Online aplikacija za slanje obavijesti u preddefinirano vrijeme

Antunović, Anto

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

Stručni studij

ONLINE APLIKACIJA ZA SLANJE OBAVIJESTI U PREDDEFINIRANO VRIJEME

Završni rad

Anto Antunović

Osijek, 2016.

SADRŽAJ

1. UVOD

Tema završnog rada je Online aplikacija za slanje obavijesti u preddefinirano vrijeme. Cilj aplikacije je osim samog uređivanja i slanja obavijesti omogućiti korisnicima odabir vremena u kojem žele da se prethodno spremljene obavijesti ili poruke pošalju. S obzirom na to da se radi o online ili web aplikaciji, za izradu aplikacije će se koristiti PHP programski jezik. Kako bi se skratilo vrijeme potrebno za izradu aplikacije, odabrana je Laravel razvojna okolina (eng. Laravel *framework*)[1]. Dalje u tekstu će se koristiti riječ *framework*. Glavni dio rada podijeljen je u dvije veće cjeline. U prvoj cjelini su predstavljeni alati koji su korišteni za izradu aplikacije. U drugoj cjelini je predstavljeno postignuto rješenje i opisani su najbitniji dijelovi aplikacije koji su implementirani korištenjem navedenih alata.

1.1 Zadatak završnog rada

Zadatak ovog završnog rada je napraviti web aplikaciju. Aplikacija mora omogućiti korisnicima uređivanje i slanje obavijesti koje će se slati putem emaila ili SMS-a te omogućiti korisniku odabir vremena u kojem će se prethodno spremljene poruke slati.

2. ALATI

U ovom poglavlju su opisane neke bitnije komponente i alati koji su korišteni za izradu aplikacije. S obzirom da se radi o online aplikaciji i tome da je za izradu odabran PHP kao programski jezik, kao serverska komponenta koristi se Apache 2.4 HTTP poslužitelj. Za potrebe lokalnog razvojnog okruženja koristi se XAMPP. XAMPP je besplatan program za Windows operacijski sustav s kojim jednostavno možemo instalirati Apache poslužitelj i ostale alate za potrebe lokalnog razvoja web aplikacija.[2]

2.1 Laravel

Laravel je razvojna okolina otvorenog koda pisana u PHP programskom jeziku. Autor Laravel razvojne okoline je Taylor Otwell. Glavne komponente koje cine ovaj *framework* su preuzete iz Symfony *frameworka*, te je u vrijeme pisanja rada jedan od najpopularnijih *frameworka* za razvoj web aplikacija u PHP programskom jeziku. Projekti koji se rade u Laravel *frameworku* imaju MVC (eng. *Model View Controller*) arhitekturu[3]. Korištenjem MVC arhitekture poslovna logika aplikacije je razdvojena od korisničkog sučelja i samim time imamo puno čitljiviji kod te možemo testirati poslovnu logiku neovisno o korisničkom sučelju. Laravel koristi Eloquent ORM (eng. *Object Relational Mapper*) za rad s bazama podataka. Eloquent ORM eliminira potrebu pisanja SQL upita nad bazom te nam omogućuje pristup podatcima preko PHP objekata na način da jedna klasa predstavlja jednu tablicu u bazi podataka. Na taj način za unos novog reda u tablicu umjesto pisanja SQL upita potrebno je kreirati novu instancu objekta, postaviti vrijednosti varijabli te pozvati odgovarajuću metodu koja će spremiti naš objekt u tablicu.

Aplikacija je pisana u Laravel 5.2 verziji.

2.2 PhpStorm

PhpStorm je komercijalno integrirano razvojno okruženje za PHP jezik tvrtke JetBrains s.r.o . PhpStorm nam pruža skup alata za napredno uređivanje PHP koda. Osim podrške za PHP u vidu automatskog kompletiranja koda i drugih stvari, PhpStorm podržava HTML i JavaScript programske jezike. Osim podrške za programske jezike imamo i podršku za repozitorije koda, npr. Git i SVN. Za pisanje aplikacije je korištena probna verzija PhpStorma 10.0.4.

2.3 Git

Git je distribuirani sustav za upravljanje verzijama koda koji je koristi u programiranju.[4] Autor Git sustava je Linus Torvalds. Stvorio ga je 2005. godine u svrhu razvoja jezgre Linux sustava. Postoji nekoliko online servisa koji služe kao repozitoriji za Git kao što su github.com, gitlab.com i bitbucket.com. Odabran je bitbucket.com iz razloga što pruža stvaranje privatnih Git repozitorija bez naplate.

2.4 Mailgun

Za slanje email poruka odabran je servis Mailgun. Mailgun je komercijalni servis za slanje email poruka. Omogućuje nam slanje email poruka preko SMTP (Simple Mail Transfer Protocol) protokola ili REST APIa. Iako je Mailgun komercijalni servis, preko njega možemo besplatno poslati 10000 email poruka svaki mjesec.[5]

2.5 Nexmo

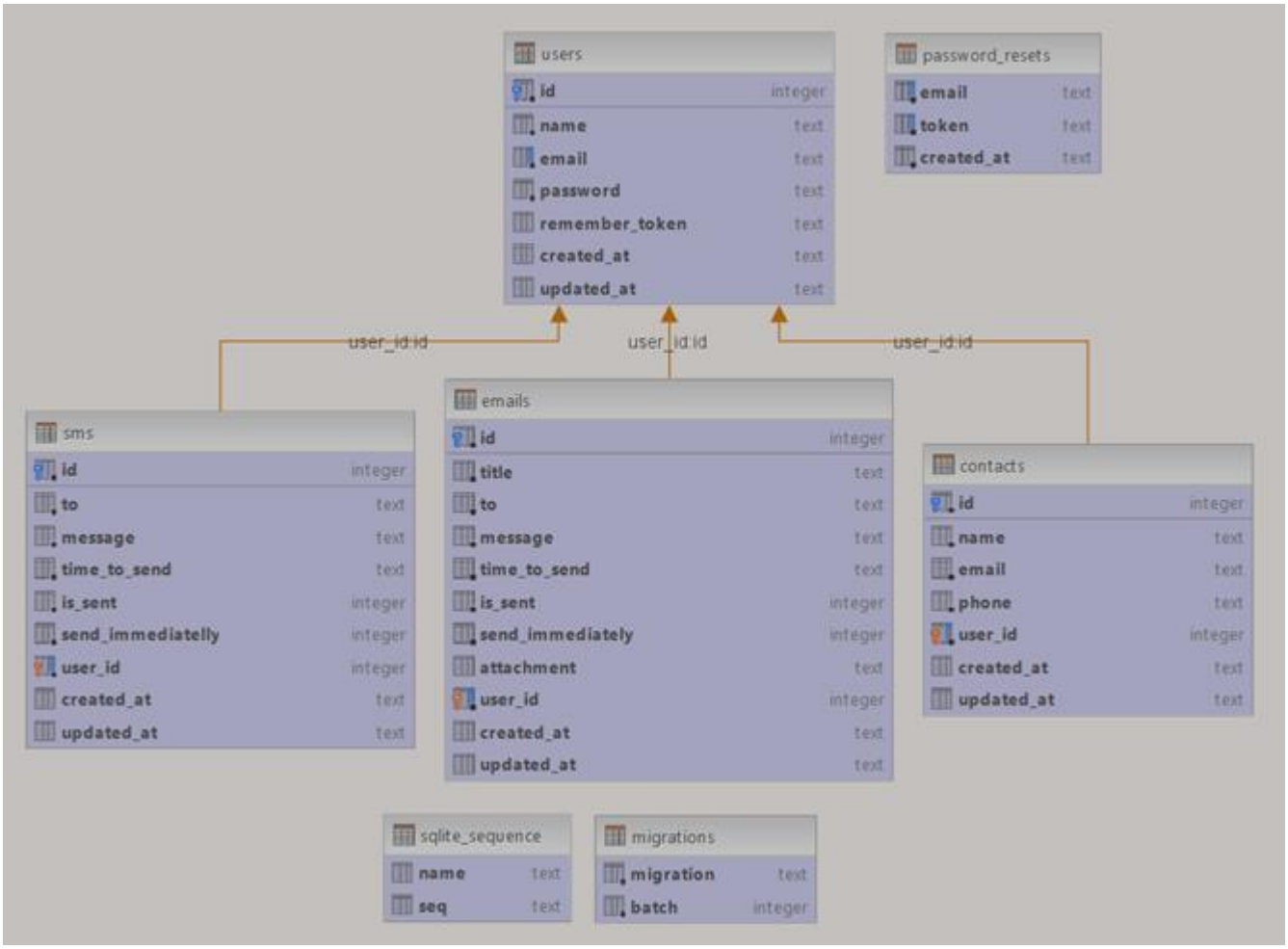
Nexmo je servis sličan Mailgunu. Razlika je u tomu što nam Nexmo omogućava slanje SMS poruka u gotovo sve zemlje svijeta. Također je komercijalan servis, ali omogućuje dodavanje 9 testnih brojeva mobitela na koje na koje dopušteno slanje poruka tokom razvoja aplikacije.[6]

3. APLIKACIJA

Struktura projekta je podijeljena u tri sloja. To su sloj modela, upravljača (eng. *controller)* i pogleda, ili korisničkog sučelja (eng. *view)*. U sloju modela se nalazi poslovna logika, baza podataka te klase entiteta. U upravljačkom sloju se nalaze klase upravljači koje nam služe kao poveznica između korisničkog sučelja i sloja modela. Korištenjem ovakve strukture projekta postignuto je implementacija poslovne logike razdvojena od korisničkog sučelja i rezultat toga je kod koji je puno lakše za održavati i proširivati.

3.1 Baza podataka

Za spremanje podataka aplikacija koristi SQLite bazu podataka. Na slici 3.1. je prikazan dijagram baze podataka na kojem se vide najbitnije tablice koje se koriste u aplikaciji, te veze između njih. Tablica *users* služi za pohranu podataka o korisnicima te se na osnovu nje vrši prijava ili registracija korisnika u aplikaciju. Nadalje, tablice *sms, emails i contacts* su vezane za tablicu *users* pomoću stranog ključa *user_id* kako bi kod unosa novih podataka u tim tablicama mogli jednoznačno označiti kojem korisniku aplikacije pripadaju ti podatci, odnosno koji je korisnik poslao sms ili email poruku, te kojem korisniku pripadaju kontakti iz tablice *contacts*. Tablica *password_resets* je potrebna kako bi se korisniku omogućilo da promijeni lozinku ukoliko to bude potrebno. Tablica *migrations* je posebna tablica koja nam služi za pohranu informacija o ostalim tablicama u aplikaciji koje treba stvoriti i sastavni je dio Laravel razvojnog okruženja.



Sl. 3.1. Baza podataka

3.2 Migracije i modeli

Za stvaranje tablica koje su opisane u prethodnom potpoglavlju, implementirane su klase modela koje će predstavljati tablice i migracije koje služe za stvaranje samih tablica. Migracije u Laravel razvojnoj okolini omogućuju objektno orijentiran pristup stvaranju tablica unutar baza podataka. One su zapravo nacrt tablica podataka i sadrže sve informacije o stupcima, tipovima podataka, primarnom i stranim ključevima. Klase modela predstavljaju entitete unutar samih tablica, i one se koriste za manipulaciju samim podatcima.

```php
class CreateEmailsTable extends Migration
{

    public function up()
    {

        Schema::create('emails', function (Blueprint $table) {
            $table->increments('id');
            $table->string('title');
            $table->json('to');
            $table->text('message');
            $table->dateTime('time_to_send');
            $table->boolean('is_sent');
            $table->boolean('send_immediately');
            $table->string('attachment')->nullable();
            $table->integer('user_id')->unsigned()->index();
            $table->foreign('user_id')->references('id')->on('users')->onDelete('cascade');
            $table->timestamps();
        });

    }

    public function down()
    {
        Schema::drop('emails');
    }
}
```

Sl. 3.2. Primjer migracije

Na slici 3.2. je prikazan primjer jedne migracije. U ovom slučaju radi se o definiciji strukture tablice za email poruke. Iz slike je vidljivo da sve što je potrebno napraviti je naslijediti klasu *Migration* i preopteretiti njene metode *up i down.* Nakon ovog koraka pomoću konzolne naredbe *php artisan migrate* su stvorene tablice u bazi podataka. Na ovaj način su stvorene i sve ostale tablice. Nakon što su implementirane migracije za sve tablice, za potrebe manipulacije podatcima u tablicama implementirane su klase koje će predstavljati entitete. Na slici 3.3. se vidi primjer implementacije za email tablicu.

```php
class Email extends Model
{

    /**
     * The database table used by the model.
     *
     * @var string
     */
    protected $table = 'emails';

    /**
     * Attributes that should be mass-assignable.
     *
     * @var array
     */
    protected $fillable = ['title', 'to', 'message', 'time_to_send', 'is_sent', 'send_immediately', 'user_id'];

    public function user()
    {
        return $this->belongsTo('App\User');
    }
}
```

Sl. 3.3. Primjer modela

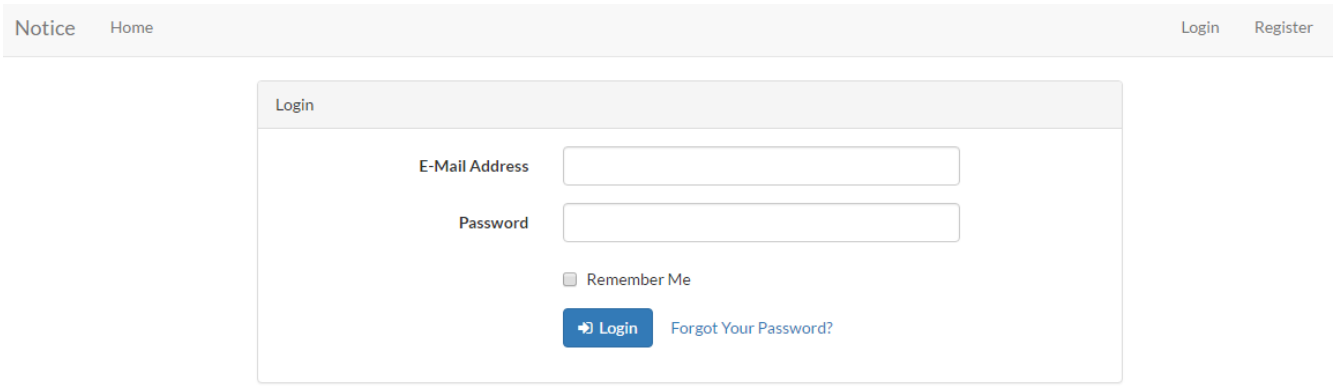
3.3 Putanje i upravljači

Nakon što su implementirane migracije i modeli te stvorene tablice, definirane su rute do pojedinih stranica aplikacije i povezane su sa metodama upravljača koji su prethodno implementirani. Upravljači u *frameworku* služe kao poveznica između sloja modela i korisničkog sučelja, na način da svaki zahtjev koji dođe na neku od stranica aplikacije se delegira upravljaču, odnosno odgovarajućoj metodi upravljača i taj zahtjev se na obradu šalje modelu koji je odgovoran za izvršenje zahtjeva.

3.4 Korisničko sučelje

Korisničko sučelje aplikacije se sastoji od trinaest stranica, gdje imamo po jednu stranicu za prijavu i registraciju korisnika, te izmjenu lozinke. Nakon što se korisnik prijavi dolazi na početnu stranicu gdje su prikazane poveznice na ostale stranice, na kojima može vidjeti kontakte, sms i email poruke koje su spremljene. Korisniku je omogućen unos novih kontakata, sms i email poruka, te njihovo pojedinačno uređivanje.

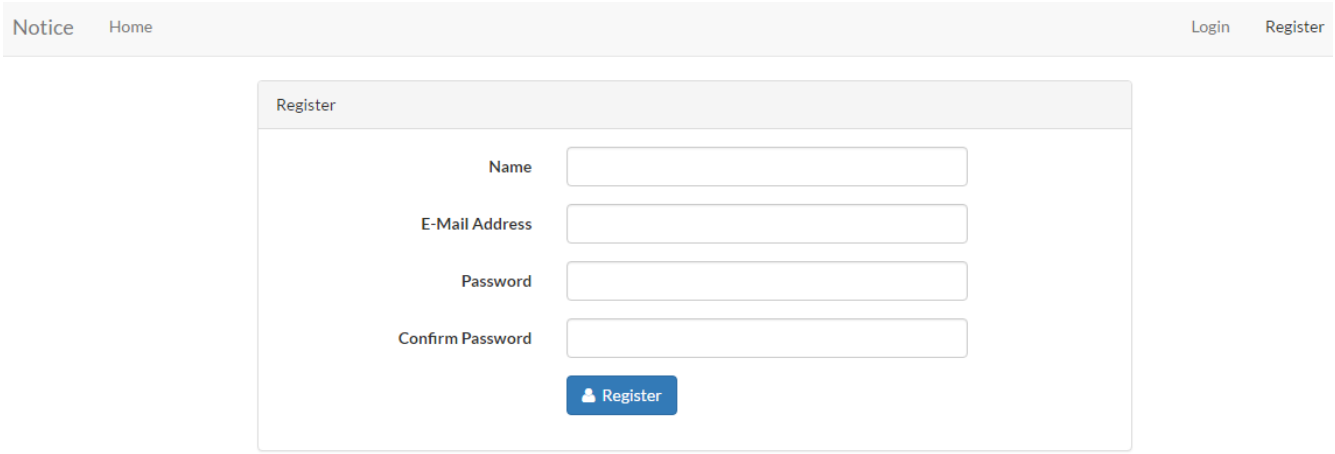
3.4.1 Stranica za prijavu



Sl. 3.5. Stranica za prijavu korisnika

Na slici 3.2. je prikazan izgled stranice za prijavu korisnika na kojoj vidimo formu za unos *email* adrese i lozinke. Nakon što korisnik unese korisničke podatke i klikne na gumb za prijavu podatci iz forme se putem *http* zahtjeva šalju na server, gdje se upitom nad bazom vrši provjera valjanosti podataka. Ukoliko u tablici *users* postoji korisnik sa *email* adresom i lozinkom koja se poklapa sa podatcima unesenim u formu, aplikacija stvara novu sjednicu (eng. *session)* i korisnika preusmjerava na početnu stranicu za korisnike sa aktivnom sjednicom. U slučaju da u bazi podataka ne postoji korisnik sa podatcima koji su uneseni u formu, aplikacija neće stvoriti novu sjednicu te će vratiti korisnika na istu stranicu sa porukom o grešci.

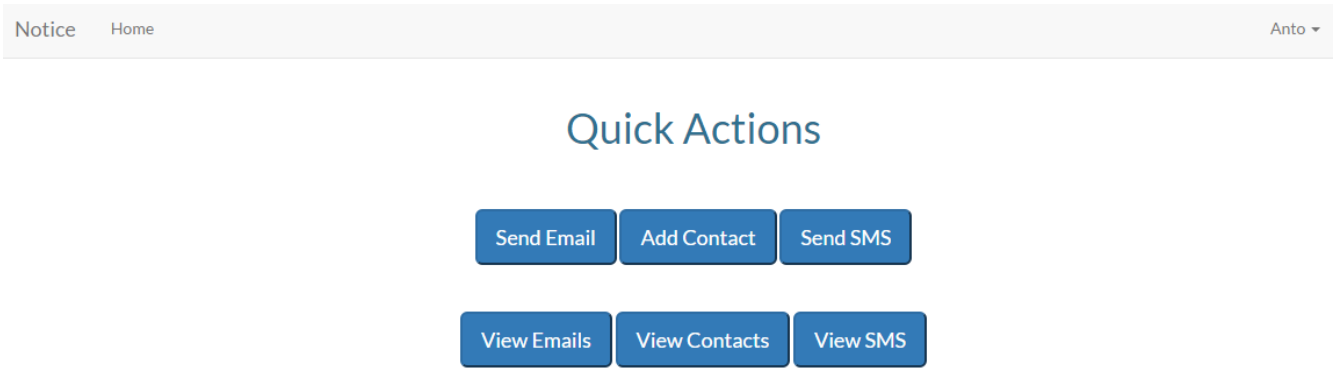
3.4.2 Stranica za registraciju



Sl. 3.6. Stranica za registraciju korisnika

Na ovoj stranici imamo formu stvaranje novog korisnika u bazi podataka, gdje korisnik unosi željeno ime, svoju *email* adresu te željenu lozinku. Iz sigurnosnih razloga korisnikova lozinka se ne šalje u izvornom obliku nego se nad njom vrši enkripcija ili raspršivanje (eng. *hashing*) pomoću *bcript* algoritma te se kao takva šalje i sprema u bazu podataka. Kod kasnije prijave korisnika procedura je ista, radi se enkripcija lozinke i dobivena vrijednost se uspoređuje sa vrijednošću u bazi podataka. Na taj način ni osobe koje imaju pristup bazi podataka ne mogu znati koju lozinku pojedini korisnici koriste.

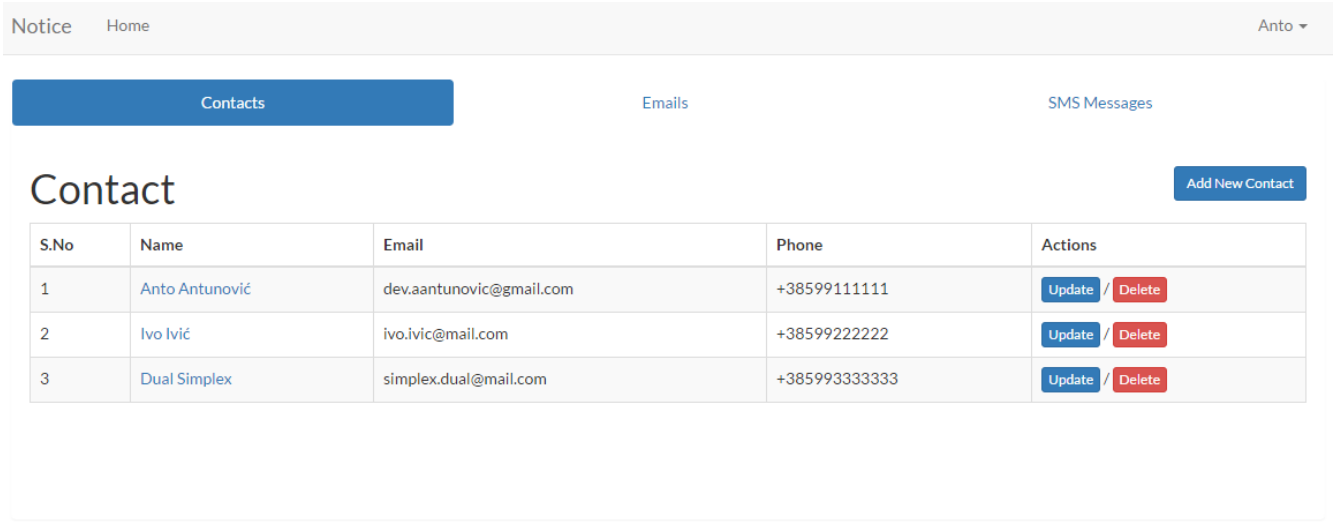
3.4.3 Početna stranica



Sl. 3.7 Početna stranica

Nakon što se korisnik registrira i uspješno prijavi u aplikaciju, aplikacija ga preusmjerava na početnu stranicu gdje se nalaze poveznice na ostale stranice unutar aplikacije. Ova stranica je napravljena kako bi korisnik mogao sa jednog mjesta imati pristup svim bitnim stranicama unutar aplikacije. Na slici 3.7. je prikazan izgled početne stranice, te ona osim poveznica na ostale stranice nema posebnih funkcionalnosti.

3.4.4 Stranice vezane za kontakte

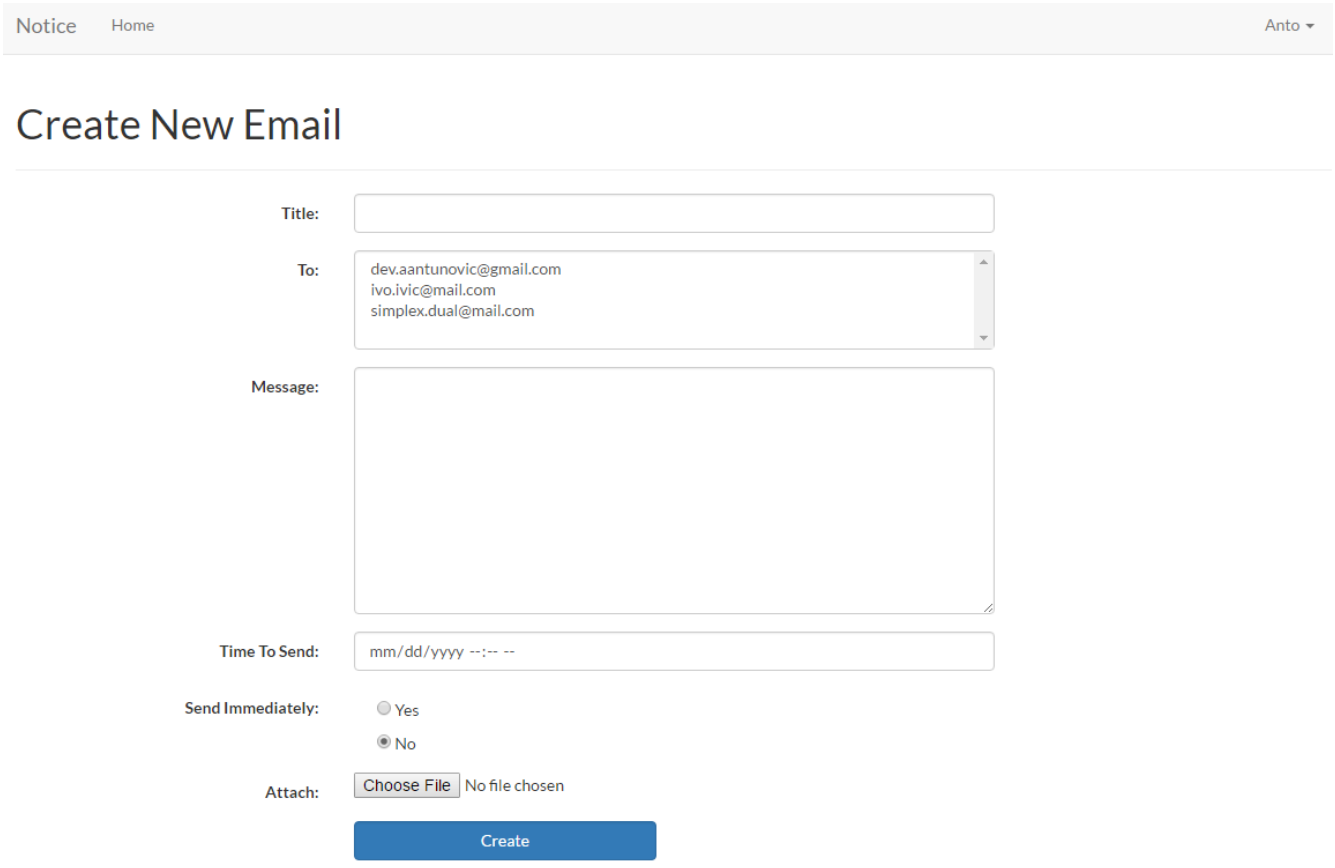


Sl. 3.8 Pregled kontakata

Sa početne stranice preko poveznica na ostale stranice korisnik može doći i na stranicu koja mu omogućuje uvid u postojeće kontakte koji su pohranjeni u bazi. Podatci o kontaktima su prikazani u tabličnom formatu, gdje su prikazani sve važnije informacije o kontaktu. S ove stranice korisnik ima mogućnost prelaska na detalje o pojedinom kontaktu klikom na ime kontakta. Nadalje, u tablicu su dodane i poveznice na stranice za modifikaciju kontakata koje se dinamički generiraju i podatci o kontaktu koji je odabran su već popunjeni unutar forme. Klikom na poveznicu za brisanje kontakt će se obrisati iz baze podataka te će se osvježiti lista.

3.4.5 Stranice vezane za poruke

Nakon što je korisnik spremio barem jedan kontakt u aplikaciju, ima mogućnost slanja sms ili email poruke, te njihov pregled. Stranice za pregled email i sms poruka su gotovo identične. Razlika koja se može vidjeti od strane korisnika je u stupcu *to*. Kod pregleda email poruka se u *to* stupcu prikazuje email adrese na koje će se pojedina poruka poslati, dok se kod pregleda sms poruka prikazuje puno ime kontakta te broj telefona u uglatim zagradama. Kod stvaranja nove sms ili email poruke korisnik ima mogućnost unosa teksta koji će se poslati, unosi vrijeme u kojem želi da se poruka pošalje, može naznačiti da se poruka šalje odmah, i u tom slučaju ne mora unositi vrijeme i datum slanja. U odnosu na stranicu za unos sms poruke, stranica za unos email poruke ima dva dodatna polja za unos, a to su naslov koji je obavezan i polje za dodavanje privitka koji je neobavezan. Na slici 3.9. se može vidjeti izgled stranice za unos email poruke.



Sl. 3.9. Unos nove email poruke

Zbog velikog zauzeća prostora a i sličnosti s prikazanim stranicama ostale stranice nisu prikazane.

3.5 Slanje poruka

Samo slanje poruka je implementirano korištenjem koncepta planiranih naredbi (eng. *scheduled commands)* unutar *frameworka.* Planirane naredbe nam omogućuju definiranje poslova koji će se izvršavati pozivanjem naredbe u konzoli. U tu svrhu su implementirane dvije klase koje nasljeđuju klasu *Command* iz *frameworka, SendEmails i SendSMSMessages.* Na slici 3.10. se vidi implementacija metode *handle* iz klase *SendEmails.* Iz slike je vidljivo da se na početku metode dohvaća kolekcija email poruka iz baze podataka. Nadalje, dobivena kolekcija se filtrira kako bi se eliminirale poruke koje se trebaju poslati u nekom vremenu u budućnosti, te se nakon filtriranja sve poruke koje imaju vrijeme slanja u prošlosti u odnosu na vrijeme izvršavanja šalju preko Mailgun servisa.

```php
/**
 * Execute the console command.
 *
 * @return mixed
 */
public function handle()
{
    //fetch unsent emails, and filter out the ones that don't need to be sent yet
    $unsentEmails = \App\Email::where('is_sent', false)->get();
    $filtered = $unsentEmails->filter(function($email,$key){
        return \Carbon\Carbon::parse($email->time_to_send)->isPast();
    });


    $emails = [];
    foreach ($filtered as $unsentEmail) {

        Log::info("Trying to send emails!");
        $ids = json_decode($unsentEmail->to);
        $contacts = \App\Contact::whereIn('id', $ids)->get();

        //append recipients
        foreach ($contacts as $contact) {
            $emails[] = $contact->email;
        }

        //try to send
        try {

            Mail::raw($unsentEmail->message, function ($message) use ($emails, $unsentEmail) {

                $message->to($emails);
                $message->subject($unsentEmail->title);
                if(!empty($unsentEmail->attachment)) $message->attach($unsentEmail->attachment);
            });

        } catch (Exception $e) {
            Log::error($e->getTraceAsString());
            dd($e->getMessage());
        }

        //update
        $unsentEmail->is_sent = true;
        $unsentEmail->save();
    }
}
```

Sl. 3.11. Metoda handle klase SendEmails

Na gotovo identičan način je implementirana i klasa *SendSMSMessages*, razlika je jedino u tome što se za slanje sms poruka koristi servis Nexmo. Nadalje, nakon što su implementirane klase naredbi, one su

registrirane u klasi *Kernel* kao naredbe koje se trebaju izvršiti svake minute. Način registracije je vidljiv na slici 3.10.

```
class Kernel extends ConsoleKernel
{
    /**
     * The Artisan commands provided by your application.
     *
     * @var array
     */
    protected $commands = [
        Commands\SendEmails::class,
        Commands\SendSMSMessages::class
    ];

    /**
     * Define the application's command schedule.
     *
     * @param  \Illuminate\Console\Scheduling\Schedule  $schedule
     * @return void
     */
    protected function schedule(Schedule $schedule)
    {
        $schedule->command('notice:sendemails')->everyMinute();
        $schedule->command('notice:sendsms')->everyMinute();
    }
}
```

Sl. 3.11. Registracija naredbi za slanje poruka

Naredbe se registriraju na način da njihove klase deklariramo u varijabli *commands* klase *Kernel* te definiramo učestalost njihovog pozivanja u metodi *schedule*.

Nakon što su naredbe registrirane, pozivanjem konzolne naredbe *php artisan schedule:run,* sam *framework* prolazi kroz sve registrirane naredbe i provjerava da li se trebaju izvršiti. Ukoliko je rezultat provjere istinit stvara se nova instanca trenutno promatrane naredbe te se poziva njena metoda *handle.* Kako bi se osiguralo periodično izvršavanje implementiranih naredbi, na serveru pogonjenom Linux operacijskim sustavom na kojem će se nalaziti aplikacija potrebno je postaviti kronološki posao (eng. *cron job)* koji će se izvršavati svaku minutu i pokretati naredbu *php artisan schedule:run.*

Kronološki posao se postavlja izvršenjem sljedeće naredbe u Linux konzoli :

**** * * * * php /path/to/artisan schedule:run >> /dev/null 2>&1***

Napomena, "/path/to/artisan" dio naredbe je potrebno zamijeniti do putanje početnog direktorija aplikacije.

4. Zaključak

Cilj rada je bila izrada online aplikacije s kojom će korisnici moći slati sms i email poruke, s tim da je trebalo uzeti u obzir i to da korisnici možda ne žele slati poruke u trenutku spremanja nego odgoditi slanje za neko buduće vrijeme. Vodeći se zahtjevima aplikacije, odabrana je tehnologija i pripadajući alati s kojima je bilo moguće udovoljiti zahtjevima te su oni predstavljeni u uvodnom dijelu rada.

Nakon toga se pristupilo opisu implementirane aplikacije s osvrtom na arhitekturu i funkcioniranje pojedinih dijelova. S obzirom na to da se odabrana razvojna okolina temelji na *MVC* arhitekturi čija je glavna karakteristika razdvajanje pojedinih funkcionalnosti po slojevima, dan je pregled implementiranih funkcionalnosti po slojevima i to redom od sloja modela gdje se nalazi sama baza podataka te klase koje su njena apstrakcija, upravljačkog sloja gdje se nalaze klase koje služe za zaprimanje zahtjeva i prosljeđivanje zahtjeva u niže slojeve (sloj modela), pa sve do korisničkog sučelja. Na samom kraju je opisana procedura slanja poruka i način na koji je omogućeno slanje istih u preddefinirano vrijeme.

Korištenjem odabrane tehnologije i za nju vezanih alata implementirana je potpuno funkcionalna aplikacija koja je spremna za upotrebu. Jedina ograničenja u pogledu slanja poruka su vezana uz servise koji se koriste za samo slanje poruka. Naime, za razvoj aplikacije su korišteni besplatni računi na Mailgun i Nexmo servisima koji omogućuju slanje deset tisuća email poruka u obračunskom razdoblju od mjesec dana i SMS poruke se mogu slati samo na maksimalno devet brojeva mobilnih telefona koji moraju biti prethodno registrirani na servisu.

5. Literatura

[1] Laravel, https://laravel.com/docs/5.2 - 5.7.2016.

[2] Xampp, https://www.apachefriends.org/index.html - 2.7.2016.

[3] MVC, https://en.wikipedia.org/wiki/model-view-controller - 7.6.2016.

[4] Git, https://git-scm.com/ - 20.5.2016.

[5] Mailgun, https://documentation.mailgun.com/ - 3.5.2016.

[6] Nexmo, https://docs.nexmo.com/ - 2.5.2016.

SAŽETAK

U radu se daje opis implementacije aplikacije za slanje poruka u preddefinirano vrijeme. Na početku su predstavljeni alati koji su bili potrebni za implementaciju. Zatim je opisana arhitektura koju koristi razvojna okolina u kojoj je aplikacija razvijena. Opisane su pojedine komponente aplikacije idući redom od dizajna baze podataka i njene apstrakcije, upravljačkog dijela pa sve do korisničkog sučelja. Na samom kraju je opisana implementacija samog slanja poruka. Rezultat rada je potpuno funkcionalna aplikacija s kojom korisnici mogu slati SMS i email poruke.

Ključne riječi: SMS, email, razvojna okolina, model, upravljač, korisničko sučelje

ABSTRACT

Web application for sending scheduled messages

This paper explains implementation details of a web application that can send scheduled messages. It presents the tools which were required for implementing the application. After presenting the tools, the framework architecture is explained. Afterwards, individual components of the application are described going from the bottom layer upwards. Going from describing the database design and the classes that are used as its abstraction to controllers and to the user interface. At the very end, the implementation details for sending the messages are described. The result of this paper is a working web application which users can use for sending scheduled SMS and email messages.

Keywords: SMS, email, framework, model, controller, interface, scheduled messages

ŽIVOTOPIS

Anto Antunović rođen je 31.01.1989. godine u Gradačcu. Osnovnoškolsko obrazovanje završio je u OŠ Davorina Trstenjaka u Hrvatskoj Kostajnici. Srednješkolsko obrazovanje je stekao u Elektrotehničkoj i prometnoj školi u Osijeku. Nakon srednjoškolskog obrazovanja, 2008. godine upisuje stručni studij elektrotehnike na Elektrotehničkom fakultetu u Osijeku, na kojem polaže sve ispite s prosječnom i kao apsolvent počinje raditi u tvrtki Samurai digital d.o.o. . Prvu godinu u tvrtki radi kao web programer razvijajući platformu za objavljivanje web stranica prilagođenih za mobilne uređaje, a posljednje dvije godine kao programer mobilnih aplikacija gdje je razvio više od deset komercijalnih aplikacija za android operacijski sustav, pretežno za njemačko i švicarsko tržište. U slobodno vrijeme voli voziti bicikl, te ponekad zaigrati nogomet.

PRILOZI

- CD na kojem se nalazi kompletan izvorni kod aplikacije te ovaj dokument u *pdf* i *doc* formatu.