FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA

Diplomski sveučilišni studij

INTERNET APLIKACIJA ZA ORGANIZACIJU TIMSKOG RADA

Diplomski rad

Luka Stošić

Osijek, 2018.

Sadržaj

1. UVOD

Cilj ovog diplomskog rada je napraviti „Internet aplikaciju za organizaciju timskog rada“. Ideja za odabirom teme proizlazi iz toga da su u današnjem vremenu gotovo sva zanimanja i poslovi bazirani na rad u timovima. U svrhu poboljšanja i napretka organizacije timskog rada razvijaju se različite strategije, a sve popularniji alat za organizaciju timskog rada su internet aplikacije namijenjene za istu. Od velike važnosti za timski rad je pregledna podjela poslova, preciznost, brza komunikacija, jednostavnost komuniciranja i informiranja suradnika. Za razvoj internet aplikacije fokus je stavljen na jednostavnost korištenja aplikacije, preciznost u postavljanju novih zadataka članovima projekta, označavanje stupnja izvršenosti pojedinih projekata, komentiranje zadataka te praćenje radnog vremena. Na temelju sličnih aplikacija koje također služe za organizaciju timskog rada razmotrene su prednosti i nedostaci u svrhu kreiranja funkcionalne i timovima prihvatljive aplikacije za organizaciju timskog rada. Za programiranje i dizajn internet aplikacije potrebno je koristiti Laravel, Javascript, HTML, CSS, Bootstrap. Također potrebno je kreirati bazu podataka u MySQL-u i odrediti relacije, odnosno povezati tablice iz baze. Za testiranje internet aplikacije na vlastitom računalu korištena je multi-platforma XAMPP. Prije početka rada na aplikaciji bilo je potrebno proširiti znanje iz navedenih programskih jezika i proučiti slične internet aplikacije (Jira, Smartsheet, Asana i td.).

1.1. Zadatak diplomskog rada

U ovom diplomskom radu je potrebno napraviti internet aplikaciju za organizaciju timskog rada. Neke od mogućnosti ovakve aplikacije su: dodavanje zadataka, komentiranje, praćenje radnog vremena, ugrađene e-mail poruke, dopisivanje, projekt management i vremensko praćenje rada. Aplikaciju je potrebno napraviti pomoću Laravel sučelja. Potrebno je navesti mogućnosti postojećih aplikacija kao što su: timr, Slack, Smartsheet, Asana i td.

2.PRIMIJENJENE TEHNOLOGIJE I ALATI

2.1. Laravel

Laravel je open-source PHP okvir (engl.Framework) koji služi za izradu MVC (Model View Controller) internet aplikacija. PHP okvir je skup gotovih modula iz programskog jezika PHP koja zajedno predstavljaju platformu za razvoj aplikacija [1]. Skup gotovih modula iz PHP-a omogućava programeru korištenje istih u različitim projektima te nema potrebe za razvojem koda iz temelja za svaki projekt što uvelike ubrzava rad. Uz ubrzani razvoj aplikacija, PHP okvir osigurava izgradnju stabilnijih aplikacija i uvelike smanjuje količinu kodiranja. Laravel je PHP okvirrazvijen 2011. godine te je ubrzo postao i najkorišteniji PHP okvir čiji se cijeli izvorni kod nalazi na GitHub-u [1]. Kako bi bilo moguće koristiti Laravel na osobnom računalu potrebno je instalirati Composer koji omogućuje upravljanje komponentama od kojih se sastoji Laravel. Odlike Laravela su čist i pregledan kod, jednostavnost, detaljna podrška dokumentacije i dosljednost MVC arhitekturi. Model View Controller arhitektura koristi se za odvajanje pojedinih dijelova koda u komponente ovisno o njihovoj funkcionalnost, odnosno namjeni. Model predstavlja podatke i poslovnu logiku aplikacije te informira poglede(View), odnosno upravitelje (Controller) kada je došlo do promjene u njegovom stanju. Pogled služi za prikaz podataka, a informacije potrebne za prikaz dolaze od modela. Upravitelj (Controller) upravlja korisničkim zahtjevima i sadrži logiku kojom ažurira stanje modela. Također je povezan s pogledom tako da prima unose korisnika i odrađuje logiku te ažurira stanje modela.Za izradu diplomskog rada korištena je Laravel 5.6 inačica.

2.2. JavaScript

JavaScript je najpopularniji besplatni skriptni jezik kojeg podržavaju gotovo svi preglednici (Microsoft Internet Explorer, Opera, Firefox, Google Chrome, Netscape Navigator, Safari i td.) [2]. Koristi se za dodavanje interaktivnog sadržaja HTML stranicama. Skriptni jezici se koriste jer omogućavaju znatno jednostavniji razvoj programa. Može se pisati u bilo kojem uređivaču teksta koji podržava ASCII standard. Jednostavno se dodaje u HTML stranice, a ključna oznaka je <script>. Preporučeno definiranje funkcija je u zaglavlju stranice, iako to nije nužan uvjet, a mogu biti definirane i u tijelu stranice [2].

2.3. HTML

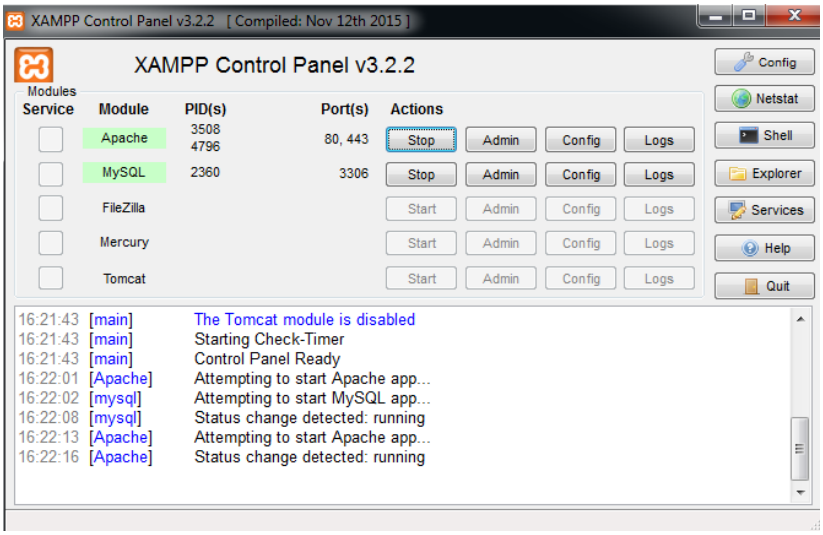
HTML je kratica za HyperText Markup Language, znači prezentacijski jezik za izradu web stranica. To je besplatan jezik koji zbog velike jednostavnosti uživa i veliku popularnost u izradi internetskih aplikacija. Ne spada u skupinu programskih jezika jer služi isključivo za opisivanje hipertekstualnih dokumenata [3]. HTML dokumenti za prikaz koriste internetske preglednike. HTML dokumenti završavaju ekstenzijom .html i sastoje se od elemenata koji su podijeljeni na oznake (tags). Svaka HTML oznaka počinje <tag>, a završava </tag> (npr. tijelo, odnosno sadržaj stranice se nalazi između oznaka <body> i </body>).

2.4. CSS

CSS je kratica za Cascading Style Sheets. Radi se o jeziku koji služi za uređivanje izgleda dokumenta napisanog pomoću opisnog jezika HTML [4]. Pisanje CSS-a je jednostavno te može biti izvedeno unutar HTML dokumenta ili u posebnom dokumentu koji će završavati ekstenzijom .css.

2.5. XAMPP

XAMPP je multi-platforma kojom se instaliraju MySQL, Apache, PHP, phpMyAdmin i ostalo potrebno za razvoj, odnosno testiranje dinamičkih internetskih stranica na osobnom računalu [5].



Sl. 2.1. – Kontrolna ploča razvojnog okruženja XAMPP

2.6. PHP

PHP (Hypertext Preprocessors) je programski jezik namijenjen programiranju dinamičnih internetskih stranica. Kod PHP-a u odnosu na klijentske skriptne jezike npr. JavaScript razlika je u tome što se izvodi na poslužiteljskoj strani, a klijentu se šalje samo rezultat [6]. Za izradu diplomskog rada koristi se Laravel (PHP okvir), ali su potrebna znanja iz PHP-a.

2.7. MySQL

MySQL je open-source sustav za upravljanje bazom podataka. MySQL baza je vrlo stabilna i ima dobro dokumentirane module i ekstenzije te podršku od brojnih programskih jezika: PHP, Java, Perl, Python i td [7].

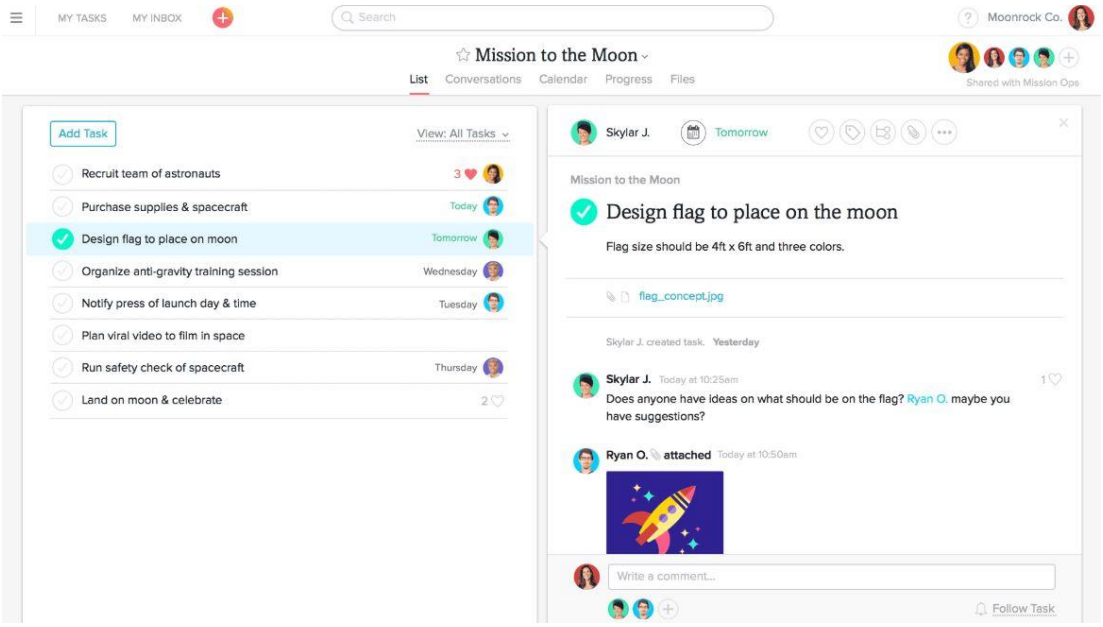
2.8 Bootstrap

Bootstrap je open-source alat za izradu internetskih aplikacija. Sadrži gotove predloške iz HTML-a, CSS-a i JavaScript-a koji ubrzavaju i olakšavaju razvoj internetskih stranica. Bootstrap je usmjeren na front-end i predlošci u Bootstrap-uomogućavaju stranicama responzivan dizajn [8].

2.9. Asana

Jedan od često korištenih alata programske podrške za upravljanje projektima (organizaciju ili raspoređivanje zadataka) jest Asana. Asana je besplatan program koji olakšava organizaciju i koordinaciju poslovnih i privatnih obaveza te olakšava upravljanje vremenom. U usporedbi s ostalim programima za upravljanje projektima, Asana omogućuje praćenje i rad na više projekata u istom trenutku. Uz to, postoji mogućnost kreiranja različitih zadataka (taskova), projekata te razgovora što Asanu čini pogodnom za koordinaciju sastanaka. Ovaj program također nudi još jednu korisnu mogućnost, a to je povezivanje sa servisima kao što su Dropbox, Google Drive, Slack imnogi drugi. Samim time omogućava se učitavanje (upload) dokumenata veličine do 100 MB [9]. Funkcije koje Asana sadrži su owerview – prikaz svih bitnih informacija na jednom mjestu, workspaces – radni prostor za organizaciju podataka i zadataka, notes – bilješke i komentari, e-mails – dijeljenje poruka s korisnicima unutar radnog prostora, contacts – upravljanje kontaktima vezanim za projekt, calendar – služi za bilježenje

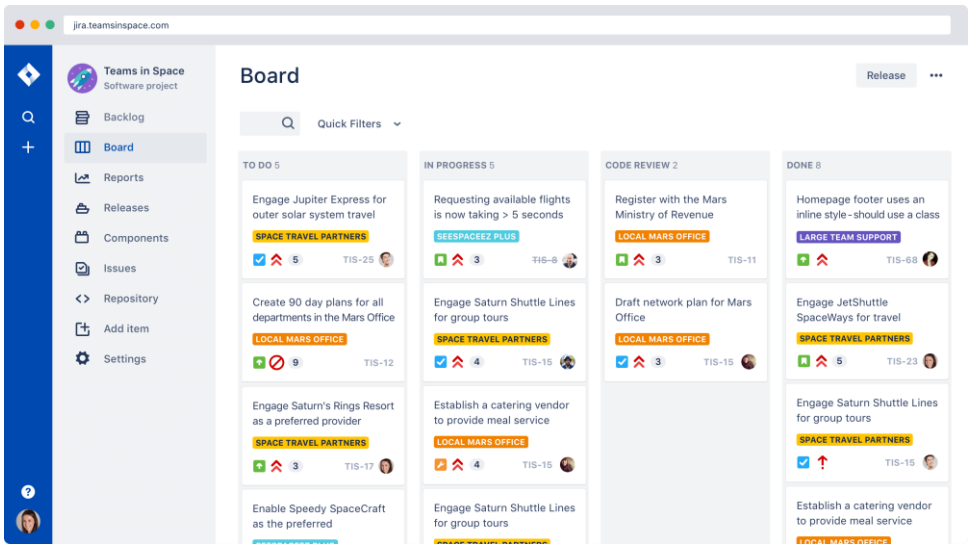
sastanaka, documents – pohrana i dijeljenje svih dokumenata, tasks – praćenje određenih zadataka koji su dio projekta te web links – poveznice na vanjske web stranice [9].



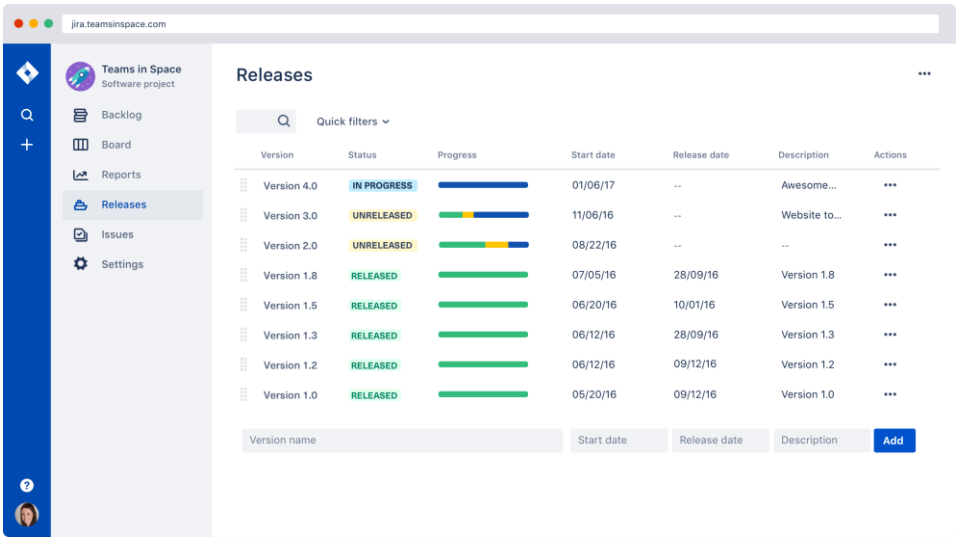
Sl. 2.2. – Asana

2.10. Jira

Jira Project Management Software je alat za praćenje rada na projektima u kojima se rješavaju određeni zadaci u projektnom timu tvrtke Atlassian [10]. To je jedan od najnaprednijih sustava upravljanja projektima kojim je moguće pratiti rad na projektima od početka do kraja. Osigurava da se svaki projekt izvrši u određeno vrijeme uz raspoložive resurse. Najvažnije značajke koje Jira uključuje su praćenje projektnih aktivnosti, bugova, problema, analiza, integraciju koda i mnoge druge. Najčešće se koristi u tvrtkama koje razvijaju programsku podršku za praćenje i popravljanje grešaka u programima. Jira omogućuje predstavljanje trenutnog tijeka rada kao skup zadataka i jednostavno upravljanje, bilo to njima zajedno i posebno[10]. Svaki kreirani zadataka sastoji se od nekoliko osnovnih dijelova. To su naziv projekta kojemu zadatak pripada, vrsta zadatka, prioritet rješavanja zadatka, komponente zadatka, njegov status, sudionici koji trebaju izvršiti zadatak te detaljan opis zadatka. Platforma na kojoj je Jira temeljena jest Java EE. Prednost Jira alata je ta da je moguće dodavati različita proširenja koji olakšavaju rješavanje određenih zadataka. Neka od poznatijih proširenja su ScriptRunner for Jira (služi za automatizaciju), Zephyr for Jira (test management), Tempo Timesheets (praćenje vremena), Capture for Jira (sučelje za vizualnu povratnu informaciju), eazyBI Reports and Charts for Jira Cloud (grafovi i izvještaji) [10].



Sl. 2.3. – Jira Project Managment



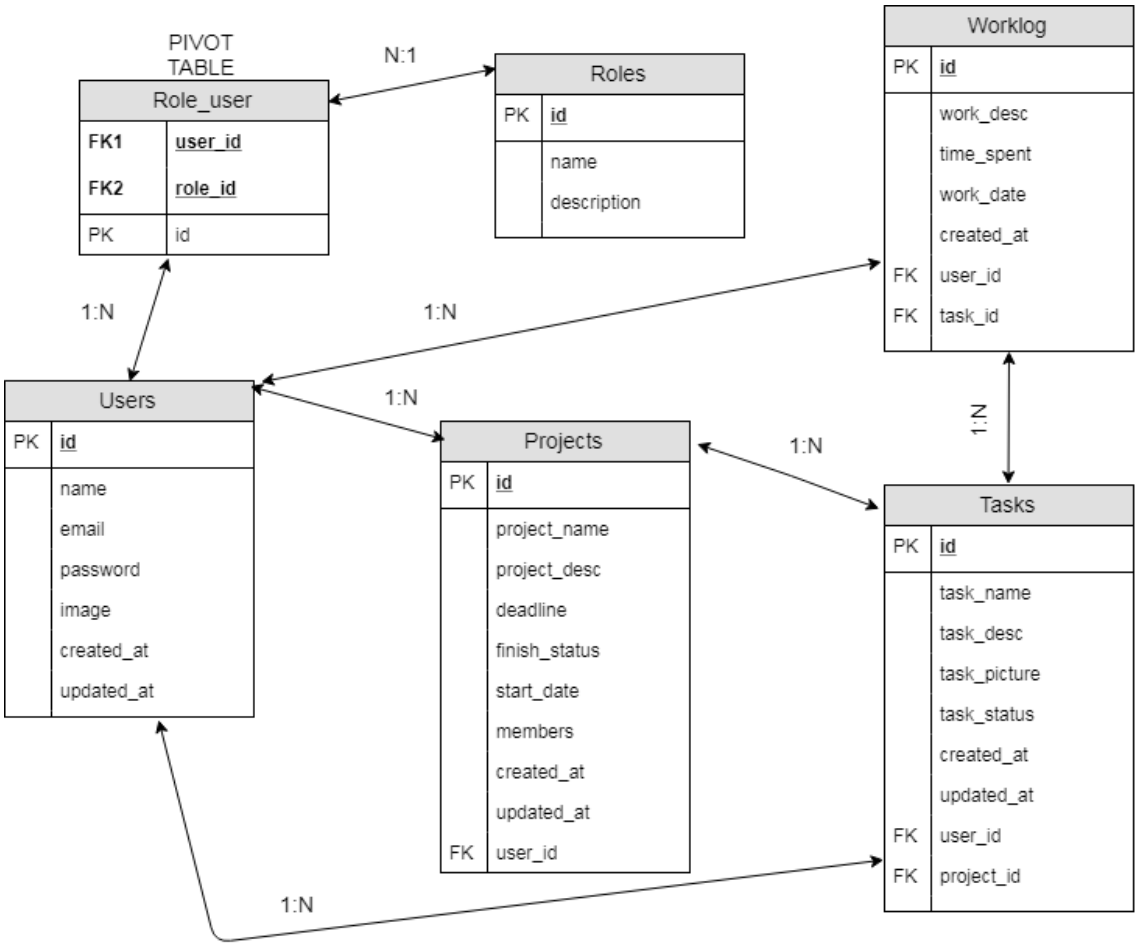
Sl. 2.4. – Jira Project Managment

3. REALIZACIJA SUSTAVA

3.1. Baza podataka

Za izradu internetske aplikacije prvotno je potrebno dizajnirati i kreirati bazu podataka. Baza podataka će omogućiti upravljanje i pristup potrebnim podacima bez stvaranja nepotrebne redundancije. U bazu podataka dodaju se tablice s atributima koje se povezuju i čine skup povezanih tablica odnosno relacije, a stvaraju se ovisno o zahtjevima internetske aplikacije. Relacijske baze podataka sastoje se od tri glavne vrste veza, a to su 1:1 (one-to-one), 1:N (one-to-many), N:M (many-to-many). Kod veza 1:1 i 1:N primarni ključ jedne tablice nalazi se u drugoj tablici kao vanjski ključ te na taj način povezuje tablice. Kod veze N:M potrebno je kreirati treću tablicu poznatu kao pivot tablica, a primarni ključevi iz obje tablice dodaju se u pivot tablicu.

Shema baze podataka internetske aplikacije na temelju koje se kreiraju relacije u modelima Laravel okvira prikazana je na slici 3.1.



Sl. 3.1. – Shema za izradu baze podataka aplikacije

Tablice od kojih se sastoji baza podataka aplikacije su: Users, Projects, Tasks, Roles, Worklogs i pivot tablica Role_user. Users i Roles tablice povezane su 1:N vezom s pivot tablicom Role_user, što znači da više korisnika može imati istu ulogu, odnosno ista uloga može biti dodijeljena različitim korisnicima. Tablica Users prema preostalim tablicama u bazi (Projects, Tasks, Worklogs) ima vezu 1:N što znači da jedan korisnik može kreirati više različitih projekata, zadataka i radnih bilješki, ali s druge strane projekti, zadaci i radne bilješke ne mogu imati više kreatora. U odnosu između Projects i Tasks tablica postavljeno je da jedan projekt može imati više zadataka, ali zadatak može pripadati samo jednom projektu. Također tablice Tasks i Worklogs su u odnosu 1:N i identično se odnose kao i tablice Projects i Tasks.

Migracije u Laravel-u služe kao sheme za jednostavno kreiranje i modificiranje stvarnih baza podataka. Kreiranje migracije izvodi se u komandnom prozoru upisivanjem naredbe make:migration. Jedna migracija služi za kreiranje jedne tablice u bazi podataka, stoga je potrebno napraviti migracije na temelju sheme baze podataka sa slike 3.1.. Upisivanjem naredbe php artisan make:migration create_users_table kreira se migracija u kojoj je potrebno napraviti preinake kako bi izgledala identično prema tablici iz sheme.

```
1.  class CreateUsersTable extends Migration
2.  {
3.      public function up()
4.      {
5.          Schema::create('users', function (Blueprint $table) {
6.              $table->increments('id');
7.              $table->string('name');
8.              $table->string('email')->unique();
9.              $table->string('password');
10.             $table->string('image')->default(„user.jpg“);
11.             $table->rememberToken();
12.             $table->timestamps();
13.         });
14.     }
15.
16.     public function down()
17.     {
18.         Schema::dropIfExists('users');
19.     }
20. }
```

Programski kod 3.1. – Izgled migracije za tablicu Users

Kod iz migracije je prilagođen tablici Users dodavanjem novih redaka. $table->string('name') predstavlja redak u koji će se unositi ime korisnika nakon što se migracija migrira u stvarnu bazu podataka. $table->timestamps() predstavlja datume kada je kreiran korisnik i datum kada

je odrađen update korisnika, odnosno uređivanje korisničkog računa. Koriste se različiti tipovi u izradi baze podataka ovisno o potrebi: int, string, date, text, float i td. U navedenom primjeru za brisanje tablice koristi se Schema::dropIfExists('users'). Kako bi se stvorila stvarna tablica u bazi podataka potrebno je migraciju migrirati naredbom php artisan migrate.

#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra
1	id	int(10)		UNSIGNED	No	None		AUTO_INCREMENT
2	name	varchar(191)	utf8mb4_unicode_ci		No	None		
3	email	varchar(191)	utf8mb4_unicode_ci		No	None		
4	password	varchar(191)	utf8mb4_unicode_ci		No	None		
5	image	varchar(191)	utf8mb4_unicode_ci		No	user.jpg		
6	remember_token	varchar(100)	utf8mb4_unicode_ci		Yes	NULL		
7	created_at	timestamp			Yes	NULL		
8	updated_at	timestamp			Yes	NULL		

Sl. 3.2. – Tablica users u bazi podataka nakon migriranja

Za preostale tablice, migracije se izvode na identičan način.

Sve tablice iz baze podataka imaju odgovarajuće modele koji se koristi za interakciju s tablicama. Model omogućava pretraživanje podataka iz tablica, ali isto tako i zapis novih podataka u tablicu. Stoga je potrebno kreirati i modificirati modele kako bi se uspostavile relacije. Spajanje modela na bazu podataka izvršava se po zadanoj konekciji aplikacije na bazu podataka.

Najjednostavniji način za stvaranje novog modela je unos naredbe php artisan make:model u komandni prozor. Na primjer, za slučaj kreiranja modela User naredba je php artisan make:model User. Shodno modelu User kreirani su i ostali modeli: Project, Role, Task, Worklog.

Nakon kreiranja modela potrebno je odrediti koji parametri imaju svojstvo popunjivosti odnosno označiti ih kao engl. fillable. Svojstvo popunjivosti imaju parametri koje korisnik može mijenjati putem HTTP zahtjeva, a služe kako bi spriječili pogrešnu manipulaciju bazom podataka. Za User model svojstvo popunjivosti imaju name, email, password i image.

```
1.  protected $fillable = [
2.          'name', 'email', 'password', 'image'
3.      ];
```

Programski kod 3.2. – Isječak koda iz modela Users

Sljedeći korak kod uređivanja modela je postavljanje relacija.

Za model User postavljaju se sljedeće relacije shodno shemi za izradu baze podataka.

```
1.  public function tasks()
2.  {
3.  return $this->hasMany('App\Task');
4.  }
5.  public function worklogs()
6.  {
7.  return $this->hasMany('App\Worklog');
8.  }
9.  public function roles()
10. {
11. return $this->belongsToMany(Role::class);
12. }
13. public function projects(){
14. return $this->hasMany('App\Project');
15. }
```

Programski kod 3.3. – Isječak koda iz modela Users za relacije

Relaciju many-to-many predstavlja metoda belongsToMany(), a ista metoda se poziva i u modelu Role. Metoda hasMany dodijeljena je između Usera i modela Project, Worklog i Task jer korisnik može imati više projekata, zadataka i radnih bilješki. Suportno hasMany metodi se poziva belongsTo metoda u modelima Project, Task, i Worklog. Također hasMany metoda se koristi između modela Project – Task i Task – Worklog, što znači da se inverzno postavljaju metode belongsTo.

```
1.  public function user()
2.   {
3.       return $this->belongsTo('App\User');
4.   }
```

Programski kod 3.4. – Isječak koda iz modela Project za relaciju

3.2. Izrada internet aplikacije

Kod izrade internet aplikacija u Laravelu od velike važnosti je pravilno podijeliti dijelove aplikacije u komponente ovisno o njihovoj namjeni. Podjela koda između modela, pogleda i upravitelja omogućava nezavisan razvoj aplikacije i bolju preglednost te u konačnici lakše održavanje aplikacije. U prethodnom poglavlju opisan je postupak kreiranja modela koji služi za interakciju s bazom podataka. Model kao sam nije funkcionalan te je potrebno dodati poglede i upravitelje koji će obavljati svoj dio posla i zajedno s modelima sačinjavati uspješno realiziranu Model-View-Controller arhitekturu na kojoj se temelji Laravel. Pogled (engl. View) koristi se za interakciju korisnika s aplikacijom, a istovremeno odvaja prezentacijsku logiku od aplikacijske koja je sadržana u upravitelju (engl. Controller). Prije izrade upravitelja potrebno je definirati rute (engl. Routes) u routes/web.php datoteci. Definirane rute su poveznica između upravitelja i URL-a, a uz to pružaju mogućnost CSRF zaštite.

```
1. Route::resource('admin','AdminController')->middleware('IsAdmin');
```

Programski kod 3.5. - Primjer definiranja rute za AdminController

Resource metoda za definiranje rute je korištena jer se radi o CRUD (Create, Read, Update, Delete) funkcijama u upravitelju. AdminController pokriva sve korisničke zahtjeve kao što su POST, GET, DELETE, PUT/PATCH i td, a upravo zbog toga je u web.php datoteci definirana ruta kao resource metoda.

Dio koda->middleware('IsAdmin') omogućava samo prijavljenom korisniku koji ima ulogu imena „admin“ pristup URL-u, u suprotnom će korisniku izbaciti poruku „Unauthorized“ ili ga vratiti na prethodnu stranicu. U $UserRoles sprema se ime uloge koju ima korisnik, te se u if($role == 'admin') izvodi provjera podudaranja uloga.

```
1.  $UserRoles = DB::table('roles')-
    >join('role_user','role_id', '=', 'roles.id')-
    >where('user_id', '=', Auth::user()->id)->pluck('name');
2.     $isAdmin = false;
3.     foreach($UserRoles as $role)
4.     {
5.         if($role == 'admin')
6.         {
7.             $isAdmin = true;
8.         }
9.     }
10.    if( ! $isAdmin )
11.    {
12.            return response('Unauthorized.', 401);
```

```
        } else {
            return redirect()->back();
        }
    }
```

Programski kod 3.6. – Međusloj za autorizaciju admina

Laravel sadrži autorizaciju u međusloju (engl. Middleware) koja sprječava korisničke radnje prije nego što one dosegnu rutu ili upravitelj. Zbog toga su autorizacije za određenim rutama odrađene u međusloju.

Nakon postavljenih ruta kreiraju se upravitelji. Postupak kreiranja upravitelja s CRUD metodama izvodi se u komandnom prozoru upisivanjem naredbe php artisan make:controller AdminController –resource. AdminController nakon kreiranja sadrži osnovne funkcije index(), store(), create(), show(), edit(), update() i destroy().Za potrebe internet aplikacije kreirani su upravitelji AdminController, ProjectController, HomeController, UserController, TaskController i WorklogController.

Unutar index() metode definira se prikaz početne stranice za određenu rutu, a najčešće se koristi za dohvaćanje podataka potrebnih za prikaz u pogledu. Dohvaćeni podaci spremaju se u varijable, koje predstavljaju kolekcije, te se iste šalju u pogled metodom compact().Show() metoda identično funkcionira kao i index() metoda, ali se koristi za specificiranje određenog modela po id-u iz baze podataka.

```
public function index()
  {
      $users = User::all();
      return View('admin.index',compact('users'));
  }
```

Programski kod 3.7. – index() metoda u AdminController-u

```
public function show($id)
    {
        $tasks = Task::find($id);
        $projects = Project::find($tasks->project_id;);

        return view('tasks.show',compact('tasks', 'id', 'projects'));
    }
```

Programski kod 3.8. – show() metoda u TaskController-u

U varijablu users spremljeni su podaci svih korisnika koji su registrirani. Podaci spremljeni u varijablu users su proslijeđeni u views/admin/index.blade.php odnosno u pogled index za admin rutu.

Metoda create() koristi se isključivo za poziv pogleda za stvaranje novog modela, a spremanje u bazu podataka izvršava se u metodi store() zbog save() metode koja se poziva za spremanje podataka. Zbog argumenta zahtjev (engl. Request), metoda store() sadrži sve podatke koji su poslani iz forme na zahtjev.

```
1.  public function store(Request $request)
2.     {
3.              $projects= new Project;
4.              $projects->user_id= Auth::user()->id;
5.              $projects->project_name=$request->get('project_name');
6.              $projects->project_desc=$request->get('project_desc');
7.              $projects->start_date = date('Y-m-d');
8.              $projects->deadline_date=$request->get('deadline_date');
9.              $projects->project_leader = Auth::user()->name;
10.             $projects->members = Auth::user()->id;
11.
12.             $projects->save();
13.
14.             return redirect('projects')-
    >with('success', 'Information has been added');
15.        }
```

Programski kod 3.9. – store() metoda u ProjectController-u

Iz koda je vidljivo da metoda store() prima argument zahtjev te ima mogućnost pristupa svim podacima iz forme koju je primila. Da bi se moglo pristupiti, odnosno popunjavati parametre koji u modelu imaju svojstvo popunjivosti, potrebno je napraviti instancu modela Project ($projects = New Project). Ispunjavanje parametara instance modela Project odrađeno je npr. $projects->project_name=$request->get('project_name'), a $request->get('project_name') predstavlja podatak iz forme pod nazivom „project_name“. Kada se pozove save() metoda parametri instance modela se spremaju u bazu podataka.

Edit() i update()su usko povezane metode. U metodi edit() se poziva pogled za specificirani model po id-u jer kao argument prima varijablu id, odnosno identifikator zapisa. Za spremanje podataka modela po id-u u varijablu koristi se metoda find(). Varijabla u kojoj se nalazi kolekcija podataka se metodom compact() proslijedi pogledu gdje se podaci iz kolekcije prikazuju korisniku te ih je moguće mijenjati u formama. Izmjena podataka vrši se u update() metodi koja kao argument prima zahtjev iz forme pogleda za uređivanje.

```
1.  public function edit($id)
2.     {
3.         $tasks = Task::find($id);
4.         return view('tasks.edit',compact('tasks','id'));
5.     }
```

Programski kod 3.10. – edit() metoda u TaskController-u

Zahtjev i id kao argumenti u metodi update() omogućavaju pristup svim podacima iz forme za određeni model. Identično kao u store() metodi se vrši popunjavanje, odnosno izmjena postojećih parametara modela koji imaju svojstvo popunjivosti. U update() metodi za TaskController if petljom provjerava se postoji li zahtjev iz forme 'task_picture' kako ne bi došlo do pogreške prilikom spremanja u Laravel storage direktorij. Ukoliko postoji zahtjev iz forme 'task_picture' isti se sprema u varijablu „image“ te se sprema u direktorij storage/uploads. U varijablu „storagePath“ spremljena je putanja do slike s prefiksom uploads/ te se zbog toga koristi basename() metoda kako bi se prefiks odvojio od imena slike. Pozivom metode save() se spremaju izmjene u bazu podataka.

```
1.  public function update(Request $request, $id)
2.      {
3.          $tasks= Task::find($id);
4.          $tasks->task_name=$request->get('task_name');
5.          $tasks->task_desc=$request->get('task_desc');
6.          $tasks->task_status = $request->input('status');;
7.
8.          if($request->hasFile('task_picture')){
9.              $image = $request->file('task_picture');
10.
11.             $storagePath = Storage::disk('public')->put('uploads', $image);
12.             $storageName = basename($storagePath);
13.             $tasks->task_picture = $storageName;
14.           };
15.
16.         $tasks->save();
17.         return redirect('projects/'. $tasks->project_id);
18.     }
```

Programski kod 3.11. – update() metoda u TaskController-u

Metoda destroy() koristi se za brisanje postojećih modela iz baze podataka. Kao argument metoda prima identifikator zahtjeva te se metodom find() pronalazi instanca za određeni model. Kreiranu instancu modela iz baze podataka se briše metodom delete(). U AdminController-u je primjer metode delete() za brisanje korisnika iz baze podataka.

```
1.  public function destroy($id)
2.    {
3.        $users = User::find($id);
4.        $users->delete();
5.        return redirect('admin')->with('success','User has been  deleted');
6.    }
```

Programski kod 3.12. – destroy() metoda u AdminController-u

U razvoju internet aplikacije glavne metode kod svih upravitelja koriste se identično, ali su djelomično izmijenjene jer je logika unutar njih drugačija ovisno o onome što je potrebno odraditi u pojedinom upravitelju.

Da bi modeli i upravitelji bili funkcionalni potrebno je dodati poglede koji će omogućiti interakciju korisnika s aplikacijom. Upravitelji mogu predati podatke pogledu kroz metodu compact(), ali isto tako se u upravitelju podaci mogu dohvatiti iz formi pogleda kao zahtjev gdje se iste obrađuje. Poglede se sprema u direktoriju resource/views s ekstenzijom .blade.php. U poglede je moguće dodati „layout“ koji se može shvatiti kao pogled koji je moguće naredbama @extends() i @section() uključiti u druge poglede. „Layout“ je obično dio prikaza stranice koji će se ponavljati na više stranica, a koristi se u svrhu izbjegavanja nepotrebnog dupliciranja koda. U izradi internetske aplikacije bočni izbornik je potreban na svim stranicama te se isti kreira kao „layout“.

Primjer koda za pogled koji služi za prikaz korisničkog profila prikazan je u tablici 3.13. U pogledu se mogu koristiti PHP funkcije i petlje dodavanjem @ znaka ispred (npr. @foreach @endforeach). Varijablu „users“ poslanu metodom compact() iz upravitelja koristi se u foreach petlji kako bi se prošlo kroz cijelu kolekciju koja je pohranjena u varijablu i pristupilo pripadajućim projektima. Na primjer, za ispis naziva projekta koji pripadaju prijavljenom korisniku u foreach-u koristi se {{$project->project_name}}, a na identičan naziv bi se pristupilo i ostalim atributima u projektu. Da se radi o prijavljenom korisniku definirano je prethodno u upravitelju prilikom instanciranja modela User po identifikatoru, što se pohranilo u varijablu „users“. Procedura za prikaz atributa za zadatke izvodi se na isti način kao i za projekte.

```
1.  <body>
2.  @extends('layouts.sidebar')
3.
4.  @section('content')
5.
6.  <div class = „row“>
7.
8.    <div class = „col-lg-6“>
9.    @foreach($users->projects->sortByDesc('created_at') as $project)
10.   <a href=“{{action('ProjectController@show', $project['id'])}}“ class=“list-group-
    item list-group-item-action flex-column align-items-start“>
11.     <div class=“d-flex w-100 justify-content-between“>
12.       <h5 class=“mb-1“>{{$project->project_name}}</h5>
13.       <small>{{$project->created_at->diffForHumans()}}</small>
14.     </div>
15.     <p class=“mb-
    1“>{{ str_limit($project['project_desc'], $limit = 100, $end = '...') }}</p>
16.     <small>Deadline: {{$project->deadline_date}}</small>
17.   </a>
18.   @endforeach
19.   </div>
20.
21.   <div class = „col-lg-6“>
22.   @foreach($users->tasks->sortByDesc('created_at') as $task)
23.   <a href=“{{action('TaskController@show', $task['id'])}}“ class=“list-group-
    item list-group-item-action flex-column align-items-start“>
24.     <div class=“d-flex w-100 justify-content-between“>
```

```
25.       <h5 class="mb-1">{{$task->task_name}}</h5>
26.       <small>{{$task->created_at->diffForHumans()}}</small>
27.     </div>
28.     <p class="mb-
    1">{{ str_limit($task['task_desc'], $limit = 100, $end = '...') }}</p>
29.     <small>Task status: @if($task->task_status == 0) In process @elseif($task-
    >task_status == 1) Completed @else Aborted @endif</small>
30.   </a>
31.
32.   @endforeach
33.   </div>
34. </div>
35.
36.   @endsection
37.   </body>
```

Programski kod 3.13. – Dio koda u pogledu za korisnički profil

Izrada ugrađenih e-mail poruka izvodi se uz pomoć „Mailtrap“. Mailtrap oponaša rad stvarnog SMTP (Simple Mail Transfer Protocol) poslužitelja, odnosno uklanja bilo koju mogućnost da testni e-mail dospije u stvarni poštanski sandučić (engl. Mailbox). Slanje e-mail-a korisniku koji je vlasnik zadatka potrebno je odraditi prilikom kreiranja nove radne bilješke. Prilikom izvođenja store() metode u WorklogController-u potrebni podaci za ispuniti formu e-maila spremaju se u niz podataka (engl. Array).

```
1.  $emailUser = User::find($tasks->user_id)->email;
2.
3.  $data =[
4.            'task_name' => $tasks->task_name,
5.            'created_at' => $tasks->created_at->format('d/m/Y'),
6.            'username' => Auth::user()->name
7.         ];
8.
9.  Mail::to($emailUser)->send(new CommentMail($data));
```

Programski kod 3.14. – Dio koda u WorklogController-u za slanje e-mail

Mail::to($emailUser) označava na koju adresu će e-mail biti isporučen, a niz podataka spremljen pod nazivom „data“ pridružuje se klasi CommentMail i na adresu korisnika šalje se poruka. Klasa CommentMail kreira se pisanjem naredbe php artisan make:mail CommentMail u komandnom prozoru i sastoji se od dvije metode __construct() i build(). Metoda __construct() služi za inicijalizacijuobjekta koji se koristi u pogledu za oblikovanje izgleda e-maila. U metodi build() se proslijedi compact() metodom inicijalizirani objekt pogledu i određuju se detalji e-maila.U ovom diplomskom radu potrebno je znati e-mail adresu pošiljatelja i ime korisnika koji šalje.

```
1.  public function __construct($data)
2.     {
3.         $this->data = $data;
4.     }
```

```
5.  public function build()
6.     {
7.         $email = Auth::user()->email;
8.         $name = Auth::user()->name;
9.
10.    return $this->from($email, $name)
11.        ->subject('Worklog info')
12.        ->view('mail.comment')->with(['data' => $this->data]);
13.    }
```

Programski kod 3.15. – Metode __construct() i build() u klasi CommentMail

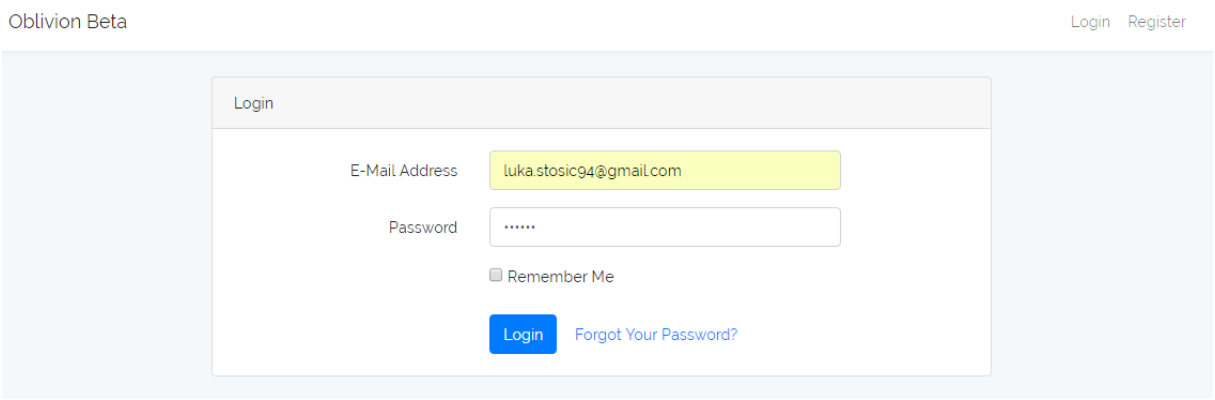
U pogledu se kroz HTML oblikuje izgled e-maila koji će biti isporučen korisniku.

```
1.  <body>
2.      <p>User: {{$data['username']}} added comment to your task: {{$data['task_name']}
    }</p>
3.      <p>Task created at: {{$data['created_at']}}</p>
4.  </body>
```

Programski kod 3.16. – Pogled comment.blade.php

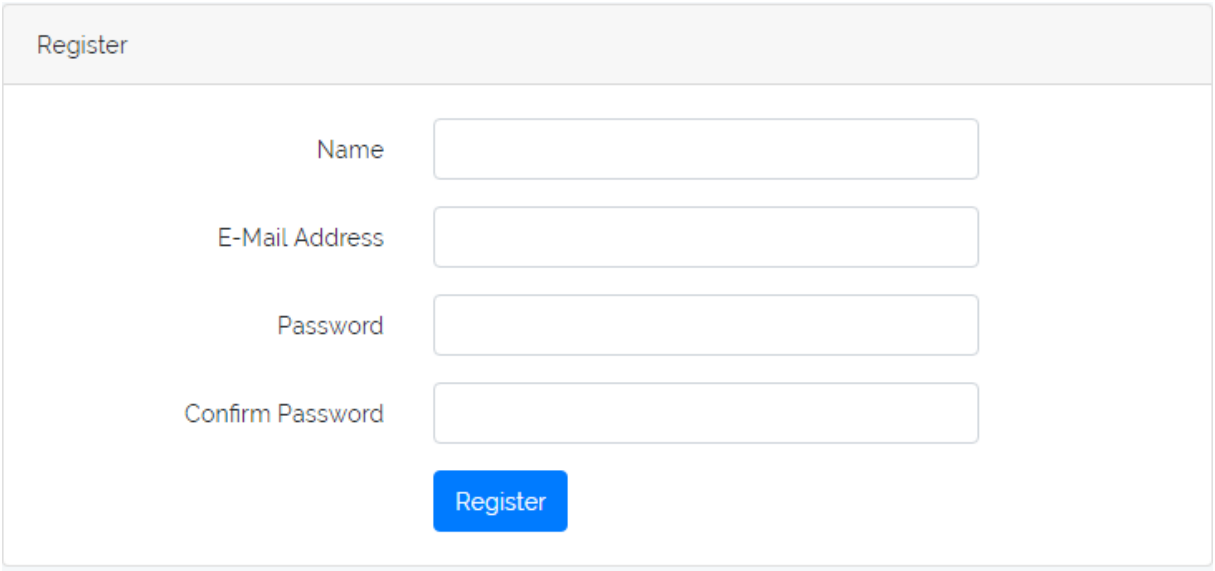
3.3. Opis rada aplikacije

Ulaskom na početnu stranicu aplikacije korisniku se prikazuju polja za prijavu, a za prijavu potrebno je unijeti jedinstvenu e-mail adresu korisnika i lozinku. U slučaju da je riječ o novom korisniku potrebno je obaviti registraciju. Registracija se obavlja klikom na „Register“ koji se nalazi u gornjem desnom uglu. Na slici 3.3. prikazana je početna stranica za prijavu.



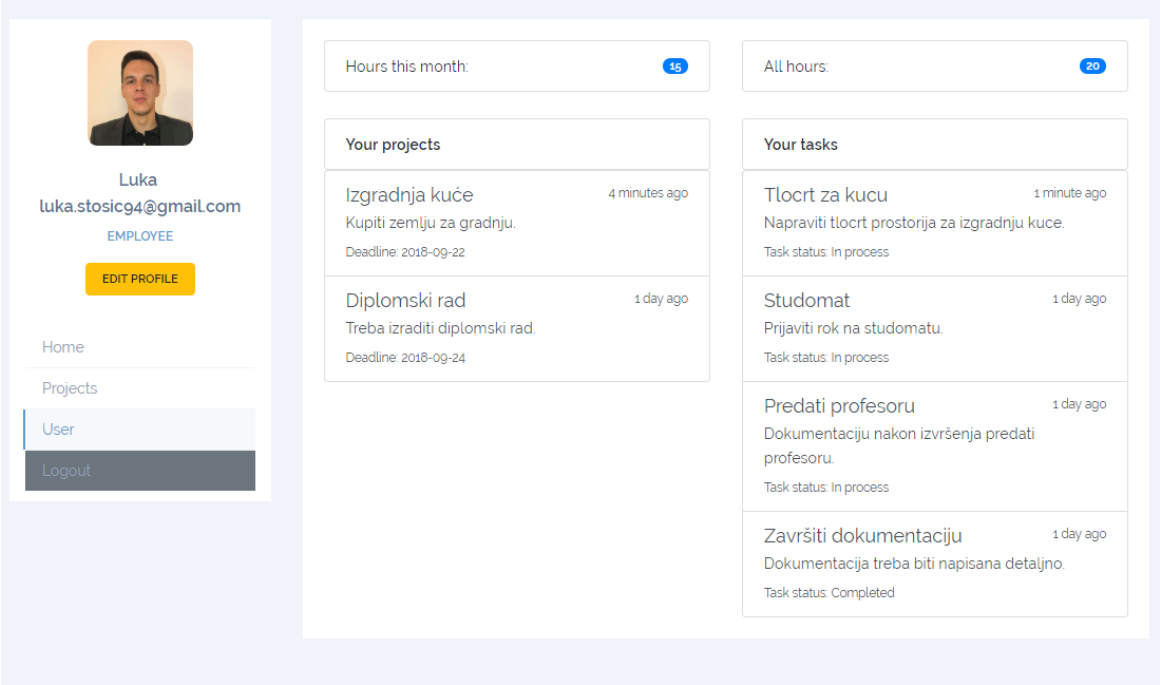
Sl. 3.3. – Stranica za prijavu

Na stranici za registraciju potrebno je unijeti ime korisnika, e-mail korisnika koji mora biti jedinstven, odnosno ne smije se podudarati s već postojećim e-mailovima u bazi podataka, lozinku i potvrditi lozinku. Na slici 3.4. prikazana je stranica za registraciju korisnika.



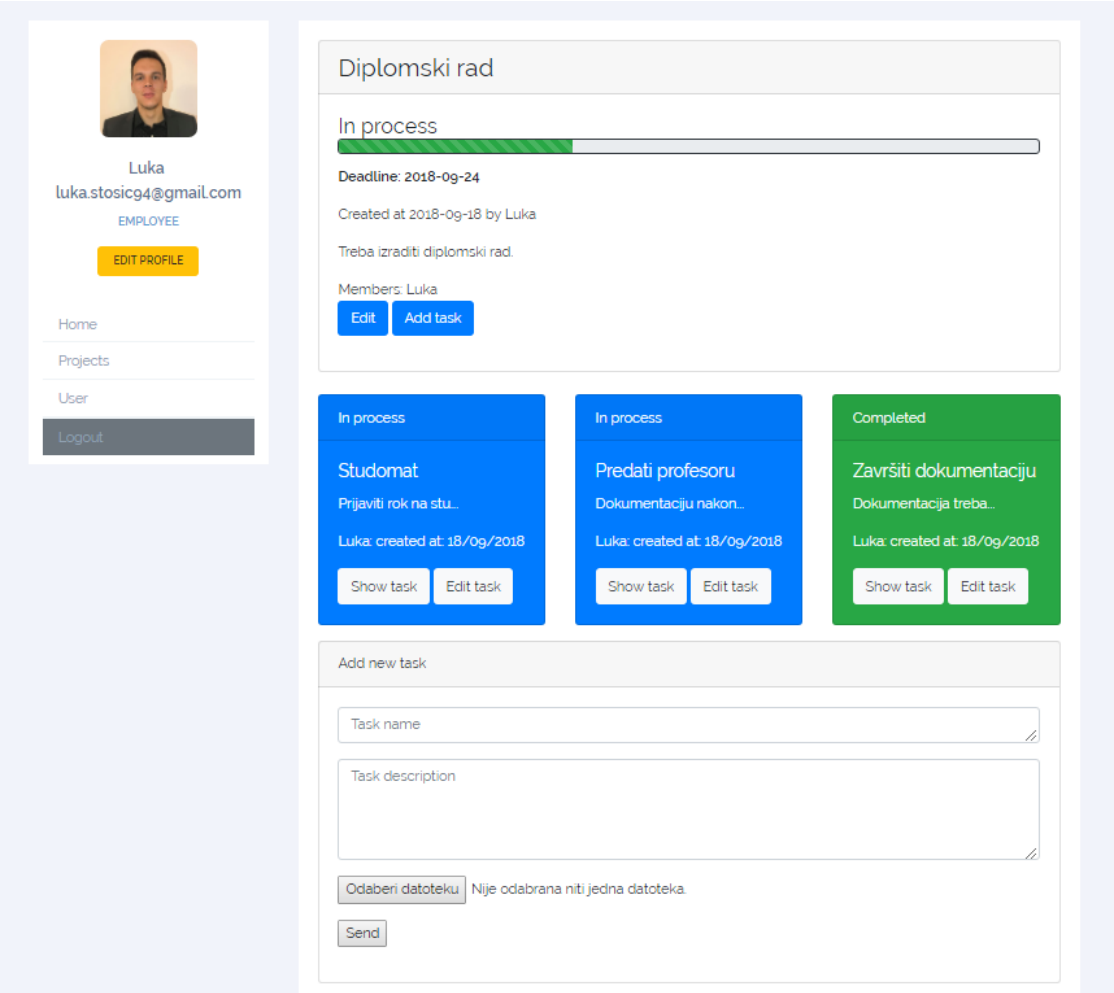
Sl. 3.4. – Stranica za registraciju

Nakon uspješne prijave slijedi preusmjeravanje te prikaz stranice korisničkog profila. Na stranici korisničkog profila s lijeve strane prikazan je bočni izbornik u kojem se nalaze informacije o korisniku i poveznice na druge stranice aplikacije. Bočni izbornik nalazi se na svim stranicama aplikacije. Unutar bočnog izbornika nalazi se korisnikova slika koju je moguće dodati ili promjeniti prilikom čega se slika može izrezati pomoću „cropper-a“ klikom na „Edit profile“. Ta Glavni dio stranice prikazuje broj odrađenih sati u tekućem mjesecu, ukupan broj odrađenih sati od postojanja računa, projekte i zadatke. Korisniku se prikazuju samo projekti i zadaci koje je on sam kreirao.



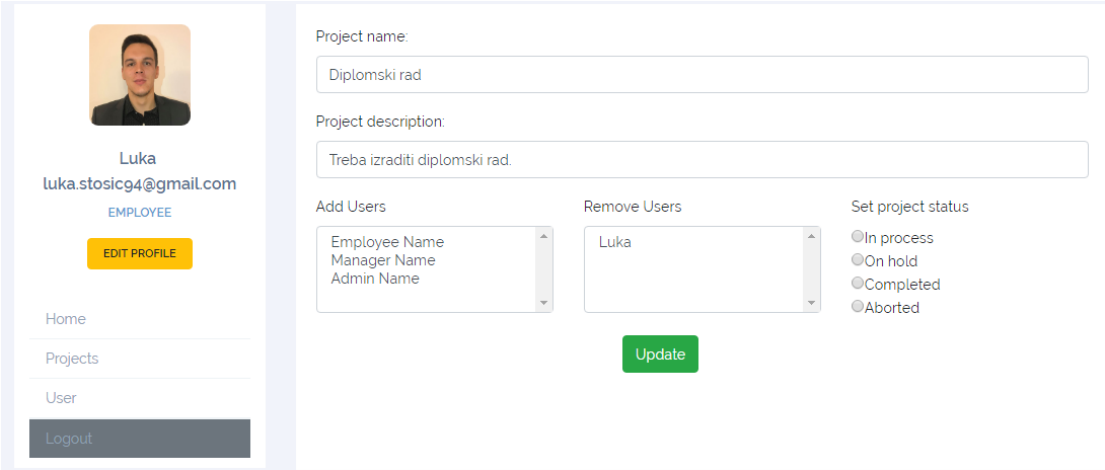
Sl. 3.5. – Stranica korisničkog profila

Odabirom određenog projekta ili zadatka korisnika se preusmjerava na stranicu odabranog projekta gdje su vidljive samo informacije o tom projektu. Na stranici projekta prikazan je naslov, stanje u kojem se nalazi, rok završetka, datum kreiranja, vlasnik, članovi i zadaci zaduženi za isti projekt. S obzirom na stanje u kojem se zadatak nalazi kartica zadatka može poprimiti tri različite boje. Zelena boja predstavlja završen zadatak, plava boja predstavlja zadatak u izvođenju dok crvena boja predstavlja obustavljen zadatak. Ispod prikaza postojećih zadataka nalazi se forma u kojoj je moguće dodati novi zadatak. Također su vidljivi gumbovi za uređivanje projekta, dodavanje zadataka, prikaz zadataka i uređivanje zadataka.



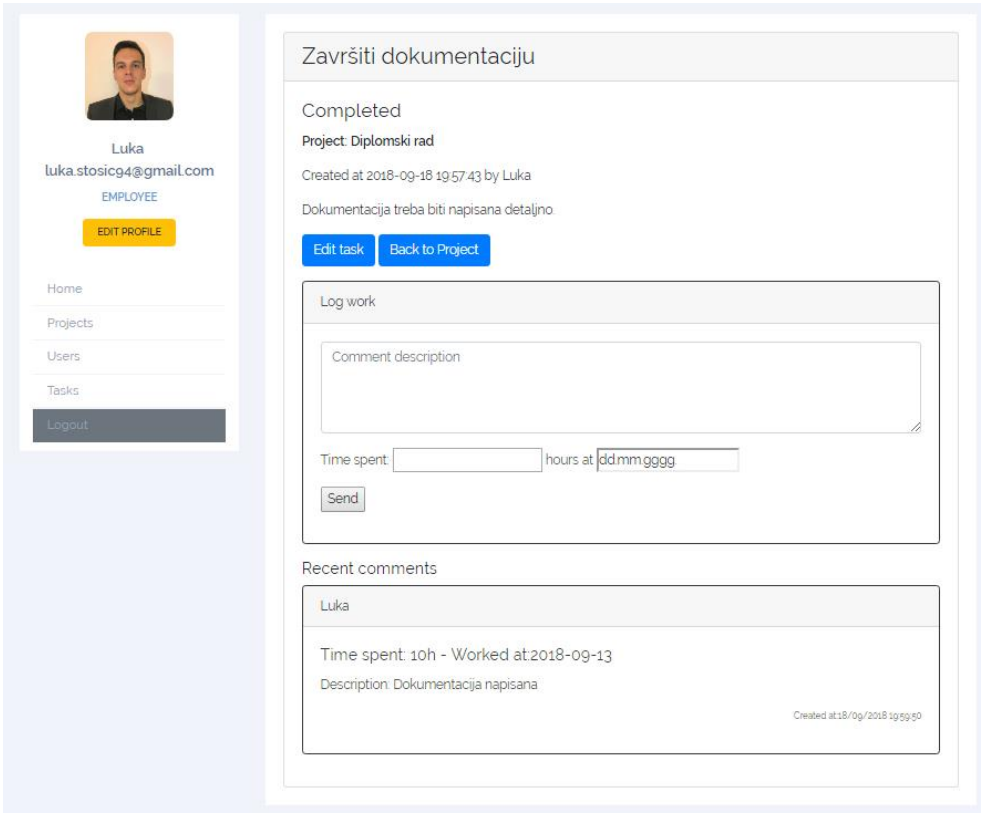
Sl. 3.6. – Prikaz stranice projekta

Odabirom gumba „Edit“ preusmjerava na stranicu za uređivanje trenutnog projekta. Izmjene na projektu mogu raditi članovi i vlasnik projekta. Forma za uređivanje omogućava promjenu naziva i opisa projekta, te dodavanje novih članova i uklanjanje postojećih. U formi se također nalaze četiri „radio“ gumba koji omogućuju promjenu stanja projekta. Nakon odrađenih potrebnih izmjena odabirom na gumb „Update“ pohranjuju se iste.



Sl. 3.7. – Prikaz stranice za uređivanje projekta

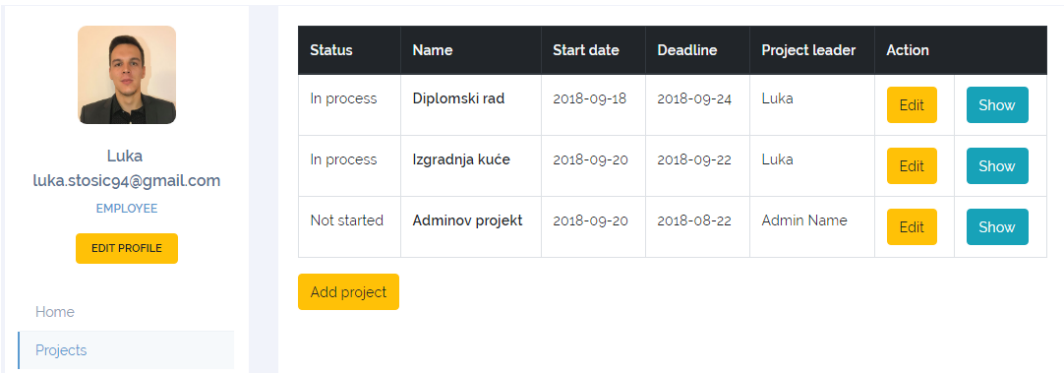
Ulaskom u prikaz pojedinog zadatka vidljive su detaljne informacije. One sadrže naziv, stanje, projekt pod koji pripada zadatak, datum kreiranja, sliku zadatka ako je dodana, vlasnika i opis zadatka. Uz informacije o zadatku na stranici se nalazi forma za dodavanje radnih bilješki. Da bi se ispravno popunile radne bilješke potrebno je unijeti komentar, vrijeme potrebno da se obavi posao i datum za koji je posao obavljen. Potvrdom na tipku „Send“ dodaje se radna bilješka i šalje se e-mail vlasniku zadatka s informacijama iz radne bilješke. Radne bilješke prikazane su na istoj stranici kronološkim redoslijedom, počevši od posljednje kreirane.



Sl. 3.8. – Prikaz stranice pojedinog zadatka

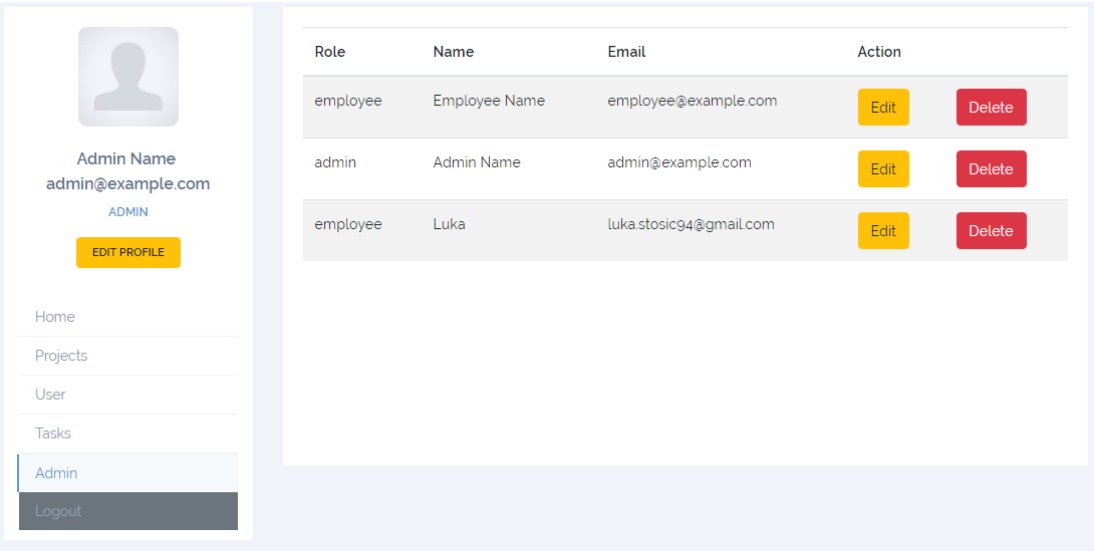
Odabirom gumba „Edit task“ korisnik se preusmjerava na stranicu za uređivanje zadatka. Na stranici za uređivanje zadatka moguće je izmijeniti naziv, opis i status zadatka.

Korisniku je moguće vidjeti sve postojeće projekte uz najosnovnije informacije na stranici „Projects“, no može pristupiti samo onim projektima koje je on sam kreirao ili je njihov član. Na istoj stranici korisnik može odabrati tipku „Add project“ i otići na novu stranicu za kreiranje novog projekt. Stranica za kreiranje novog projekta sadrži forme za dodavanje naziva, opisa i roka za završetak projekta.



Sl. 3.9. – Prikaz stranice svih projekata

Ovlaštene osobe za mijenjanje informacija o bilo kojem korisniku su admini. Samo korisnici s ulogom admin mogu pristupiti stranici na kojoj je popis svih registriranih korisnika u internet aplikaciji. Admin može raditi izmjene na korisničkom računu, odnosno promijeniti ulogu, ime i e-mail korisnika. Također jedina osoba koja može brisati korisničke račune iz baze podataka je admin potvrdom nagumb „Delete“.



Sl. 3.10. – Prikaz Admin stranice

4. ZAKLJUČAK

U ovom diplomskom radu izrađena je internet aplikacija za organizaciju timskog rada. Prvi korak bio je proučavanje funkcionalnosti sličnih aplikacija Asana i Jira kako bi se utvrdilo na koji način funkcioniraju i što će biti potrebno obuhvatiti u izradi internet aplikacije. Nakon toga osmišljena je shema za izradu baze podataka u aplikaciji. Na temelju sheme za izradu baze podataka radi se razvoj u Laravel okviru. Prvi korak razvoja u Laravelu bilo je kreiranje migracija koje su se koristile za stvaranje stvarnih tablica u bazi podataka. Na temelju sheme baze podataka modificirani su modeli, odnsono atributi kojima je bilo potrebno dodati svojstvo popunjivosti i odrediti relacije. Da bi modeli imali svrhu potrebno je bilo dodati rute, upravitelje i poglede. Rute su definirane ovisno o potrebi za stranicama kojima će korisnik pristupati. U upraviteljima se odradio dio koda koji obavlja logiku aplikacije i koriste se isključivo CRUD metode. Pogledi se koriste za interakciju korisnika i aplikacije te je fokus stavljen na funkcionalnost i jednostavnost izgleda. U aplikaciji je uspješno odrađeno registriranje i prijava korisnika u sustav koji mogu imati različite uloge (admin, zaposlenik).Korisnici koji se registriraju imaju mogućnost stvaranja i uređivanje projekata, dodjeljivanje novih zadataka projektu, komentiranje zadataka pomoću radnih bilješki. Uz komentiranje zadataka pomoću radnih bilješki prati se vrijeme rada korisnika koje se ispisuje na korisničkom profilu. Također su ugrađene e-mail poruke koje obavještavaju vlasnike zadataka da je stvorena nova radna bilješka na njihovom zadatku. Sporedne opcije su mogućnost dodavanja slike za zadatke i korisničke profile. Korisnici koji nisu admini imaju ograničen pristup određenim dijelovima aplikacije. Kao nedostatak naveo bih mogućnost dodavanja više opcija, a razlog je širok spektar opcija koje sadrže već postojeće aplikacija.

LITERATURA

[1] – Laravel, https://laravel.com, pristupljeno rujan 2018.

[2] – JavaScript, https://www.javascript.com, pristupljeno lipanj 2018.

[3] – HTML, https://html.com/, pristupljeno lipanj 2018.

[4] – CSS, https://www.w3schools.com/css/, pristupljeno lipanj 2018.

[5] – XAMPP, https://www.apachefriends.org/index.html, pristupljeno lipanj 2018.

[6] – PHP, http://php.net/, pristupljeno lipanj 2018.

[7] – MySQL, https://hr.wikipedia.org/wiki/MySQL, pristupljeno lipanj 2018.

[8] – Bootstrap, https://getbootstrap.com/, pristupljeno rujan 2018.

[9] – Asana, https://asana.com/, pristupljeno rujan 2018.

[10] – Jira, https://www.atlassian.com/software/jira, pristupljeno rujan 2018.

SAŽETAK

Naslov: Internet aplikacija za organizaciju timskog rada

Za uspješnu izradu diplomskog rada bilo je potrebno koristiti Laravel, HTML, CSS, PHP, Bootstrap, Javascript, te proučiti aplikacije slične funkcionalnosti (Asana, Jira). Prije početka izrade aplikacije kreirana je shema baze podataka. Izrada aplikacije u Laravel-u odrađuje se na temelju MVC arhitekture budući da je to i smisao Laravel-a. Aplikacija se sastoji od sučelja za dodavanje i uređivanje projekata, zadataka i radnih bilješki, prikaza korisničkog profila, slanja e-mailova te administratorskog sučelja. Registrirani korisnici, odnosno oni koji postoje u bazi podataka imaju uloge koje samo admin može mijenjati. Uloge su uvedene u svrhu ograničavanja mogućnosti korisnika. Svaki od korisnika unošenjem radnih bilješki koje su vezane za projekt, unosi i vrijeme rada na istom te se na taj način prati broj odrađenih sati za svaki mjesec. Projekti, zadaci i radne bilješke mogu se shvatiti kao princip stabla čvorova, gdje je projekt korijen čvor, zadaci se nalaze na nivou (dubini) 1,a radne bilješke pripadaju nivou (dubini) 2. Time se korisniku omogućavadodavanje jednog ili više zadataka u svaki projekt, te jednu ili više radnih bilješki u zadatak.

Ključne riječi: Laravel, MVC, organizacija timskog rada, CRUD, baza podataka

ABSTRACT

Title: Internet application for teamwork organizing

For a successful graduate thesis, it was necessary to use Laravel, HTML, CSS, PHP, Bootstrap, Javascript, and to learn about applications with similar functionality (Asana, Jira). Before creating the application, a database schema was created. Creating an application in Laravel is based on MVC architecture. The application consists of interfaces for adding and editing projects, tasks, and work notes, user profile views, emails, and administrative interfaces. Registered users, the ones who are allready in the database, have roles that only the administrator can change. Roles have been introduced in this project to limit the user's ability. Every user inputs notes that are related to the project, the number of hours that he worked on a project, which keeps track of the number of hours worked for each month. Projects, tasks and work notes can be shown as a node tree, where the project is the root node, the tasks are at the first level, and the work notes belong to the second level. This form allows the user to add one or more tasks to each project, and one or more work notes for each task.

Keywords: Laravel, MVC, teamwork organization, CRUD, database

ŽIVOTOPIS

Luka Stošić rođen je 10. kolovoza 1994. godine u Đakovu. Godine 2009. upisuje Opću gimnaziju u Đakovu. 2013. godine završava srednju školu te upisuje preddiplomski studij računarstva na Fakultetu elektrotehnike, računarstva i informacijskih tehnologija u Osijeku. 2016. godine upisan diplomski studij Računarstva.

Luka Stošić

PRILOZI

Kod na DVD-u priloženom uz rad.

Seminar u .doc i .pdf formatu.