Moderne tehnologije u izradi sučelja Web aplikacija na primjeru poslovne aplikacije

Špoljarić, Mihaela

Undergraduate thesis / Završni rad

2017

Degree Grantor / Ustanova koja je dodijelila akademski / stručni stupanj: **Josip Juraj Strossmayer University of Osijek, Faculty of Electrical Engineering, Computer Science and Information Technology Osijek / Sveučilište Josipa Jurja Strossmayera u Osijeku, Fakultet elektrotehnike, računarstva i informacijskih tehnologija Osijek**

Permanent link / Trajna poveznica: https://urn.nsk.hr/urn:nbn:hr:200:561672

Rights / Prava: In copyright / Zaštićeno autorskim pravom.

Download date / Datum preuzimanja: **2024-03-29**

Repository / Repozitorij:

Faculty of Electrical Engineering, Computer Science and Information Technology Osijek

zir.nsk.hr

Image not found or type unknown

SVEUČILIŠTE JOSIPA JURJA STROSSMAYERA U OSIJEKU
ELEKTROTEHNIČKI FAKULTET

Stručni studij

Moderne tehnologije u izradi sučelja Web aplikacija na primjeru poslovne aplikacije

Završni rad

Mihaela Špoljarić

Osijek, 2017.

Sadržaj rada

1. UVOD 3
1.1. Zadatak i struktura rada 3
1.2. Povijest PHP-a 3
2. KORIŠTENE TEHNOLOGIJE 4
2.1. Laravel 4
2.2. JavaScript + Ajax 4
2.3. MySQL 5
2.4. Bootstrap 5
2.5. Docker 6
2.6. TravisCI 6
3. OPIS APLIKACIJE ZA PRAĆENJE ZAPOSLENIKA 7
3.1. Entitet/veze, atributi baze podataka 7
3.2. ER dijagram 8
3.3. Korisničko sučelje 9
3.4. Nadzorna ploča 9
3.5. Pregled zaposlenika 10
3.6. Pregled zadataka 11
3.7. Pregled dokumenata 12
3.8. Sučelje zaposlenika 13
3.9. Dijagram uvida upotrebe aplikacije 13
4. IMPLEMENTACIJA APLIKACIJE ZA PRAĆENJE ZAPOSLENIKA 14
4.1. Fizičko oblikovanje 14
4.2. Funkcija dodavanje novog zaposlenika 15
4.3. Funkcija dodavanja novog zadatka 16
4.4. Funkcija dodavanja novog dokumenta 19
5. UPRAVLJANJE ISPORUKOM I NEPREKIDNA INTEGRACIJA APLIKACIJE 22
5.1. Docker 22
5.2. TravisCI 24
6. ZAKLJUČAK 25
LITERATURA 26
SAŽETAK 27
ABSTRACT 28
ŽIVOTOPIS 29
PRILOZI 30

1. UVOD

Cilj ovog završnog rada je istražiti korištenje modernih tehnologija izradom web aplikacije za praćenje zaposlenika. Prvi korak uključuje izradu data modela i baze podataka na temelju data modela, zatim određivanje željene funkcionalnosti web aplikacije i na kraju implementacija backenda i frontenda. Za razvoj web aplikacije korištene su moderne tehnologije: Laravel koji je baziran na PHP-u, JavaScript + Ajax, MySQL, Bootstrap, Docker i TravisCI. U prvom dijelu rada opisane su sve tehnologije i način na koje su primijenjene u razvoju aplikacije. U drugom dijelu rada prikazana je sama aplikacija i dio koda pomoću kojeg su napravljene funkcionalnosti, a također je opisana i svrha Docker-a i TravisCI-a.

1.1. Zadatak i struktura rada

Zadatak završnog rada je objasniti korištene moderne tehnologije i zašto su baš one odabrane za izradu te koja je njihova namjena u razvoju web aplikacije.

1.2. Povijest PHP-a

Puni naziv PHP-a glasi *Hypertext Preprocessor*. Prva verzija ovog popularnog programskog jezika nazivala se PHP/FI (Personal Home Page Tools/Forms Interpreter) te je nastala 1995. godine, a razvio ju je Rasmus Lerdorf. Prva verzija je bila skup Perl skripti i koristila se za brojanje posjeta na web stranici. Kako tehnologija zahtijeva stalni razvoj, PHP je dobio svoju novu verziju razvijenu pomoću programskog jezika C. Nova verzija PHP-a mogla je raditi s bazama podataka, a korisnicima je također omogućena izrada jednostavnih dinamičnih web stranica. Nakon toga Rasmus Lerdorf odlučio je objaviti PHP kao slobodan softver kako bi ga ostali korisnici mogli razvijati.

PHP programski jezik je namijenjen programiranju dinamičkih web stranica. Ono što razlikuje PHP od npr. JavaScripta, gdje se kod izvršava na klijentskoj strani – pregledniku, je to što se kod izvršava na poslužitelju te generira HTML koji se zatim šalje klijentu. Tako klijent prima rezultate pokretanja skripte, ali ne vidi izvorni kod.

2. KORIŠTENE TEHNOLOGIJE

Tehnologije korištene za izradu ove web aplikacije su: Laravel, JavaScript + Ajax, MySQL, Bootstrap, Docker, CircleCI.

2.1. Laravel

Laravel je web okvir koji je baziran je na PHP-u te je nastao s ciljem lakšeg razvoja web aplikacija pomoću MVC arhitekture (model-view-controller). MVC arhitektura je razdvajanje web aplikacija na dijelove ovisno o njihovoj namjeni. Model se sastoji od podataka, poslovnih pravila, logike i funkcija. View ili pogled sastoji se od prikaza podataka kao što su obrazac, dijagram ili tablica. Također, moguće ih je prikazati upotrebom nekoliko različitih pogleda. Controller ili upravitelj prihvaća ulazne podatke i pretvara ih u naredbe koji se prosljeđuje na model ili pogled.

Laravel se koristi zbog toga što je zadatak ovog završnog rada bio predstaviti stariji programski jezik pomoću modernih tehonologija, a Laravel je jedan od poznatijih web framework-a baziran na PHP-u te ga se sve više koristi za web razvoj.

2.2. JavaScript + Ajax

JavaScript programski je jezik koji se izvršava u web pregledniku na strani korisnika, a koristi se za izradu interaktivnih web stranica. Uz HTML i CSS, JavaScript je jedan od temeljnih tehnologija za izradu web razvoja.

Ajax (engl. *Asynchronous JavaScript And XML*) koristi kombinaciju JavaScript i HTML za prikaz ili korištenje podataka te ugrađeni XMLHttpRequest u preglednik za zahtijevanje podataka od web poslužitelja. Ajax nam dopušta ažuriranje web stranica bez da ju moramo ponovo učitati, odnosno Ajax dopušta ažuriranje web stranice asinkrono razmjenom podataka s web poslužitelja iza scene.

U ovom završnom radu korištena je kombinacija JavaScripta + Ajaxa za dodavanje zaposlenika, dokumenata i zadataka.

2.3. MySQL

Baza podataka (engl. *Database*) je zbirka organiziranih zapisa koja je pohranjena u računalo na sustavan način. Njezina namjena je pohrana, analiza i pretraživanje grupe srodnih i povezanih podataka. Navedeni podaci mogu biti na primjeru ove web aplikacije, podaci o zaposlenicima, njihove adrese, status.

Svaka baza podataka razvijena je na temelju tablica koje se sastoje od entiteta i atributa. Entitet može biti stvar ili objekt u kojemu skupljamo podatke. Atribut je taj koji opisuje tu stvar, objekt, odnosno služi da se taj atribut pobliže identificira, odredi. Baze podataka mogu sadržavati i sheme, upite, poglede i druge objekte koji omogućavaju upravljanje podacima.

Baza podataka za ovu aplikaciju kreirana je pomoću MySQL jezika, a bazom podataka se upravlja pomoću sustava phpMyAdmin.

MySQL (engl. *My Structured Query Language*) sustav je za upravljanje bazom podataka. MySQL baze su relacijskog tipa, a taj tip pokazao se najboljim izborom kod baza s velikom količinom podataka. Relacijski tip baza podataka omogućava brzo pretraživanje podataka. Relacijske baze raspoređuju podatke po tablicama umjesto da te podatke stavljaju u jedan prostor. Struktura baze podataka raspoređena je u fizičke datoteke.

2.4. Bootstrap

Bootstrap je front-end okvir za dizajniranje web stranica i web aplikacija. Sadrži HTML, CSS i JavaScript predefinirane predloške koje su dizajnirane za pomoć u izgradnji komponenti korisničkog sučelja. Umjesto da pišemo CSS klase, jednostavno se može uključiti Bootstrap datoteka i spomenuti Bootstrap predefinirane nazive klasa za naše elemente na web stranici ili aplikaciji i bit će automatski oblikovane pomoću Bootstrapa.

U ovom završnom radu Bootstrap je korišten kako bi se iskoristila još jedna od modernih tehnologija.

2.5. Docker

Docker je platforma kontejnerskog softvera. Razvojni programeri mogu koristiti Docker za pakiranje, isporuku i pokretanje aplikacije kao lagane, prijenosne kontejnere koji mogu raditi bilo gdje. Docker nam omogućava postavljanje lokalnih razvojnih okruženja koja su baš poput izvornog poslužitelja, pokretanje više razvojnih okruženja iz istog izvora koji imaju svaki jedinstveni softver, operacijske sustave i konfiguracije, testne projekte na novim ili različitim poslužiteljima i dopuštaju da svatko može raditi na istom projektu s istim postavkama, bez obzira na lokalno okruženje domaćina.

Pod izrazom platforma kontejnerskog softvera misli se na način pakiranja softvera u obliku koji se može izvoditi izoliranim na dijeljenom operativnom sustavu. Kontejneri ne zahtijevaju puni operativni sustav, nego samo biblioteke i postavke koje su potrebne za izradu softvera.

2.6. TravisCI

TravisCi je softver koja nam omogućava proces razvoja softvera pomoću kontinuirane integracije (engl. Continuous Integration) i kontinuirane isporuke (engl. Continuous Deployment).

Kontinuirana integracija je razvojna praksa koja zahtjeva integraciju nakon svake promjene koda više puta dnevno. Nakon svake integracije softver se provjerava automatiziranim testovima, radi otkrivanja pogrešaka integracije. Cilj kontinuirane integracije je brzo identificiranje pogrešaka u kodu, što ujedno omogućava i njihovo brzo ispravljanje.

3. OPIS APLIKACIJE ZA PRAĆENJE ZAPOSLENIKA

Funkcionalnosti koje web aplikacija mora sadržavati su: dodavanje zaposlenika, uređivanje podataka o zaposlenicima, dodavanje novih zadataka, dodjeljivanje zadataka, dodavanje potrebnih dokumenata te mogućnost da se podsjeti određenog zaposlenika koji dokument mora dostaviti.

3.1. Entitet/veze, atributi baze podataka

Tablica 3.1. *Entiteti/veze i atributi*

ENTITETI/VEZE	ATRIBUTI
documents	id, name
employee_tasks	id, employee_id, task_id, done
employees	id, user_id, status_id, birth_date, address, employment_date
migrations	id, migration, batch
password_resets	email, token, created_at
permission_role	permission_id, role_id
permissions	id, name, display_name, description, created_at, updated_at
required_documents	id, employee_id, document_id, delivered
role_user	user_id, role_id
roles	id, name, display_name, description, created_at, updated_at
statuses	id, name
tasks	id, name, deadline
users	id, first_name, last_name, email, password, created_at, updated_at

U bazi podataka pohranjuju se podaci koji se odnose na dokumente, zaposlenike, njihove zadatke, dokumente koje moraju dostaviti, uloge, statuse, zadatke i korisnike.

Za svakog korisnika bitno je definirati njegov id, npr. admin ili samo korisnik, nakon toga njegovo ime i prezime, e-poštu, lozinku, kada je profil korisnika kreiran i kada je ažuriran.

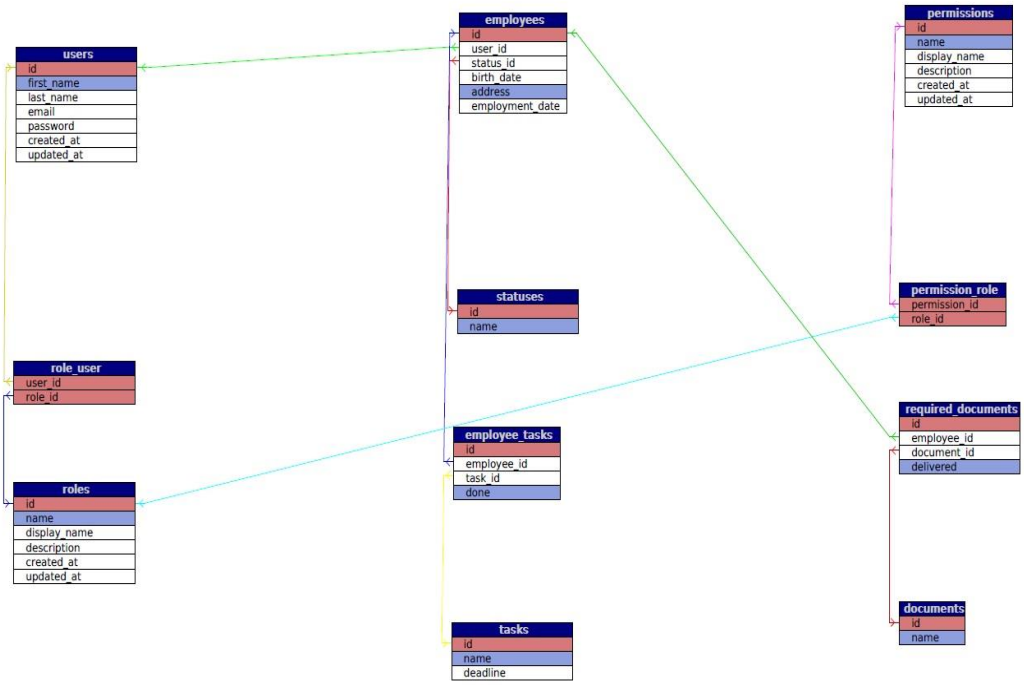
Za svakog zaposlenika potrebno je definirati njegovo ime, prezime, datum rođenja, adresu, e-poštu, lozinku, status (npr. frontend developer, web dizajner, backend developer) i datum zapošljavanja.

Za dodavanje novog zadatka bitno je definirati naziv zadatka i rok za izvršavanje.

Za dodavanje nove vrste dokumenta bitno je definirati naziv dokumenta.

Administrator je osoba npr. CEO tvrtke koji ima pristup svim podacima. Također, administrator je osoba koja dodaje nove zaposlenike, ispunjava podatke o njima, dodaje nove zadatke i dodjeljuje ih određenom zaposleniku, a dodaje i podatke o dokumentima koji moraju biti dostavljeni i dodjeljuje koji zaposlenik to mora obaviti te ima uvid u popis svih zaposlenika.

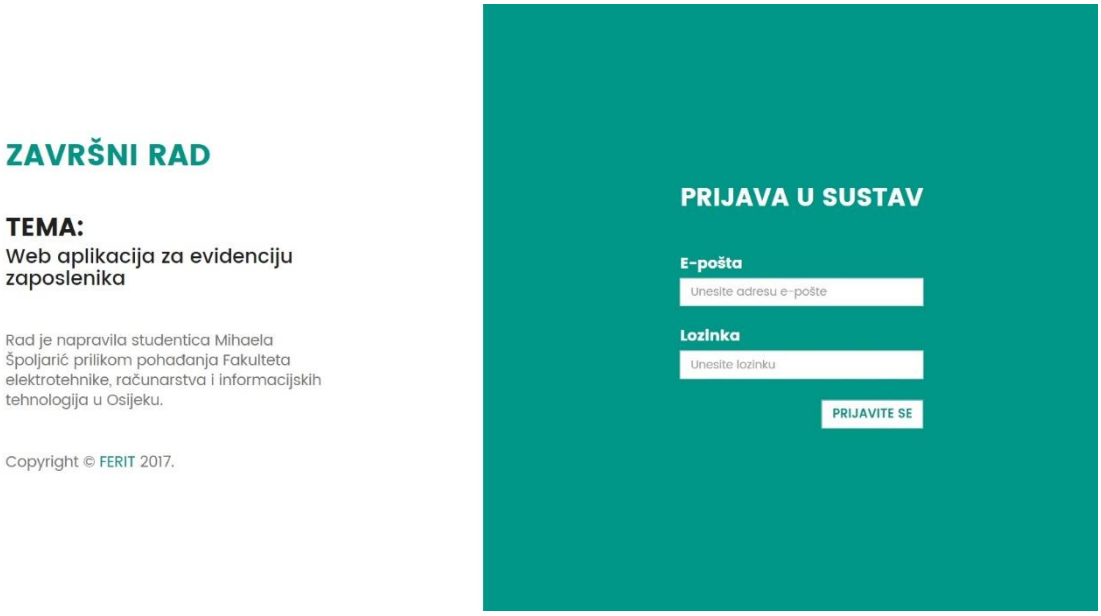
3.2. ER dijagram



Sl 3.1. *ER dijagram*

Na temelju *Tablice 3.1.* napravljena je relacijska shema baze podataka koja je predstavljena slikom *3.1.*

3.3. Korisničko sučelje



Sl 3.2. *Korisničko sučelje*

Korisničko sučelje napravljeno je pomoću Laravela, Bootstrapa. Sastoji se od samo jednog dijela na kojemu se nalazi prijava u sustav te gdje se administrator ili zaposlenik mogu prijaviti u sustav pomoću e-pošte i lozinke. Korisničko sučelje možemo vidjeti na slici *3.2.*

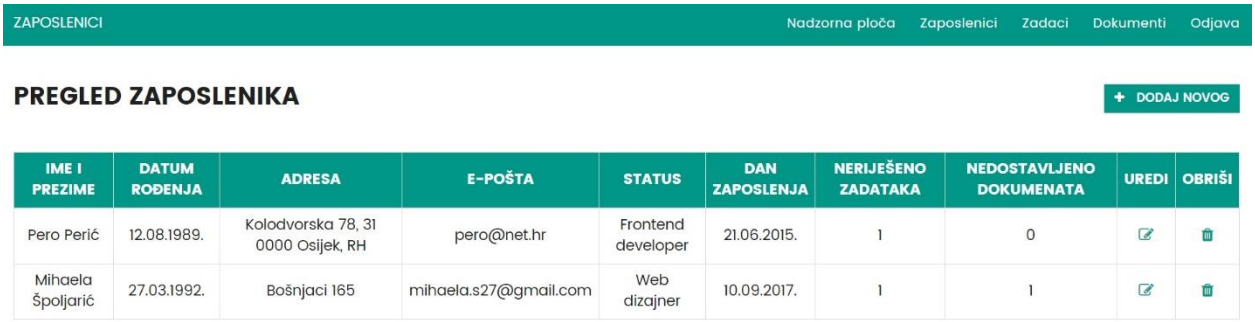
3.4. Nadzorna ploča



Sl 3.3. *Nadzorna ploča administratora*

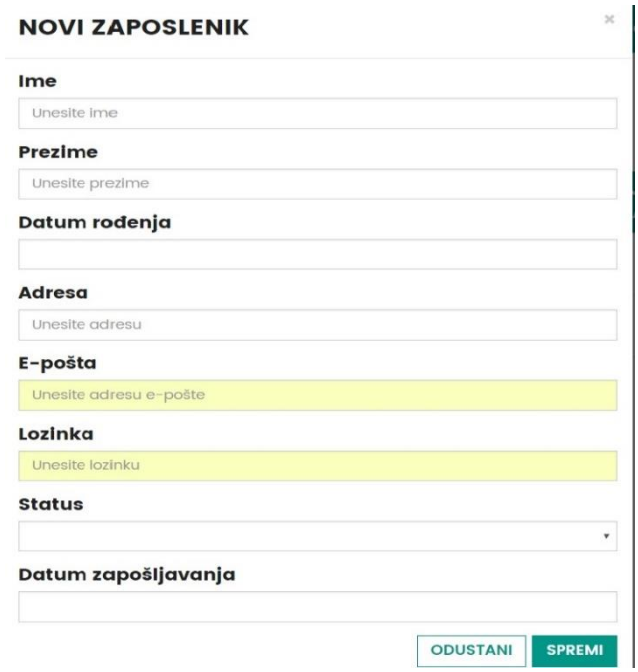
Nadzorna ploča administratora vidi se na slici *3.3.*, a napravljena je pomoću Laravel, Bootstrap-a. Sadrži pregled statistike, koliko ima sveukupno zaposlenika, zadataka, neriješenih zadataka te koliko dokumenata nedostaje. Uz navedeno, sadrži i pregled neriješenih zadataka.

3.5. Pregled zaposlenika



Sl 3.4. *Pregled zaposlenika*

Sučelje pregleda zaposlenika vidi se na slici *3.4.*, a sadrži uvid u već dodane zaposlenike te se ovdje može i kreirati novi zaposlenik. Sučelje za dodavanje novog zaposlenika može se vidjeti na slici *3.5.* Dodavanje novog zaposlenika može napraviti samo administrator koji unosi ime, prezime, datum rođenja, adresu, e-poštu, lozinku, status i datum zapošljavanja zaposlenika. Navedenim podacima zaposlenik se može prijaviti u sustav.



Sl 3.5. *Dodavanje novog zaposlenika*

3.6. Pregled zadataka



Sl 3.6. *Sučelje pregleda zadataka*

Na slici *3.6.* može se vidjeti pregled zadataka, zadani zadaci, koji je rok izvršavanja, pregled dodijeljenih zadataka, kome su zadaci dodijeljeni, koji je rok izvršavanja, a administrator može označiti kvačicu izvršeno ili obrisati zadatak.

Dodavanje novog zadatka i dodjeljivanje novog zadatka može se vidjeti na slici *3.7.* Kod novog zadatka administrator mora unijeti naziv zadatka te odabrati rok za izvršenje. Kod dodjeljivanja zadatka zaposleniku administrator mora odabrati kojem zaposleniku želi dodati određeni zadatak.



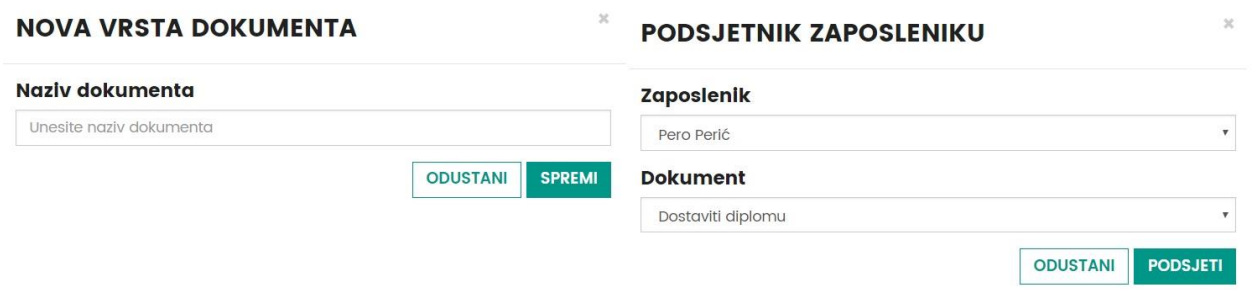
Sl 3.7. *Sučelje novog zadatka i dodjeljivanja zadatka zaposleniku*

3.7. Pregled dokumenata



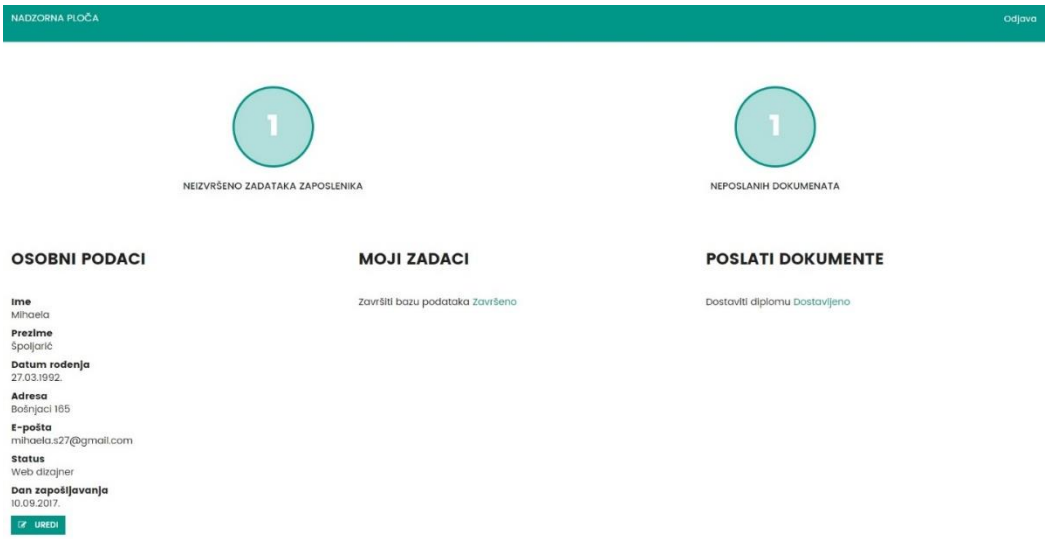
Sl 3.8. *Sučelje pregleda dokumenata*

Slika *3.8.* prikazuje pregled dokumenata, naziv dokumenta, a administrator može označiti kvačicu svi prikupljeni dokumenti te može obrisati naziv dokumenta. Također, na slici *3.9.* može se vidjeti kako izgleda dodavanje novog dokumenta i podsjetnik zaposleniku. Na sučelju nova vrsta dokumenta administrator mora upisati naziv dokumenta, a na sučelju podsjetnik zaposleniku može odabrati koji zaposlenik mora dostaviti koji dokument.



Sl 3.9. *Sučelje nove vrste dokumenta i podsjetnik zaposleniku*

3.8. Sučelje zaposlenika

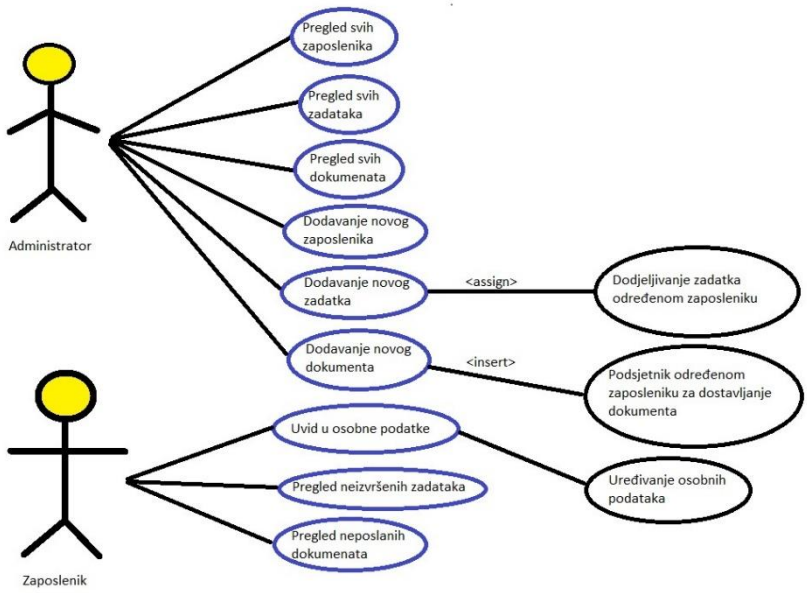


Sl 3.10. *Sučelje zaposlenika*

Nakon prijave zaposlenika u sustav, zaposlenik dobiva informacije o neizvršenim zadacima i dokumentima koje treba dostaviti te uvid u svoje osobne podatke koje ima pravo uređivati. Također, ima pravo promijeniti status zadatka u završeno, a za dostavljene dokumente ima pravo promijeniti status u završeno.

3.9. Dijagram uvida upotrebe aplikacije

Na slici *3.11.* je prikazan dijagram uvida upotrebe aplikacije, koje pristupe imaju korisnici, odnosno u čega imaju uvid



Sl 3.11. *Dijagram uvida upotrebe aplikacije*

4. IMPLEMENTACIJA APLIKACIJE ZA PRAĆENJE ZAPOSLENIKA

4.1. Fizičko oblikovanje

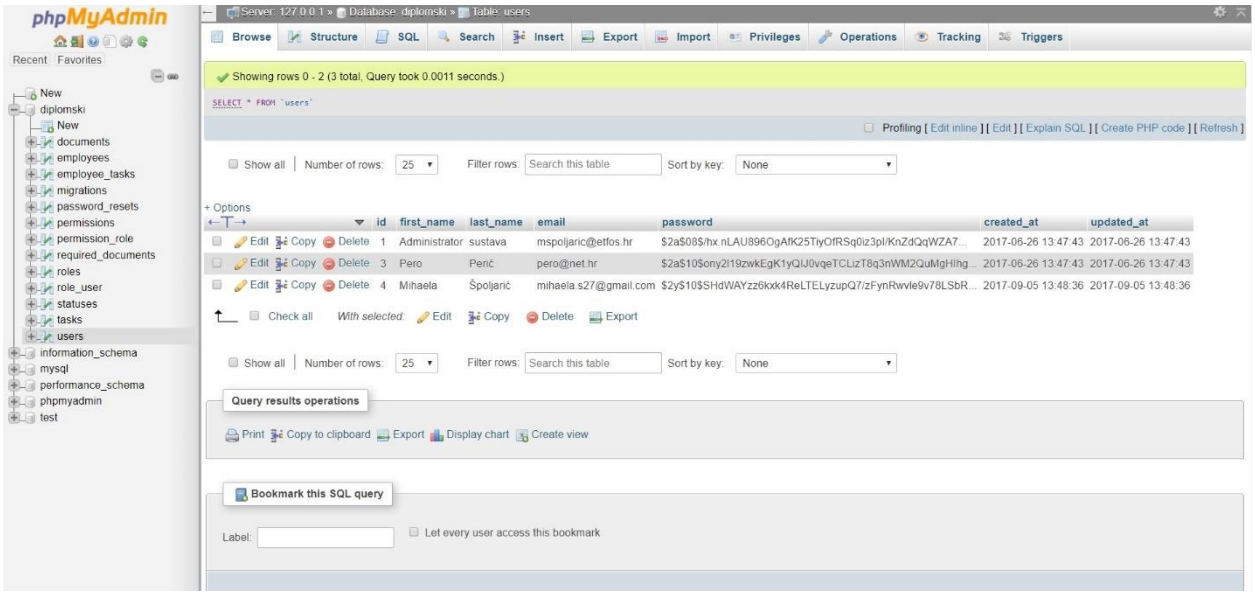
Za kreiranje tablica korišten je MySQL. Na slici *4.12.* može se vidjeti kako izgleda kreiranje tablice zaposlenika gdje se pomoću naredbe CREATE TABLE definira naziv tablice 'employees'. Na slici *4.13.* definiran je 'id' kao primarni ključ tablice 'employees'. Na slici *4.14.* je prikazano sučelje phpmyadmin, gdje se vide sve tablice.

```
CREATE TABLE `employees` (
  `id` int(10) UNSIGNED NOT NULL,
  `user_id` int(10) UNSIGNED NOT NULL,
  `status_id` int(10) UNSIGNED NOT NULL,
  `birth_date` date NOT NULL,
  `address` varchar(255) COLLATE utf8mb4_unicode_ci NOT NULL,
  `employment_date` date NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_unicode_ci;
```

Sl 4.12. *Kreiranje tablice 'employees'*

```
ALTER TABLE `employees`
  ADD PRIMARY KEY (`id`),
  ADD KEY `employees_user_id_foreign` (`user_id`),
  ADD KEY `employees_status_id_foreign` (`status_id`);
```

Sl 4.13. *Određivanje primarnog ključa*



Sl 4.14. Sučelje phpmyadmin

4.2. Funkcija dodavanje novog zaposlenika

Na slici *3.5.* nakon što administrator unese tražene podatke pritisne se dugme spremi, a nakon toga poziva se metoda koju prikazuje slika *4.15.*

```
function insertEmployee()
{
    $.ajax({
        url: ajax_url + 'admin/employees/insert',
        type: 'post',
        dataType: 'json',
        contentType: false,
        processData: false,
        beforeSend: function(request) {
            return request.setRequestHeader('X-CSRF-Token', $("meta[name='_token']").attr('content')); },
        data: new FormData($('.employee-form')[0]),
        success: function(data) {

            var responseStatus = data.status;

            switch (responseStatus)
            {
                case 1:
                    location.href = ajax_url + 'admin/employees';
                    break;
                case 2:
                    alert('Nepravilno popunjena forma');
                    break;
                case 0:
                    alert('Greška');
                    break;
            }
        },
        error: function() {
            alert('Greška');
        }
    });
}
```

Sl 4.15. *Funkcija dodavanja zaposlenika*

Nakon što je pozvana metoda sa slike *4.15.*, poziva se ajax i šalju se podaci s forme na rutu 'admin/employees/insert' što se može vidjeti na slici *4.16.* Ta ruta poziva metodu 'insertEmployee()' koja se nalazi u AdminController-u, a ima svrhu unosa novog zaposlenika. Navedena metoda može se vidjeti na slici *4.17.*

```
Route::group(['prefix' => 'employees'], function() {

    Route::get('/', ['as' => 'AdminEmployees', 'uses' => 'AdminController@getEmployees']);

    Route::post('insert', ['uses' => 'AdminController@insertEmployee']);

    Route::post('details', ['uses' => 'AdminController@getEmployeeDetails']);

    Route::post('update', ['uses' => 'AdminController@updateEmployee']);

    Route::get('delete/{id}', ['as' => 'DeleteEmployee', 'uses' => 'AdminController@deleteEmployee']);
});
```

Sl 4.16. *Slanje podataka na rutu employee*

```php
public function insertEmployee(Request $request)
{
    $first_name = $request->first_name;
    $last_name = $request->last_name;
    $email = $request->email;
    $password = $request->password;
    $status = $request->status;
    $birth_date = $request->birth_date;
    $address = $request->address;
    $employment_date = $request->employment_date;

    $validator = Validator::make($request->all(), Employee::rules());

    if (!$validator->passes())
    {
        return response()->json(['status' => 2]);
    }

    $response = $this->repo->insertEmployee($first_name, $last_name, $email, $password, $status, $birth_date, $address,
        $employment_date);

    return response()->json($response);
}
```

Sl 4.17. *Funkcija insertEmployee*

4.3. Funkcija dodavanja novog zadatka

```javascript
function insertTask()
{
    var name = $('.name').val();
    var deadline = $('.deadline').val();

    var date_test = /^[0-9]{2}\.[0-9]{2}\.[0-9]{4}\.$/;

    if (name == '')
    {
        $('.name').css('border', '1px solid #FF0000');

        return 0;
    }
    else
    {
        $('.name').removeAttr('style');
    }

    if (!date_test.test(deadline))
    {
        $('.deadline').css('border', '1px solid #FF0000');

        return 0;
    }
    else
    {
        $('.deadline').removeAttr('style');
    }

    $.ajax({
        url: ajax_url + 'admin/tasks/insert',
        type: 'post',
        dataType: 'json',
        beforeSend: function(request) {
            return request.setRequestHeader('X-CSRF-Token', $("meta[name='_token']").attr('content')); },
        data: {'name': name, 'deadline': deadline},
        success: function(data) {

            var responseStatus = data.status;

            switch (responseStatus)
            {
                case 1:
                    location.href = ajax_url + 'admin/tasks';
                    break;
                case 0:
                    alert('Greška');
                    break;
            }
        },
        error: function() {
            alert('Greška');
        }
    });
}
```

Sl 4.18. Funkcija dodavanje zadatka

Na slici *3.7.* može se vidjeti što se događa kada se pritisne na dugme spremi novi zadatak, a nakon popunjavanja naziva zadatka i roka izvršavanja, poziva se funkcija 'insertTask()' *(Slika 4.18.)*. Nakon toga navedena funkcija poziva Ajax te se šalju podaci s forme dodavanje novog zadatka na rutu 'admin/tasks/insert' koja se može vidjeti na slici *4.19.*

```php
Route::group(['prefix' => 'tasks'], function() {

    Route::get('/', ['as' => 'AdminTasks', 'uses' => 'AdminController@getTasks']);

    Route::post('insert', ['uses' => 'AdminController@insertTask']);

    Route::post('assign', ['uses' => 'AdminController@assignTask']);

    Route::get('delete/{id}', ['as' => 'DeleteTask', 'uses' => 'AdminController@deleteTask']);

    Route::get('employee/delete/{id}', ['as' => 'DeleteEmployeeTask', 'uses' => 'AdminController@deleteEmployeeTask']);
});
```

Sl 4.19. *Slanje podataka na rutu tasks*

Nakon što su podaci s forme poslani na rutu 'admin/tasks/insert', ta ruta poziva metodu koja se nalazi u AdminController-u 'insertTask()' koja unosi novi zadatak. Funkcija se može vidjeti na slici *4.20.*

```php
public function insertTask(Request $request)
{
    $name = $request->name;
    $deadline = $request->deadline;

    $validator = Validator::make($request->all(), Task::$rules);

    if (!$validator->passes())
    {
        return response()->json(['status' => 2]);
    }

    $this->repo = new TaskRepository;
    $response = $this->repo->insertTask($name, $deadline);

    return response()->json($response);
}
```

Sl 4.20. *Funkcija insertTask*

Iz slike *3.7.* vidi se da na aplikaciji postoji mogućnost dodjeljivanja određenog zadatka određenom zaposleniku. Kada se popuni forma za unos te se pritisne dugme spremi, poziva se funkcija 'assignTask()' koja se može vidjeti na slici *4.21.*

```javascript
function assignTask()
{
    var employee = $('.employee').val();
    var task = $('.task').val();

    $.ajax({
        url: ajax_url + 'admin/tasks/assign',
        type: 'post',
        dataType: 'json',
        beforeSend: function(request) {
            return request.setRequestHeader('X-CSRF-Token', $("meta[name='_token']").attr('content')); },
        data: {'employee': employee, 'task': task},
        success: function(data) {

            var responseStatus = data.status;

            switch (responseStatus)
            {
                case 1:
                    location.href = ajax_url + 'admin/tasks';
                    break;
                case 0:
                    alert('Greška');
                    break;
            }
        },
        error: function() {
            alert('Greška');
        }
    });
}
```

Sl 4.21. *Funkcija dodjeljivanja zadatka*

Nakon pozivanja funkcije 'assignTask()' poziva se Ajax koji šalje podatke na rutu 'admin/tasks/assign', a ta ruta prikazana je na slici *4.19.* Nakon što su podaci poslani na rutu 'admin/tasks/assign', ona poziva metodu 'assignTask()' koja se može vidjeti na slici *4.22.* Navedena metoda nalazi se u AdminController-u koja dodjeljuje zadatak određenom zaposleniku.

```php
public function assignTask(Request $request)
{
    $employee = $request->employee;
    $task = $request->task;

    $this->repo = new TaskRepository;
    $response = $this->repo->assignTask($employee, $task);

    return response()->json($response);
}
```

Sl 4.22. *Metoda 'assignTask'*

4.4. Funkcija dodavanja novog dokumenta

Na slici *3.9.* nakon što se popuni forma za unos naziva dokumenta i pritisne se dugme spremi poziva se funkcija 'insertDocument()' koja se može vidjeti na slici *4.23.* Navedena funkcija poziva Ajax i šalju se podaci s forme za unos naziva dokumenta na rutu 'admin/documents/insert' koja se može vidjeti na slici *4.24.* Ta ruta poziva metodu 'insertDocuments()' koja se nalazi u AdminController-u koja unosi novi dokument. Metoda se može vidjeti na slici *4.25.*

```
function insertDocument()
{
    var name = $('.name').val();

    if (name == '')
    {
        $('.name').css('border', '1px solid #FF0000');

        return 0;
    }
    else
    {
        $('.name').removeAttr('style');
    }

    $.ajax({
        url: ajax_url + 'admin/documents/insert',
        type: 'post',
        dataType: 'json',
        beforeSend: function(request) {
            return request.setRequestHeader('X-CSRF-Token', $("meta[name='_token']").attr('content')); },
        data: {'name': name},
        success: function(data) {

            var responseStatus = data.status;

            switch (responseStatus)
            {
                case 1:
                    location.href = ajax_url + 'admin/documents';
                    break;
                case 0:
                    alert('Greška');
                    break;
            }
        },
        error: function() {
            alert('Greška');
        }
    });
}
```

Sl 4.23. *Funkcija dodavanja dokumenata*

```
    Route::group(['prefix' => 'documents'], function() {

        Route::get('/', ['as' => 'AdminDocuments', 'uses' => 'AdminController@getDocuments']);

        Route::post('insert', ['uses' => 'AdminController@insertDocument']);

        Route::post('insertRequiredDocument', ['uses' => 'AdminController@insertRequiredDocument']);

        Route::get('delete/{id}', ['as' => 'DeleteDocument', 'uses' => 'AdminController@deleteDocument']);
    });
});
```

Sl 4.24. *Slanje podataka na rutu documents*

```php
public function insertDocument(Request $request)
{
    $name = $request->name;

    $validator = Validator::make($request->all(), Document::$rules);

    if (!$validator->passes())
    {
        return response()->json(['status' => 2]);
    }

    $this->repo = new DocumentRepository;
    $response = $this->repo->insertDocument($name);

    return response()->json($response);
}
```

Sl 4.25. *Funkcija insertDocument*

Na slici *3.9.* vidi se da postoji podsjetnik zaposleniku koji funkcionira tako da se izabere zaposlenik kojeg se želi podsjetiti koji dokument mora dostaviti ili poslati. Nakon što je popunjena forma za unos i pritisne se dugme spremi poziva se funkcija 'insertRequiredDocument()' koja se može vidjeti na slici *4.26.* Nakon što je funkcija pozvana, poziva se ajax i šalju se podaci s forme na rutu ' admin/documents/insertRequiredDocument' . Ruta se može vidjeti na slici *4.24.* Navedena ruta poziva metodu 'insertRequiredDocument()' koja se nalazi u AdminController-u i ona dodjeljuje podsjetnik zaposleniku. Metoda 'insertRequiredDocument' može se vidjeti na slici *4.27.*

```javascript
function insertRequiredDocument()
{
    var employee = $('.employee').val();
    var document = $('.document').val();

    $.ajax({
        url: ajax_url + 'admin/documents/insertRequiredDocument',
        type: 'post',
        dataType: 'json',
        beforeSend: function(request) {
            return request.setRequestHeader('X-CSRF-Token', $("meta[name='_token']").attr('content')); },
        data: {'employee': employee, 'document': document},
        success: function(data) {

            var responseStatus = data.status;

            switch (responseStatus)
            {
                case 1:
                    location.href = ajax_url + 'admin/documents';
                    break;
                case 0:
                    alert('Greška');
                    break;
            }
        },
        error: function() {
            alert('Greška');
        }
    });
}
```

Sl 4.26. *Funkcija podsjetnika*

```php
public function insertRequiredDocument(Request $request)
{
    $employee = $request->employee;
    $document = $request->document;

    $this->repo = new DocumentRepository;
    $response = $this->repo->insertRequiredDocument($employee, $document);

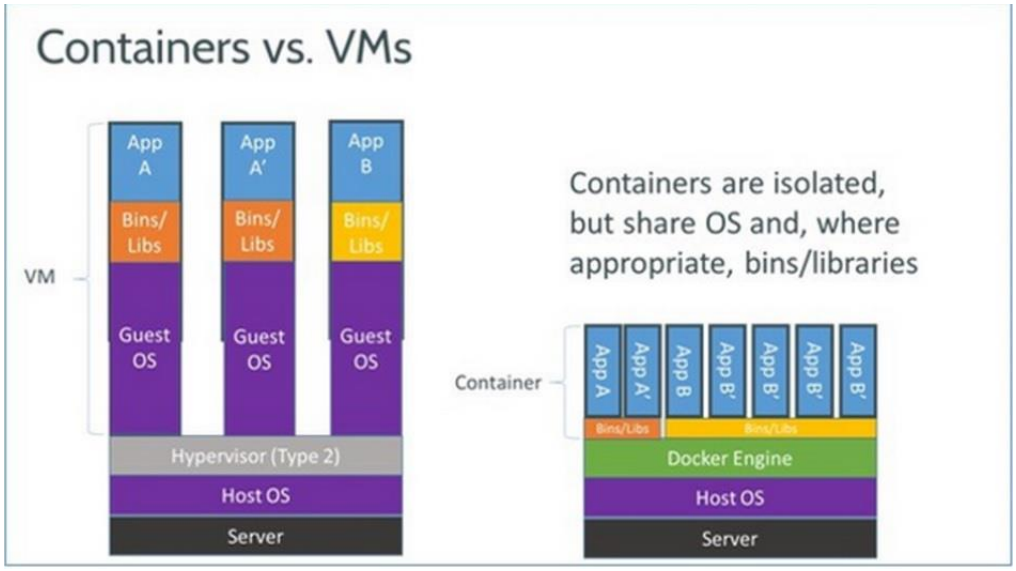
    return response()->json($response);
}
```

SI 4.27. *Metoda insertRequiredDocument*

5. UPRAVLJANJE ISPORUKOM I NEPREKIDNA INTEGRACIJA APLIKACIJE

5.1. Docker

Kao što je rečeno u poglavlju *2.5*, Docker je platforma kontejnerskog softvera. Razvojni programeri ga koriste za pakiranje, isporuku i pokretanje aplikacije, a aplikacija nakon toga može raditi na bilo kojem operativnom sustavu. Takvu svrhu Docker je imao u razvoju ove web aplikacije. Kontejneri omogućuju razvojnim programerima pakiranje aplikacija sa svim potrebnim dijelovima, ali ih isporučuju kao jedan paket. Programer može biti siguran da će se aplikacija na drugom računalu pokrenuti u istom obliku kao što je i na njegovom računalu, neovisno o operativnom sustavu ili drugačijim konfiguracijama. Docker je jednim dijelom poput virtualnog stroja, ali za razliku od virtualnog stroja on ne stvara cijeli virtualni operativni sustav, već omogućava postavljanje lokalnih razvojnih okruženja koja su baš poput živog poslužitelja.



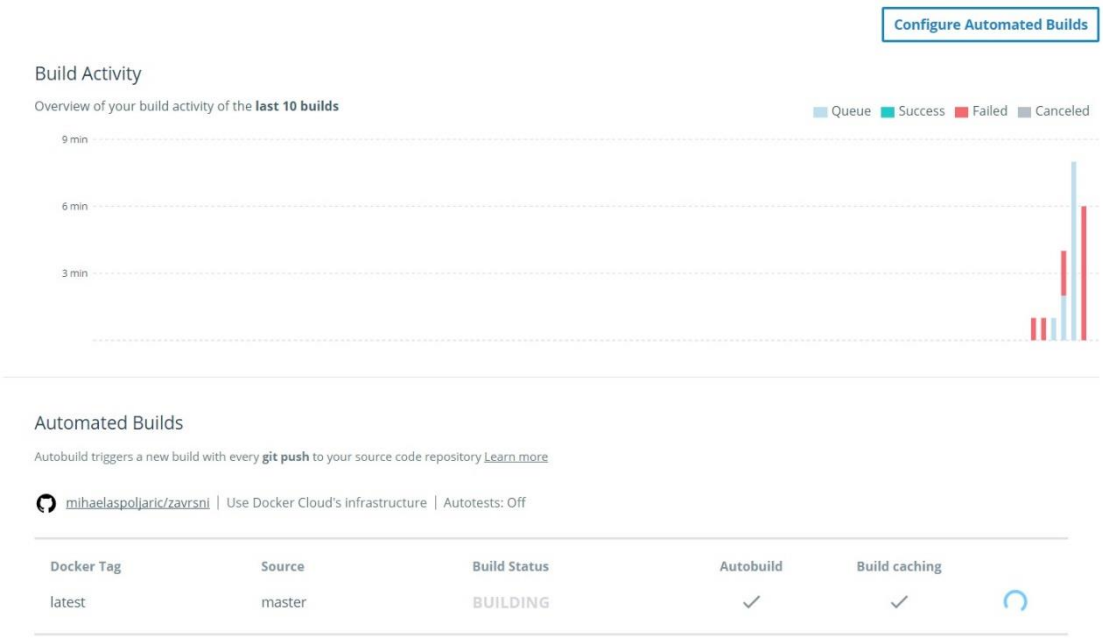
Sl 5.28. *Kontejneri vs. VMs*

Na slici *5.28.* može se vidjeti razlika u zauzimanju prostora između Dockerovih kontejnera i virtualnog stroja. Virtualni stroj zauzima duplo više prostora nego kontejneri upotrebom operativnog sustava 'gosta'. Kod pohrane kontejnera nije potreban operativni sustav 'gosta'.

Docker u suradnji s GitHub omogućuje lakšu pohranu, odnosno učitavanje aplikacije na Docker Cloud. Za ovu web aplikaciju prvo je napravljeno postavljanje projekta na GitHub što se može vidjeti na slici *5.29.* Nakon što je projekt postavljen na GitHub ostalo je samo povezivanje Docker-a s GitHub-om. Na slici *5.30.* vidi se da je nakon povezivanja projekt postavljen na Docker Cloud.



Sl 5.29. *GitHub*

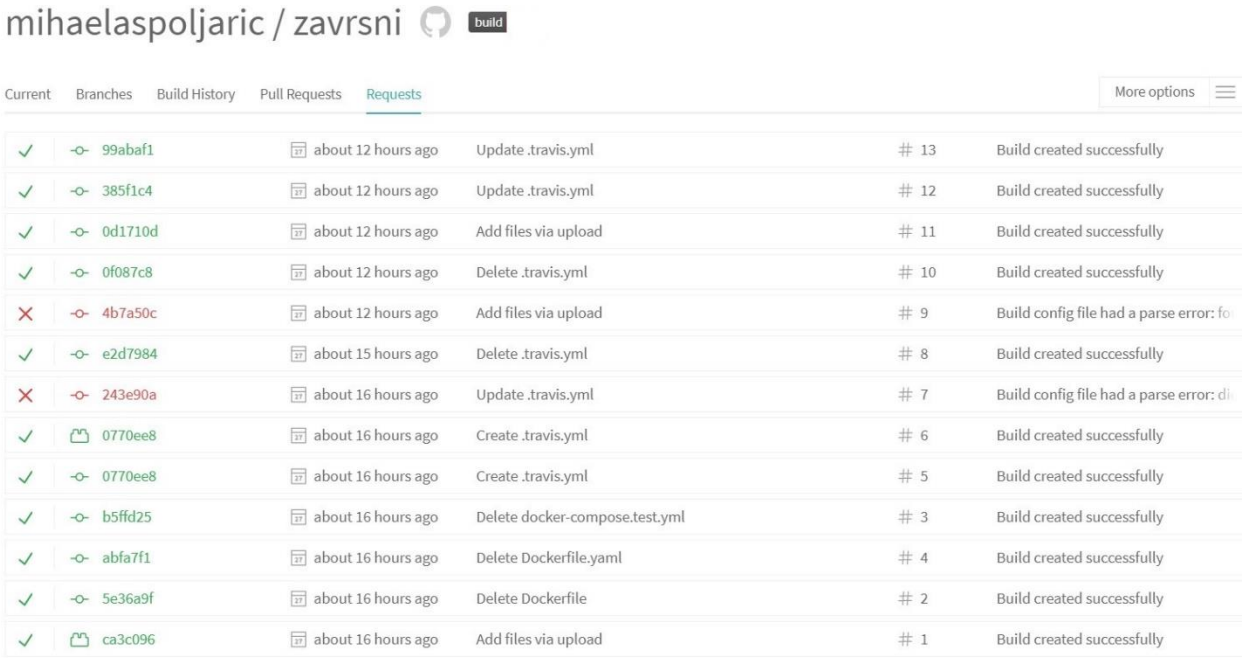


Sl 5.30. *Sučelje Docker Cloud*

5.2. TRAVISCI

TravisCI je softver za kontinuiranu integraciju usluga koje se koriste za izgradnju i testiranje softverskih projekata. Kako bi se TravisCI mogao koristiti, potrebno je imati otvoren račun na GitHub-u. Nakon što su računi sinkrozirani na sučelju TravisCI pokazuju se vlastiti projekti koji su spremni za testiranje. Kako bi testiranje funkcioniralo, potrebno je izraditi datoteku pod nazivom .travis.yml koja mora biti sačuvana u direktoriju projekta. U navedenu datoteku upisuje se ono što se želi testirati. Za potrebu ovog završnog rada napravljeni su testovi samo za provjeru PHP verzije koja je korištena za razvoj web aplikacije. TravisCI nudi puno više nego samo testiranje verzije PHP te podržava izgradnju softvera na brojnim programskim jezicima.

Sučelje TravisCI prikazan je na slici *5.31.*, a kod za testiranje verzije PHP-a prikazan je na slici *5.32.*



Sl 5.31. *Testiranje projekta*

```
language: php

php:

- 5.5
- 5.7
- 7.0

```

Sl 5.32. *Kod za testiranje*

6. ZAKLJUČAK

Laravel je web okvir, koji je baziran na PHP-u, koristi se većinom kako bi se ubrzao tok razvoja aplikacija. Web okviri daju strukturu kodu, aplikacija razvijena na web okviru daje bolji, čitljiviji kod. Laravel olakšava razvoj aplikacije u pogledu autorizacije, ruta i još puno ostalih stvari. Kod razvoja ove web aplikacije najveći problem je bio nepoznavanje svih mogućnosti Laravel-a.

Docker je platforma kontejnerskog softvera. Koristi se za pakiranje, isporuku i pokretanje aplikacije. Nakon što je aplikacija postavljena na Docker Cloud, spremna je za daljni razvoj, drugi programeri mogu preuzeti aplikaciju kao jedan paket i bit će pokrenuta u istom obliku kao što je bila pokrenuta na izvornom računalu.

TravisCI je softver koji koristi koncept kontinuirane integracije softvera za bolju izgradnju i testiranje softverskih projekata. Prednost korištenja TravisCI je ako se radi na nekom velikom projektu, na kojem radi više osoba gdje se rade česte promjene na kodu, TravisCI radi integraciju kodova u dijeljeno spremište nekoliko puta dnevno. Nakon svake integracije moguće je pokretanje raznih testova, te samim time i otkrivanje pogrešaka u kodu. Programerima dolazi obavijest ukoliko postoji pogreška u kodu, samim time programerima olakšava posao otkrivanja pogreške.

Web aplikacija za praćenje podataka o zaposlenicima, zadacima koje moraju obaviti te dokumentima koje moraju dostaviti može se još dosta poboljšati. Prikazana verzija napravljena je samo za prikaz upotrebe modernih tehnologija. Trenutna verzija aplikacije radi samo na lokalnom serveru zbog toga što nije postavljena na web tako da je pokretanje moguće samo korištenjem programa za simulaciju lokalnog servera. Aplikaciji imaju pristup administrator i zaposlenik. Na sučelju administratora unose se svi bitni podaci, tko su zaposlenici, koji su njihovi statusi, koje zadatke moraju odraditi i koje dokumente moraju dostaviti. Zaposlenici imaju uvid samo u sučelje koje im daje prikaz u zadatke koje moraju obaviti, koji su njihovi osobni podaci, uvid u njihove osobne podatke koje mogu promijeniti te uvid u dokumente koje još moraju dostaviti.

Web aplikacija razvijena je pomoću Laravel-a koji je baziran na PHP-u, Ajaxa pomoću kojeg su napravljene funkcionalnosti, a baza podataka napravljena je pomoću MySQL-a.

LITERATURA

[1] Ajax https://www.w3schools.com/xml/ajax_intro.asp Datum zadnje posjete: 1.9.2017.

[2] Docker https://www.docker.com/what-docker Datum zadnje posjete: 1.9.2017.

[3] MySql https://www.mysql.com/ Datum zadnje posjete: 25.8.2017.

[4] PHP https://www.w3schools.com/php/ Datum zadnje posjete: 20.8.2017.

[5] Preuzeta slika sa stranice http://www.zdnet.com/article/what-is-docker-and-why-is-it-so-darn-popular/ Datum zadnje posjete: 21.8.2017.

SAŽETAK

Web aplikacija za praćenje podataka o zaposlenicima može raditi na bilo kojem operacijskom sustavu. Aplikacija može biti korištena u većim tvrtkama koje zahtijevaju organiziranost, a ne zahtijeva napredno znanje korisnika za rad s računalom te je jednostavna za korištenje. Aplikaciji imaju pristup administrator i zaposlenici. Za kreiranje klijentskog dijela aplikacije korišteni su Laravel, JavaScript+Ajax, Bootstrap, a na poslužiteljskoj strani korišten je MySQL.

Ključne riječi: Bootstrap, Docker, Laravel, PHP, TravisCI,

ABSTRACT

Modern technologies in developing web application interfaces for business applications

Web application for monitoring data about employees can work on any operating system. Application can be used in bigger companies which require organization. This application doesn't require advance level of computer knowledge, and it's simple to use. Administrator and employee have access to the application. Client side of the application was created with Laravel, JavaScript+Ajax, Bootstrap, and on server side MySQL.

Keywords: Bootstrap, Docker, Laravel, PHP, TravisCI,

ŽIVOTOPIS

Mihaela Špoljarić rođena je 27. ožujka 1992. u Starim Mikanovcima. Osnovnu školu je završila u Ivankovu, a srednju strukovnu školu u Vinkovcima. Akademske godine 2013/14 upisala je stručni studij Informatike na Fakultetu elektrotehnike, računarstva i informacijskih tehnologija u Osijeku.

Mihaela Špoljarić

_____

PRILOZI

Na optičkom disku nalazi se .doc verzija završnog rada i u datoteci sav kod web aplikacije napravljene u ovom radu.