Web aplikacija za praćenje stanja projekata

Borovica, Igor

SVEUČILIŠTE JOSIPA JURJA STROSSMAYERA U OSIJEKU
ELEKTROTEHNIČKI FAKULTET

Sveučilišni studij

Web aplikacija za praćenje stanja projekta

Diplomski rad

Igor Borovica

Osijek, 2016.

D1 OBRAZAC

IZJAVA O ORIGINALNOST

SADRŽAJ

1. UVOD

Svaki razvoj novog projekta, bilo da je u pitanju veliki ili mali projekt, zahtijeva praćenje stanja razvoja. Sve osobe koje sudjeluju u razvoju projekta trebaju imati mogućnost uvida u stanje projekta. Najjednostavniji način da se omogući praćenje stanja svim sudionicima u razvoju projekta je pomoću jednostavnih alata. U ovom diplomskom radu izrađena je jednostavna aplikacija za praćenje stanja projekta kreiranjem sastanaka i dodavanjem zadataka koje je potrebno obaviti. Svaki sudionik projekta ima mogućnost u bilo kojem trenutku vidjeti trenutno stanje projekta. Izrada aplikacije podijeljena je u dva dijela: teorijski i praktični dio.

U drugom poglavlju opisane su tehnologije i alati potrebni za izradu aplikacije. Spominju se relacijske baze podataka, te alati za izradu baze podataka. Opisane su Internet tehnologije pomoću kojih je aplikacija izrađena, a to su: PHP, Javascript, HTML i CSS. U trećem poglavlju opisano je ostvareno programsko rješenje. Opisan je postupak izrade baze podataka i prikazan izgled relacijske i fizičke sheme baze podataka. Nakon toga opisana je izrada aplikacijskog programskog sučelja pomoću koje korisničko sučelje komunicira s bazom podataka te je prikazan izgled MVC strukture. Na kraju poglavlja opisana je izrada korisničkog sučelja. Korisničko sučelje je izrađeno kao jednostranična aplikacija te je dan uvid u jednostranične aplikacije. Također je prikazan način korištenja ostvarenog programskog rješenja.

1.1. Zadatak diplomskog rada

U ovom radu je potrebno izraditi web aplikaciju za praćenje stanja projekata. Svaki projekt treba imati vremensku liniju i bojama naznačene događaje. Između ostalog treba napraviti login, upload datoteka, termine sastanaka te evidenciju i realizaciju ideja tijekom rada na projektu.

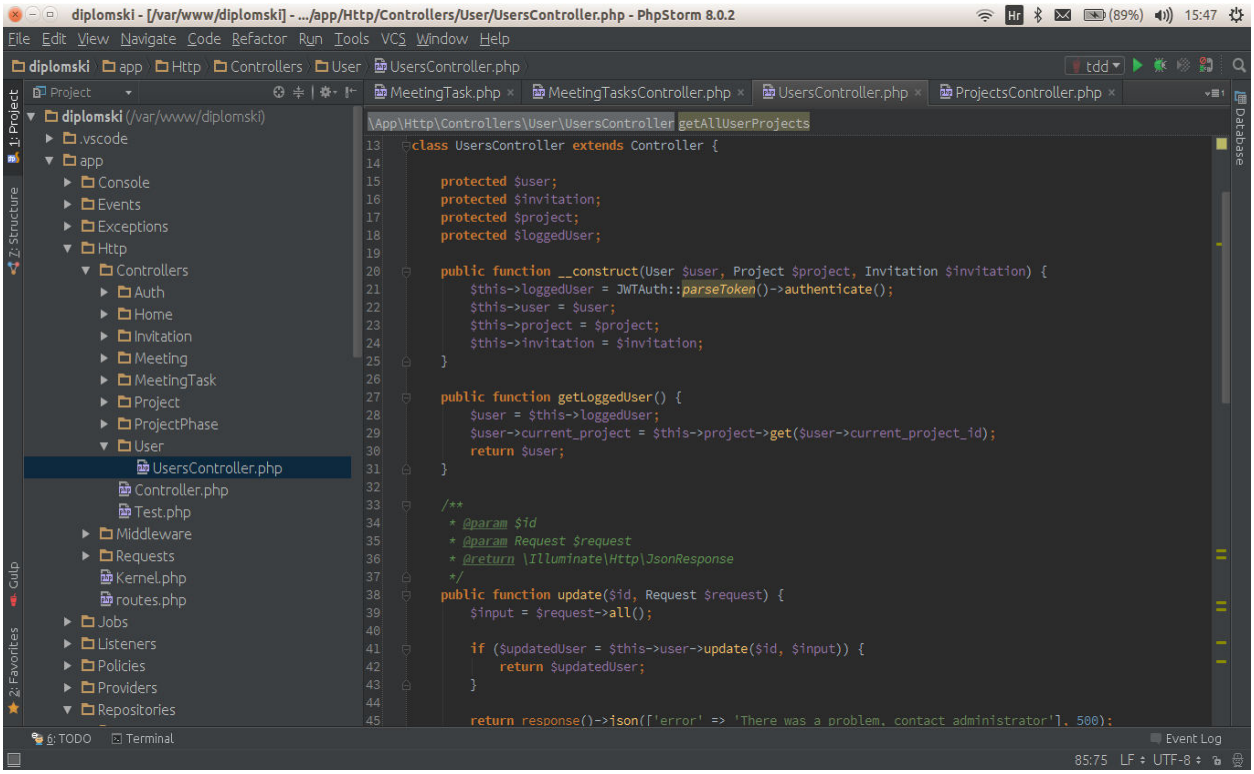
2. PRIMJENJENE TEHNOLOGIJE I ALATI

Tehnologije za izradu Internet aplikacije podijeljene su u dvije vrste:

- Tehnologije za izradu i komunikaciju s bazom podataka
- Tehnologije za izradu korisničkog sučelja

2.1. Tehnologije za izradu komunikacije s bazom podataka

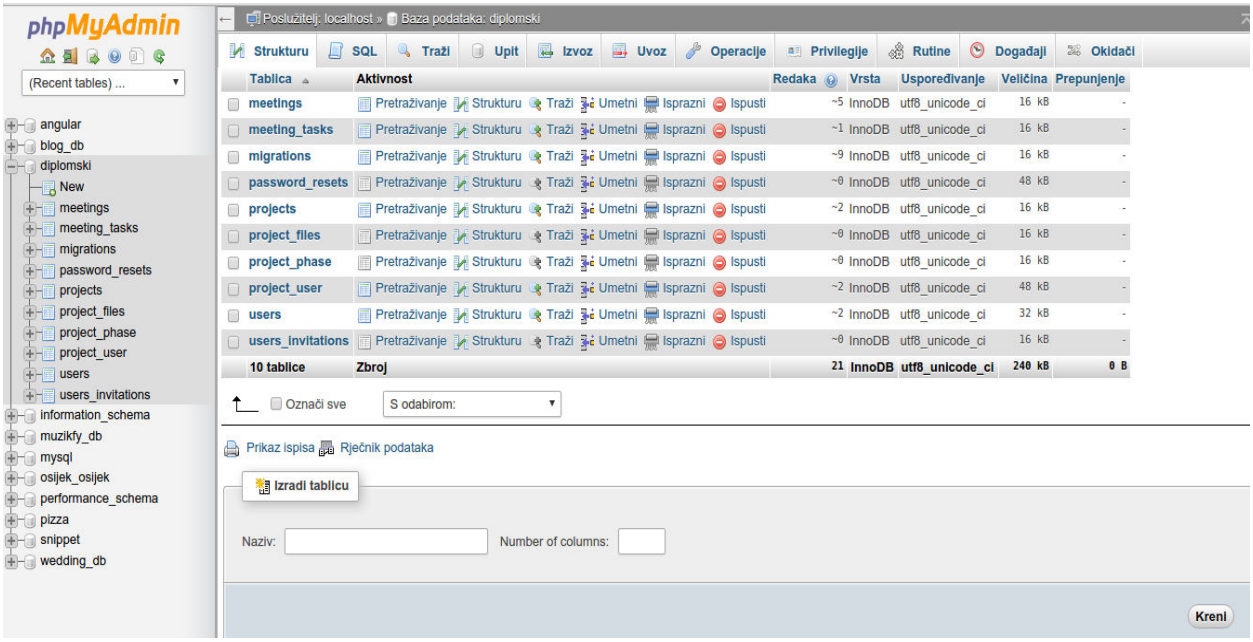
Komunikacija s bazom podataka vrši se preko aplikacijskog programskog sučelja (engl. *Application programming interface*, u nastavku samo API), te je za izradu API-a izabran programski jezik PHP, te njegov Laravel *framework.* Web API će biti zadužen za spremanje, dohvaćanje i dodatnu obradu podataka iz baze podataka. API kontroleri implementiraju REST (engl. *Representational State Transfer)* pozive na bazu pomoću koji se dobivaju podaci. Server šalje odgovore u JSON (engl. *JavaScript Object Notation)* formatu. PHP je skriptni programski jezik za izradu dinamičkih Internet stranica, čiju je početnu inačicu razvio Rasmus Lerdorf 1994. godine [1]. Zbog toga što je PHP skriptni programski jezik nije potreban prevodilac. Jedna od najvećih prednosti PHP programskog jezika je ta što podržava većinu baza podataka (MySQL, PostgreSQL, Oracle, MongoDB, itd). Sintaksa je slična programskom jeziku C [1]. Laravel je trenutno najpoznatiji PHP *framework* za razvoj Internet aplikacija, Laravel je besplatan, te je njegov izvorni kod dostupan svima. Za pokretanje aplikacije koristi se Apache HTTP (engl. *Hypertext Transfer Protocol*, dalje tu tekstu HTTP) server. PHP nema svoje službeno okruženje za razvoj, potrebno je samo napraviti praznu .php datoteku, te je moguće koristiti bilo koji uređivač teksta, također je moguće izvođenje preko komandne linije. Radna okruženja postoje, najpoznatije radno okruženje je PhpStorm, te je za izradu ovog diplomskog rada korišten PhpStorm (Sl. 2.1.).



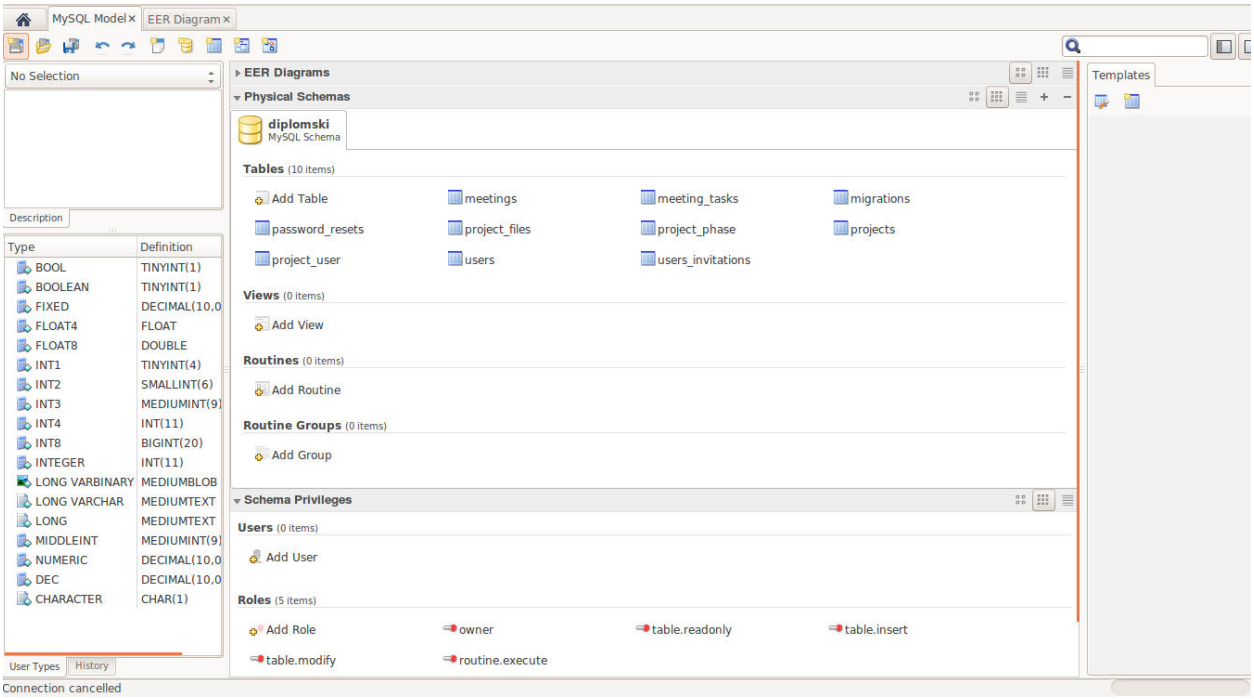
Slika 2.1. Razvojno okruženje PhpStorm.

2.2. Tehnologije za izradu i administraciju baze podataka

Baza podataka služi za spremanje podataka o kreiranom projektu kako bi se kasnije ti isti podaci mogli dohvaćati iz baze te obrađivati i prikazati krajnjem korisniku. Za izradu baze podataka u ovom diplomskom radu izabran je MySQL sustav za upravljanje bazom podataka. MySQL je besplatan sustav otvorenog koda, te je uz PostgreSQL najčešći izbor za izradu baze podataka. MySQL baza podataka je relacijska baza podataka. U relacijskim bazama podataka podaci se organiziraju u skup relacija koje su međusobno povezane određenim vezama. Svaka relacija može imati primarni i strani ključ pomoću kojih se ostvaruju veze između relacija. Alat za upravljanje i administraciju baze podataka u ovom diplomskom radu je phpMyAdmin. PhpMyAdmin je besplatan alat otvorenog koda izrađen u PHP-u koji olakšava administraciju baze podataka. Upravljanje se vrši preko Internet preglednika (Sl.2.2), te nakon instalacije može se otvoriti odlaskom na Internet adresu: http://localhost/phpmyadmin. Za pristup korisničkom sučelju potrebno je unijeti korisničko ime i lozinku (korisničko ime je root, dok polje za lozinku ostaje prazno). Pomoću phpMyAdmin alata moguće je brzo i jednostavno izraditi bazu podataka. Za izradu strukture baze podataka korišten je MySQLWorkbench (Sl. 2.3.) [4].



Slika 2.2. Alat za upravljanje i administraciju baze podataka.

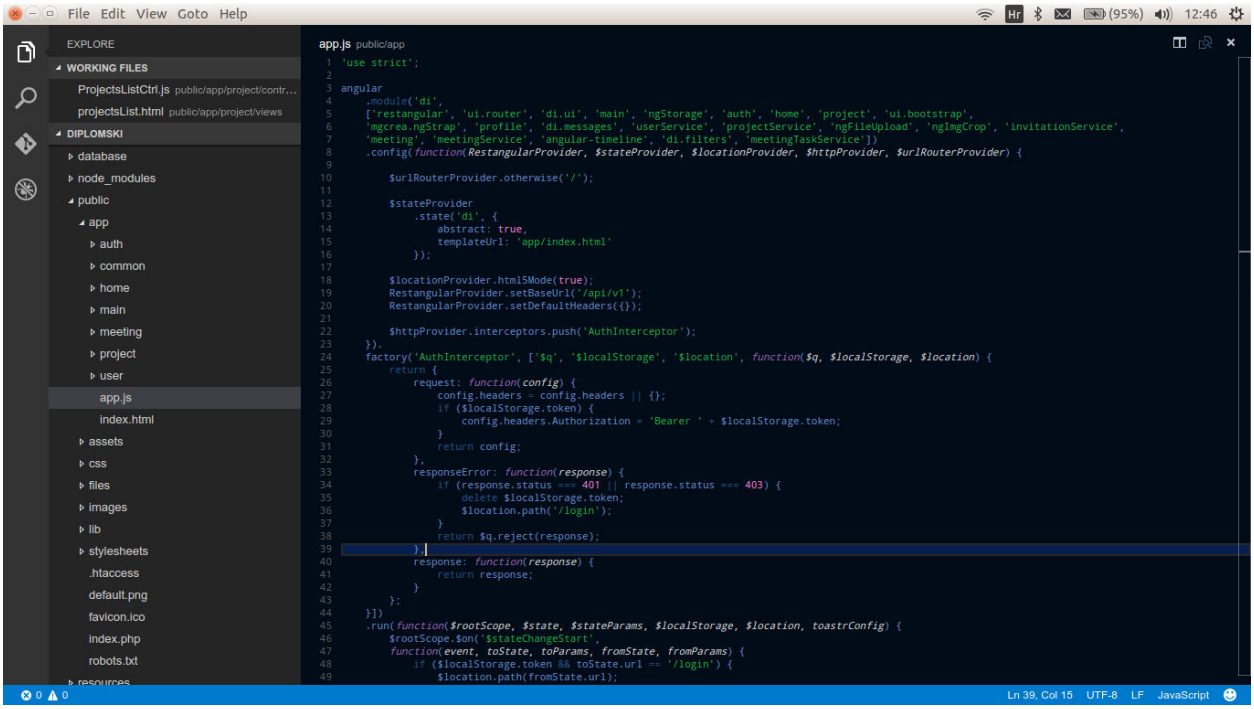


Slika 2.3. Radno okruženje MySQLWorkbench.

2.3. Tehnologije za izradu korisničkog sučelja

Korisničko sučelje omogućuje korisniku kreiranje projekta, te praćenje stanja projekta. Za izradu ovog diplomskog rada korištene su standardne tehnologije koje se koriste za izradu Internet aplikacija, a to su: HTML, CSS i JavaScript. HTML (engl. *HyperText Markup Language*) služi za opisivanje Internet dokumenata. HTML nije programski jezik, te pomoću njega nije moguće izvoditi jednostavne operacije kao što su zbrajanje ili oduzimanje, ne mogu se izvršavati petlje te ispitivati uvjeti. Sastoji se od HTML oznaka pomoću kojih se opisuje HTML dokument. Glavne oznake HTML-a su: <html>, <head>, <body>, <title>. Za uređivanje HTML dokumenta u ovom diplomskom radu koristi se CSS (engl. *Cascading Style Sheets*), te njegova ekstenzija Sass (engl. *Syntactically Awesome Stylessheets*). CSS opisuje kako će se elementi prikazivati na ekranu. Pomoću CSS-a definira se između ostalog razmak između tekstova, veličina i boja fonta, širina i boja linija. Pomoću Sass-a moguće je definirati varijable, te jednostavnije gniježđenje CSS pravila, također je moguće kreiranje funkcija koje se mogu koristiti kroz aplikaciju, a samim time se izbjegava ponavljanje koda [2]. Sintaksa Sass-a je jednaka sintaksi CSS-a. JavaScript je skriptni programski jezik koji služa za komunikaciju između klijenta i API-a. JavaScript je objektno orijentirani programski jezik koji se izvršava u Internet pregledniku, odnosno na strani klijenta [3]. Rad sa JavaScriptom može biti izazovan zbog toga jer se aplikacija mora pravilno izvoditi ne samo na različitim tipovima računala, nego i u nekoliko različitih Internet preglednika kao što su Chrome, Firfox, Opera, Safari i Internet Explorer [3]. JavaScript ima veliku popularnost zbog svog laganog dodavanja u Internet stranice. Sve što je potrebno je uključiti HTML element <script> u stranicu, zadati "text/javascript" atribut za *type*, te dodati JavaScript kod [3]. JavaScript kod može se pisati unutar <script> elementa ili se može izvoditi učitavanjem vanjskih JavaScript datoteka tako da se unutar <script> elementa unosi putanja do vanjske JavaScript datoteke (za postavljanje putanje koristi se *src* atribut). JavaScript kod dodaje se unutar head ili body oznaka, ili se čak može dodati u jedan i drugi element. Smještanjem JavaScript koda unutar head elementa može se zaustaviti učitavanje Internet stranice, radi toga preporuča se spremanje skripte na dno body elementa [3]. U ovom diplomskom radu za što bolju komunikaciju između klijenta i poslužitelja korišten je Ajax. Ajax (engl. *Asynchronous JavaScript and XML*) funkcionira tako da se poslužitelju pošalje upit, usluga se pozove i podaci se vrate nazad te se prikažu korisniku. Prednost Ajax-a je ta što nema ponovnog učitavanja stranice nakon što je upit poslan i obrađen, nego se sve obavlja unutar konteksta Internet stranice [3]. Ajax omogućuje asinkronu komunikaciju između klijenta i poslužitelja, što znači da korisnik ne mora čekati da se zahtjev završi nego može nastaviti sa

korištenjem aplikacije dok se zahtjev izvršava, te nakon što se zahtjev izvrši rezultati će biti prikazani korisniku. Radno okruženje za izradu korisničkog sučelja je Visual Studio Code (Sl. 2.4.). Visual Studio Code je nastao od strane Microsoft-a, te je besplatan i može se izvoditi na svim platformama.



Slika 2.4. Visual Studio Code.

3. OSTVARENO PROGRAMSKO RJEŠENJE

Ostvareno programsko rješenje podijeljeno je u slijedeća tri dijela:

- Izrada baze podataka za potrebe Internet aplikacije
- Izrada aplikacijskog programskog sučelja
- Izrada korisničkog sučelja

3.1. Izrada baze podataka za potrebe Internet aplikacije

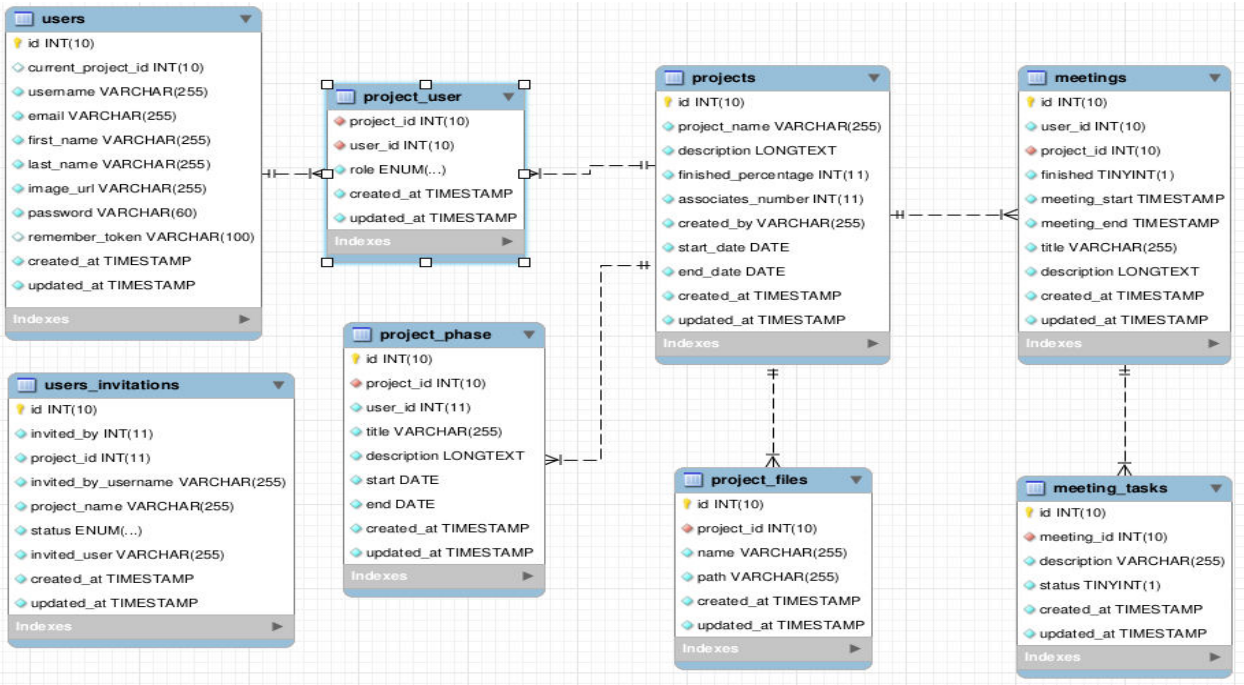
Baza podataka je osnova svake moderne Internet aplikacije. Pri izradi svake Internet aplikacije potrebno je na početku dobro postaviti bazu podataka jer kako aplikacija raste stvari se mogu dodatno zakomplicirati ako je baza podataka loše postavljena. Analizom aplikacije i zahtjeva koje aplikacija za praćenje stanja projekta treba obavljati postavljeni su sljedeći entiteti i atributi:

- Entitet *USERS* ima sljedeće atribute: *ID, CURRENT_PROJECT_ID, USERNAME, EMAIL, FIRST_NAME, LAST_NAME, IMAGE_URL, PASSWROD, REMEMBER_TOKEN, CREATED_AT, UPDATED_AT*.
- Entitet *PROJECTS* ima sljedeće atribute: *ID, PROJECT_NAME, DESCRIPTION, FINISHED_PERCENTAGE, START_DATE, END_DATE , ASSOCIATES_NUMBER, CREATED_BY, CREATED_AT, UPDATED_AT*.
- Entitet *MEETINGS* ima sljedeće atribute: *ID, USER_ID, PROJECT_ID, FINISHED, MEETING_START, MEETING_END, TITLE, DESCRIPTION*.
- Entitet *USERS_INVITATIONS* ima sljedeće atribute: *ID, INVITED_BY, PROJECT_ID, INVITED_BY_USERNAME, PROJECT_NAME, STATUS, INVITED_USER, CREATED_AT, UPDATED_AT*.
- Entitet *PROJECT_PHASE* ima sljedeće atribute: *ID, PROJECT_ID, USER_ID, TITLE, DESCRIPTION, START, END, CREATED_AT, UPDATED_AT*.
- Entitet *PROJECT_FILES* ima sljedeće atribute: *ID, PROJECT_ID, NAME, PATH, CREATED_AT, UPDATED_AT*.
- Entitet *MEETING_TASKS* ima sljedeće atribute: *ID, MEETING_ID, DESCRIPTION, STATUS, CREATED_AT, UPDATED_AT*.

Kao što je već navedeno u poglavlju 2.2. za izradu prikaza baze podataka korišten je alat pod nazivom MySQLWorkbench. Pomoću MySQLWorkbench alata moguće je također spajanje na localhost i phpMyAdmin. Baza podataka aplikacije za praćenje projekta sastoji se od osam tablica koje su međusobno povezane relacijama (Sl. 3.1.). U relacijskim bazama podataka postoje tri osnovne relacije, a to su:

- 1:1 - jedan na prema jedan relacija, svaki entitet može biti povezan samo sa jednim entitetom drugog tipa i obrnuto.
- 1:N - jedan prema mnogo relacija, jedan entitet može biti u relaciji sa 0 ili više entiteta drugog tipa, ali drugi entitet može biti u relaciji samo sa jednim tipom drugog entiteta.
- M:N - mnogo na mnogo relacija, jedan entitet može biti povezan sa 0 ili više entiteta drugog tipa i obrnuto.

U ovom diplomskom radu korištene su 1:N i M:N relacije. M:N relacija korištena je pri povezivanju *USERS* i *PROJECTS* tablice. Kod kreiranja M:N relacije potrebno je koristiti pomoćnu tablicu u koju se spremaju strani ključevi povezanih tablica, u ovom slučaju to je *PROJECT_USER* tablica u koju se spremaju *USER_ID* i *PROJECT_ID*. U svim ostalim slučajevima korištena relacija između tablica je 1:N, osim kod *USERS_INVITATIONS* tablice koja služi kao pomoćna tablica za pozivanje korisnika s projektom u koji je pozvan.



Slika 3.1. Prikaz baze podataka.

Svaki korisnik može kreirati više projekta, te na svakom projektu može sudjelovati više korisnika. Svaki kreirani projekt može imati više faza projekta, datoteka potrebnih za izradu projekta, te sastanaka vezanih za projekt, svaki od tih entiteta pripadaju jednom projektu. Svaki sastanak može imati više zadataka koji se trebaju obraditi na svakom sastanku, te svaki zadatak može pripadati jednom sastanku. Na temelju zadanih veza dobivena je sljedeća relacijska shema prikazana je na slici 3.2.

USERS *(ID, CURRENT_PROJECT_ID, USERNAME, EMAIL, PASSWORD, FIRST_NAME, LAST_NAME, IMAGE_URL, PASSWORD, REMEMBER_TOKEN, CREATED_AT, UPDATED_AT*

PROJECT_USER *(PROJECT_ID, USER_ID, ROLE)*

PROJECTS *(ID, PROJECT_NAME, DESCRIPTION, FINISHED_PERCENTAGE, ASSOCITAES_NUMBER, CREATED_BY, START_DATE, END_DATE, CREATE_AT, UPDATED_AT)*

USER_INVITATIONS *(ID, INVITED_BY, PROJECT_ID, INVITED_BY_USERNAME, PROJECT_NAME, STATUS, INVITED_USER, CREATED_AT, UPDATED_AT)*

PROJECT_PHASE *(ID, PROJECT_ID, USER_ID, TITLE, DESCRIPTION, START, END, CREATED_AT, UPDATED_AT)*

PROJECT_FILES *(ID, PROJECT_ID, PATH, NAME, CREATED_AT, UPDATED_AT)*

MEETINGS *(ID, PROJECT_ID, USER_ID, FINISHED, MEETING_START, MEETING_END, TITLE, DESCRIPTION, CREATED_AT, UPDATED_AT)*

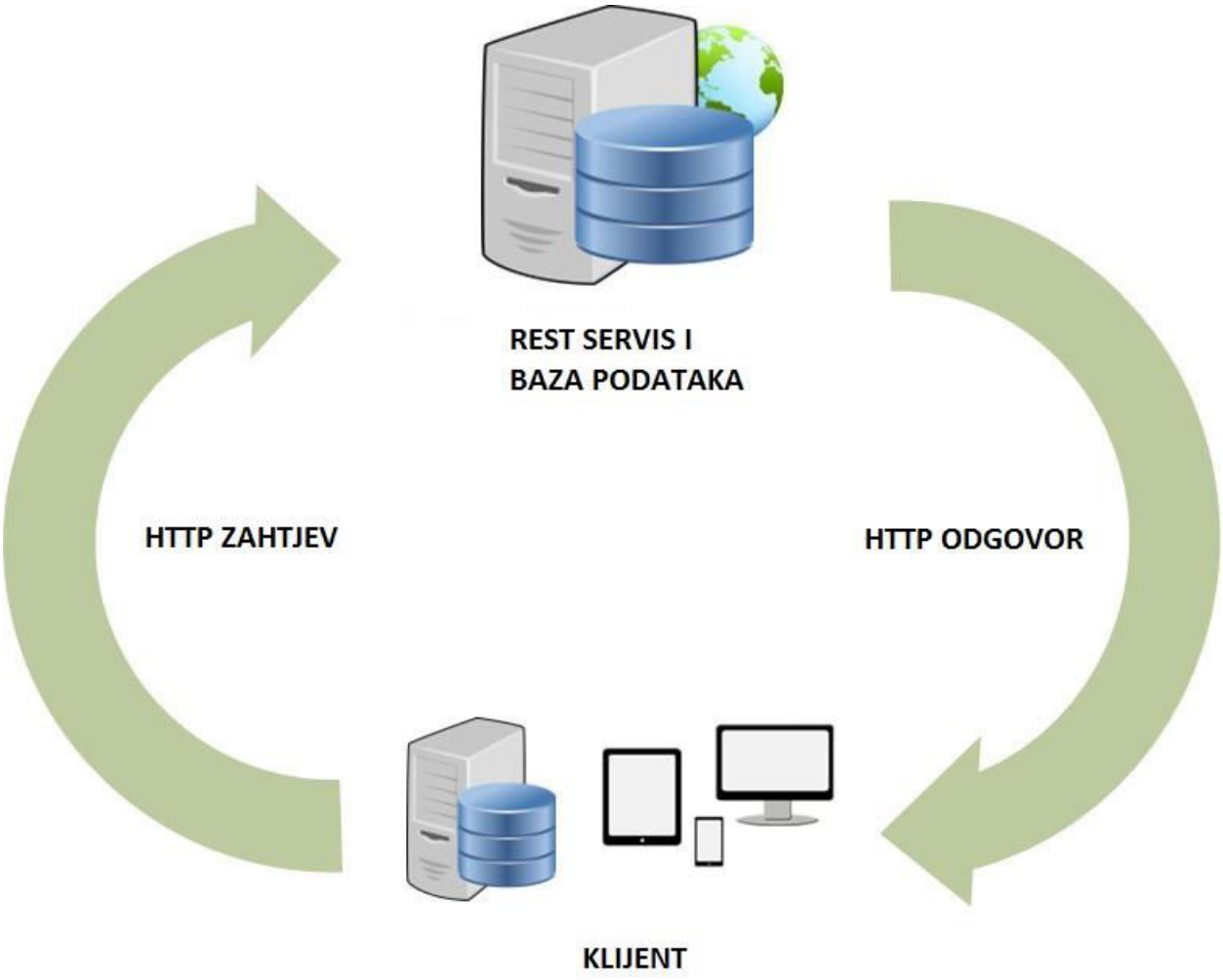
MEETING_TASKS *(ID, MEETING_ID, DESCRIPTION, STATUS, CREATED_AT, UPDATE_AT)*

Slika. 3.2. Relacijska shema baze podataka.

Na temelju relacijske baze podataka izrađena je fizička shema baze podataka (prilog A).

3.2. Izrada aplikacijskog programskog sučelja

Kao što je u poglavlju 2.1. navedeno aplikacijsko programsko sučelje izrađeno je u PHP-u i Laravel *framework-u.* U današnje vrijeme sve se više aplikacija razvija kao REST API zbog visokih performansi i jednostavne održivosti. REST servisi komuniciraju sa serverom preko HTTP glagola *(POST, GET, UPDATE, DELETE)*. Pomoću ovih HTTP glagola moguće je slati upite na server i primati odgovore. Najveća prednost REST servisa je ta što nije bitno da li klijent šalje upit sa mobilne, desktop ili Internet aplikacije (Sl. 3.3.), sve dok je putanja za poslani upit točna. Dovoljno je da korisnik pošalje upit na server i ako nije došlo do greške tokom slanja upita, korisniku će biti prikazani rezultati na aplikaciji koju koristi.



Slika 3.3. Prikaz toka REST servisa.

U ovom diplomskom radu korisnik upite šalje preko Internet aplikacije. U tablici 3.1. prikazane su putanje za kreiranje, dohvaćanje, uređivanje i brisanje projekta. Primjer koda za tablicu 3.1. prikazan je u prilogu B.

Tablica 3.1. Primjer REST HTTP upita.

HTTP UPIT	**PUTANJA**	OPIS
GET	*users*/1/*projects*	Dohvati sve projekte za korisnika koji ima ID 1
GET	*users*/1/*projects*/1	Dohvati projekt koji ima ID 1
POST	*users*/1/*projects*	Kreiranje novog projekta
PUT	*users*/1/*projects*/1	Uređivanje projekta koji ima ID 1
DELETE	*users*/1/*projects*/1	Brisanje projekta koji sa ID 1

Kada korisnik pošalje *GET* upit na putanju *users*/1/*projects*/1 tada se u *ProjectsController* klasi okida *get* metoda koja prima jedan parametar (ID traženog projekta), te na osnovu primljenog parametra pronalazi i vraća projekt iz baze podataka kojem je ID jednak proslijeđenom parametru. Ako korisnik želi spremiti novi projekt u bazu tada se šalje *POST* upit na putanju *users*/1/*projects* i okida se *post* metoda koja prima dva parametra, prvi parametar je ID korisnika koji želi kreirati projekt, dok je drugi parametar *ProjectsFormRequest* klasa koja provjerava da li je korisnik pravilno ispunio formu za kreiranje projekta. Ako sve provjere prođu, tada se u bazu sprema novi projekt koji se zatim vraća kao odgovor korisniku, te se taj projekt postavlja za trenutni projekt. Za uređivanje kreiranog projekta šalje se *PUT* upit na putanju *users*/1/*projects*/1, te se zatim okida *put* metoda koja prima dva parametra, prvi parametar je ID

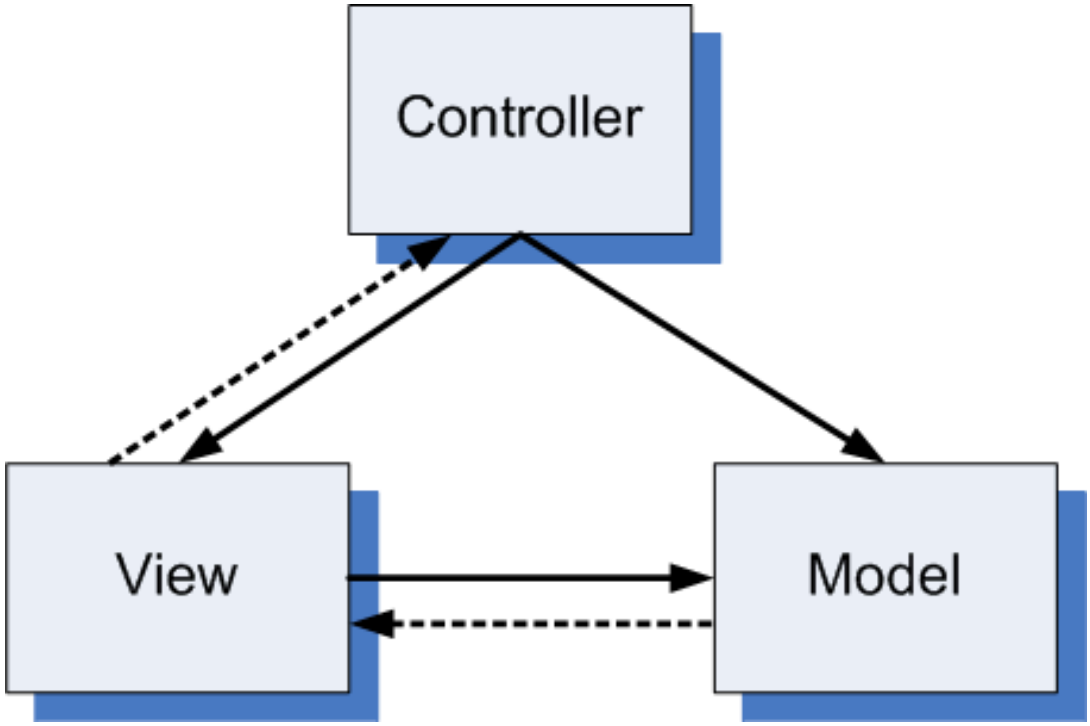
projekta, drugi parametar je *ProjectsFormRequest* klasa. Pomoću prvog parametra metoda pronalazi projekt u bazi te ga uredi sa novim parametrima koje je korisnik unio u formu. I na kraju ako korisnik želi izbrisati projekt šalje se *DELETE* upit na putanju *users*/1/*projects*/1, te se okida *delete* metoda koja prima dva parametra. Prvi parametar je ID korisnika koji briše projekt, drugi parametar je ID projekta kojega korisnik želi izbrisati. *Delete* metoda pomoću ta dva parametra pronalazi projekt u bazi i briše ga iz baze, te vraća korisniku poruku da je projekt uspješno izbrisan (prilog B). Kako bi servis prepoznao koju metodu treba okinuti kada primi određenu putanju potrebno je u *routes.php* datoteci definirati sve putanje koje servis treba prepoznati. U *routes.php* datoteci se definiraju putanje i njihove metode, na slici 3.4. prikazane su putanje za primjer *ProjectsController* klase.

```
get('projects/{id}', 'Project\ProjectsController@get');

get('projects/{projectId}/getUploadedFiles', 'Project\ProjectsController@getUploadedFiles');

post('users/{userId}/projects', 'Project\ProjectsController@post');

post('projects/upload', 'Project\ProjectsController@upload');

post('projects/{projectId}/uploadFiles', 'Project\ProjectsController@uploadFiles');

put('projects/{projectId}', 'Project\ProjectsController@put');

delete('users/{userId}/projects/{projectId}', 'Project\ProjectsController@delete');
```

Slika 3.4. Putanje HTTP upita za *ProjectsController.*

Routes.php datoteka je sastavni dio Laravel *framework-a* u kojoj se definiraju putanje servisa koji komunicira s bazom podataka. Ovisno o HTTP upitu postavljaju se metode za izvršavanje HTTP upita (npr. za *GET* zahtjev potrebno je pozvati *get* metodu). Prvi parametar predstavlja putanju, drugi parametar je metoda koja se treba okinuti kada se pozove ta putanja. Cijeli API napravljen je po MVC (engl. *Model-View-Controller*, dalje u tekstu samo MVC) strukturi. U današnjem svijetu razvoja aplikacija najbolji način izrade aplikacija je upravo MVC, zbog svoje

održivosti i modularnosti. Sve vrste aplikacija mogu se razvijati pomoću MVC strukture. Kod razvoja Internet aplikacija *Model* predstavlja svaku tablicu koja se kreira u bazi, te se pomoću *Model-a* podaci spremaju ili dohvaćaju iz baze podataka. *View* služi za prikazivanje rezultata korisniku, te korisnik preko *View-a* može pristupiti metodama u *Controlleru*. *Controller* služi za prijenos podataka između *Model-a* i *View-a*, također *Controller* služi za dodatnu obradu podataka. U *Controlleru* se pristupa *Modelu* pomoću kojeg se dohvaćaju ili spremaju podaci u bazu. Slika 3.5. prikazuje izgled MVC strukture.



Slika 3.5. MVC struktura.

U prilogu B prikazan je kod *ProjectsControllera*, a u prilogu C prikazan je kod *Project* modela. Kada se pozove post metoda prvo svi podaci se spreme u varijablu koja se zatim proslijedi *Project* modelu, te *Project* model pomoću *create* metode sprema podatke za novi projekt u bazu podataka. Korištenjem ovakvog načina programiranja također dolazi do puno jednostavnijeg pregledavanja koda.

3.3. Izrada korisničkog sučelja

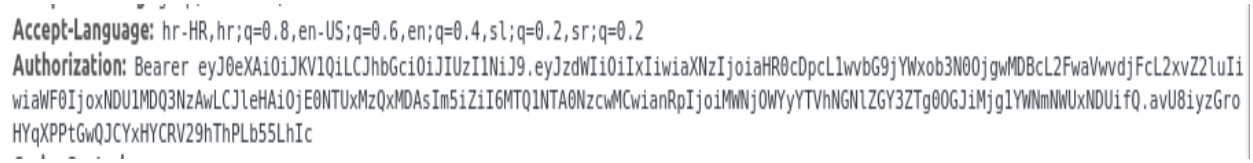
Korisničko sučelje predstavlja glavni dio aplikacije za praćenje stanja projekta. Kao što je navedeno u poglavlju 2.3. za izradu korisničkog sučelja korištene su sljedeće tehnologije: HTML, CSS i JavaScript. Korisničko sučelje napravljeno je kao jednostranična aplikacija (engl. *Single Page Application,* dalje u tekstu SPA).

3.3.1. Jednostranične aplikacije

Sve bržim razvojem Interneta i potrebe za boljim i bržim aplikacijama dolazi do razvoja jednostraničnih aplikacija. Jednostranične aplikacije izvode se na jednoj stranici što korisniku daje osjećaj korištenja Internet aplikacije kao desktop aplikacije. Glavne tehnologije za razvoje SPA su: HTML, CSS i JavaScript. Kod SPA sve se odvija u pozadini, a korisniku se samo prikazuju rezultati nakon obrade podataka,. Primjer jednostranične aplikacije je Gmail.

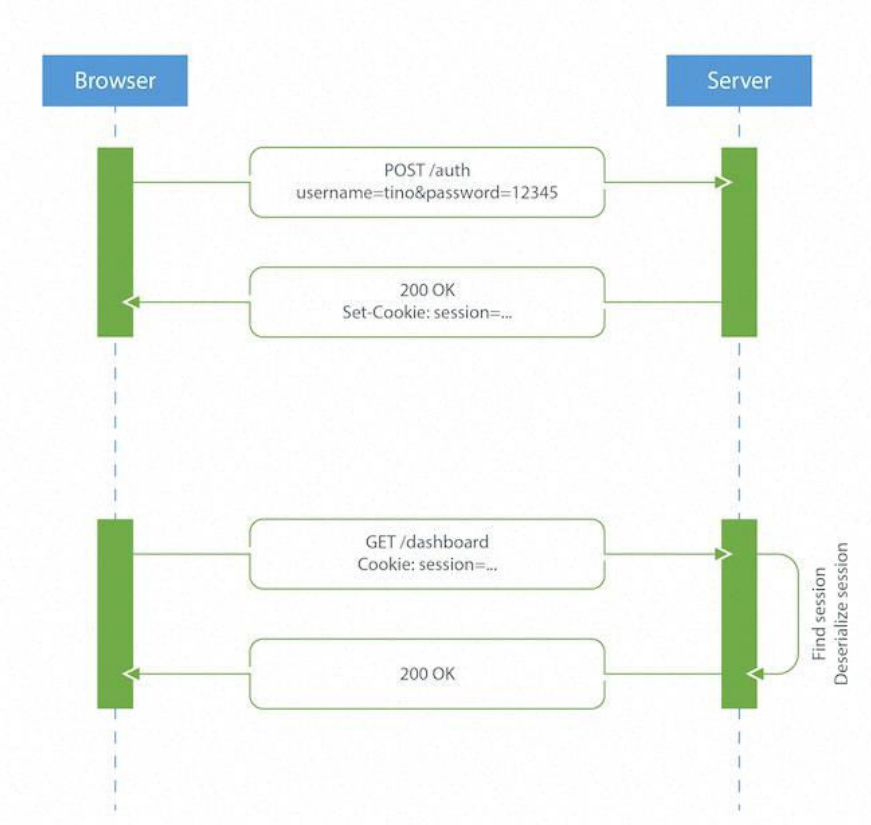
3.3.2. Registracija i prijava korisnika

Zbog toga što je aplikacija razvijena kao jednostranična aplikacija nije moguće koristiti standardni oblik prijave korisnika preko servera koristeći sesije i kolačiće. U ovom diplomskom radu za prijavu korisnika korišten je JWT (engl. *JSON Web Token,* dalje u tekstu JWT). JWT koristi se za prijenos informacija te se može provjeriti prema digitalnom potpisu. JWT je predstavljen kao *base64url* (vidi [5]) kodirane vrijednosti koje su odvojene točkom. Zbog svoje kompaktne strukture JWT se koristi u HTTP *Authorization* zaglavlju, na slici 3.6. prikazan je izgled JWT nakon što se korisnik prijavi u aplikaciju [6].



Slika 3.6. Izgled JWT.

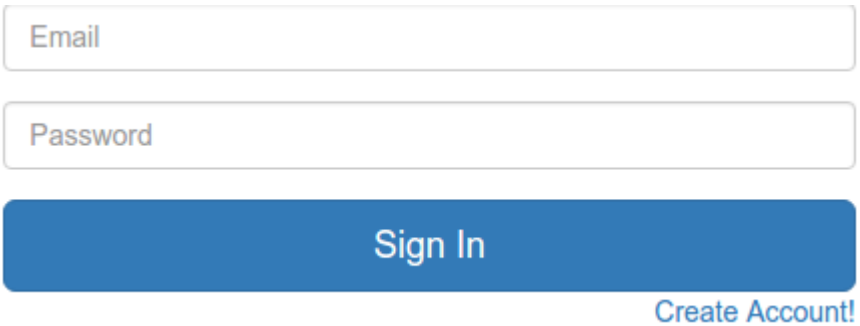
Na slici 3.7. prikazan je tok prijave korisnika pomoću servera koristeći sesije i kolačiće. Korisnik napravi *POST* upit prema serveru koji sadrži korisničko ime i lozinku, server odgovara postavljajući kolačić u korisnikov preglednik zajedno sa ID-em sesije koja je nastala na temelju korisničkog imena i lozinke. Na svakom zahtjevu server pregleda i deserijalizira sesiju zbog toga što su podaci o korisniku spremljeni na serveru.



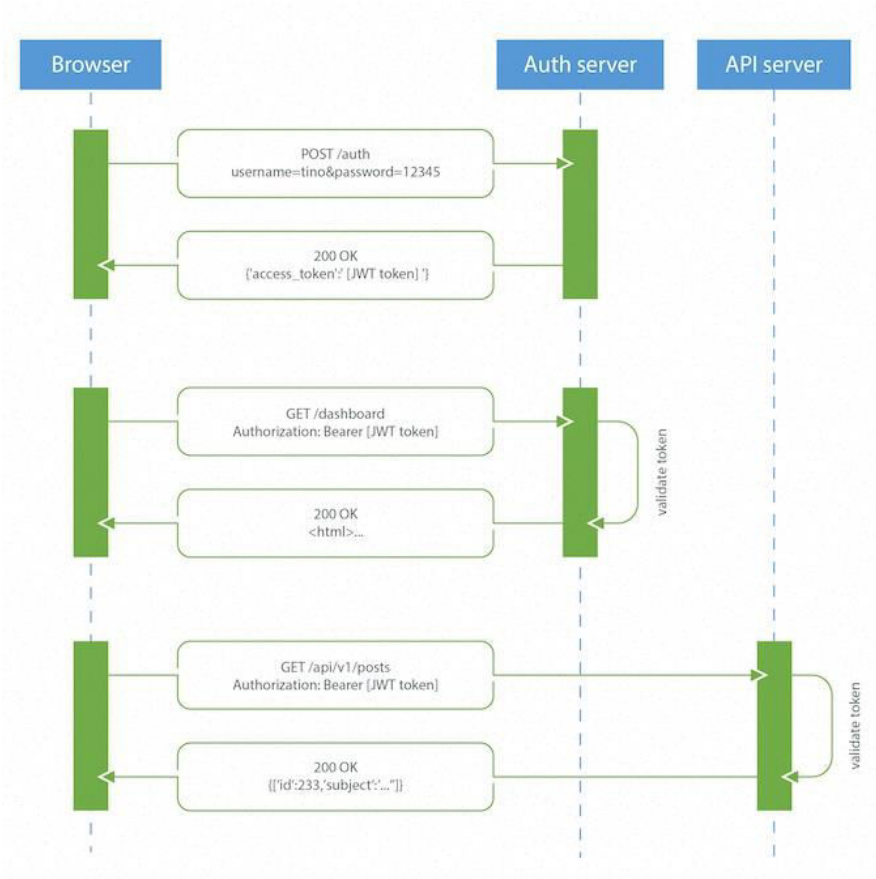
Slika 3.7. Tok autentikacije preko servera [6].

Za razliku od server autentikacije JWT autentikacija ne sprema informacije u sesiju, što dovodi do izrade skalabilnijih aplikacija. Korisnik napravit upit na server koji sadrži korisničko ime i lozinku, nakon toga server generira JWT koji se vraća klijentu. Svakim upitom korisnik šalje JWT unutar *Authorization* zaglavlja koji server provjerava te šalje sigurni izvor klijentu (slika 3.9) [6].

U ovom diplomskom radu za registraciju i prijavu korisnika potrebno je unijeti ispravno *email* adresu i lozinku (pomoću *email* adrese i lozinke generira se JWT). Na slici 3.8. prikazan je izgled obrasca za prijavu korisnika.



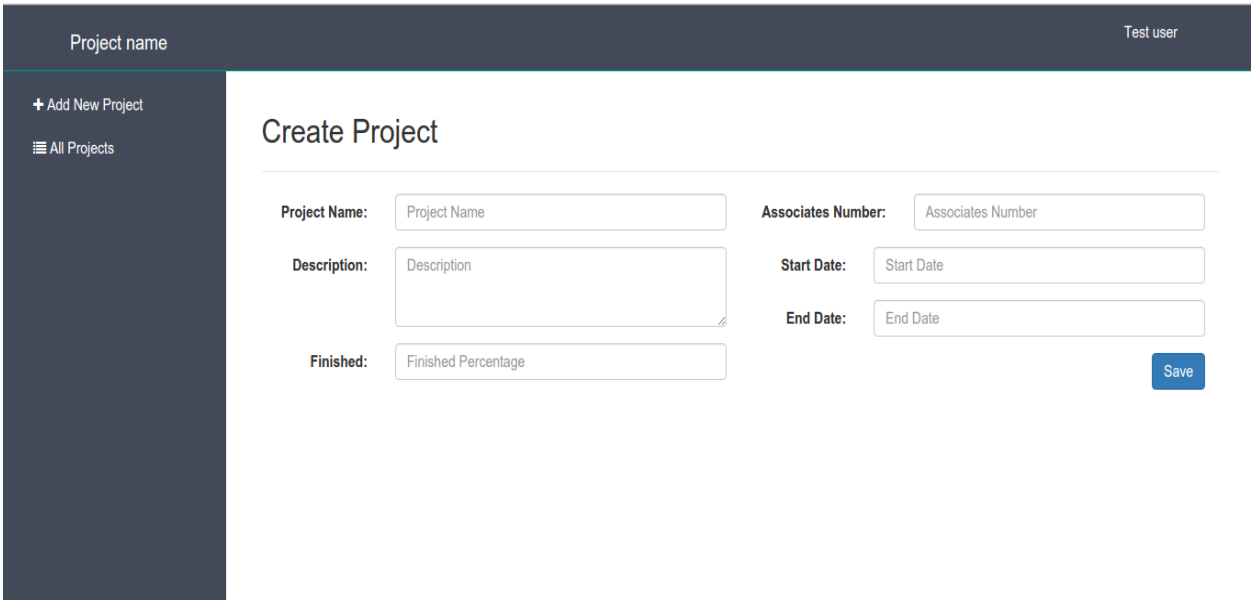
Slika 3.8. Obrazac za prijavu korisnika.



Slika 3.9. Tok autentikacije pomoću JWT [6].

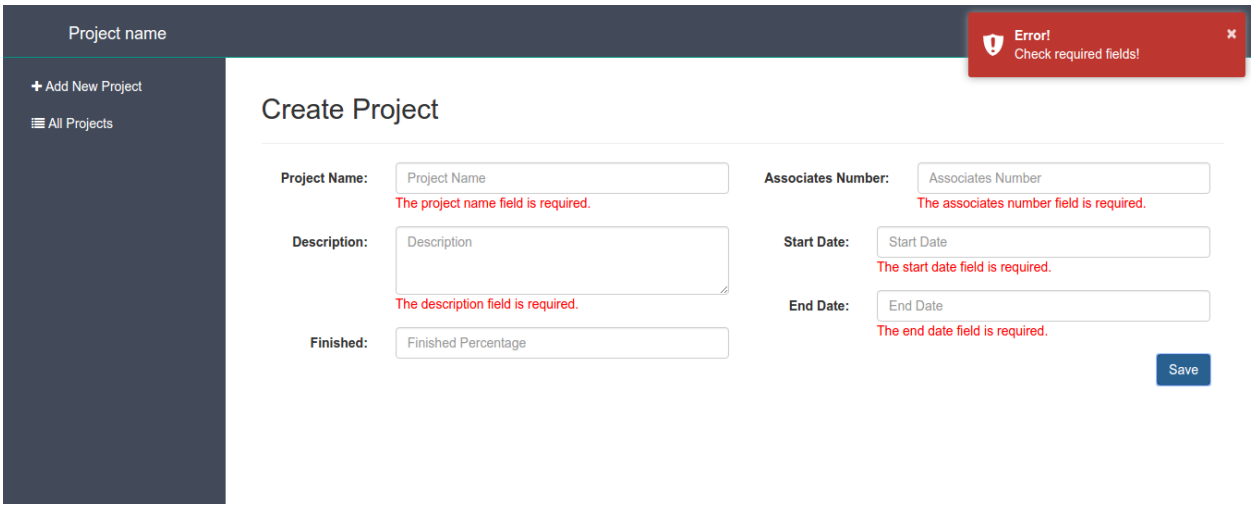
3.3.3. Kreiranje i upravljanje projektom

Nakon što se korisnik uspješno prijavi u aplikaciju ima mogućnost kreiranja novog projekta. Klikom na *Add New Project* korisniku se otvara obrazac za kreiranje novog projekta, *Add New Project* nalazi se na lijevoj strani u izborniku (Sl. 3.10.). Kako bi korisnik uspješno kreirao novi projekt potrebno je unijeti sve potrebne podatke, ako korisnik nije unio potrebne podatke ispod svakog polja u obrascu će se pojaviti poruka (Sl. 3.11.). Provjera da li je korisnik unio sve potrebne podatke obavlja se na server strani te ako korisnik nije unio potrebne podatke poruke se prikazuju pomoću JavaScript-a i HTML-a. Primjer koda za slanje upita i hvatanje grešaka pomoću JavaScripta prikazan u prilogu D.



Slika 3.10. Obrazac za unos novog projekta.

Kreiranje novog projekta obavlja se unutar *ProjectsControllera* okidanjem *post* metode (prilog B). Nakon što se novi projekt kreira, isti se postavlja za trenutni projekt korisnika, također nakon kreiranja projekta nema ponovnog učitavanja stranice zbog toga što je aplikacija napravljena kao jednostranična aplikacija. U slučaju da korisnik unese krive ili ne unese podatke, korisnik neće biti u mogućnosti kreirati novi projekt te će biti prikazane greške (Sl. 3.11.).

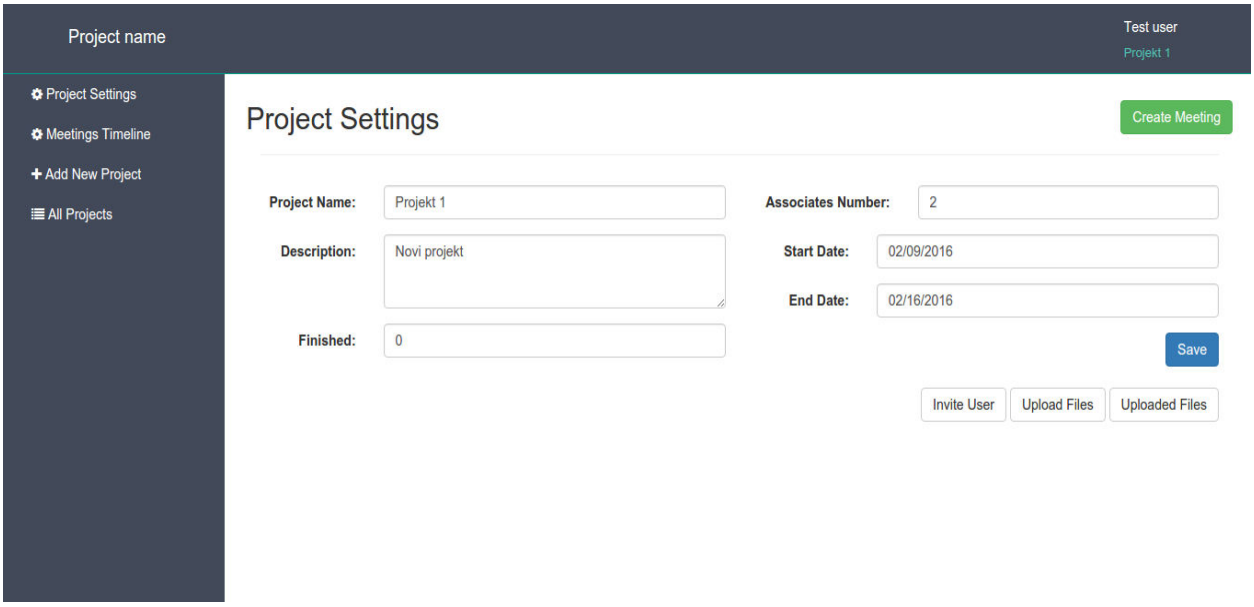


Slika 3.11. Prikaz grešaka u slučaju da korisnik krivo unese podatke.

Nakon što korisnik uspješno kreira novi projekt u lijevom izborniku pojavljuju se još dvije dodatne mogućnosti:

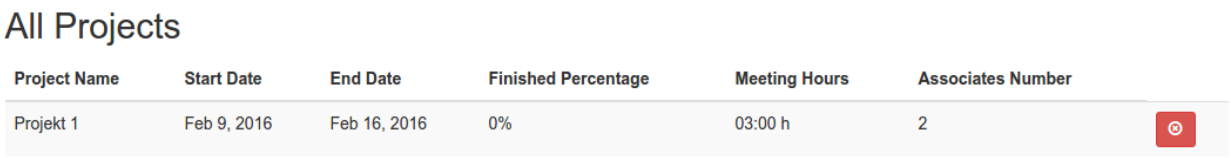
- Praćenje stanja projekta
- Mogućnost izmjena postavki projekta

Ime trenutnog projekta prikazano je ispod korisničkog imena u desnom gornjem kutu (Sl. 3.12). Klikom na *Project Settings* otvara se obrazac sa podacima koje je korisnik unio pri kreiranju projekta. Korisnik ima mogućnost izmjene podataka o projektu, te također ima dodatne mogućnosti dodavanja datoteka, pozivanja novih korisnika u projekt pomoću *email-a*, te kreiranja sastanaka vezanih za projekt (Sl. 3.12.). Dodavanje datoteka izvodi se pomoću *uploadFiles* metode (prilog B), dodane datoteke se spremaju u folder koji dobije naziv ovisno o ID-u projekta. Putanja do datoteka sprema se u bazu zbog lakšeg dohvaćanja.



Slika 3.12. Obrazac za uređivanje projekta.

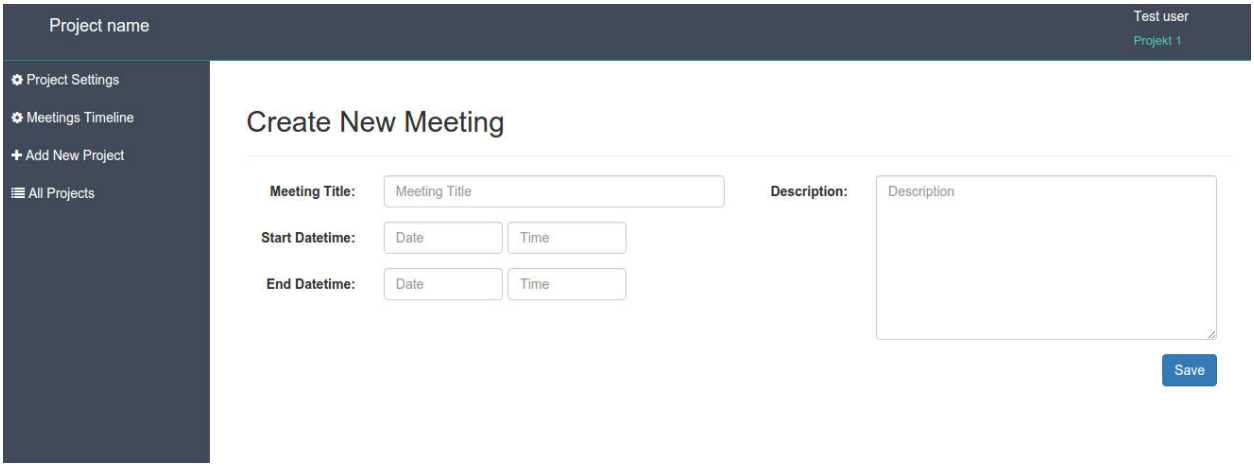
Na slici 3.13. prikazana je lista svih projekata koje je korisnik kreirao. U listi projekata su prikazani svi podaci o projektu, te ukupan broj sati kreiranih sastanaka na pojedinom projektu. Korisnik ima mogućnost brisanja projekta klikom na crveno dugme koje se nalazi na zadnjem mjestu u redu.



Slika 3.13. Lista projekata.

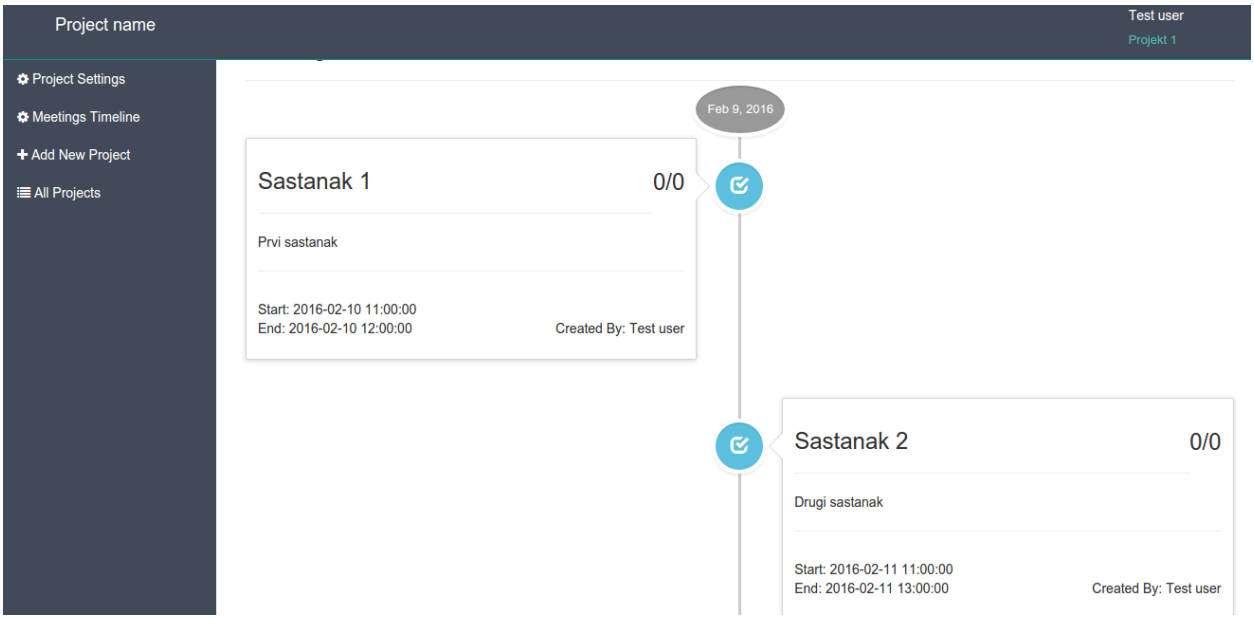
3.3.4. Kreiranje i upravljanje sastancima

Kao što je prikazano na slici 3.12. korisnik ima mogućnost kreiranja sastanaka za projekt. U desnom kutu korisnik može klikom na *Create Meeting* dugme otvoriti formu za kreiranje sastanka (Sl. 3.14.).



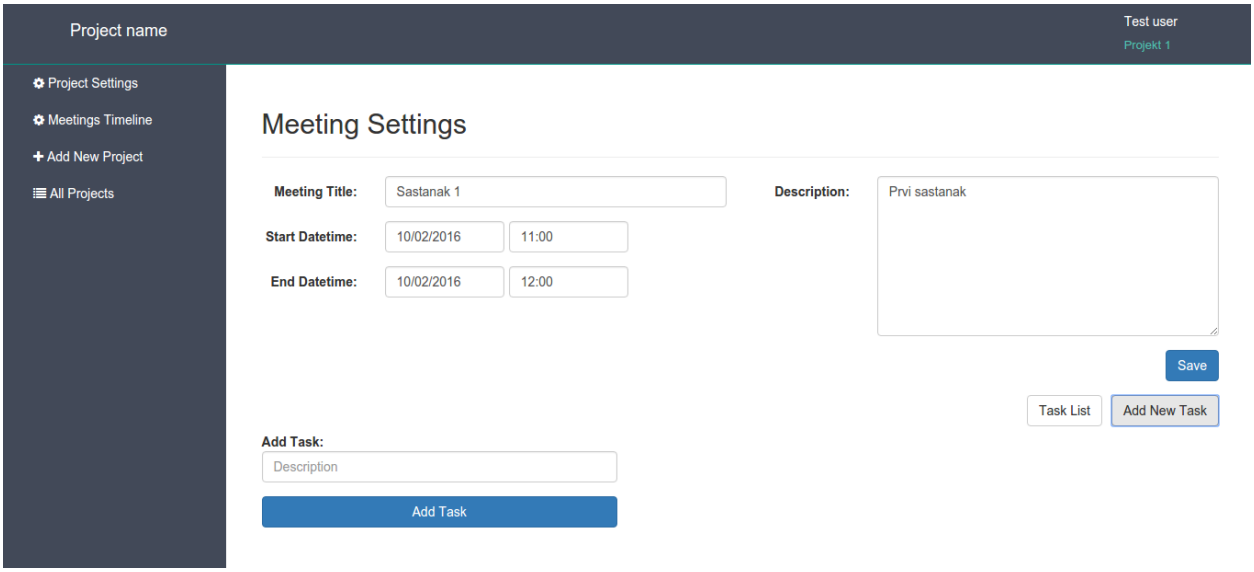
Slika 3.14. Kreiranje novog sastanka.

Kako bi korisnik uspješno kreirao sastanak potrebno je unijeti sve podatke, ako korisnik ne unese sve podatke neće biti u mogućnosti kreirati novi sastanak te će mu se prikazati poruke jednake onima pri kreiranju novog projekta. Nakon što korisnik kreira jedan ili više sastanaka klikom na *Meetings Timeline* može vidjeti vremensku liniju sastanaka (sastanci su poredani po datumu), te pratiti stanje razvijanja projekata kroz sastanke (Sl. 3.15.).



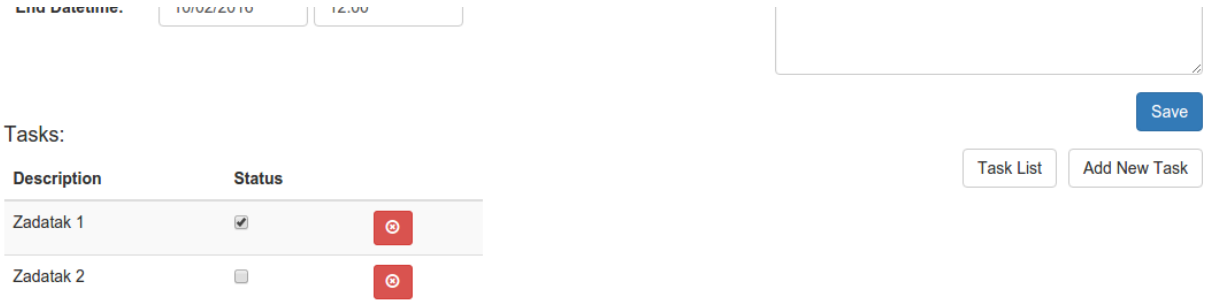
Slika 3.15. Vremenska lista sastanaka za praćenje stanja projekta kroz sastanke.

Svaki sastanak ima poseban oblak u kojem se nalaze informacije o sastanku. Desno od naslova sastanka nalazi se omjer ukupnog broja zadataka i broja zadataka koji su izvršeni. Ako je taj broj jednak, tada je ikona sastanaka obojana plavom bojom, a ako je broj izvršenih zadataka manji od ukupnog broja sastanak tada je ikona sastanaka obojana u žuto. Klikom na ikonu sastanak otvara se prozor gdje korisnik ima mogućnost izmjene podataka o sastanku, te dodavanje ili brisanje zadataka (Sl. 3.16.). Također možemo vidjeti da zbroj trajanja sastanak iznosi 3 sata, što je jednako broju sati u listi projekt pod imenom Projekt 1 (pogledati sliku 3.13.).



Slika 3.16. Postavke kreiranog sastanka.

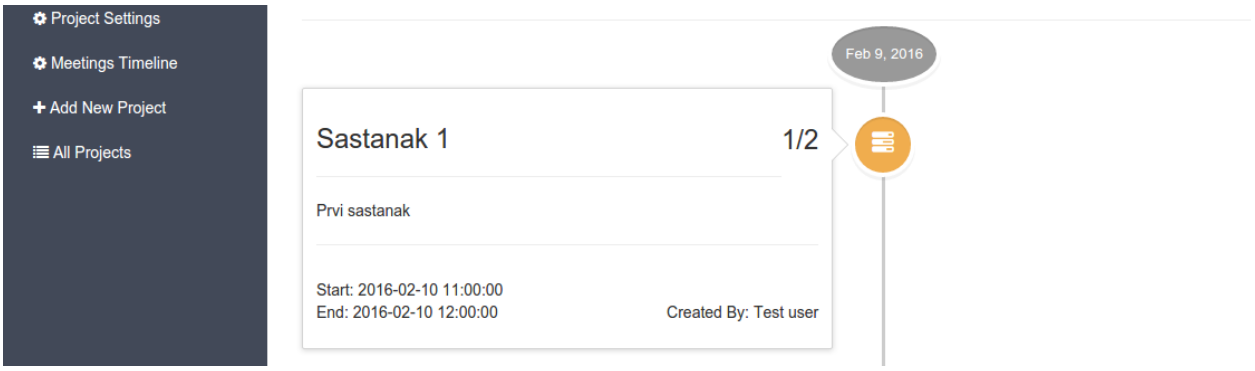
Na slici 3.16. se vidi obrazac za dodavanje zadataka za kreirani sastanak. Nakon što korisnik kreira zadatke, listu zadataka može vidjeti klikom na *Task List* dugme (Sl. 3.17).



Slika 3.17. Lista kreiranih zadataka za sastanak.

U listi zadataka može se vidjeti opis, status i dugme za brisanje zadatka. Ako je osoba uspješno obavila zadataka tada se može označiti *checkbox*. Također se sada na vremenskoj liniji projekta

treba izmijeniti boja sastanka u žutu jer nisu svi zadaci obavljeni, što se može vidjeti na slici 3.18.



Slika 3.18. Omjer obavljenih zadataka i ukupan broj zadataka.

Korisniku je zabranjen unos datuma izvan okvira početnog i krajnjeg datuma projekta. Korisnik ima mogućnost pozivanja drugih korisnika za praćenje stanja projekta, ali pozvani korisnici nemaju mogućnost izmjene podataka projekta. Korisnik može pozvati samo one korisnike koji su već registrirani.

4. ZAKLJUČAK

U današnje vrijeme zbog velike konkurencije javlja se potreba za što bržom i kvalitetnijom izradom projekta. Da bi krajnji rezultat bio što kvalitetniji i bolji potrebno je u svakom trenutku imati mogućnost uvida u trenutno stanje. Najfleksibilnije rješenje moguće je ostvariti pomoću Internet tehnologija. U ovom diplomskom radu izrađena je web aplikacija pomoću koje svi sudionici projekta mogu pratiti stanje projekta kroz sastanke i zadatke koje je potrebno obaviti. Korisniku je omogućeno kroz par klikova mišem kreirati novi projekt, te organizirati sastanke i dodavati zadatke koje treba obaviti za uspješan razvoj projekta. Korisnik kroz korisničko sučelje također može dodavati datoteke potrebne za pojedinu fazu razvoja projekta. Što dovodi do zaključka da je praćenje stanja projekta puno jednostavnije korištenjem jednostavnih alata nego usmenim ili pisanim dogovorom. Zbog toga što je aplikacija napravljena kao aplikacijsko programsko sučelje, moguće unaprjeđenje je izrada mobilne aplikacije koja bi se spajala na server i dohvaćala potrebne podatke za prikaz stanja, tako bi korisnici mogli imati uvid u stanje preko mobilne aplikacije.

LITERATURA

[1] R. Lerdorf, K Tatroe i P. MacIntyre, Programiranje PHP, Drugo izdanje, Dobar Plan, Zagreb, 2009.

[2] Sass http://sass-lang.com/, veljača 2016

[3] Shelly Powers, Naučite JavaScript, Drugo izdanje, Dobar Plan, Zagreb, 2009.

[4] MySQL Workbench https://www.mysql.com/products/workbench/, veljača 2016

[5] JWT https://jwt.io/introduction/

[6] Tino Tkalce, JWT, http://www.toptal.com/web/cookie-free-authentication-with-json-web-tokens-an-example-in-laravel-and-angularjs

SAŽETAK

U ovom diplomskom radu izrađena je aplikacija za praćenje stanja projekta. Razvoj aplikacije podijeljen je u tri dijela: izrada baze podataka, izrada aplikacijskog programskog sučelja i izrada korisničkog sučelja. Rad se sastoji od teorijskog i praktičnog dijela. U teorijskom dijelu opisane su korištene tehnologije i potrebni alati za izradu aplikacije. U radu je poseban naglasak stavljen na razvijanje jednostranične aplikacije koja uz pomoć aplikacijskog programskog sučelja komunicira sa serverom i bazom. Nakon opisa tehnologija i alata prikazano je ostvareno programsko rješenje, te problemi koji su se javljali prilikom izrade aplikacije. Rezultat ovog diplomskog rada je brza i pouzdana Internet aplikacija koja korisniku omogućuje praćenje stanja projekta.

Ključne riječi: programiranje, PHP, JavaScript, jednostranične aplikacije, aplikacijsko programsko sučelje, Laravel, web aplikacija

ABSTRACT

In this master‘s thesis, an application for project status monitoring was designed. The application development was divided into three parts: database construction, making the application programming interface and designing the user interface. The thesis consist of the theoretical and practical part. The theoretical part describes the technology used and tools needed for building the application. In the thesis, special emphasis was put on the development of a single page application, which, with the assistance of the application programming interface, communicates with the server and database. After the technologies and tools have been described, the realised software solution is shown, as well as the problems which occurred during application development. The result of this master’s thesis is a fast and reliable internet application which enables the user to monitor the project status.

Keywords: programming, PHP, JavaScript, Single Page Application, Application programming interface, Laravel, web application

ŽIVOTOPIS

Igor Borovica rođen je 06.06.1990. godine u Beogradu. Nakon rođenja seli se u Županju. U Županji završava osnovnu školu Ivana Kozarca, te upisuje četverogodišnju Tehničku školu u Županji, smjer elektrotehnika. 2009. godine upisuje Elektrotehnički fakultet u Osijeku, smjer računarstvo. Na prvoj godini fakulteta se prvi put susreo s programiranjem. Na drugoj godini fakulteta se počinje ozbiljnije baviti Internet programiranje, te po završetku treće godine počinje raditi kao *freelance* programer. Nakon tri godine rada kao *freelance* programer zapošljava se kao *Software* programer za strani *startup* Aquicore gdje razvija *software* za praćenje potrošnje električne energije, plina i vode u zgradama.

PRILOZI

Izvorni kod aplikacije i svi dodatni materijali nalaze se na DVD-u priloženom uz ovaj rad. U nastavku su dani dijelovi koda za praćenje rada.

Prilog A. Baza.sql

```
-- phpMyAdmin SQL Dump
-- version 4.0.10deb1
-- http://www.phpmyadmin.net
--
-- Računalo: localhost
-- Vrijeme generiranja: Velj 08, 2016 u 01:42 PM
-- Verzija poslužitelja: 5.5.47-0ubuntu0.14.04.1
-- PHP verzija: 5.5.9-1ubuntu4.14

SET SQL_MODE = "NO_AUTO_VALUE_ON_ZERO";
SET time_zone = "+00:00";



/*!40101 SET @OLD_CHARACTER_SET_CLIENT=@@CHARACTER_SET_CLIENT */;
/*!40101 SET @OLD_CHARACTER_SET_RESULTS=@@CHARACTER_SET_RESULTS */;
/*!40101 SET @OLD_COLLATION_CONNECTION=@@COLLATION_CONNECTION */;
/*!40101 SET NAMES utf8 */;

--
-- Baza podataka: `diplomski`
--

-- --------------------------------------------------------

--
-- Tablična struktura za tablicu `meetings`
--

CREATE TABLE IF NOT EXISTS `meetings` (
  `id` int(10) unsigned NOT NULL AUTO_INCREMENT,
  `user_id` int(10) unsigned NOT NULL,
```

```
  `project_id` int(10) unsigned NOT NULL,
  `finished` tinyint(1) NOT NULL DEFAULT '0',
  `meeting_start` timestamp NOT NULL DEFAULT '0000-00-00 00:00:00',
  `meeting_end` timestamp NOT NULL DEFAULT '0000-00-00 00:00:00',
  `title` varchar(255) COLLATE utf8_unicode_ci NOT NULL,
  `description` longtext COLLATE utf8_unicode_ci NOT NULL,
  `created_at` timestamp NOT NULL DEFAULT '0000-00-00 00:00:00',
  `updated_at` timestamp NOT NULL DEFAULT '0000-00-00 00:00:00',
  PRIMARY KEY (`id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8 COLLATE=utf8_unicode_ci
AUTO_INCREMENT=9 ;

-- --------------------------------------------------------

--
-- Tablična struktura za tablicu `meeting_tasks`
--

CREATE TABLE IF NOT EXISTS `meeting_tasks` (
  `id` int(10) unsigned NOT NULL AUTO_INCREMENT,
  `meeting_id` int(11) NOT NULL,
  `description` varchar(255) COLLATE utf8_unicode_ci NOT NULL,
  `status` tinyint(1) NOT NULL DEFAULT '0',
  `created_at` timestamp NOT NULL DEFAULT '0000-00-00 00:00:00',
  `updated_at` timestamp NOT NULL DEFAULT '0000-00-00 00:00:00',
  PRIMARY KEY (`id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8 COLLATE=utf8_unicode_ci
AUTO_INCREMENT=8 ;

-- --------------------------------------------------------

--
-- Tablična struktura za tablicu `projects`
--

CREATE TABLE IF NOT EXISTS `projects` (
```

```
  `id` int(10) unsigned NOT NULL AUTO_INCREMENT,
  `project_name` varchar(255) COLLATE utf8_unicode_ci NOT NULL,
  `description` longtext COLLATE utf8_unicode_ci NOT NULL,
  `finished_percentage` int(11) NOT NULL,
  `associates_number` int(11) NOT NULL,
  `created_by` varchar(255) COLLATE utf8_unicode_ci NOT NULL,
  `start_date` date NOT NULL,
  `end_date` date NOT NULL,
  `created_at` timestamp NOT NULL DEFAULT '0000-00-00 00:00:00',
  `updated_at` timestamp NOT NULL DEFAULT '0000-00-00 00:00:00',
  PRIMARY KEY (`id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8 COLLATE=utf8_unicode_ci
AUTO_INCREMENT=4 ;

-- --------------------------------------------------------

--
-- Tablična struktura za tablicu `project_files`
--

CREATE TABLE IF NOT EXISTS `project_files` (
  `id` int(10) unsigned NOT NULL AUTO_INCREMENT,
  `project_id` int(11) NOT NULL,
  `name` varchar(255) COLLATE utf8_unicode_ci NOT NULL,
  `path` varchar(255) COLLATE utf8_unicode_ci NOT NULL,
  `created_at` timestamp NOT NULL DEFAULT '0000-00-00 00:00:00',
  `updated_at` timestamp NOT NULL DEFAULT '0000-00-00 00:00:00',
  PRIMARY KEY (`id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8 COLLATE=utf8_unicode_ci
AUTO_INCREMENT=1 ;

-- --------------------------------------------------------

--
-- Tablična struktura za tablicu `project_phase`
--
```

```
CREATE TABLE IF NOT EXISTS `project_phase` (
  `id` int(10) unsigned NOT NULL AUTO_INCREMENT,
  `project_id` int(11) NOT NULL,
  `user_id` int(11) NOT NULL,
  `title` varchar(255) COLLATE utf8_unicode_ci NOT NULL,
  `description` longtext COLLATE utf8_unicode_ci NOT NULL,
  `start` date NOT NULL,
  `end` date NOT NULL,
  `created_at` timestamp NOT NULL DEFAULT '0000-00-00 00:00:00',
  `updated_at` timestamp NOT NULL DEFAULT '0000-00-00 00:00:00',
  PRIMARY KEY (`id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8 COLLATE=utf8_unicode_ci
AUTO_INCREMENT=1 ;

-- --------------------------------------------------------

--
-- Tablična struktura za tablicu `project_user`
--

CREATE TABLE IF NOT EXISTS `project_user` (
  `project_id` int(10) unsigned NOT NULL,
  `user_id` int(10) unsigned NOT NULL,
  `role` enum('ADMIN','INVITED') COLLATE utf8_unicode_ci NOT NULL,
  `created_at` timestamp NOT NULL DEFAULT '0000-00-00 00:00:00',
  `updated_at` timestamp NOT NULL DEFAULT '0000-00-00 00:00:00',
  KEY `project_user_project_id_index` (`project_id`),
  KEY `project_user_user_id_index` (`user_id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8 COLLATE=utf8_unicode_ci;

-- --------------------------------------------------------

--
-- Tablična struktura za tablicu `users`
--
```

```
CREATE TABLE IF NOT EXISTS `users` (
  `id` int(10) unsigned NOT NULL AUTO_INCREMENT,
  `current_project_id` int(10) unsigned DEFAULT NULL,
  `username` varchar(255) COLLATE utf8_unicode_ci NOT NULL,
  `email` varchar(255) COLLATE utf8_unicode_ci NOT NULL,
  `first_name` varchar(255) COLLATE utf8_unicode_ci NOT NULL,
  `last_name` varchar(255) COLLATE utf8_unicode_ci NOT NULL,
  `image_url` varchar(255) COLLATE utf8_unicode_ci NOT NULL,
  `password` varchar(60) COLLATE utf8_unicode_ci NOT NULL,
  `remember_token` varchar(100) COLLATE utf8_unicode_ci DEFAULT NULL,
  `created_at` timestamp NOT NULL DEFAULT '0000-00-00 00:00:00',
  `updated_at` timestamp NOT NULL DEFAULT '0000-00-00 00:00:00',
  PRIMARY KEY (`id`),
  UNIQUE KEY `users_email_unique` (`email`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8 COLLATE=utf8_unicode_ci
AUTO_INCREMENT=3 ;

-- --------------------------------------------------------

--
-- Tablična struktura za tablicu `users_invitations`
--

CREATE TABLE IF NOT EXISTS `users_invitations` (
  `id` int(10) unsigned NOT NULL AUTO_INCREMENT,
  `invited_by` int(11) NOT NULL,
  `project_id` int(11) NOT NULL,
  `invited_by_username` varchar(255) COLLATE utf8_unicode_ci NOT NULL,
  `project_name` varchar(255) COLLATE utf8_unicode_ci NOT NULL,
  `status` enum('NOT_ACCEPTED','ACCEPTED','DENIED') COLLATE utf8_unicode_ci NOT NULL,
  `invited_user` varchar(255) COLLATE utf8_unicode_ci NOT NULL,
  `created_at` timestamp NOT NULL DEFAULT '0000-00-00 00:00:00',
  `updated_at` timestamp NOT NULL DEFAULT '0000-00-00 00:00:00',
  PRIMARY KEY (`id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8 COLLATE=utf8_unicode_ci
```

```
AUTO_INCREMENT=1 ;


--
-- Ograničenja za izbačene tablice
--


--
-- Ograničenja za tablicu `project_user`
--
ALTER TABLE `project_user`
  ADD CONSTRAINT `project_user_project_id_foreign` FOREIGN KEY (`project_id`) REFERENCES `projects` (`id`) ON DELETE CASCADE,
  ADD CONSTRAINT `project_user_user_id_foreign` FOREIGN KEY (`user_id`) REFERENCES `users` (`id`) ON DELETE CASCADE;

/*!40101 SET CHARACTER_SET_CLIENT=@OLD_CHARACTER_SET_CLIENT */;
/*!40101 SET CHARACTER_SET_RESULTS=@OLD_CHARACTER_SET_RESULTS */;
/*!40101 SET COLLATION_CONNECTION=@OLD_COLLATION_CONNECTION */;
```

Prilog B. ProjectsController.php

```
<?php

namespace App\Http\Controllers\Project;

use App\Http\Controllers\Controller;
use App\ProjectFiles;
use Illuminate\Support\Facades\File;
use Illuminate\Http\Request;
use App\Http\Requests\ProjectsFormRequest;
use App\Repositories\Contracts\User\UserInterface as User;
use App\Repositories\Contracts\Project\ProjectInterface as Project;

class ProjectsController extends Controller {

    protected $project;
```

```
protected $user;

public function __construct(Project $project, User $user) {
    $this->project = $project;
    $this->user = $user;
}

public function get($id) {
    return $this->project->get($id);
}

public function post($userId, ProjectsFormRequest $request) {
    $input = $request->all();
    $user = $this->user->get($userId);

    if ($project = $this->project->create($user, $input)) {
        return $project;
    }

    return response()->json(['error' => 'There was a problem!', 500]);
}

public function put($projectId, ProjectsFormRequest $request) {
    $input = $request->all();

    if ($project = $this->project->update($projectId, $input)) {
        return $project;
    }

    return response()->json(['error' => 'There was a problem!', 500]);
}

public function delete($userId, $projectId) {
    $this->project->delete($userId, $projectId);
    return response()->json(['success' => 'Deleted successfully!', 200]);
}
```

```
    public function uploadFiles($projectId, Request $request) {
        $path = 'files/projects/' . $projectId;
        $destinationPath = public_path($path);

        if (!File::exists($destinationPath)) {
            File::makeDirectory($destinationPath, 0777, true);
        }

        foreach ($request->file('files') as $file) {
            $fileName = $file->getClientOriginalName();
            $data['project_id'] = $projectId;
            $data['name'] = $fileName;
            $data['path'] = $path . '/' . $fileName;

            if (ProjectFiles::create($data)) {
                $file->move($destinationPath, $fileName);
            }
        }
    }

    public function getUploadedFiles($projectId) {
        $files = $this->project->get($projectId)->files()->get()->toArray();
        return $files;
    }

}
```

Prilog C. Project.php

```
<?php

namespace App;

use Illuminate\Database\Eloquent\Model;

class Project extends Model {
```

```
    protected $table = 'projects';

    protected $casts = [
        'associates_number' => 'integer',
        'finished_percentage' => 'integer'
    ];

    protected $fillable = [
        'project_name', 'start_date', 'end_date', 'project_logo',
        'associates_number', 'created_by', 'description', 'finished_percentage'
    ];

    public function users() {
        return $this->belongsToMany('App\User')
                    ->withPivot('role')
                    ->withTimestamps();
    }

    public function invitations() {
        return $this->hasMany('App\Invitation');
    }

    public function files() {
        return $this->hasMany('App\ProjectFiles');
    }

    public function meetings() {
        return $this->hasMany('App\Meeting');
    }
}
```

Prilog D. CreateProjectCtrl.js

```
function CreateProjectCtrl(ProjectService, $state, Messages, UserService) {

    var vm = this;
```

```
    vm.projectData = {};
    vm.user = UserService.user;

    vm.createProject = function() {
        vm.projectData.created_by = vm.user.email;
        ProjectService.createProject(vm.user, vm.projectData).then(function(project) {
            vm.user.projects.push(project);
            UserService.changeCurrentProject(vm.user, project.id).then(function() {
                $state.go('di.main.home');
                Messages.success('Your project is created successfully');
            });
        }).catch(function(error) {
            vm.validationErrors = error.data;
            Messages.error("Check required fields!");
        });
    }
}
```