uvesti tu klasu u projekt. U core mapi nalazi se 'init.php' datoteka u kojoj se nalazi globalna varijabla koja sadrži podatke o konekciji na bazu i varijabli sesije, te autoloader funkcija. Autoloader

funkcija služi da automatski uključimo klase u datoteku prilikom kreiranja objekta te klase [19]. To je korisno ukoliko postoji veliki broj klasa koje koristimo pa da ne moramo pomoću require naredbe ručno uključivati svaku klasu u datoteku, samo na početku svake datoteke uključimo putanju /core/init.php.

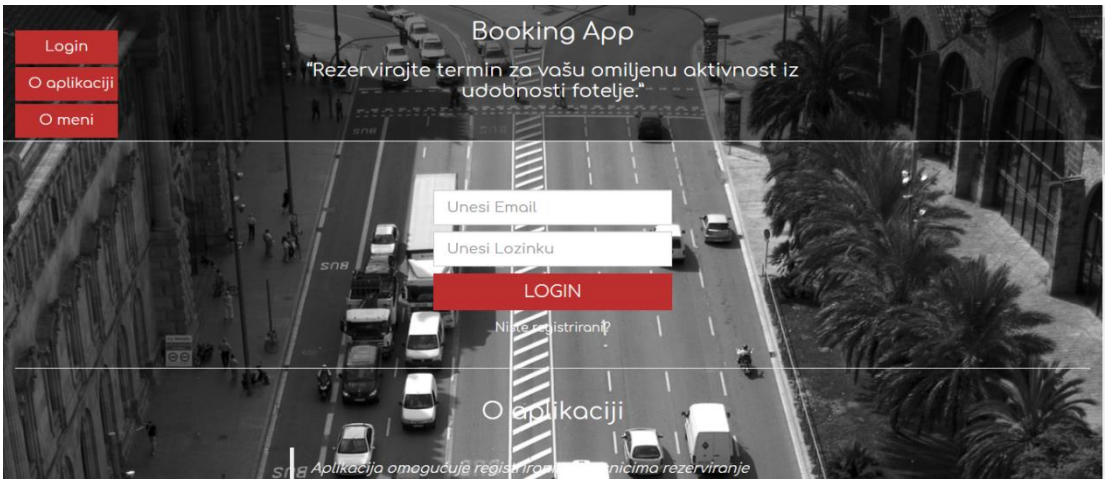
```
require_once('../core/init.php');
```

Sl. 3.2. Uključivanje inicijalizacijske datoteke

Kao što i samo ime kaže, u css, js i images mapama se nalaze CSS i JavaScript datoteke, te slike. U includes mapi se nalaze open source projekti koji su preuzeti s interneta kako bi omogućili neke funkcionalnosti kao npr. slanje e-maila. U layouts mapi se nalaze HTML predlošci, a u logic mapi sva potrebna logika aplikacije.

3.2. Početna stranica

Na početnoj stranici nalaze se obrasci za registraciju i prijavu korisnika i administratora te informacije o aplikaciji i kreatoru aplikacije. Obrazac za prijavu je jedinstven za oba režima rada, tj. jedan obrazac se koristi za prijavu u oba režima rada.



Sl. 3.3. Obrazac za prijavu

3.2.1. Prijava

Prijava je realizirana pomoću JavaScripta i PHP-a. Klikom na gumb za prijavu pokreće se JavaScript kod koji poziva AJAX funkciju koja zatim šalje podatke iz polja za prijavu u PHP skriptu 'login.php' koja se nalazi u logic mapi.

```
$.ajax({
type:"POST",
url: "logic/login.php",
data:$(".loginForm").serialize(),
dataType: 'json',
success:function(array)
```

Sl. 3.4. Struktura AJAX poziva

Data postavka šalje varijable url-u. Sva polja obrasca za prijavu su serijalizirana kako se pojedinačne tekstualne vrijednosti polja ne bi morale spremati u varijable pa unositi u data postavku. Url specificira datoteku kojoj se prosljeđuju varijable definirane u data postavci. *Success* funkcija se pokreće ukoliko su varijable prosljeđene PHP datoteci i ako je datoteka uspješno izvedena, tj. ako se negdje nije pojavila greška pri izvođenju. Type predstavlja tip HTTP zahtjeva kojim šaljemo podatke na server. *DataType* definira kakav odziv se može očekivati kada se url datoteka izvede. U ovom slučaju je to JSON, jer u 'login.php' datoteci *echo* naredbom šaljemo odziv u obliku JSON niza.

```php
<?php

require_once('../core/init.php');

if(Input::exists()){
    $validate = new Validation();
    $validate->check($_POST, array(
        'emailLogin' => array(
            'name' => 'Email',
            'required' => true
        ),
        'passwordLogin' => array(
            'name' => 'Lozinka',
            'required' => true
        )
    ));

    $user = new User();
    $login = $user->login(Input::get('emailLogin'), Input::get('passwordLogin'));

    if($login->hasPassed())
        Session::put('email', Input::get('emailLogin'));

    echo json_encode(array_merge($validate->getMessages(), $login->getMessages()));
}
```

Sl. 3.5. 'login.php' datoteka

Skripta 'login.php' izvodi validaciju prijave pomoću objekta klase *Validation*. Metoda *check* klase *Validation* poziva funkciju *addError* koja u niz pogrešaka dodaje pogrešku ukoliko ona postoji. Metoda *getMessage* vraća sve pogreške pa tu metodu koristimo u *echo* naredbi 'login.php' datoteke kako bi mogli manipulirati s porukama pogrešaka u AJAX *success* funkciji. AJAX *success* funkcija manipulira s porukama pogreške primljenima iz 'login.php' skripte na način da dodaje crveni obrub ukoliko je polje prazno ili dodaje tooltip sa tekstom pogreške.

Statična metoda *get* klase *Input* služi za dohvaćanje vrijednosti POST ili GET varijable, a u ovom slučaju služi za dohvaćanje teksta iz polja za prijavu. Metoda *hasPassed* klase *Validation* provjerava postoje li pogreške prilikom prijave. Ukoliko one ne postoje, metoda vraća true, te se u tom slučaju, pomoću statične metode *put* klase *Session* postavlja varijabla sesije nazvana 'email' na vrijednost teksta polja 'emailLogin' u obrascu prijave. Nakon toga, AJAX *success* funkcija koja se nalazi u 'login.js' datoteci, preusmjerava web aplikaciju na 'admin.php' ili 'user.php', ovisno je li se prijavio korisnik ili administrator.

```
if(array.passed == true && array.admin == true)
    window.location = 'admin.php';
else if(array.passed == true)
    window.location = 'user.php';
```

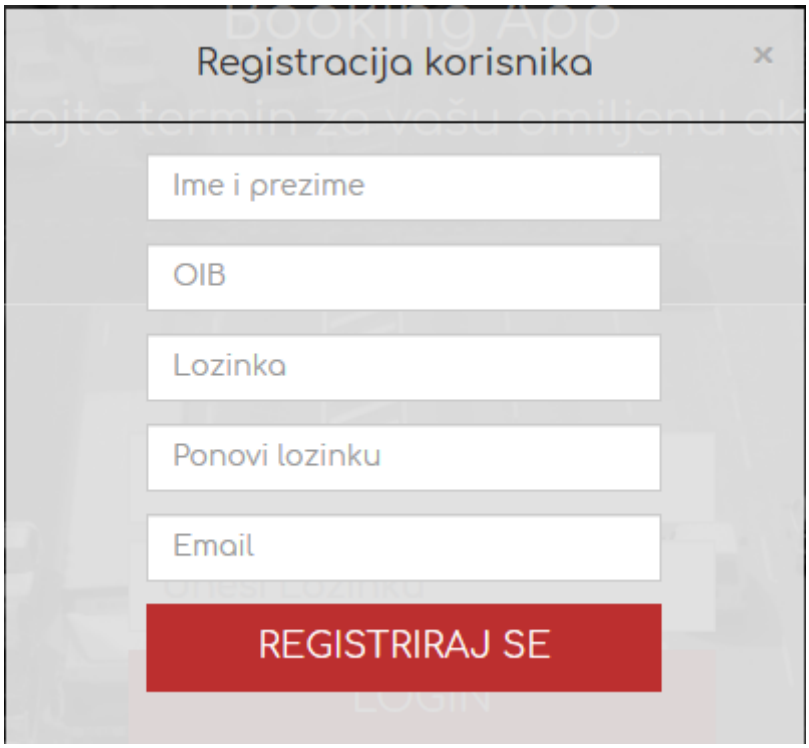
Sl. 3.6. Preusmjeravanje aplikacije ovisno o režimu rada

AJAX *success* funkcija kao parametar prima asocijativni niz koji se sastoji od elemenata 'inputNames' koji služi za dodavanje pogrešaka ukoliko su polja prazna, 'errorLogin' za dodavanje pogrešaka ako polja nisu prazna, npr. ako je lozinka pogrešna ili ako korisnik ne postoji u bazi, 'passed' tipa boolean koji ima vrijednost ovisno o tome je li prošla validacija ili ne, i 'admin', također boolean, koji ima vrijednost ovisno o tom je li prijavljena osoba korisnik ili administrator.

3.2.2. Registracija

Klikom na link 'Niste registrirani?' otvara se modalni prozor u kojemu se nalazi obrazac za registraciju. Modalni prozor je realiziran u Bootstrapu uz određene izmjene dizajna koje se nalaze u modal.css datoteci. Obrazac se sastoji od pet polja: ime i prezime, OIB, lozinka,

ponovljena lozinka i email. Sva polja su obavezna. Ime i prezime nema restrikcija prilikom validacije. OIB mora biti niz od 11 znamenki. Lozinka mora sadržavati između 6 i 32 znaka.



Sl. 3.7. Obrazac za registriranje

Ukoliko je validacija uspješna, lozinka se hashira pomoću statične metode *make* klase *Hash* koja vraća rezultat PHP funkcije password_hash koja koristi bcrypt algoritam. Hashiranje je postupak kreiranja nerazumljivog niza znakova od početnog niza znakova. Na taj način prilikom hakiranja baze podataka, lozinka ostaje otporna na krađu. Ponovljena lozinka se mora poklapati sa upisanom lozinkom, te email mora biti validnog formata. Funkcija filter_var kao drugi parametar prima FILTER_VALIDATE_MAIL pomoću kojega se ispituje format emaila. Također, e-mail ne smije biti dulji od 50 znakova jer polje u tablici korisnika u koje se pohranjuje e-mail može spremiti maksimalno 50 znakova. Isto to se i odnosi i na polje u koje se unosi ime i prezime.

Klikom na gumb 'Registriraj se' pokreće se JavaScript kod u datoteci 'register.js' koji poziva AJAX funkciju koja prosljeđuje vrijednosti polja u obrascu PHP skripti 'register.php'. Skripta 'register.php' vraća JSON niz JavaScript datoteci 'register.js' koja u *success* funkciji izvršava određene akcije na klijentskoj strani aplikacije ovisno o porukama koje je proslijedila PHP skripta. Ukoliko su polja prazna prilikom klika na gumb, ona poprimaju crveni obrub.

Ako su se pojavile neke od pogrešaka, *success* funkcija dodaje tooltip s odgovarajućom porukom.

Ukoliko je validacija prošla, šalje se email registriranom korisniku kako bi ga potvrdio i mogao se prijaviti u aplikaciju. Klasa *Email* služi kao klasa spremnik od PHPMailer klase koja je skinuta s interneta. *PHPMailer* klasa ima već predefinirane funkcije za postavljanje lozinke, korisničkog imena, porta, hosta, sadržaja i naslova emaila. Ova klasa je korištena zbog svoje jednostavnosti jer se lokalno „čistim“ PHP-om ne može slati e-mail bez podešavanja konfiguracije u XAMPP-u. Važno je napomenuti da se slanje emaila ne događa asinkrono sa unosom korisnika u bazu kako ne bi došlo do zastoja aplikacije. Najprije se korisnik unese u bazu nakon što validacija prođe, pa se nakon toga pomoću novog AJAX poziva prosljeđuju parametri u PHP datoteku 'sendmail.php' koja šalje e-mail. U suprotnome, kada bi se email slao u 'register.php' datoteci, došlo bi do zastoja od nekoliko sekundi dok se e-mail ne pošalje u pozadini aplikacije.

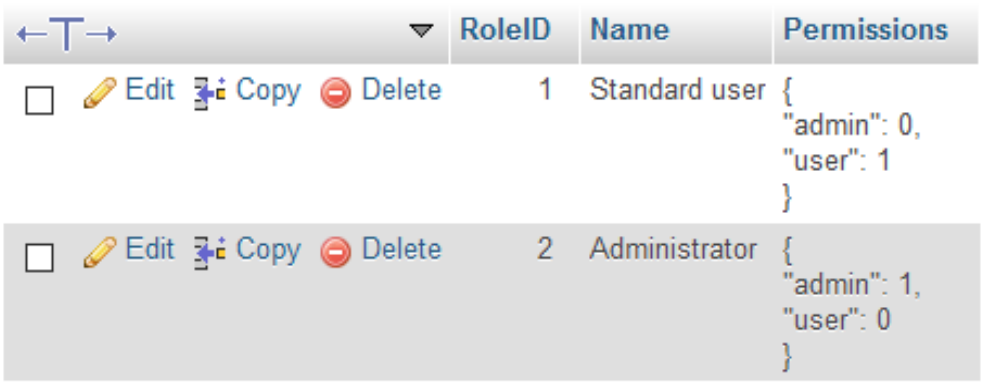
Poslani e-mail sadrži aktivacijski link koji u url-u ima dvije GET varijable koje predstavljaju e-mail i jedinstveni kod. Jedinstveni kod se kreira pomoću md5 funkcije koja kao parametar prima trenutno vrijeme u mikrosekundama izraženo u UNIX vremenskoj oznaci. Klikom na aktivacijski link, korisnik se preusmjerava na 'activate.php' datoteku koja poziva metodu *activate* klase *Email*. Ta metoda izmjenjuje vrijednost polja Confirmed u tablici korisnika s 0 na 1, kako bi se korisnik mogao prijaviti u aplikaciju. Prilikom aktivacije e-maila, u datoteci 'activate.php' uključen je predložak 'activate.phtml' koji omogućuje povratak na početnu stranicu. Ukoliko je e-mail već potvrđen, 'activate.php' datoteka se neće učitati te će se pojaviti *Object Not Found! Error 404* pogreška.

```
$user = new User();
$email = new Email();
$emailCode = md5(Input::get('name').microtime());
$validate->setEmailData(Input::get('email'), $emailCode, Input::get('name'));
try{
    $user->create(array(
            Input::get('name'),
            Input::get('oib'),
            Input::get('email'),
            Hash::make(Input::get('password')),
            $emailCode
       ));
} catch (InsertException $ex) {
    $validate->addError($ex->getMessage());
}
```

Sl. 3.8. Unos korisnika u bazu preko metode *create* klase *User*

3.3. Korisnički režim rada

Nakon uspješne prijave u aplikaciju, ukoliko korisnik aplikacije u tablici korisnika ima polje RoleID postavljeno na 1, on će se prijaviti u korisnički režim rada. Polje RoleID je strani ključ u tablici korisnika koji pokazuje na tablicu prava pristupa. Postoje dva prava pristupa, korisnički i administratorski. Prava pristupa su zabilježena u obliku JSON niza u kojemu postoje dva elementa, admin i user. Kada je element postavljen na 1, tada korisnik ima pravo pristupa u tom režimu rada. Korisnički režim rada se sastoji od tri sekcije: uputa za naručivanje usluga, obrasca za naručivanje usluga te arhive svih prošlih usluga prijavljenog korisnika. Na ovoj stranici, tj. režimu rada, korišten je projekt otvorenog koda za paginaciju tablice arhive koja će biti objašnjena kasnije u ovom poglavlju.



Sl. 3.9. Tablica prava pristupa

Na početku 'user.php' datoteke uključene su 'isLoggedIn.php' i 'hasPermission.php' datoteke. U obje datoteke je kreiran objekt klase *User*. U konstruktoru klase *User* provjerava se je li postavljena sesija čija je vrijednost e-mail korisnika, te ukoliko jest, poziva se metoda *find* koja traži korisnika s tom e-mail adresom [17]. Ukoliko je pronađen korisnik s tom e-mail adresom, varijabla instance *isLoggedIn* se postavlja na true. Ta varijabla služi kako bi se na svakoj stranici aplikacije provjerilo je li korisnik prijavljen. Ako nije, aplikacija preusmjerava korisnika na početnu stranicu. U 'hasPermission.php' datoteci poziva se *hasPermission* metoda klase *User*. Ta metoda provjerava ima li prijavljena osoba prava u određenom režimu rada aplikacije. Metoda funkcionira na način da najprije pronađe prava pristupa prijavljene osobe koji je pronađen metodom *find*. Nakon toga, Permissions stupac dobiven upitom se pomoću json_decode funkcije pretvori iz JSON niza u PHP niz kako bi se moglo pristupati pojedinačnim elementima niza. Na kraju, provjerava se je li user element

niza postavljen na 1, te ukoliko jest, prijavljena osoba ima pravo pristupa u korisničkom režimu rada.

```
public function hasPermission($key){
    $role = $this->_db->query
            ("SELECT * FROM Role WHERE RoleID=?",
            array($this->
                    getResultSet()->
                    getRoleID()));

    $permissions = json_decode($role->
            getFirstResult()->
            getPermissions(), true);

    if($permissions[$key] == 1)
        return true;

    return false;
}
```

Sl. 3.10. Metoda *hasPermission* koja provjerava ima li korisnik pravo na određeni režim rada

Korisnik se odjavljuje klikom na gumb 'Odjavi se' koji poziva skriptu 'logout.php'. Ta skripta kreira objekt tipa *User* i poziva metodu *logout* koja uništava trenutnu sesiju. Nakon toga, poziva statičnu metodu *to* klase *Redirect* koja preusmjerava korisnika na početnu stranicu.

```
$user = new User();
$user->logout();

Redirect::to('../index.php');
```

Sl. 3.11. Skripta 'logout.php'

3.3.1. Naručivanje korisnika

Naručivanje korisnika za usluge koje pruža pružatelj usluga se sastoji od odabira usluge, datuma i vremena te trajanja same usluge. Naručivanje se odvija sekvencijalno, odnosno korisnik mora odabirati navedene stavke tim redosljedom budući da je svaka stavka onemogućena za odabir dok prethodna stavka nije odabrana. Polje za odabir usluge se sastoji

od stavki koje su povučene iz tablice usluga nazvane 'Activity' iz baze podataka. Povlačenje se odvija pri osvježavanju stranice putem AJAX-a. Nakon odabira usluge, polje za odabir datuma se omogućuje za odabir. Klikom na polje za odabir datuma, pojavljuje se JQuery kalendar. Kalendar je podešen tako da su svi datumi, osim raspona između dva i trideset dana od današnjeg dana, onemogućeni. U 'activityPicker.js' skripti nalazi se JQuery funkcija koja se pokreće na odabir usluge. Unutar te funkcije nalazi se AJAX poziv koji šalje podatke o ID-u odabrane usluge i trajanju radnog vremena PHP skripti 'getDisabledDates.php'.

```
$('.datePicker').datepicker({
    altField  : '.actualDate',
    altFormat : 'yy-mm-dd',
    minDate: 2,
    maxDate: 30,
    beforeShowDay: function(date){
        var string = jQuery.datepicker.formatDate('yy-mm-dd', date);
        return [ disabledDates.indexOf(string) == -1 ]
    },
```

Sl. 3.12. Inicijalizacija JQuery kalendara

PHP skripta 'getDisabledDates.php' vraća sve datume na koje ne postoji slobodan termin za odabranu uslugu. Trajanje radnog vremena dobije se oduzimanjem sati zadnjeg i početnog sata radnog vremena. Zadnji i početni sat radnog vremena su pohranjeni u tablici 'Options' u bazi podataka. U PHP skripti 'getDisabledDates.php' datumi na koje određena usluga nema slobodnih termina dobiju se tako da se pomoću GROUP BY naredbe u SQL-u grupiraju retci po datumu, a pomoću naredbe HAVING filtriraju samo oni datumi na čiji dan je zbroj trajanja svih usluga jednak trajanju radnog vremena. Usluge čija se trajanja zbrajanju moraju biti pregledane i potvrđene. Datumi koji se dobiju upitom na bazu, spremaju se unutar for petlje u PHP niz koji se kasnije pretvori u JSON niz te pošalje nazad na stranu klijenta u 'activityPicker.js' skriptu gdje se taj niz spremi u *disabledDates* varijablu. Ta varijabla se potom koristi u *beforeShow* funkciji JQuery kalendara koja određene datume onemogućava za odabir.

```
$result = Database::getInstance()->query(
        "SELECT * FROM Book "
        . "WHERE ActivityID=? AND "
        . "Validated=? AND Valid=? "
        . "GROUP BY Date "
        . "HAVING CAST(SUM(TimeEnd - TimeBegin) as TIME)=?",
        array($selectedID, 1, 1, $workingDuration))->getResultSet();
```

Sl. 3.13. Upit na bazu koji vraća zauzete datume

Nakon što je odabran datum, gumb za odabir termina je omogućen. Klikom na gumb pojavljuje se lista termina koji su slobodni na odabrani datum za odabranu uslugu. Lista termina je definirana na temelju radnog vremena, gdje početak radnog vremena predstavlja prvi termin, a kraj radnog vremena posljednji termin u listi. Razlika između pojedinačnih termina u listi je pola sata. Početno i krajnje vrijeme radnog vremena se povlače iz skripte 'getWorkingTime.php' preko AJAX poziva koji se pokreće pri osvježavanju stranice. Dobivenim vremenima se oduzimaju minute i sekunde pomoću *getHours* metode. Završnom satu radnog vremena oduzeto je pola sata budući da je razlika između pojedinih termina u listi pola sata.

```
success: function(data) {
  startTime = data.startTime;
  endTime = data.endTime;
  var hour = getHour(endTime) - 1;
  lastTime = hour + ':30';
```

Sl. 3.14. AJAX *success* funkcija za dohvaćanje radnog vremena

U *onSelect* funkciji JQuery kalendara nalazi se AJAX poziv koji povlači sve zauzete termine tako da šalje odabrani datum i odabranu uslugu PHP skripti 'getDisabledTimes.php', a ona mu vraća JSON multidimenzionalni niz koji se sastoji od dva reda. Prvi red predstavlja početke, a drugi red završetke zauzetih termina. Takav niz se šalje natrag u 'dateTimePicker.js' skriptu gdje se formatira u niz termina gdje je svaki termin predstavlja par njegovog početnog i završnog vremena u obliku sati. Na temelju tog niza, onemogućuju se pojedini rasponi termina u listi pomoću opcije *disableTimeRanges*.

```
success: function(array) {
    pair = [[]];
    var disabledTimeBegin = array.timeBegin;
    var disabledTimeEnd = array.timeEnd;

    for(var i = 0;i < disabledTimeBegin.length; i++){
        pair[i] = [disabledTimeBegin[i], disabledTimeEnd[i]];
    }

    $('.timePicker').timepicker( 'option', {
        disableTimeRanges: pair
    });
```

Sl. 3.15. Onemogućavanje zauzetih termina u JQuery-u

Nakon odabira termina, polje za odabir trajanja usluge je omogućeno. Termin usluge se ne smije preklapati sa već potvrđenim terminima za istu uslugu na isti datum, pa su takva trajanja onemogućena. Prilikom odabira termina usluge, AJAX poziv šalje podatke o datumu, ID-u usluge te vremenu početka termina usluge PHP skripti 'getDisabledDuration.php'. Skripta sadrži upit na bazu koji se nalazi u *foreach* petlji koja prolazi kroz sve elemente niza *duration* koja sadrži trajanja usluga. Upit se izvršava svaki put posebno za svako pojedino trajanje. Trajanja su izražena u obliku minuta. Upit dodaje minute trajanja svakom terminu u listi na određeni datum za određenu uslugu. Ukoliko se termin preklapa sa već potvrđenim terminom ili prelazi završetak radnog vremena nakon dodavanja određenih minuta trajanja, to trajanje se dodaje u PHP niz. Nakon što *foreach* petlja prođe kroz sva trajanja, PHP skripta šalje enkodirani niz u obliku JSON-a nazad u JavaScript skriptu 'dateTimePicker.js'. U AJAX *success* funkciji, onemogućuju se određena trajanja na temelju enkodiranog JSON niza primljenog iz PHP skripte 'getDisabledDuration.php'.

```
$count = Database::getInstance()->query("SELECT *
    FROM book JOIN Options WHERE Date=? AND ActivityID=?
    AND Valid=? AND Validated=? AND
    (TimeBegin BETWEEN ? AND
    DATE_ADD(CAST(? AS TIME), INTERVAL ? MINUTE)
    OR Options.EndTime < DATE_ADD(CAST(? AS TIME), INTERVAL ? MINUTE))",
```

Sl. 3.16. Upit koji vraća zauzete termine

Nakon što je odabrano trajanje usluge, gumb 'Naruči se' je omogućen. Klikom na taj gumb resetira se obrazac za naručivanje te se pokreće AJAX funkcija koja šalje sve podatke iz obrasca PHP skripti 'insertOrder.php'. Ta skripta unosi podatke o narudžbi u tablicu 'Booking'. Stupci 'Validated' i 'Valid' tablice Booking se postavljaju na 0, sve dok administrator ne potvrdi ili ne odbije narudžbu. Nakon što se PHP skripta izvede, iskoči Bootstrap modalni prozor koji obaviještava korisnika da je njegova narudžba zaprimljena i da je potrebno pričekati dok administrator ne pregleda narudžbu.

3.3.2. Arhiva korisnika

Arhiva korisnika je tablica koja prikazuje sve usluge prijavljenog korisnika koje je administrator prihvatio i koje su se već dogodile. Sastoji se od pet stupaca koji govore o kojoj se usluzi radi, na koji datum je ona izvršena te u koliko sati je ona počela i završila. Sama tablica je realizirana pomoću tri datoteke. HTML predložak predstavlja datoteka 'archiveUser.phtml' u kojoj se nalazi samo „kostur“ tablice. Predložak se sastoji samo od prazne Bootstrap tablice koja se kasnije dinamički pri osvježavanju stranice puni pomoću AJAX poziva u JQuery skripti 'archiveUser.js'. AJAX povlači podatke iz baze tako što poziva PHP skriptu 'getArchiveUser.php'. U toj skripti, kreiran je upit koji povezuje tablicu svih narudžbi sa tablicom korisnika SQL naredbom JOIN, kako bi se filtrirale narudžbe po korisniku na temelju sesije čija je vrijednost e-mail koji se nalazi u tablici korisnika. Narudžbe su filtrirane po stupcu Validated i Valid jer su potrebne samo one narudžbe koje su prihvaćene od strane administratora, tj. one narudžbe koje navedene stupce imaju postavljene na 1. Također, uzete su samo one narudžbe čiji je datum u prošlosti što je realizirano pomoću uvjeta *Date < NOW()*.

```
$result = Database::getInstance()
        ->query("SELECT * FROM Book "
                . "JOIN User ON Book.UserID=User.UserID "
                . "JOIN Activity "
                . "ON Book.ActivityID = Activity.ActivityID "
                . "WHERE Validated=? AND Valid=? AND Date < NOW() "
                . "AND User.Email=?", array(1, 1, $email))
        ->getResultSet();
```

Sl. 3.17. Upit na bazu koji vraća arhivu korisnika

Određene stupce rezultata upita potrebno je formatirati. Datum u bazi podataka je spremljen u formatu kakav se ne koristi na području Europe, pa se on formatira u prihvatljivi oblik. Isto vrijedi i za vrijeme početka i kraja usluga, kojima se oduzimaju sekunde kako bi format bio prihvatljiviji za ispis u tablicu. Nakon formatiranja, rezultati upita se spremaju multidimenzionalni niz. Postupak formatiranja i spremanja rezultata upita u multidimenzionalni niz odvija se u *foreach* petlji. Nakon što je *foreach* petlja prošla sve iteracije rezultata upita, on se pomoću json_encode naredbe pretvara u JSON niz. Takav niz se šalje natrag na stranu klijenta gdje se njime manipulira u AJAX *success* funkciji kao i u ranijim slučajevima. U *success* funkciji dinamički se generira tablica arhive pomoću JQuery *each* metode. Ta metoda funkcionira poput *foreach* petlje u PHP-u, tj. obilazi sve elemente nekog niza.

```
$.each(array.activityName, function(i, item) {
    $('#archiveUser tbody').append(
            '<tr class=' + i + '>\n\
             <td class=' + i + 'activities></td>\n\
             <td class=' + i + 'date></td>\n\
             <td class=' + i + 'timeBegin></td>\n\
             <td class=' + i + 'timeEnd></td>\n\
             </tr>');
});
```

Sl. 3.18. Dinamičko kreiranje tablice arhive u JQuery-u

U AJAX *success* metodi, najprije se kreiraju svi retci tablice na način prikazan na prethodnoj slici. Svaki redak tablice se razlikuje na temelju klase koja ima vrijednost rednog broja elementa u nizu podataka. Svaki stupac pripada klasi koja u imenu sadrži redni broj elementa u nizu i ime samog stupca. Dinamičko dodavanje sadržaja po stupcima svakog retka odvija se u *each* metodi nakon kreiranja tablice. Na slici u nastavku je prikazano dinamičko dodavanje teksta u svaki redak stupca 'activities'. Na isti način dodavan je tekst za svaki od stupaca tablice arhive.

```
$.each(array.activityName, function(i, item) {
    $('.' + i + 'activities').text(item);
});
```

Sl. 3.19. Dinamičko dodavanje elemenata po stupcima tablice arhive

Tablica ima paginaciju što znači da je podjeljena na stranice koje sadrže određeni broj redaka. Konkretno, u ovom radu je određeno da je limit tablice 10 redaka. To znači da se prvih 10 redaka nalazi na prvoj stranici tablice, zatim sljedećih 10 redaka prelazi na drugu stranicu, pa sljedećih na treću, itd. sve dok postoji redaka u tablici. Brojevi za paginaciju su prikazani samo u slučaju da postoji barem jedan redak u tablici. To je realizirano na temelju varijable *numItem* koja prebrojava stupce multidimenzionalnog niza enkodiranog u JSON poslanog iz PHP skripte 'getArchiveUser.php'. Aktivna stranica je postavljena 0, što znači da će tablica prvo prikazivati retke od 1 do 10. Paginacija je realizirana metodom *paging* pomoću ID selektora tablice na koju se primjenjuje.

```
if(numItem > 0){
    $('#archiveUser').paging({
        limit:10,
        rowDisplayStyle: 'block',
        activePage: 0,
        rows: []
    });
}
```

Sl. 3.20. Paginacija tablice arhive korisnika

3.4. Administratorski režim rada

Ukoliko je validacija prijave u aplikaciju uspješna te ako prijavljena osoba ima polje RoleID u tablici korisnika postavljeno na 2, ta osoba se preusmjerava u administratorski režim rada. Administratorska stranica sastoji se od pet sekcija: liste čekanja, obrasca za postavljanje radnog vremena, obrasca za dodavanje usluga, kalendara svih prihvaćenih usluga i arhive svih usluga koje su prihvaćene te završene.

Sl. 3.21. Tablica liste čekanja

3.4.1. Lista čekanja

Generiranje liste čekanja omogućuju tri datoteke: 'waitingList.phtml', 'waitingList.js' i 'getWaitingList.php'. Datoteka 'waitingList.phtml' služi kao HTML predložak koji se sastoji od Bootstrap tablice koja se dinamički popunjava AJAX pozivom pri osvježavanju stranice. Skripta 'waitingList.js' sadrži taj AJAX poziv u čijoj *success* funkciji se popunjava Bootstrap tablica iz 'waitingList.phtml' datoteke. Ta skripta popunjava podatke na temelju enkodiranog JSON niza poslanog iz 'getWaitingList.php' skripte. JSON niz se dobije popunjavanjem rezultatima upita na bazu kojim se dobiju sve usluge koje nisu pregledane, a tek predstoje, odnosno usluge u budućnosti.

JOIN naredba se koristi kako bi se povezalo više tablica po određenim uvjetima, a to je potrebno kako bi se dobile sve potrebne informacije o uslugama koje su na čekanju, kao što su ime i prezime osoba koje su zatražile usluge, naziv te vrijeme i datum usluga. WHERE naredba služi kako bi se dobiveni rezultati filtrirali po stupcima Validated, Valid i Date. Znak upitnika u upitu je karakterističan za PDO. PDO upit je kreiran i poslan bazi tako da su vrijednosti parametara neodređeni i označeni u obliku upitnika. Nakon toga, baza parsira, kompajlira, optimizira i sprema upit bez njegovog izvođenja. Kasnije, aplikacija pridružuje vrijednosti parametrima i upit se izvršava. Aplikacija može izvršiti upit više puta sa drugačijim vrijednostima parametara. Na taj način smanjuje se vrijeme parsiranja jer se priprema upita izvrši jednom iako se sam upit može više puta izvršiti.

```
$result = Database::getInstance()
->query("SELECT * FROM Book "
        . "JOIN User ON Book.UserID = User.UserID "
        . "JOIN Activity ON Book.ActivityID = Activity.ActivityID "
        . "WHERE Validated=? AND Valid=? AND Date > NOW()",
        array(0, 0))
->getResultSet();
```

Sl. 3.22. Upit kojim se dobiju sve usluge na čekanju

Klasa *ResultSet* služi kao spremnik rezultata upita na bazu. Ona omogućuje da se na objektno orijentirani način pristupa rezultatima upita. Metoda *getResultSet* vraća sve retke upita spremljene u obliku objekata klase ResultSet čije varijable predstavljaju vrijednosti pojedinih stupaca. Svaki objekt predstavlja jedan redak rezultata upita, a pojedini stupci njegove varijable instance. Svaka varijabla instance ima svoj *getter* pomoću kojeg se može

dohvatiti vrijednost pojedinog stupca u određenom retku rezultata upita. Vrlo je važno da sve varijable instance klase ResultSet imaju identična imena kao i stupci tablice koju mapiraju, npr. varijabla *Email* mapira stupac Email, itd.

```
public function getDate() {
    return $this->Date;
}

for($i = 0; $i < sizeof($result); $i++){
    $date = $result[$i]->getDate();
```

Sl. 3.23. Primjer korištenja metode ResultSet klase

Svaki termin u tablici liste čekanja sastoji se od podataka o imenu i prezimenu korisnika koji se naručio, nazivu usluge, datumu te vremenu početka i kraja usluge. Također, svaki termin sadrži simbole za prihvaćanje ili odbijanje termina. Kada se termin prihvati ili odbije, on nestane iz liste čekanja bez potrebe za osvježavanjem stranice što je postignuto *fadeOut* metodom u JQuery-u. Svaki redak tablice ima svoj ID koji se dohvaća klikom na jedan od simbola, te se taj ID prosljeđuje *fadeOut* metodi. Metoda *fadeOut* nalazi se u AJAX *success* funkciji koja se izvršava nakon što PHP skripte, 'accept.php' ili 'deny.php' uspješno završe. Ukoliko se dogodi greška u kodu u tim PHP skriptama, *success* funkcija se nikada neće izvesti.

```
success: function(data) {
    if(data != "overlap"){
        var parent = $('#' + data).parent().parent();
        parent.fadeOut("fast", function() {
            $(this).remove();
        });
    }
```

Sl. 3.24. Dinamičko uklanjanje redaka iz tablice čekanja

Može se dogoditi da su zaprimljene narudžbe za istu uslugu koje se vremenski preklapaju, pa je nemoguće prihvatiti obje narudžbe. Prva odobrena narudžba se stavlja u kalendar, dok klikom na simbol za prihvaćanje druge narudžbe, javlja se poruka da već postoji odobrena narudžba u tom terminu. Ovisno o tome klikne li se na simbol za prihvatiti

ili odbiti termin, pokreću se skripte 'accept.js' ili 'deny.js' koje pozivaju PHP skripte 'accept.php', tj. 'deny.php' te manipuliraju uklanjanjem redaka iz liste čekanja. Skripta 'accept.php' mijenja vrijednost stupca Valid iz 0 u 1 ukoliko se termin ne preklapa sa već potvrđenim terminom. Obje skripte, i 'accept.php' i 'deny.php', mijenjaju vrijednost stupca Validated iz 0 u 1. Ako se termin preklapa sa potvrđenim terminom, bit će automatski odbijen te uklonjen iz liste čekanja pri sljedećem osvježavanju stranice. Kao i kod ranije spomenute arhive korisnika, i kod liste čekanja se koristi paginacija sa limitom od deset redaka po stranici. Također kao i kod arhive korisnika, paginacija se ne prikazuje ukoliko ne postoji niti jedan redak u tablici liste čekanja.

3.4.2. Radno vrijeme

Kod odabira radnog vremena, aplikacija prisiljava administratora da prvo odabere početak radnog vremena. Odabir kraja radnog vremena je onemogućen ukoliko nije odabran početak radnog vremena. Nakon odabira kraja radnog vremena omogućava se gumb za podešavanje i unos radnog vremena u tablicu Options baze podataka. Odabir radnog vremena se sastoji od punih sati, tj. raspon između ponuđenih opcija je točno sat vremena. Sat početka radnog vremena ne smije biti poslije sata kraja radnog vremena jer radni dan ne može biti na prijelazu iz jednog dana u drugi, npr. početak ne može biti u 19:00h, a kraj u 02:00h. Ukoliko administrator pokuša napraviti takav odabir, javlja se modalni prozor s porukom „Početak radnog vremena ne smije biti poslije kraja radnog vremena“. Prilikom klika na gumb za podešavanje radnog vremena, pokreće se AJAX funkcija koja uzima vrijednosti početka i kraja radnog vremena i šalje ih PHP skripti 'setWorkingTime.php'.

```
$(".setWorkingTime").click(function(e){
    e.preventDefault();
    e.stopImmediatePropagation();
    $.ajax({
        type:"POST",
        url: "logic/setWorkingTime.php",
        data:{timePickerStart : $(".timePickerStart").val(),
            timePickerEnd : $(".timePickerEnd").val()},
```

Sl. 3.25. AJAX poziv koji se pokreće klikom na gumb za podešavanje radnog vremena

U skripti 'setWorkingTime.php' kreira se instanca klase *Options* koja sadrži metode za provjeru je li početno radno vrijeme poslije kraja radnog vremena te postoje li aktivne narudžbe izvan radnog vremena koje je administrator pokušao podesiti. Metoda *checkOrders* provjerava postoje li prihvaćene narudžbe u budućnosti koje se nalaze unutar radnog vremena koje je administrator pokušao podesiti. Ukoliko upit na bazu vrati barem jedan redak, znači da postoje narudžbe izvan radnog vremena te radno vrijeme nije moguće podesiti na takve vrijednosti. Metoda *checkTimes* provjerava je li podešeni početak radnog vremena prije kraja radnog vremena, te ukoliko jest, moguće je podesiti radno vrijeme uz prethodno ispunjavanje uvjeta za provjeru preklapanja aktivnih narudžbi.

Sl. 3.26. Obrazac za postavljanje radnog vremena

Ukoliko su obje provjere o ispravnosti radnog vremena prošle, 'setWorkingTime.php' skripta šalje string „ok“ AJAX *success* funkciji koja na temelju tog odziva prikazuje modalni prozor sa porukom o uspješnom postavljanju radnog vremena. Nakon uspješnog postavljanja radnog vremena, obrazac za postavljanje radnog vremena se resetira, tj. onemogućuje se odabir kraja radnog vremena i gumb za postavljanje radnog vremena. Ukoliko validacija nije prošla, PHP skripta šalje string „nerijeseno“ u slučaju da postoje neriješene narudžbe izvan radnog vremena ili „not“ ako je početak radnog vremena poslije kraja radnog vremena. Ovisno o stringu pogreške ispisuje se poruka u modalnom prozoru.

```
public function checkTimes($startTime, $endTime){
    if($endTime <= $startTime)
        return false;

    return true;
}
```

Sl. 3.27. *checkTimes* funkcija za provjeru ispravnosti radnog vremena

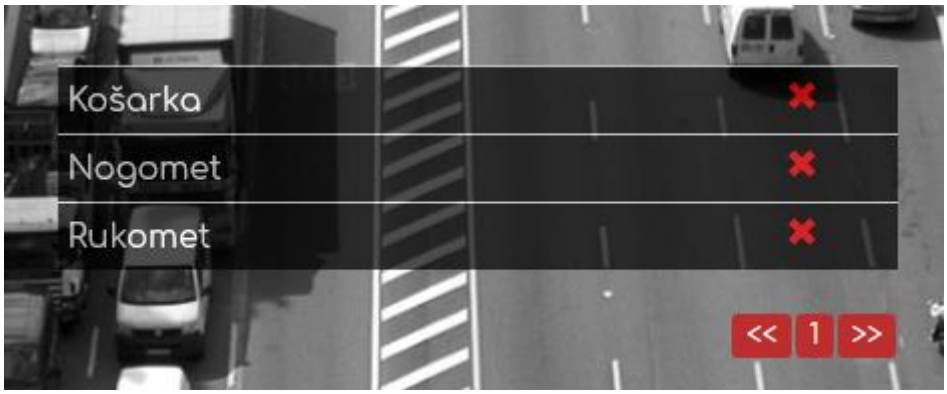
3.4.3. Unos usluga

Sekcija za usluge sastoji se od obrasca za unos usluga i liste svih ponuđenih usluga. Gumb za dodavanje usluge u obrascu je onemogućen sve dok se ne unese barem jedan znak unutar polja za unos usluge. To je postignuto JQuery *keyup* metodom koja se pokreće pritiskom na bilo koju tipku na tipkovnici. Unutar metode potrebno je i ispitati duljinu teksta unesenog u polje za unos usluge. Ukoliko je duljina teksta jednaka nuli, gumb za dodavanje usluge se onemogućuje. Klikom na gumb za unos usluge, poziva se AJAX funkcija koja šalje vrijednost unesenog teksta skripti 'addActivity.php'. U toj PHP skripti instancira se objekt klase *Options* koji poziva metodu *insertActivity* koja unosi uslugu u tablica usluga. Usluga neće biti unesena ukoliko ta usluga već postoji u tablici. Metoda *activityExists* provjerava postoji li unesena usluga u tablici te vraća *true* ukoliko postoji. Ako je unos usluge u tablicu uspješno izvršen, PHP skripta vraća *true* AJAX *success* funkciji koja prikazuje modalni prozor sa porukom o uspješnosti. Nakon što je usluga unesena u bazu, obrazac za unos usluga se resetira.

Sl. 3.28. Obrazac za unos usluga

U listi usluga, moguće je izbrisati uslugu ukoliko ne postoje narudžbe za tu uslugu. U 'denyActivity.js' skripti nalazi se AJAX funkcija koja poziva 'denyActivity.php' skriptu koja provjerava postoje li takve narudžbe za određenu uslugu. AJAX funkcija šalje toj skripti ID usluge. Ako postoje narudžbe za tu uslugu javlja se poruka da nije moguće izbrisati usluge iz liste, već da je potrebno se javiti administratoru baze podataka kako bi se obrisale narudžbe za tu uslugu iz tablice. Nakon toga, administrator može obrisati uslugu iz liste. Razlog zbog kojeg se ne mogu obrisati usluge za koju postoje narudžbe je strani ključ jer ID usluge postoji u tablici narudžbi, tj. tablica narudžbi je povezana sa tablicom usluga preko stranog ključa.

Nakon što je izvršena PHP skripta koja briše uslugu na temelju njezinog ID-a u tablici, u AJAX *success* funkciji poziva se metoda koja dinamički uklanja redak u kojem se nalazi usluga iz tablice usluga. Kao i kod liste čekanja i arhive usluga, i u listi usluga postoji paginacija koja funkcionira kao i u prijašnje navedenim slučajevima.



Sl. 3.29. Lista usluga

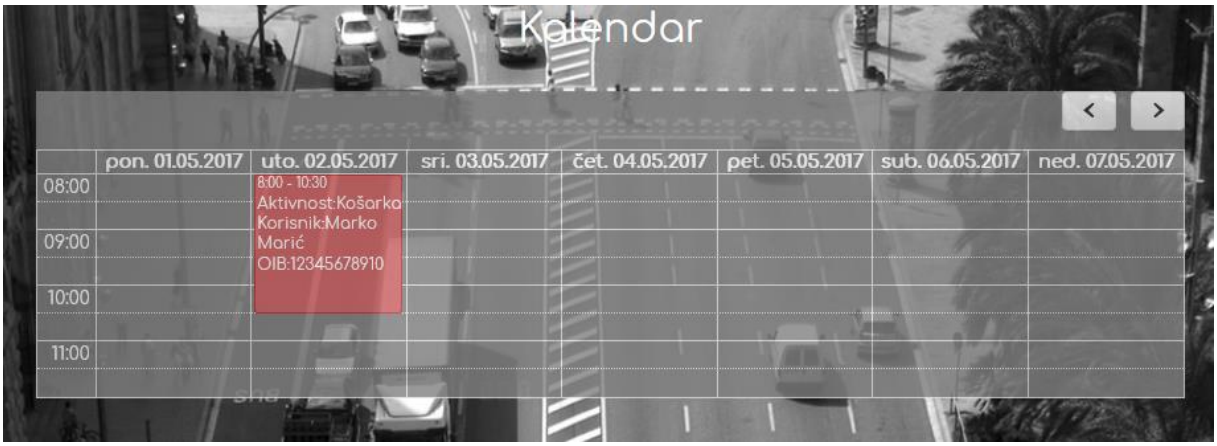
3.4.4. Kalendar

Tjedni kalendar sastoji se od HTML predloška 'weekCalendar.phtml' koji se prilikom učitavanja stranice dinamički popunjava prihvaćenim narudžbama. Skripta 'weekCalendar.js' manipulira tim popunjavanjem te sadrži dvije AJAX funkcije. Prva služi za dohvaćanje radnog vremena koje je potrebno za prikaz raspona sati u tjednom kalendaru. Kalendar je responzivan na promjenu radnog vremena, tj. promjenom radnog vremena, mijenja se i prikazano početno i krajnje vrijeme svakog pojedinog dana. Druga funkcija poziva PHP skriptu 'getAcceptedOrders.php' koja dohvaća sve prihvaćene narudžbe te njima popunjava JSON niz. Naredbom *echo* šaljemo taj niz u AJAX *success* funkciju gdje vrijednosti niza pridružujemo JavaScript varijabli *events*. Varijabla *events* sadrži podatke o narudžbama, kao što su ime i prezime naručitelja, njegov OIB te o kakvoj se usluzi radi.

```
success: function(array) {
  var event = {};
  for(var i = 0; i < array.start.length; i++){
      event = {title: 'Aktivnost:' + array.activity[i] +
               '\nKorisnik:' + array.username[i] + '\nOIB:'
               + array.oib[i], start: array.start[i],
      end: array.end[i]};
      events.push(event);
  }
}
```

Sl. 3.30. Popunjavanje niza events vrijednostima prihvaćenih narudžbi

Funkcija *fullCalendar* renderira kalendar tako da se u obliku JSON niza unesu parametri kalendara te njihove vrijednosti. Neki od parametara su format datuma i vremena, jezik i boja kalendara, koji je prvi dan u tjednu, početno i krajnje vrijeme pojedinog dana, itd. Parametar *events* služi za generiranje događaja u kalendaru u obliku JSON niza. U ovom slučaju, varijabla *events,* koja sadrži JSON niz gdje svaki element predstavlja jednu prihvaćenu narudžbu, pridružena je parametru *events*. Funkcija *eventClick* definira što će se dogoditi prilikom klika na određenu narudžbu generiranu u kalendaru. U ovom slučaju, iskočiti će modalni prozor sa podacima o prihvaćenoj narudžbi. Kalendar generira sve prihvaćene narudžbe, bilo one u prošlosti ili u budućnosti. Klikom na neku od strijelica u zaglavlju, generiraju se tjedni u prošlosti ili u budućnosti. Nema ograničenja što se tiče raspona datuma, a najmanja vremenska jedinica je pola sata, tj. svaki pojedini dan se sastoji od više vremenskih jedinica od pola sata.



Sl. 3.31. Tjedni kalendar

3.4.5. Arhiva usluga

Arhiva usluga funkcionira po sličnom principu kao i arhiva korisnika. Funkcionalnost arhive je također postignuta pomoću Bootstrap tablice koja se dinamički generira AJAX pozivom na PHP skriptu. Isto kao i kod arhive usluga, MySQL upit na bazu spaja tablice User, Activity i Book kako bi se dobili svi potrebni podaci o narudžbama. Jedina razlika između tih dvaju upita je u filtriranju WHERE naredbom. Kod arhive korisnika, uz filtriranje po stupcima Validate i Valid kako bi se dobile samo prihvaćene naredbe te filtriranjem po datumu kako bi se dobile samo narudžbe koje tek predstoje, radi se i filtriranje po emailu kako bi se dobile samo one prihvaćene narudžbe koje se odnose na prijavljenog korisnika. Kod

arhive usluga, takvo filtriranje nije potrebno jer se moraju prikazati sve prihvaćene narudžbe koje su prošle. Kao i kod svih izlistanja sadržaja u obliku tablice u aplikaciji, i ovo izlistanje sadrži paginaciju u istom formatu kao i u prethodno navedenim slučajevima, npr. arhive korisnika, liste čekanja ili liste dostupnih usluga.

Sl. 3.32. Arhiva usluga

4. ZAKLJUČAK

Ova web aplikacija mogla bi biti korisna za olakšavanje poslovanja u različitim malim obrtima, recimo frizerskim ili kozmetičkim salonima, ili kao platforma za iznajmljivanje određenih poslovnih ili sportskih objekata. Aplikacija je prilagodljiva za bilo kakav tip poslovnog subjekta, tj. sam vlasnik odlučuje kakve će usluge biti dostupne za narudžbu preko aplikacije. Isto tako, vrlo je korisno što aplikacija omogućuje postavljanje radnog vremena. Zbog činjenice da je za pristup web aplikacijama potrebna internet veza koja je u današnje vrijeme dostupna gotovo svugdje i običan web preglednik koji se može preuzeti besplatno na internetu, web aplikacije će imati sve veću primjenu u budućnosti.

LITERATURA

[1] J., Duckett, HTML and CSS: Design and Build Websites, John Wiley & Sons, Inc., Indianapolis, 2011.

[2] HTML, https://developer.mozilla.org/en-US/docs/Web/HTML, 19.04.2017.

[3] Introduction to HTML, https://www.w3schools.com/html/html_intro.asp, 19.04.2017.

[4] Cascading Style Sheets, https://www.w3.org/Style/CSS/Overview.en.html, 20.04.2017.

[5] CSS, https://developer.mozilla.org/en-US/docs/Web/CSS, 20.04.2017.

[6] CSS Tutorial, https://www.w3schools.com/css/, 20.04.2017.

[7] Bootstrap - The world's most popular mobile first and responsive front-end framework, http://getbootstrap.com/, 22.04.2017.

[8] Bootstrap 3 Tutorial, https://www.w3schools.com/bootstrap/, 22.04.2017.

[9] Introduction – Javascript, https://developer.mozilla.org/en-US/docs/Web/JavaScript/Guide/Introduction, 23.04.2017.

[10] JavaScript Tutorial, https://www.w3schools.com/js/, 23.04.2017.

[11] M., Haverbeke, Eloquent JavaScript: A Modern Introduction to Programming, William Pollock, San Francisco, 2011.

[12] J., Ducket, JavaScript and JQuery: Interactive Front-End Web Development, John Wiley & Sons, Inc., Indiana, 2014.

[13] PHP: What is PHP?, http://php.net/manual/en/intro-whatis.php, 26.04.2017.

[14] B., McLaughlin, PHP & MySQL: The Missing Manual, O'Reilly Media, Inc., Sebastopol, 2012.

[15] M.E. Davis, J.A. Phillips, Learning PHP and MySQL, O'Reilly Media, Inc., Sebastopol, 2007.

[16] JQuery Tutorial, https://www.w3schools.com/jquery/, 25.04.2017.

[17] PHP 5 Sessions, https://www.w3schools.com/php/php_sessions.asp, 26.04.2017.

[18] Primary and Foreign Key Constraints, https://docs.microsoft.com/en-us/sql/relational-databases/tables/primary-and-foreign-key-constraints, 28.04.2017.

[19] PHP: spl_autoload_register – Manual, http://php.net/manual/en/function.spl-autoload-register.php, 28.04.2017.

SAŽETAK

Cilj aplikacije je da pojednostavi naručivanje korisnika za usluge koje pružaju različiti poslovni subjekti. Aplikacija je podjeljena na dva režima rada, administratorskog i korisničkog. Korisnici se mogu prijaviti za usluge na odabrani datum i vrijeme te biraju vremensko trajanje usluge.

Nakon toga, administratoru su vidljive sve prijave te ih on može odobriti ili odbiti. Ukoliko ih odobri, email potvrde će biti poslan korisniku koji se naručio za tu uslugu te će događaj biti generiran u kalendaru vidljivom samo administratoru. Nadalje, administrator može dodavati usluge te mijenjati raspon radnog vremena. Svaki režim rada sadrži i arhiv. U korisničkom režimu, vidljive su sve prošle usluge prijavljenog korisnika koje su prihvaćene. U administratorskom režimu, administratoru su vidljive sve prošle prihvaćene usluge za sve registrirane korisnike.

Ključne riječi: lista čekanja, kalendar, naručivanje korisnika, vremensko raspoređivanje poslova, web aplikacija

ABSTRACT

Web application for user booking and job time scheduling

The main task of this application is to simplify user booking for services which are provided by various bussinesses. Application is divided into two work regimes: administration and user. Users can apply for services on selected date and time and they can choose duration of the service.

After that, all bookings are visible to administrator and he can approve or deny them. Confirmation email is sent to user if booking is approved by administrator. Also, event is generated in calendar which is visible only to administrator. Furthermore, administrator can add more services to application or change working time. Each work regime contains archive. In user regime, all previous and approved bookings from logged in user are visible to him. In administration regime, all previous bookings from all registered users are visible to administrator.

Keywords: calendar, job time scheduling, user booking, waiting list, web application

ŽIVOTOPIS

Šimun Gogić rođen je 27. Veljače 1993. godine u Osijeku. Završio je osnovnu školu „Retfala“ u Osijeku s odličnim uspjehom. Nakon osnovne škole, upisao je I. Gimnaziju u Osijeku koju je završio s vrlo dobrim uspjehom.

Godine 2011. upisao je preddiplomski studij računarstva na Elektrotehničkom fakultetu u Osijeku koji završava 2014. godine. Iste godine upisuje diplomski studij procesnog računarstva na istom fakultetu.

PRILOZI

Na CD-u priloženom uz diplomski rad nalaze se:

Dokumenti:

- Web aplikacija za naručivanje i vremensko raspoređivanje poslova.docx
- Web aplikacija za naručivanje i vremensko raspoređivanje poslova.pdf

Datoteke:

- Izvorni kodovi aplikacije