Kriptiranje korisničkih datoteka na serveru upotrebom PHP-a i OpenSSL-a

Cerovski, Nikola

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:200:629292

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

FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA

Stručni studij, smjer informatika

KRIPTIRANJE KORISNIČKIH DATOTEKA NA SERVERU UPOTREBOM PHP-A I OPENSSL-A

Završni rad

Nikola Cerovski

Osijek, 2017.

FERIT
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA OSIJEK

Obrazac Z1S: Obrazac za imenovanje Povjerenstva za obranu završnog rada na preddiplomskom stručnom studiju

Osijek, 31.08.2017.

Odboru za završne i diplomske ispite

Imenovanje Povjerenstva za obranu završnog rada na preddiplomskom stručnom studiju

Ime i prezime studenta:	Nikola Cerovski
Studij, smjer:	Preddiplomski stručni studij Elektrotehnika, smjer Informatika
Mat. br. studenta, godina upisa:	AI4448, 16.10.2015.
OIB studenta:	83294567920
Mentor:	Doc.dr.sc. Ivica Lukić
Sumentor:	
Sumentor iz tvrtke:	
Predsjednik Povjerenstva:	Izv. prof. dr. sc. Krešimir Nenadić
Član Povjerenstva:	Doc.dr.sc. Mirko Köhler
Naslov završnog rada:	Kriptiranje korisničkih datoteka na serveru upotrebom PHP-a i OpenSSL-a
Znanstvena grana rada:	**Programsko inženjerstvo (zn. polje računarstvo)**
Zadatak završnog rada	Napraviti kriptiranje korisničkih datoteka i dekriptiranje s Laravel razvojnim okruženjem i OpenSSL-om. Korisnik postavlja datoteke i kriptira ih. Daotetkama može pristupiti i dekriptirati ih samo ovlašteno osoblje. Koristiti Open SSL ili Mcrypt.
Prijedlog ocjene pismenog dijela ispita (završnog rada):	Izvrstan (5)
Kratko obrazloženje ocjene prema Kriterijima za ocjenjivanje završnih i diplomskih radova:	Primjena znanja stečenih na fakultetu: 3 bod/boda Postignuti rezultati u odnosu na složenost zadatka: 3 bod/boda Jasnoća pismenog izražavanja: 3 bod/boda Razina samostalnosti: 3 razina
Datum prijedloga ocjene mentora:	31.08.2017.

Potpis mentora za predaju konačne verzije rada u Studentsku službu pri završetku studija:	Potpis:
	Datum:

FERIT
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA
I INFORMACIJSKIH TEHNOLOGIJA **OSIJEK**

IZJAVA O ORIGINALNOSTI RADA

Osijek, 15.09.2017.

Ime i prezime studenta:	Nikola Cerovski
Studij:	Preddiplomski stručni studij Elektrotehnika, smjer Informatika
Mat. br. studenta, godina upisa:	AI4448, 16.10.2015.
Ephorus podudaranje [%]:	0%

Ovom izjavom izjavljujem da je rad pod nazivom**: Kriptiranje korisničkih datoteka na serveru upotrebom PHP-a i OpenSSL-a**

izrađen pod vodstvom mentora Doc.dr.sc. Ivica Lukić

i sumentora

moj vlastiti rad i prema mom najboljem znanju ne sadrži prethodno objavljene ili neobjavljene pisane materijale drugih osoba, osim onih koji su izričito priznati navođenjem literature i drugih izvora informacija.
Izjavljujem da je intelektualni sadržaj navedenog rada proizvod mog vlastitog rada, osim u onom dijelu za koji mi je bila potrebna pomoć mentora, sumentora i drugih osoba, a što je izričito navedeno u radu.

Potpis studenta:

SADRŽAJ

1. UVOD 1
2. TEHNOLOGIJE KORIŠTENE PRI IZRADI APLIKACIJE 2
2.1. HTML 2
2.2. CSS 3
2.3. JavaScript i jQuery 4
2.4. SweetAlert2 biblioteka 5
2.5. PHP 6
2.6. Laravel radno okruženje 7
2.6.1. Podešavanje Laravel radnog okruženja 8
2.6.2. Osnovni koncepti Laravel radnog okruženja 8
2.6.3. Rad sa bazom podataka 11
3. IZRADA APLIKACIJE 14
3.1. Izrada obrazaca za prijavu i registraciju 15
3.2. Prijenos datoteka na server 16
3.2.1. Enkripcija datoteke 18
3.2.2. Preuzimanje i dekripcija datoteke 20
3.3. Brisanje datoteke 21
3.4. Prikaz javnih korisničkih datoteka 22
3.5. Izrada i dizajn korisničkog profila 23
3.6. Slanje poruka 24
3.7. Izmjena podataka 25
4. POSTAVLJANJE WEB APLIKACIJE NA WEB SERVER 26
5. ZAKLJUČAK 28
LITERATURA 29
SAŽETAK 31
ABSTRACT 32
ŽIVOTOPIS 33

1. UVOD

U današnje doba, s razvojem web tehnologija, javljaju se i novi načini pohrane korisničkih datoteka. Kako bi mogli pristupiti svojim datotekama ili ih podijeliti sa drugim osobama, korisnici imaju mogućnost pohrane svojih datoteka na serveru. Pohrana na serveru korisnicima omogućuje pristup datotekama u bilo koje vrijeme preko različitih uređaja. Veliki je izazov pravilno zaštititi korisničke datoteke od neželjenog (neovlaštenog) pristupa. U svrhu zaštite korisničkih podataka koriste se razne metode enkripcije među koje spada i OpenSSL (raniji naziv *SSLeay*). OpenSSL je slobodni softver (engl. open source) za *Secure Sockets Layer* (skraćeno SSL) i *Transport Layer Security* (skraćeno TLS).

Razvoj modernih metoda enkripcije započeo je pojavom elektronske trgovine (engl. *e-commerce*). Ove metode enkripcije zasnivaju se na korištenju ključa za enkripciju kojim se originalna informacija maskira u niz znakova koje ne možemo pročitati ukoliko ne posjedujemo identični ključ. Obrnut postupak enkripcije, odnosno korištenje ključa za vraćanje sadržaja datoteke u prvobitni oblik naziva se dekripcija.

Ovaj završni rad sastoji se od četiri poglavlja koja obuhvaćaju teorijski opis tehnologija, proces izrade aplikacije i zaključak. U drugom poglavlju navedene su i opisane tehnologije i alati korišteni pri izradi web aplikacije. Treće poglavlje obuhvaća opis glavnih dijelova aplikacije sa primjerima koda. U četvrtom poglavlju napravljen je osvrt na proces izrade aplikacije i dobivene rezultate u obliku zaključka.

Zadatak ovog završnog rada je izrada web aplikacije za pohranu korisničkih datoteka na serveru koristeći PHP programski jezik (i ostale web tehnologije) i Laravel razvojno okruženje (engl. framework), te provesti učinkovitu zaštitu datoteka upotrebom OpenSSL enkripcije. Za dodatnu razinu zaštite, aplikaciju mogu koristiti samo registrirani korisnici.

2. TEHNOLOGIJE KORIŠTENE PRI IZRADI APLIKACIJE

Aplikacija je izrađena kroz PHP *Laravel framework,* te su korištene JavaScript biblioteke za prikaz modalnih prozora (za upozorenja i obavijesti), te za sortiranje tablica – *SweetAlert2* i *DataTables*. Alati potrebni za izradu aplikacije su web preglednik, tekstualni editor, XAMPP radno okruženje i upravljač paketima (engl. *packet manager*) *Composer*.

Osnovne tehnologije na strani klijenta su HTML, CSS i JavaScript. Koriste se za izradu sučelja web stranica i aplikacija, i omogućavaju dodatnu funkcionalnost i interakciju sa korisnicima. Tehnologije na strani poslužitelja su PHP i MySQL, koriste se za izradu dinamičkog sadržaja, pohranu podataka na bazi podataka i slično. U nastavku su opisani alati korišteni pri izradi aplikacije.

2.1. HTML

HyperText Markup Language (skraćeno HTML) je opisni jezik koji predstavlja temelj svake web stranice. Hipertekst predstavlja poveznice koje povezuju jednu web stranicu sa drugom, što predstavlja funkcionalnost svake web stranice. HTML dokumenti imaju nastavak *.html*. Svaki HTML dokument započinje sa *doctype* deklaracijom koja označava verziju HTML-a koja je korištena pri izradi web stranice ili aplikacije. Trenutna verzija je HTML5, pa pišemo *<!DOCTYPE html>*[1]. Glavne oznake svake web stranice su *<html>* i *</html>* koje označavaju početak i kraj HTML dokumenta. Sadržaj stranice dijelimo na dva dijela pomoću *<head>* i *<body>* oznaka. Oznaka *<head>* označava dio HTML dokumenta u kojem navodimo informacije o web stranici poput naslova, referenci na vanjske dokumente (npr. poveznica na CSS dokument ili razne biblioteke) i mnoge druge podatke (informacije) o stranici koji su potrebni pregledniku. Oznaka *<body>* predstavlja korisnicima vidljiv dio web stranice koji se prikazuje na zaslonu.

HTML elementi opisuju različite tipove sadržaja na web stranici poput naslova, teksta, slika, različitih odjeljaka na stranici i slično. Sadržaj elemenata se uglavnom nalazi između početne i krajnje oznake u obliku: *<ime oznake>sadržaj</ime oznake>*. Neki od osnovnih HTML elemenata navedeni su u tablici 2.1.

Tab. 2.1. Osnovni HTML elementi i njihovo značenje

<h1>, <h2>, …., <h6>	Definiranje naslova
<p>	Pasus (engl. paragraph)
<a>	Poveznica
	Odjeljak
<header>	Oznaka za zaglavlje stranice
<footer>	Oznaka za podnožje stranice
<img>	Oznaka za sliku
<table>	Oznaka za tablicu

Kao što je i ranije navedeno, HTML je osnovni dio svake web stranice. HTML kodom povezujemo sadržaj sa vanjskim poveznicama, te kreiramo sami *kostur* web stranice i njen raspored elemenata (engl. *layout*). HTML kod prikazan pomoću web preglednika nije vizualno atraktivan, te ga je potrebno urediti sa CSS stilskim jezikom.

2.2. CSS

CSS (skraćeno od *Cascading Style Sheets*) je stilski jezik koji se koristi za oblikovanje web stranice, te omogućuje prilagođavanje web stranice prema zadanom dizajnu [3]. Upotrebom CSS stilskog jezika možemo pozicionirati pojedine HTML elemente na prozoru, njihovu širinu i visinu, pozadinsku boju, oblikovati tekst i mijenjati mnoga druga obilježja.

Trenutna inačica CSS stilskog jezika je CSS3. Stilske elemente možemo navesti u samom HTML dokumentu između oznaka *<style>*, u atributima HTML elemenata (engl. *inline*) ili u odvojenom dokumentu koji nosi nastavak *.css* (potrebno je navesti poveznicu na dokument u *<head>* dijelu HTML dokumenta). Prilikom uređivanja web stranice, izazov je prilagoditi izgled stranice za prikaz na svim uređajima različitih veličina ekrana. To se ostvaruje korištenjem relativnih jedinica (*%, em, rem*) [4] prilikom definiranja širine elemenata, te uporabom *media query* pravila koja primjenjuju određene stilove za određene širine i visine ekrana [5]. Navedeni postupci pripadaju metodama responzivnog web dizajna gdje se stranica prilagođava veličini ekrana uređaja.

Razvojem web tehnologija, u novije vrijeme su dostupni i CSS pretprocesori – alati koji dodatno proširuju svojstva CSS stilskog jezika. Najpopularniji pretprocesor je *Sass*. Uporaba

pretprocesora znatno proširuje mogućnosti CSS stilskog jezika – uvodi varijable, novu sintaksu, if, else i for petlje, nudi mogućnost kreiranja grupe stilova koje se ujedinjuju pod *mixin* pravilima i slično. Sass kod potrebno je pisati u odvojene datoteke koje nose nastavak *.scss*. Web preglednici ne mogu čitati Sass kod, pa je potrebno kod sastaviti (engl. compile) u izvorni CSS pomoću alata kao što je *gulp.js*.



Sl. 2.1. *Primjer Sass koda*

2.3. JavaScript i jQuery

JavaScript je programski jezik na strani klijenta kojim se proširuje funkcionalnost web stranica [6]. JavaScript datoteke imaju nastavak *.js*. Uporaba je moguća unutar *<script>* i *</script>* oznaka u HTML dokumentu, ili putem vanjskih datoteka koje sadrže JavaScript kod.

JavaScript kodom možemo manipulirati HTML elementima, kreirati nove elemente, mijenjati njihova obilježja i definirati njihovo ponašanje u slučaju različitih događaja i slično. Taj princip se naziva DOM manipulacija. Također, JavaScript programski jezik koristitimo za računske kalkulacije i validacije podataka kojima nije potrebno opterećivati server.

Biblioteka jQuery donosi pojednostavljeni način manipulacije dokumentima, te je jedna od najranijih i najpopularnijih biblioteka. Ukoliko želimo koristiti jQuery biblioteku, potrebno je navesti vanjsku poveznicu na biblioteku, te navesti datoteku koja sadrži naš jQuery kod. Slika 2.2. prikazuje primjer jQuery koda koji dodaje klasu "*active*" na HTML element "*mobile-nav*" ukoliko korisnik klikne na njega.

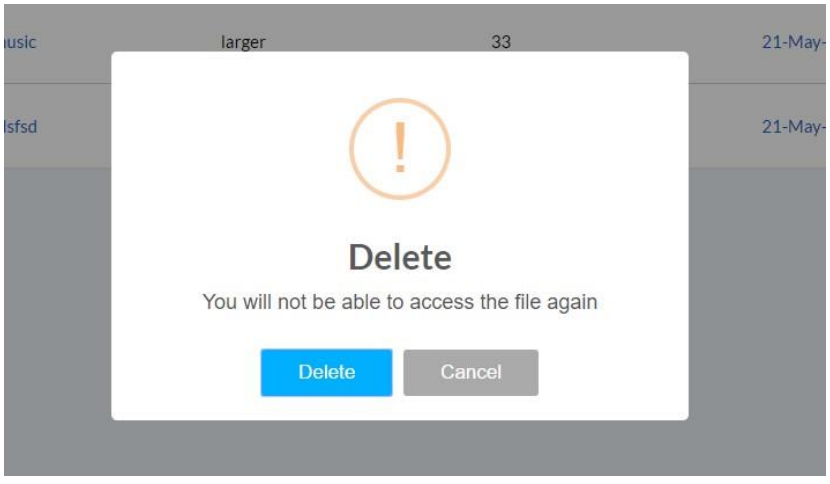
```
$('#mobile-nav').on('click', function() {
    $('#mobile-nav').toggleClass('active');
});
```

Sl. 2.2. *Primjer jQuery koda*

JavaScript programski jezik je u prošlih nekoliko godina znatno napredovao zahvaljujući raznim razvojnim okruženja (engl. *framework*) kao što su *React*, *Angular* i *Vue* [7], te je postao neophodan alat pri izradi složenih web aplikacija. Također se više ne koristi samo na strani klijenta, već i na strani poslužitelja (*Node.js*). JavaScript programski jezik je standardiziran kroz "*ECMAScript*" specifikacije, a trenutačni standard je ES5 [6].

2.4. SweetAlert2 biblioteka

Za prikaz obavijesti na web aplikaciji korištena je "*SweetAlert2*" JavaScript biblioteka. Navedena biblioteka pruža napredniji i bolje dizajnirani način prikaza modalnih prozora od standardne JavaScript funkcije "*alert()*". Za instalaciju biblioteke potrebno je u željenom direktoriju putem *npm* komandne linije preuzeti biblioteku naredbom "*npm install sweetalert2*", te referencirati potrebne CSS i JavaScript datoteke. Primjer modalnog prozora prikazan je na slici 2.3.



Sl. 2.3. *Modalni prozor prilikom brisanja datoteke*

Izrada modalnih prozora kreira se kroz JavaScript, ili po želi - jQuery. Potrebno je funkcijom "*swal*" navesti parametre za modalni prozor koji će se izvršiti u određenom trenutku. Neki od osnovnih parametara su:

- naslov,
- tekst,
- tip prozora (npr. upozorenje ili obavijest),
- tipke (npr. potvrdi, obriši), itd.

```
$('.table-icon.delete').on('click', function(e) {
    e.preventDefault();
    var file = $(this);
    swal({
            title: "Delete",
            text: "You will not be able to access the file again",
            type: "warning",
            showCancelButton: true,
            confirmButtonColor: "#00B0FF",
            confirmButtonText: "Delete",
            closeOnConfirm: true
    }).then(function () {
          $.ajax({
              $(this).css('display', 'none');
              url: $(file).attr('href')
          });
    });
});
```

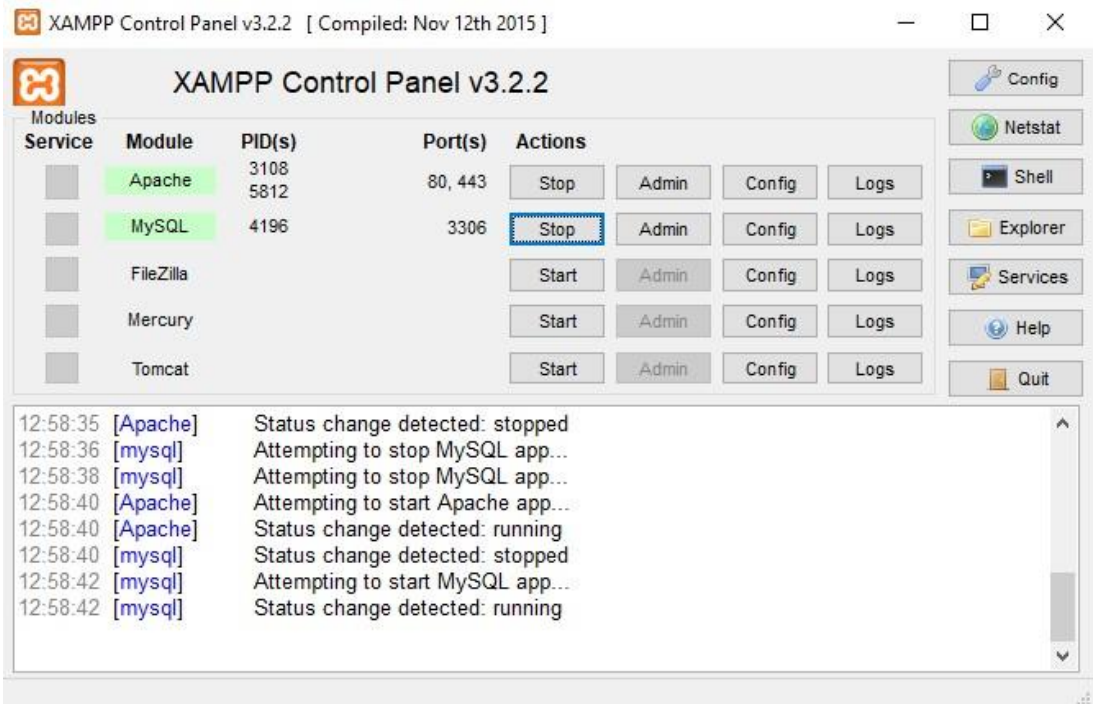
Sl. 2.4. *Izrada modalnoga prozora za brisanje datoteke*

2.5. PHP

PHP, skraćeno od *PHP Hypertext Preprocessor*, je programski jezik koji se izvodi na strani poslužitelja (servera). Dinamički generira sadržaj stranice na serveru koji se potom šalje korisniku [8]. Programski jezici na strani poslužitelja neophodni su za izradu složenijih web stranica koje zahtijevaju pristup bazi podataka. Sve datoteke koje sadržavaju PHP nose nastavak *.php*. PHP programski jezik može komunicirati sa raznim bazama podataka što omogućava dohvaćanje i pohranu podataka. PHP kod piše se unutar oznaka <*?php* i *?*>.

Unaprjeđivanjem programskog jezika i dolaskom verzije PHP5 omogućen je objektno orijentirani pristup što je unaprijedilo sigurnost i otvorilo vrata za izradu mnogih radnih okruženja koja olakšavaju i poboljšavaju izradu web stranica i aplikacija. Najpopularnija radna okruženja su *Symfony* i *Laravel*. Trenutačna verzija je PHP7, dok je još široko korištena verzija PHP 5.6.

Za rad je potrebno postaviti lokalni web server (engl. *localhost*), a najjednostavnije rješenje je pomoću besplatnog alata XAMPP. XAMPP je alat koji pruža mogućnosti rada sa Apache HTTP serverom, MySQL bazom podataka i drugim dodatnim funkcijama koje se pokreću kroz upravljačku ploču. Slika 2.3. prikazuje izgled upravljačke ploče alata XAMPP.



Sl. 2.5. *Podešavanje Apache servera i baze podataka pomoću XAMPP alata*

2.6. Laravel radno okruženje

Laravel je PHP radno okruženje (engl. *framework*) temeljeno na principima objektno orijentiranog programiranja, te je baziran na MVC arhitekturi (skraćeno od *Model View Controller*). Korištenje alata kao što je *Laravel* znatno olakšava izradu web aplikacija jer pružaju veliki broj izrađenih funkcija i metoda koje automatiziraju učestale poslove pri izradi web aplikacija. Primjerice, (uz opcionalne i minimalne korekcije) programeri imaju na raspolaganju već gotove i pouzdane sustave autorizacije i registracije korisnika, učinkovit rad sa datotekama, rukovanje putanjama (engl. *routes*) i mnoga druga rješenja za izradu web aplikacija. Također, *Laravel* nudi i jednostavno rukovanje bazama podataka pomoću ORM tehnike naziva *Eloquent* [9]. Razvoj *Laravel* radnog okruženja započeo je 2011. godine, te je redovitim ažuriranjem postao najpopularnijim radnim okruženjem za PHP programski jezik. U trenutku pisanja ovog završnoga rada, aktualna verzija je 5.4., a uskoro dolazi verzija 5.5.

Svaki *Laravel* projekt sastoji se od devet direktorija, a najbitniji su:

- App - sadrži osnovne datoteke potrebne za izradu aplikacije kao što su klase, *middleware*, razne funkcije i modeli
- Config - sadrži konfiguracijske datoteke koje je potrebno podesiti za funkcioniranje aplikacije (podaci o bazi podataka, e-mail servisu i slično)
- Database - sadrži bitne datoteke za bazu podataka kao što su migracije (datoteke kojima se kreiraju tablice u bazi podataka) i *factories* (alati kojima se automatski unosi sadržaj u bazu podataka u svrhu testiranja)
- Public - sadrži datoteke za uređivanje stranice (CSS), JavaScript, dodatne biblioteke, slike koje se prikazuju na stranici i slično,
- Resources - direktorij koji sadrži poglede (engl. views)
- Routes - sadrži datoteke koje rukuju sa putanjama

2.6.1. Podešavanje Laravel radnog okruženja

Postoji nekoliko zahtijeva koje treba ispuniti za uspješno podešavanje *Laravel* radnoga okruženja. Potrebna je PHP 5.6.4. verzija (ili novija), instalirane ekstenzije za OpenSSL, PDO, "*Mbstring*", "*Tokenizer*" i XML, te "*Composer*" upravljač paketima [10]. Instalacija započinje unošenjem naredbe *composer global require "laravel/installer"* u naredbenu liniju (CMD). Navedena naredba će preuzeti službenu instalaciju te podesiti radno okruženje na računalu. Projekt se u željenom direktoriju stvara naredbom "*laravel new ime_projekta*". Početak rada *Laravel* radnim okruženjem započinje unošenjem naredbe "*php artisan serve*" koja pokreće HTTP server [10].

2.6.2. Osnovni koncepti Laravel radnog okruženja

Kao što je ranije navedeno, *Laravel* radno okruženje pruža veliki broj funkcija i pomno razrađenih metoda koje uz određeno predznanje olakšavaju i poboljšavaju izradu web aplikacija. Neke od funkcionalnosti su rukovanje putanjama, CSRF zaštita, kontroleri, pogledi (engl. *view*), validacija, *middleware,* itd.

Putanja (engl. *Route*) vodi do određene stranice na web aplikaciji (npr. *localhost/home*). Rukovanjem putanjama moguće je odrediti koji će se kontroler i njegova metoda izvršiti u slučaju

određene putanje i specificiranog HTTP zahtijeva. Putanje u *Laravel* radnom okruženju nalaze se u direktoriju *routes*. Prilikom rukovanja putanjama, potrebno je navesti tip HTTP zahtjeva (npr. GET ili POST), putanju (URL), te kontroler i njegovu metodu koja će se izvršiti u slučaju dolaska te putanje. Slika 2.4. prikazuje primjer u kojem će se u slučaju *GET* zahtjeva nad putanjom *localhost/home* izvršiti *home* metoda u kontroleru naziva *HomeController*.

```
Route::get('/home', 'HomeController@home');
```

Sl. 2.6. *Primjer rukovanja putanjama*

Middleware je sljedeća skupina korisnih metoda koje se koriste za filtriranje HTTP zahtijeva nad aplikacijom [11]. *Middleware* možemo predočiti kao zaštitni sloj/vrata koji HTTP zahtjev mora proći prije pristupa određenom dijelu aplikacije. Primjerice, *middleware RedirectIfAuthenticated (slika 2.5.)* postavljen na jednu od putanja će proslijediti korisnika na sljedeći zahtjev ukoliko je prijavljen u aplikaciju, odnosno, vratiti na zaslon za prijavu ukoliko validacija korisničkih podataka nije uspješna.

```
class RedirectIfAuthenticated
{
    public function handle($request, Closure $next, $guard = null)
    {
        if (Auth::guard($guard)->check()) {
            return redirect('/user/'.$user->id);
        }

        return $next($request);
    }
}
```

Sl. 2.7. *RedirectIfAuthenticated middleware*

CSRF (engl. *cross-site request forgery*) zaštita sprječava CSRF napade. Ovaj tip napada odnosi se na situacije gdje se izvršavaju neželjene (štetne) naredbe preko korisnika (bez njegova znanja) kojega je aplikacija autorizirala (kojemu je dodijeljen *session cookie*). *Laravel* dodjeljuje CSRF *token* svakom aktivnom korisniku. Za provjeru *tokena* zadužen je *VerifyCsrfToken middleware* koji uspoređuje *token* unutar obrasca sa *tokenom* dodijeljenim korisniku unutar sesije. Provjerom se utvrđuje da li autorizirani korisnik stvarno izvršava neku radnju ili ne. CSRF zaštita preporučena je u svakom HTML obrascu, te se određuje naredbom *csrf_field()*. Naredba *csrf_field()* u obrascu stvara nevidljivo polje sa generiranim *tokenom*.

S obzirom da je *Laravel* MVC radno okruženje, bitna značajka su kontroleri (engl. *controller),* modeli i pogledi. Zadaća kontrolera je kao što i samo ime nalaže – kontroliranje zahtjeva i obrada različitih funkcija unutar aplikacije, te efikasnije grupiranje zahtjeva nad istim sadržajem kako se programski kod ne bi (nepotrebno) ponavljao. Kontroleri dohvaćaju podatke, obrađuju ih i prosljeđuju modelima za pohranu, te pokreću poglede. Svaki kontroler implementiran je klasom koja sadrži niz metoda [12]. Metode unutar kontrolera primjerice imaju zadaću dohvaćanja ulaznih podataka dospjelih kroz HTML obrasce, obrađuju podatke i pohranjuju ih u bazu podataka, stvaraju objekte, šalju zahtjeve i slično. Prosljeđivanje podataka između kontrolera i pogleda (engl. view) izvodi se *return* funkcijom kojoj se predaje nekoliko parametara:

- *View* - prima ime pogleda koji će se prikazati izvršavanjem metode u kontroleru,
- *With* - potrebno je navesti podatke koji će se predati navedenom pogledu, više podataka navodi se odvojenim *with* parametrima (slika 2.8.)

```
return view('pages.userprofile')
      ->with('files', $files)
      ->with('user', $user);
```

Sl. 2.8. *Prosljeđivanje podataka između kontrolera i pogleda*

Kreiranje kontrolera može biti izvršeno unošenjem naredbe *php artisan make:controller ime_kontrolera.* Slika 2.9. primjer je kontrolera *HomeController* koji sadrži dvije metode. Pojedina metoda izvršava se na temelju rukovanja putanjama gdje se navodi koja će se metoda izvršiti u kojem slučaju. Javna metoda *home* će u ovome primjeru prikazati pogled *home,* tj. naslovnu stranicu, a metoda *recentFiles* prikazati će pogled za prikaz najnovijih datoteka postavljenih na server.

```
class HomeController extends Controller
{
    public function home()
    {
        return view('/home');
    }

    public function recentFiles()
    {

        $files = File::where('file_status', '=', 'public')->orderBy('created_at', 'desc')->paginate(25);

        return view('pages.files')
              ->with('files', $files);
    }
}
```

Sl. 2.9. *Primjer kontrolera sa dvije metode*

Pogledi (engl. view) prikazuju sadržaj stranice kroz HTML i PHP kod. Pogled je implementiran kroz *Laravel blade* predloške (*blade templating engine*). Svakom pogledu moguće je predati niz podataka kroz kontroler. Podacima koji su pogledu predani preko kontrolera pristupamo naredbom *{{ $varijabla }}*. *Blade* predlošci pružaju mogućnost nasljeđivanja HTML koda iz drugih *.blade.php* datoteka kroz naredbu *@extends('ime_datoteke'),* čime je pojednostavljena izrada sučelja kojega možemo primjenjivati na svim stranicama. Dalje, HTML kod možemo dijeliti u komponente, te ih primjenjivati na različitim datotekama naredbom *@component('ime_komponente')*. Također, *blade* predlošci pružaju veliki broj izrađenih funkcija koje pojednostavljuju pisanje HTML koda. Pogled se pohranjuje u .php formatu, a puni naziv datoteke mora biti u obliku *ime_datoteke.blade.php* [13].

```
@extends('layouts.app')
@section('sidebar')
    <div class="user-info-wrap">
        <h1 id="page-logo">
            <a href="../../user/{{ Auth::user()->id}}">
                <img src="../../images/logo.svg" alt="Up!">
            </a>
        </h1>
        <div class="sidebar-content">
            <div class="user-display">
                <a class="username" href="../../user/{{ Auth::user()->id }}">
                    @if (Auth::user()->profile_picture_path === null)
                        <img class="profile-picture" src="../../{{ Auth::user()->profile_picture }}" alt="Profile picture" />
                        <h1>{{ Auth::user()->name }}</h1>
                    @else
```

Sl. 2.10. *Primjer pogleda koji nasljeđuje sučelje layouts.app*

Modeli predstavljaju objekte koji sadrže podatke, te su povezani sa određenom tablicom u bazi podataka. Primjerice, model može biti korisnik, datoteka, poruka. Modeli se kreiraju u komandnoj liniji naredbom *php artisan make:model ime_modela*.

2.6.3. Rad sa bazom podataka

Nakon instalacije *Laravel* radnoga okruženja potrebno je konfigurirati aplikaciju, te odabrati željenu bazu podataka koju će aplikacija koristiti. Podaci o bazi podataka sadržani su u datoteci koja se nalazi u direktoriju *config/database.php* dok se ostali najpotrebniji podaci za konfiguriranje aplikacije nalaze u *.env* datoteci. Podržane baze podataka su:

- MySQL
- Postgres
- SQLite
- SQL Server

Izrada tablice izvršava se naredbom *php artisan make:migration naziv_tablice*. Navedena naredba kreirati će migraciju. Migracije su datoteke koje sadrže dvije metode za rad nad bazom podataka – *Up* i *Down*. Metoda *Up* se koristi za dodavanje novih tablica te se izvršava naredbom *php artisan migrate*, a metoda *Down* isključivo kada se žele poništiti operacije iz metode *Up* te se izvršava naredbom *php artisan migrate:rollback*. Tablice se izrađuju pomoću *Schema fasade*. Fasada predstavlja klasu koja daje pojednostavljenu sintaksu za rad nad složenijim operacijama. Za izradu tablica, te dodavanje redova koristi se fasada naziva *Schema*. Slika 2.8. prikazuje primjer migracije za dodavanje novoga reda u postojeću tablicu navođenjem tipa i imena retka u *Up* metodi. Nakon izrade tablice, potrebno je potvrditi promjene naredbom *php artisan migrate* čime će se promjene postaviti na bazu podataka.

```
class AddFileTypeToFiles extends Migration
{
    public function up()
    {
        Schema::table('files', function($table)
        {
            $table->string('file_type');
        });
    }
    public function down()
    {
        Schema::table('files', function($table)
        {
            Schema::dropIfExists('file_type');
        });
    }
}
```

Sl. 2.11. *Shema za izradu migracije AddFileTypeToFile*

Pristupanje podacima iz baze podataka implementirano je *Eloquent* ORM tehinkom. *Eloquent* je tehnika objektno-relacijskog mapiranja (engl. Object-relation mapping, skraćeno ORM) koja pojednostavljuje rukovanje bazom podataka. Temelj ORM tehnike je prikaz podataka objektno orijentiranim pristupom. Tablice u bazi podataka prikazuju se kao klase, pa više nije potrebno pisati složene SQL upite, već je dovoljno podacima iz baze podataka rukovati kao objektima i njihovim metodama. *Eloquent* povezuje tablice sa modelima (objektima). Primjerice *User::find(1)->username* naredba pronaći korisnika iz baze podataka rednoga broja jedan (tablica je povezana sa modelom *User*) i ispisati njegovo korisničko ime.

Model se izrađuje naredbom *php artisan make:model ime_modela*. *Eloquent* tehnika nije obavezna, već je moguće pisati i tradicionalne SQL upite za pristup podacima. Slika 2.8. prikazuje

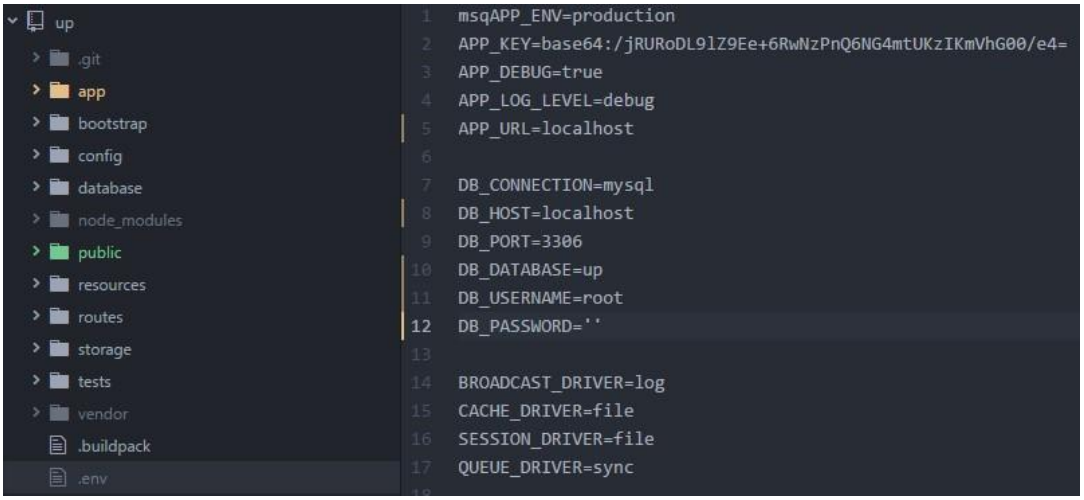
jednostavni SQL upit nad bazom podataka za dohvaćanje svih javnih datoteka određenoga vlasnika datoteke (specificiranog preko njegovog identifikacijskog broja).

```
$files = File::where('author_id', '=', $id)
       ->where('file_status', '=', 'public')->orderBy('created_at', 'desc')->paginate(25);
```

Sl. 2.12. *Pristup podacima objekta File uz određene uvjete*

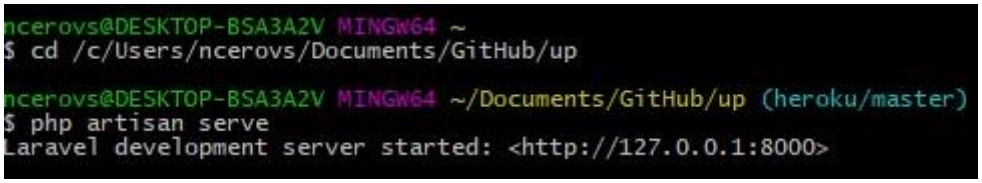
3. IZRADA APLIKACIJE

Izrada aplikacije započinje instalacijom *Laravel* radnoga okruženja. Nakon instalacije potrebno je podesiti konfiguracijske datoteke kako bi aplikacija mogla komunicirati sa bazom podataka i HTTP serverom. Za početak, aplikacija je podešena za rad na lokalnom serveru (engl. *localhost*), no prilikom postavljanja aplikacije na neki od web servera, ovu datoteku je potrebno izmijeniti odgovarajućim podacima dobivenim od odabranog web hostinga. Na slici 3.1. prikazan je dio *.env* datoteke koji sadrži osnovne podatke o web aplikaciji. Potrebno je unijeti podatke o serveru (*APP_URL*) i bazi podataka (adresa baze, korisničko ime i lozinka, te naziv baze podataka).



Sl. 3.1. *Podrešena .env datoteka*

Nakon uspješnog podešavanja datoteke *.env*, potrebno je pokrenuti web server kroz komandnu liniju naredbom "*php artisan serve*". Ukoliko nije bilo poteškoća prilikom pokretanja servera, naredba će ispisati adresu kojom se pristupa aplikaciji. Na slici 3.2. prikazan je primjer izvršavanja naredbe i ispis adrese za pristup aplikaciji. U slučaju rada sa XAMPP alatom, aplikacija će biti poslužena na *localhost* adresi.



Sl. 3.2. *Pokretanje aplikacije kroz komandnu liniju*

Pristupom adresi *localhost* otvara se prazna stranica, te razvoj aplikacije može započeti. U sljedećem koraku potrebno je izraditi sustav za prijavu i registraciju korisnika, početnu stranicu i slično.

3.1. Izrada obrazaca za prijavu i registraciju

Za izradu sučelja za prijavu (engl. login) i registraciju nije potrebno izrađivati sve dijelove *ručno*, već *Laravel* radno okruženje nudi gotovo rješenje kroz naredbu *php artisan make:auth*. Naredba će kreirati sve potrebne datoteke za validaciju korisnika kao što je kontroler, datoteke za bazu podataka (migracije), *view* (pogled) za prijavu i registraciju, te model *User* kojim se pristupa korisničkim podacima iz baze podataka. Sljedeći korak je modifikacija pogleda za prijavu i registraciju prema željenoj funkcionalnosti. Svaki od pogleda sadržava obrazac (formu) za unos podataka koja korisničke podatke šalje kontroleru. U obrasce se uz postojeće inpute mogu dodavati i drugi, ali je potrebno prethodno modificirati bazu podataka (izraditi potrebne migracije za dodatne stupce u tablicu) kako bi ti podaci bili uspješno pohranjeni. Pritiskom tipke submit, podaci se kontroleru šalju specificiranjem putanje unutar parametra *action* čime se izvršava kontroler koji će te podatke obraditi. U slučaju putanje login (vidljivo na slici 3.3.), pozvati će se *LoginController*, a u slučaju putanje register, pozvati će se *RegisterController* sa svojim specifičnim metodama. Spomenuti kontroleri će podatke pohraniti u bazu podataka (uz prethodnu validaciju), odnosno, usporediti sa postojećima i prijaviti korisnika u aplikaciju.

```
<form class="auth-form" role="form" method="POST" action="{{ route('login') }}">
    {{ csrf_field() }}

    <h1 class="form-heading">Log in</h1>

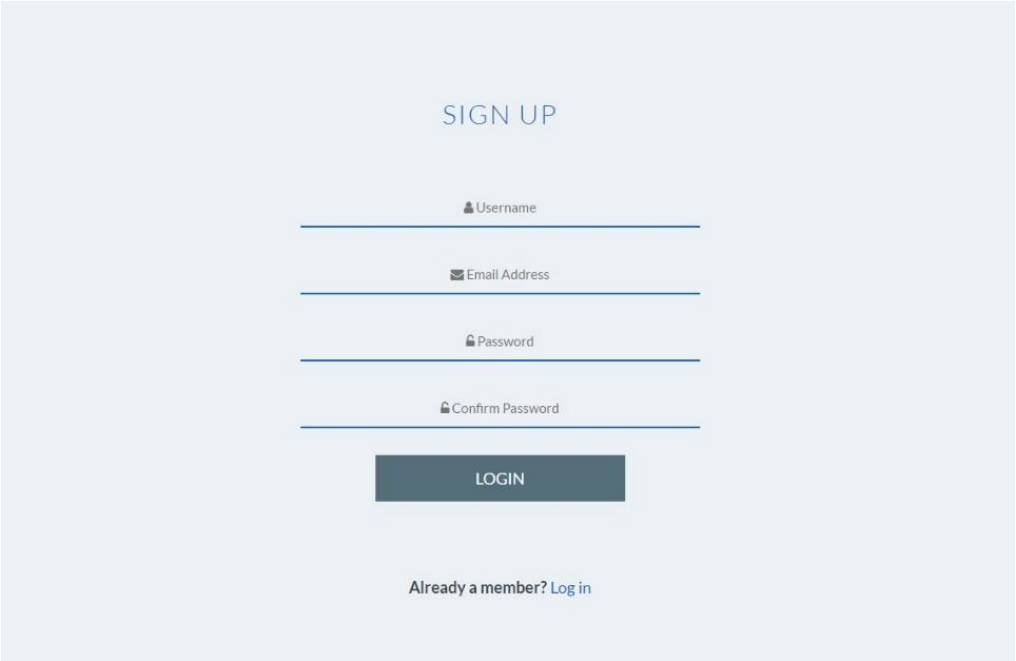
    <div class="form-group{{ $errors->has('email') ? ' has-error' : '' }}">
        <p class="wrapper">
            <input id="user" type="text" class="form-control" name="name" value="{{ old('name') }}" required autofocus>
        </p>
    </div>

    <div class="form-group{{ $errors->has('password') ? ' has-error' : '' }}">
        <p class="wrapper">
            <input id="password" type="password" class="form-control" name="password" required autofocus>
        </p>
    </div>

    <div class="form-group">
        <button type="submit" class="form-btn">Login</button>
    </div>
</form>
```

Sl. 3.3. *Izgled obrasca za prijavu korisnika i unos e-mail adrese*

Nakon izrađenih pogleda, potrebno ih je prilagoditi prema željenom dizajnu i prikazu na svim uređajima i svim veličinama zaslona. Slika 3.4. prikazuje izgled stranice za registraciju korisnika.



Sl. 3.4. *Obrazac za registraciju korisnika*

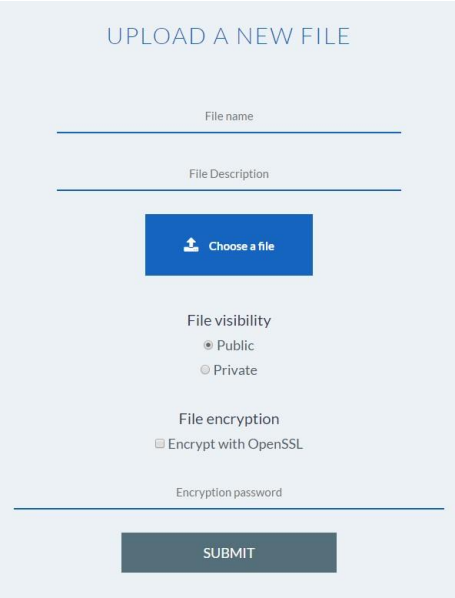
3.2. Prijenos datoteka na server

Prijenos datoteka glavni je dio ove aplikacije. Zbog djelotvornije zaštite datoteka, korisnici moraju imati kreiran korisnički račun i biti prijavljeni u aplikaciju prije pohrane datoteka na server. Prije izrade obrasca za prijenos datoteke, potrebno je izraditi model "*File*" naredbom "*php artisan make:model File*", te tablicu u bazi podataka "*files*". Na već ranije objašnjen način kreira se migracija, te se tablica postavlja na bazu podataka naredbom "*php artisan migrate*". Prilikom izrade migracije, potrebno je odabrati pravilan tip podataka. Također, kako bi se povezao vlasnik datoteke sa pohranjenom datotekom na serveru, potrebno je povezati strani ključ (engl. *foreign key*) sa stupcem "*author_id*" unutar tablice. Navedena metoda će omogućiti pristup podacima o korisniku kroz model *File*. Slika 2.5. prikazuje shemu za izradu tablice *files*. Svaka datoteka će u tablici imati osnovne informacije kao što je jedinstveni identifikacijski ključ (*id)*, naziv (*title*), opis (*description*), putanja do datoteke na serveru (*path*), id vlasnika datoteke (*author_id*), te veličina datoteke (*filesize*). Kasnije, sa dodavanjem funkcionalnosti enkripcije datoteka, ova tablica je dodatno proširena.

```php
public function up()
{
    Schema::create('files', function (Blueprint $table) {
        $table->increments('id');
        $table->string('title');
        $table->string('description')->nullable();
        $table->string('path')->unique();
        $table->integer('author_id')->unsigned();
        $table->foreign('author_id')->references('id')->on('users');
        $table->string('filesize');
        $table->timestamps();
    });
}
```

Sl. 3.5. *Izrada tablice za informacije o datotekama*

Nakon izrade tablice i modela za datoteke, sljedeći korak je izrada kontrolera i obrasca za pohranu na server. Obrazac se nalazi u pogledu naziva "*uploadfile.blade.php*". Za prijenos datoteka obavezno je odabrati željenu datoteku sa računala ili mobilnog uređaja, unijeti njeno ime i opis, odabrati vidljivost datoteke (javna ili privatna), te opcionalno, zaštitu datoteke OpenSSL enkripcijom pri čemu je potrebno unijeti i željenu lozinku. Ukoliko je vidljivost datoteke postavljena na javno (engl. *public*), biti će vidljiva svim korisnicima, a ukoliko je datoteka privatna, datoteka će biti vidljiva samo vlasniku.



Sl. 3.6. *Obrazac za prijenos datoteke*

Pritiskom na tipku *submit* podaci se šalju kontroleru "*FileController*". Zadaća ovoga kontrolera je izvršavanje svih radnji oko datoteka kao što je pohrana, brisanje i preuzimanje datoteka. Rukovanjem putanjama izvršiti će se javna metoda *create* koja pripada kontroleru

"*FileController*". Za pristup podacima iz obrasca potrebno je navesti parametar "*Request $request*". Navedena metoda će izvršiti provjeru valjanosti primljenih podataka, te pohraniti podatke u bazu podataka. Metoda će kroz if petlju utvrditi da li je korisnik odabrao enkripciju datoteke ili nije, te će se na temelju tog podatka izvršiti pohrana. Slika 3.7. prikazuje dio metode *create* koja pohranjuje podatke na bazu podataka – kreira se nova instanca modela *File* koji sadrži podatke o datoteci, te se naposljetku svi podaci pohranjuju *save()* naredbom.

```
$file = new File(array(
    'title' => $request->input('title'),
    'description' => $request->input('description'),
    'author_id' => Auth::user()->id,
    'file_type' => $fileType,
    'filesize' => $file->getClientSize(),
    'path' => $path,
    'file_status' => $request->input('file_status'),
    'enc_status' => $enc_status,
    'enc_pass' => $enc_pass,
    'vector' => $iv,
    'org_name' => $fileOrgName
));

$file->save();
```

Sl. 3.7. *Pohrana na bazu podataka*

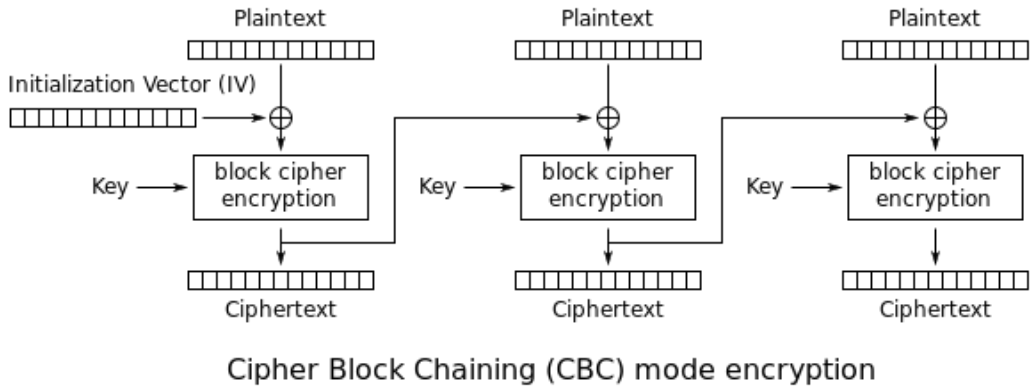
3.2.1. Enkripcija datoteke

Ukoliko je korisnik odabrao enkripciju datoteke i upisao ključ za enkripciju duži od šest znakova, metoda *create* će *if* petljom izvršiti kod za pohranu datoteke uz enkripciju. Odabrana metoda enkripcije je OpenSSL. OpenSSL enkripcija izvedena je funkcijom *openssl_encrypt()* koja prima pet parametara. Parametri potrebni za kriptiranje datoteke su sljedeći:

- Datoteka koja se kriptira,
- Metoda enkripcije,
- Ključ za enkripciju,
- OPENSSL_RAW_DATA,
- Inicijalizacijski vektor

Datoteka koja se kriptira predaje se kao prvi parametar. Podaci datoteke dobiveni su kroz *fopen* i *fread* naredbe (otvaramo datoteku, te čitamo njen sadržaj), te pohranjeni u *$plaintext* varijablu koja se predaje funkciji *openssl_encrypt*.

Metoda enkripcije je parametar koji predstavlja algoritam (engl. *cipher*) koji se koristi za enkripciju. Odabrani algoritam korišten u ovoj aplikaciji je AES-256-CBC. AES je kratica za napredni enkripcijski standard (engl. *Advanced Encryption Standard*) koji izvorne podatke dijeli u blokove od 128 bita, te radi sa veličinama ključa od 128, 192 i 256 bita. AES je simetrični tip enkripcije (koristi isti ključ za kriptiranje i dekriptiranje). Veličina ključa određuje broj transformacijskih ciklusa koji izmjenjuju sadržaj originalne datoteke, te naposljetku pohranjuju datoteku u kriptiranome obliku [14]. Kod 256 bita, kriptiranje se odvija u 14 koraka. Svaki transformacijski ciklus (serija) sastoji se od nekoliko koraka koji na razne načine izmjenjuju sadržaj datoteke. Može se izvršavati zamjena sa drugim znakovima, pomicanje redova nekoliko puta, miješanje redova, itd. Naposljetku, odabran je CBC način enkripcije (skraćeno od *Cipher Block Chaining*) kod kojega svaki blok datoteke od 128 bita prolazi kroz logičku operaciju XOR gdje se uspoređuje sa prethodno kriptiranim blokom. Nakon usporedbe, novi blok se kriptira. Enkripcija završava nakon što se usporede svi blokovi. Slika 2.8. prikazuje CBC način rada.



Sl. 3.8. *CBC način enkripcije [15]*

Inicijalizacijski vektor (skraćeno IV) je nasumično generirani niz znakova koji se mora razlikovati za svaki enkripcijski ključ. Kod CBC načina enkripcije potreban je za početak enkripcije (inicira algoritam za enkripciju), jer početni (nulti) blok algoritma nema prethodnoga bloka za usporedbu. Inicijalizacijski vektor se još naziva i inicijalizacijskim (nultim) blokom. Ključan je kako bi svaki kriptirani blok bio jedinstven. Funkcija "*openssl_random_pseudo_bytes(duljina_vektora, false)*" generira inicijalizacijski vektor, a funkcija *"openssl_cipher_iv_length(metoda_enkripcije)*" određuje duljinu inicijalizacijskog vektora, te ovisi o metodi enkripcije. Slika 2.9. prikazuje cijeli proces kriptiranja datoteke.

```php
$this->validate($request, [
    'enc_pass' => 'required|min:6'
]);
$enc_pass = $request->input('enc_pass');
$enc_status = $request->input('enc_status');
$method = "AES-256-CBC";
$iv_length = openssl_cipher_iv_length($method);
$iv = sha1(openssl_random_pseudo_bytes($iv_length, false));
$request->file('file')->move(
    base_path() . '/public/uploads/encrypted/', $randomStr . $fileOrgName
);
$path = base_path() . '/public/uploads/encrypted/' . $randomStr . $fileOrgName;
$input = fopen($path, 'rb');
$plaintext = fread($input, filesize($path));
fclose($input);
$output = fopen($path, 'w');
$ciphertext = openssl_encrypt($plaintext, $method, $enc_pass, OPENSSL_RAW_DATA, $iv);
fwrite($output, $ciphertext);
fclose($output);
```

Slika 3.9. *Kriptiranje datoteke*

3.2.2. Preuzimanje i dekripcija datoteke

U slučaju preuzimanja datoteke koja je zaštićena OpenSSL enkripcijom, korisnik mora unijeti ključ korišten prilikom kriptiranja. Prije kreiranja metode za dekripciju datoteke, potrebno je izraditi metodu za preuzimanje datoteka. Kod za preuzimanje datoteke prikazan je na slici 3.10.

```php
public function download(Request $request, $id)
{
    $file = File::find($id);
    $user_id = $file->user->id;

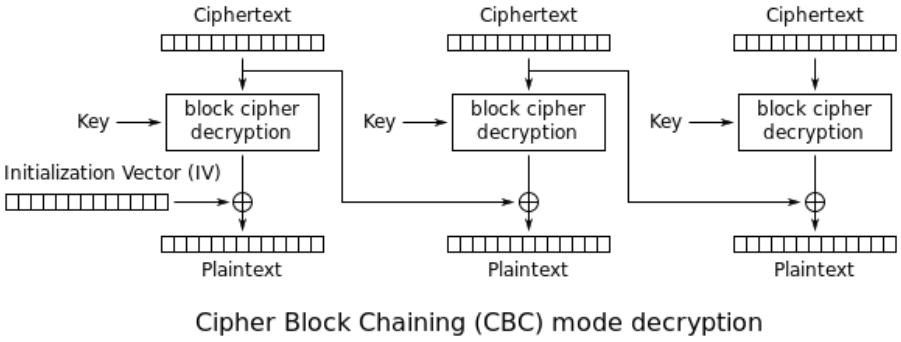
    if ( file_exists($file->path) ) {
        if ($file->enc_status === 'encrypt')
        {
            return view('pages.decryptfile', compact('file'));
        } else {
            return response()->download($file->path);
        }
```

Sl. 3.10. *Preuzimanje datoteke*

Metoda će provjeriti da li zatražena datoteka postoji (*file_exists()* PHP funkcija), te da li je zaštićena enkripcijom ili nije. Ukoliko je datoteka zaštićena, metoda će izvršiti pogled sa obrascem za unos ključa, a ukoliko datoteka nije zaštićena, izvršiti će preuzimanje kroz funkciju *download(lokacija_datoteke).*

Metoda za dekripciju datoteke izvršava se pomoću funkcije "*openssl_decrypt()*" koja prima već navedene parametre iz funkcije za enkripciju. Jedina razlika je što umjesto izvorne datoteke,

ova funkcija prima kriptiranu datoteku (*ciphertext*). Ključ i inicijalizacijski vektor moraju biti identični onima korištenima prilikom kriptiranja. Shema dekripcije pomoću CBC načina rada prikazana je na slici 2.11. Kako bi datoteka na serveru ostala kriptirana, potrebno je kreirati kopiju kriptirane datoteke, te nju dekriptirati i preuzeti. Nakon preuzimanja potrebno je obrisati pomoćnu datoteku funkcijom koju pruža Laravel radno okruženje: "*deleteFileAfterSend(true)*".



Sl. 3.11. *Dekripcija datoteke CBC načinom rada* [15]

3.3. Brisanje datoteke

Korisnik može uklanjati svoje datoteke kroz posebnu stranicu „*Edit files*“ u kojoj su prikazane sve datoteke pojedinoga korisnika sa opcijama za uređivanje i brisanje. Ukoliko je korisnik zatražio brisanje datoteke, pokreće se *SweetAlert2* prozor sa upozorenjem, te je potrebno potvrditi akciju klikom na tipku *"Delete"*. Klikom na tipku korisnik potvrđuje brisanje datoteke, te se pokreće *Ajax* zahtjev. Ajax zahtjev će izvršiti putanju *localhost/delete/file/id* koja poziva kontroler i njegovu metodu za brisanje datoteke.

```
$.ajax({
    $(this).css('display', 'none');
    url: $(file).attr('href')
});
```

Sl. 3.12. *Ajax poziv za brisanje datoteke*

Metoda za brisanje datoteke naziva se *deleteFile,* te prima parametar *id* pomoću kojega lako nalazimo pojedinu datoteku koju korisnik želi ukloniti. Parametar *id* dobiva se kroz rukovanje putanjama jer *Laravel* omogućava uzimanje informacija iz URL-a. Primjerice, *id* u slućaju putanje *localhost/delete/file/11* biti će 11. U nastavku, metoda će pronaći datoteku i vlasnika datoteke, te

kroz *Gate* fasadu provjeriti da li korisnik ima ovlasti za brisanje datoteke. Ukoliko korisnik koji šalje zahtjev nije vlasnik datoteke, *Gate* fasada neće dozvoliti daljnje izvršavanje. Posljednji korak je uklanjanje putanje sa baze podataka (*unlink()* funkcija) i uklanjanje datoteke sa servera (*$file->delete()*). Slika 3.13. prikazuje navedenu metodu.

```
public function deleteFile($id)
{
    $file = File::find($id);
    $user_id = $file->user->id;

    if ( file_exists($file->path) ) {

      if (Gate::forUser(Auth::user())->allows('temper', $file)) {
          $file_path = $file->path;
          unlink($file_path);
          $file->delete();
      }
```

Sl. 3.13. *Metoda za brisanje datoteke*

3.4. Prikaz javnih korisničkih datoteka

Datoteke pohranjene kao javne nisu zaštićene enkripcijom, te su vidljive i dostupne svim korisnicima na odvojenoj stranici. Podaci o datotekama prikazani su kroz kontroler *HomeController*, te metodu *recentFiles*. Metoda će predati podatke o svim javnim datotekama sortiranim prema datumu kreiranja na serveru, te će prikazati po 25 datoteka po stranici pomoću paginacije. *Eloquent ORM* omogućava jednostavno dohvaćanje svih podataka, potrebno je navesti model *File*, te odrediti po kojim parametrima će se tražiti datoteke i način sortiranja. Naposljetku, podaci se predaju pogledu *files* naredbom *return*.

```
public function recentFiles()
{

    $files = File::where('file_status', '=', 'public')->orderBy('created_at', 'desc')->paginate(25);

    return view('pages.files')
          ->with('files', $files);
}
```

Sl. 3.14. *Metoda za ispis svih javnih datoteka*

Podaci o datotekama se predaju pogledu "*files.blade.php*" gdje će se koristiti u HTML tablici za prikaz. Kako bi podaci bili pregledniji, korišten je jQuery dodatak *DataTables* koji omogućava sortiranje tablice po abecedi, veličini datoteke i mnogim drugim parametrima.

```
var filesTable = $('.files-table').DataTable({
    "sDom": '<"top">rt<"bottom"p><"clear">',
    "paging": false,
    "order": [[4, "table-date"]],
    "autoWidth": false,
});
```

Sl. 3.15. *Implementacija dodatka DataTables*

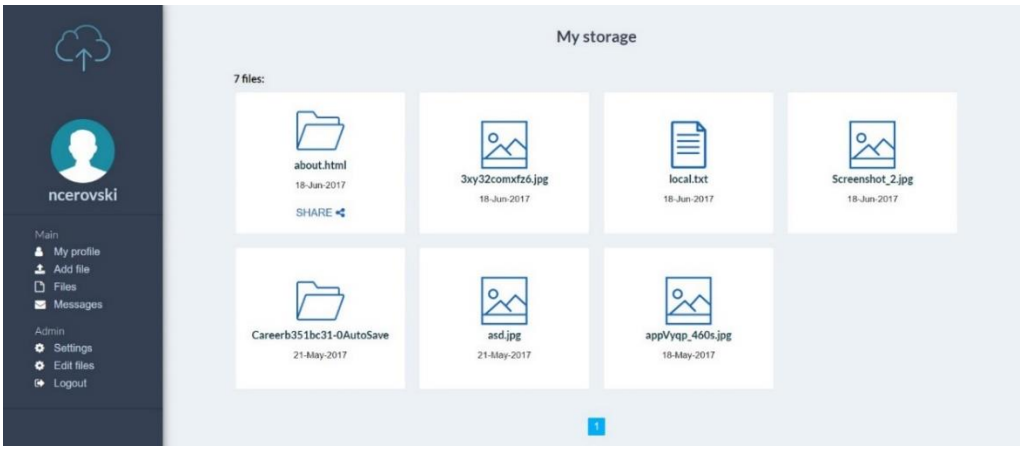
3.5. Izrada i dizajn korisničkog profila

Korisnički profil je početna stranica nakon prijave u aplikaciju gdje su prikazane sve datoteke pojedinoga korisnika. Uz svaku datoteku, dodane su funkcionalnosti preuzimanja i slanja datoteke drugim korisnicima. Svakoj datoteci pridružena je odgovarajuća ikona koja opisuje tip datoteke (npr. slika ili tekst dokument). Ikone se pridružuju pomoću JavaScript *switch(case)* uvjeta. Slika 3.16. prikazuje dio *switch(case)* uvjeta. Tip datoteke pohranjen je u bazi podataka, te se svakoj datoteci prikazanoj u HTML dokumentu pridružuje i *data-type* atribut koji sadrži tip datoteke (pridružen PHP-om), te je potreban za JavaScript dio koda.

```
var type = $(this).attr('data-type');
var file = $(this).find('.file-anchor');
switch(type) {
 case type = 'png':
  $(file).prepend("<a href=<img src='../images/files/picture.svg' alt='image'>");
  break;
 case type = 'jpg':
  $(file).prepend("<img src='../images/files/picture.svg' alt='image'>");
  break;
```

Sl. 3.16. *Primjer pridruživanja ikona sa odgovarajućim tipom datoteka*

Ukoliko korisnik promatra vlastiti profil, prikazane su sve datoteke uključujući i privatne. U suprotnom, korisnik vidi samo javne datoteke. Moguć je prikaz prvih 25 datoteka, a ostatak se prikazuje paginacijom. Slika 3.17. prikazuje korisnički profil.



Sl. 3.17. *Korisnički profil*

3.6. Slanje poruka

U aplikaciju je ugrađen sistem slanja poruka kao rješenje za razmjenu datoteka. Korisnik željenu datoteku može podijeliti sa jednim odabranim korisnikom kroz obrazac. Prelaskom miša preko određene datoteke pojavljuje se opcija za slanje datoteke, te se klikom otvara obrazac. Slanjem poruka, korisnik može dijeliti i privatne datoteke koje inače nisu vidljive i dostupne ostalim korisnicima. Uz poslanu poruku na aplikaciji, korisniku pristiže i e-mail obavijest o pristigloj poruci. Za potrebe slanja poruka izrađen je kontroler "*MessageController*" sa metodama za slanje poruke, prikaz svih poruka, prikaz pojedinačne poruke i brisanje poruka.

Slanje poruke ostvareno je preko obrasca kroz koji korisnik unosi naslov poruke, odabire korisnika kojemu će poruka biti poslana, te opcionalno, unosi kratki tekst (npr. ključ za enkripciju). Podaci iz obrasca obrađuju se u metodi "*sendFile*", te se pohranjuju na bazu podataka u novu tablicu naziva "*messages*". Svaka poslana poruka okida *Laravel fasadu Mail* koja će korisniku poslati e-mail obavijest o pristigloj poruci (fasada je ugrađena klasa u *Laravel* radnom okruženju). Fasadi *Mail* potrebno je predati e-mail adresu korisnika kojemu se šalje e-mail - *Mail::to(e-mail adresa).* Moguće je navesti i veći broj korisnika (*cc, bcc*). Sljedeći korak *Mail* fasade je *send* metoda koja šalje poruku i sve parametre koji su navedeni u *SendMail* objektu (npr. ime pošiljatelja, ime primatelja, naziv datoteke i slično). Parametri su proizvoljni, te broj parametara nije ograničen. E-mail poruka šalje se kroz *smtp driver* (*smtp.gmail.com*). Slanje e-mail obavijesti prikazano je na slici 3.17.

```
Mail::to($receiver_user->email)
  ->send(new SendMail(
    $sender_user->name, $receiver_user->name, $file_name, $message_text, $download_url
  )
);
```

Sl. 3.18. *Slanje e-mail obavijesti*

3.7. Izmjena podataka

Korisnicima je pružena mogućnost uređivanja i brisanja podataka o datotekama, porukama i vlastitim podacima. Korisnici tako mogu mijenjati ime datoteke, opis datoteke, svoje korisničke podatke (e-mail adresa, korisničko ime, ime i prezime), te mogu postaviti svoju korisničku sliku. Izmjena podataka izvršava se kroz odgovarajuće obrasce. Izmjena podataka ostvarena je kroz metodu *updateProfile* kontrolera *UserController* (u slučaju izmjene korisničkih podataka) i naredbom *$user->update($user_data)* gdje je *$user* varijabla u koju je pohranjen model korisnika sa odgovarajućim identifikacijskim brojem, a *$user_data* skup podataka koje je korisnik izmijenio kroz obrazac. Na isti princip izmjenjuju se i podaci o datotekama.

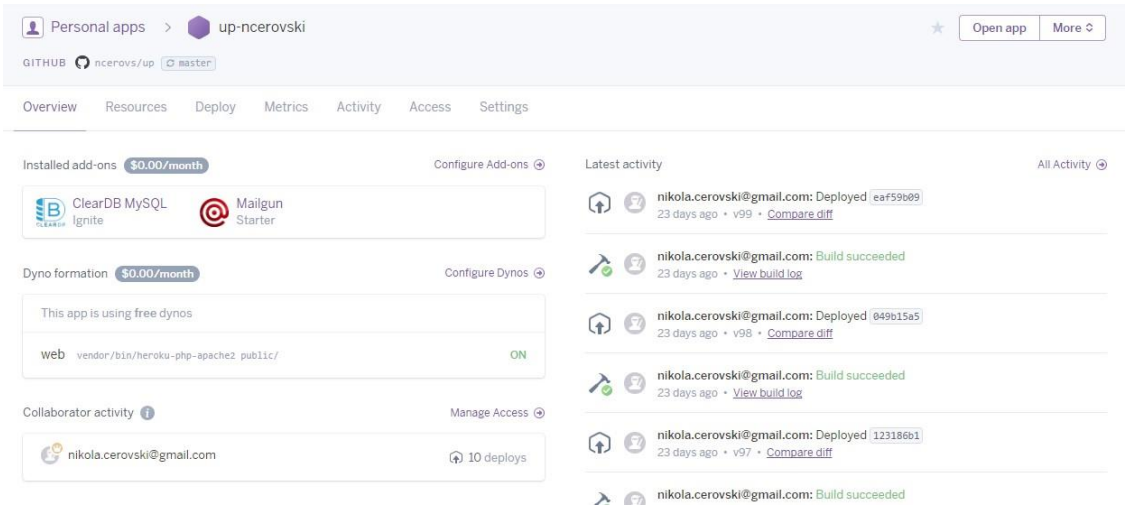
```
$user = User::find($id);
if ($request->file('profile_picture') === null) {
    $path = $user->profile_picture;
} else {
    $fileExtension = $request->file('profile_picture')->getClientOriginalExtension();
    $fileName = bin2hex(openssl_random_pseudo_bytes(7)) . '.' . $fileExtension;
    $request->file('profile_picture')->move(
        base_path() . '/public/images/user/', $fileName
    );
    $path = '/images/user/' . $fileName;
    $user->profile_picture = $path;
}
$user_data = array(
    $user->info = $request->input('info'),
    $user->profile_picture = $path,
    $user->first_name = $request->input('first_name'),
    $user->last_name = $request->input('last_name')
);
$user->update($user_data);
```

Sl. 3.19. *Izmjena korisničkih podataka*

4. POSTAVLJANJE WEB APLIKACIJE NA WEB SERVER

Za web server na kojemu će aplikacija biti smještena odabrana je *Heroku Cloud* platforma. *Heroku* platforma uz hosting koji se plaća, nudi i besplatan cloud hosting za eksperimentiranje sa aplikacijama. Platforma je pogodna za rad sa mnogim programskim jezicima, te mnogim radnim okruženjima uključujući i *Laravel*.

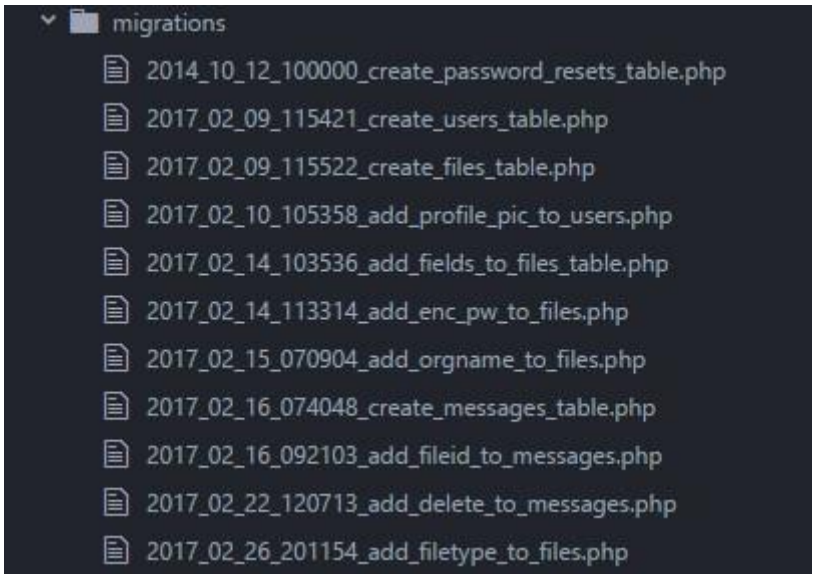
Za instaliranje aplikacije na server, potrebno je na računalu instalirati *Heroku CLI* (kratica za *Heroku Command Line*) [16] i aplikaciju postaviti na *GitHub*. Kroz naredbenu liniju potrebno je odabrati direktorij aplikacije - *GitHub* repozitorij projekta, te upisati naredbu "*heroku create*". Naredba će instalirati aplikaciju na *Heroku cloud* platformi i ispisati adresu aplikacije na webu (npr. ime-aplikacije.herokuapp.com). Kako bi aplikacija funkcionirala, potrebno je podesiti *.env* datoteku sa novim podacima kako je objašnjeno na početku trećeg poglavlja. Sve promjene na aplikaciji koje su predane na *GitHubu*, biti će reflektirane i na samoj *Heroku* aplikaciji. Slika 3.20. prikazuje kontrolno sučelje web aplikacije.



Sl. 3.20. *Heroku kontrolno sučelje*

Nakon postavljanja aplikacije na platformu, slijedi podešavanje baze podataka. Za početak je potrebno instalirati dodatak (engl. add-on) *ClearDB MySQL* kako bi se dobila mogućnost rada sa MySQL bazom podataka na platformi. Navedeni dodatak omogućava izradu baze podataka i tablica. Nakon instalacije dodatka *ClearDB*, potrebno je izraditi tablice. *Heroku* platforma podržava *Laravel artisan* naredbe, što omogućava instalaciju svih tablica koje su ranije izrađene kroz *Laravel* migracije naredbom "*heroku run php artisan migrate*". Drugi način za izradu tablica

je kroz aplikaciju za rad bazom podataka poput *MySQL Workbench*. Slika 3.20. predstavlja sve tablice (migracije) koje će biti instalirane na serveru nakon naredbe.



Sl. 3.21. *Tablice za bazu podataka*

5. ZAKLJUČAK

U ovom završnom radu izrađena je web aplikacija za pohranu korisničkih datoteka na serveru uz mogućnost zaštite datoteka OpenSSL enkripcijom. Na aplikacijama za razmjenu i pohranu datoteka, izrazito je važno obratiti pažnju na adekvatnu zaštitu korisničkih datoteka od neželjenog pristupa. Korisnicima je važna privatnost, jer koristeći aplikaciju mogu pohranjivati i povjerljive datoteke i informacije bez straha da će netko pristupiti tim podacima.

Metoda enkripcije OpenSSL pruža efikasnu razinu sigurnosti, te ima veliki izbor parametara. Međutim, kroz razvoj ove metode enkripcije (kao i drugih metoda), zabilježeni su slučajevi probijanja ove zaštite, stoga se metode redovito nadopunjuju zakrpama. Prije šire primjene OpenSSL enkripcije, koristila se Mcrypt metoda enkripcije koja je danas zastarjela, te se više ne održava.

Aplikacija je izrađena pomoću *Laravel* radnoga okruženja (engl. *framework*). Izrada aplikacije korištenjem radnih okruženja je brža i efikasnija, jer radna okruženja potiču na izradu aplikacije kroz najmodernije standarde web programiranja. Također, poboljšavaju sigurnost same aplikacije kroz razne, već izrađene metode i funkcije. Aplikacija pruža osnovne mogućnosti rada sa podacima, poput dodavanja, čitanja, brisana i izmjene podataka (engl. *Create Read Update Delete* – CRUD).

LITERATURA

[1] DOCTYPE deklaracije, 1.6.2017.
https://www.w3schools.com/tags/tag_doctype.asp

[2] HTML informacije, 1.6.2017.
https://developer.mozilla.org/en-US/docs/Web/HTML

[3] CSS informacije, 1.6.2017.
https://www.w3schools.com/css/default.asp

[4] Relativne CSS jedinice, 1.6.2017.
https://thecssworkshop.com/lessons/relative-units

[5] CSS media query, 2.6.2017.
https://www.w3schools.com/css/css_rwd_mediaqueries.asp

[6] JavaScript programski jezik, 2.6.2017.
https://en.wikipedia.org/wiki/JavaScript

[7] JavaScript radna okruženja, 2.6.2017.
https://hackernoon.com/5-best-javascript-frameworks-in-2017-7a63b3870282

[8] PHP radna okruženja, 2.6.2017.
http://www.hongkiat.com/blog/best-php-frameworks/

[9] Eloquent ORM, 2.6.2017.
https://laravel.com/docs/5.4/eloquent

[10] Laravel instalacija, 3.6.2017.
https://laravel.com/docs/5.4/installation

[11] Laravel middleware, 3.6.2017.
https://laravel.com/docs/5.4/middleware

[12] Laravel kontroleri, 3.6.2017.
https://laravel.com/docs/5.4/controllers

[13] Laravel blade pogledi, 3.6.2017.
https://laravel.com/docs/5.4/blade

[14] OpenSSL AES metoda, 8.6.2017.
http://searchsecurity.techtarget.com/definition/Advanced-Encryption-Standard

[15] OpenSSL metoda enkripcije, 9.6.2017.
http://searchsecurity.techtarget.com/definition/cipher-block-chaining

[16] Heroku cloud platforma, 18.6.2017.
https://devcenter.heroku.com/articles/heroku-cli

SAŽETAK

U ovome završnom radu obrađena je OpenSSL metoda enkripcije korisničkih datoteke, te je implementirana na web aplikaciji za pohranu datoteka na serveru. Kroz završni rad objašnjene su tehnologije u web programiranju korištenje prilikom izrade aplikacije, kao što su tehnologije na strani klijenta (engl. *front-end*), te tehnologije na strani poslužitelja (engl. *back-end*). Uz osnovne tehnologije, korišteni su i analizirani napredni alati poput *Laravel* PHP radnoga okruženja (engl. *framework*), *jQuery* biblioteke za *JavaScript* programski jezik, te dodataka *SweetAlert2* i *DataTables* za efikasniji prikaz obavijesti i sortiranja tablica. U nastavku završnoga rada, opisane su metode izrade web aplikacije koja je podijeljena po funkcionalnim komponentama. Objašnjena je izrada sustava za prijavu i registraciju korisnika. Dalje, objašnjena je izrada komponenti za prijenos i preuzimanje datoteka. Enkripcija i dekripcija datoteka usko su povezane uz ove metode, te su detaljno analizirane. Uz navedene metode, objašnjen je i sustav za razmjenu poruka koji je zamišljen kao način za razmjenu datoteka, te je implementiran e-mail servis koji šalje obavijest na adresu primatelja o pristigloj poruci. Na kraju je objašnjena instalacija web aplikacije na *cloud* platformi *Heroku*.

Ključne riječi: enkripcija, dekripcija, *Laravel* radno okruženje, web aplikacija

ABSTRACT

The title of this final paper is "*Encrypting user files stored on the server with PHP and OpenSSL encryption*". Its objective is creating a web application which will be used for storing user files on a web server. The files can be, optionally, protected with OpenSSL encryption. Throughout this final paper, basic technologies used in developing this application are described. These technologies can be divided in two main groups - front-end and back-end. Along with these, additional tools like *Laravel* PHP framework, *jQuery JavaScript* library, *SweetAlert2* and *DataTables* add-ons were used and described. Next, the development of the web application was described. Application was divided into components, and each main component is described, starting with user registration and validation. Second component is file uploading and downloading. The file encryption and decryption is located and analyzed in previously mentioned components. For file exchange, the messaging system with e-mail notifications was designed. Users can share files with other users through this system. Finally, implementation of web application on Heroku cloud platform was described.

Keywords: encryption, decryption, Laravel framework, web application

ŽIVOTOPIS

Nikola Cerovski rođen je u Osijeku 4. lipnja 1994. godine. U Našicama pohađa osnovnu školu Dore Pejačević. Također, u Našicama pohađa srednju školu Izidora Kršnjavog, smjer prirodoslovno – matematička gimnazija. Završetkom srednje škole upisuje preddiplomski studij računarstva na Fakultetu elektrotehnike, računarstva i informacijskih tehnologija, a 2014. prelazi na stručni studij elektrotehnike, smjer informatika na istome fakultetu. Znanja iz web programiranja unaprjeđuje na stručnoj praksi u firmi BetaWare.

Potpis:

_____