# Web aplikacija za obavještavanje korisnika o važnim terminima

**Krajina, Marko**



**FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA**

**Diplomski sveučilišni studij**

# WEB APLIKACIJA ZA OBAVJEŠTAVANJE KORISNIKA O VAŽNIM TERMINIMA

**Diplomski rad**

**Marko Krajina**

**Osijek, 2020.**

FERIT
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA OSIJEK

**Obrazac D1: Obrazac za imenovanje Povjerenstva za obranu diplomskog rada**

**Osijek, 20.02.2020.**

**Odboru za završne i diplomske ispite**

## Imenovanje Povjerenstva za obranu diplomskog rada

| | |
|---|---|
| **Ime i prezime studenta:** | Marko Krajina |
| **Studij, smjer:** | Diplomski sveučilišni studij Računarstvo |
| **Mat. br. studenta, godina** | D-917R, 03.10.2019. |
| **OIB studenta:** | 62411008870 |
| **Mentor:** | Prof. dr. sc. Dominika Crnjac Milić |
| **Sumentor:** | Izv. prof. dr. sc. Krešimir Nenadić |
| **Sumentor iz tvrtke:** | Dino Turopoli |
| **Predsjednik Povjerenstva:** | Izv. prof. dr. sc. Alfonzo Baumgartner |
| **Član Povjerenstva:** | Dr. sc. Tomislav Galba |
| **Naslov diplomskog rada:** | Web aplikacija za obavještavanje korisnika o važnim terminima |
| **Znanstvena grana rada:** | **Informacijski sustavi (zn. polje računarstvo)** |
| **Zadatak diplomskog rada:** | Izraditi web aplikaciju pomoću koje bi registrirani korisnici bili pravovremeno obavješteni o važnim terminima povezanim s kontaktima iz grupe ili imenika. Napraviti analizu postojećih aplikacija na tržištu, istaknuti prednosti koje omogućava novo izrađena web aplikacija te opisati tehnologije potrebne za njezinu izradu. (sumentor: izv.prof.dr.sc. Krešimir Nenadić /FERIT i Dino Turopoli, |
| **Prijedlog ocjene pismenog dijela ispita (diplomskog** | Izvrstan (5) |
| **Kratko obrazloženje ocjene prema Kriterijima za ocjenjivanje završnih i diplomskih radova:** | Primjena znanja stečenih na fakultetu: 3 bod/boda<br>Postignuti rezultati u odnosu na složenost zadatka: 3 bod/boda<br>Jasnoća pismenog izražavanja: 3 bod/boda<br>Razina samostalnosti: 3 razina |
| **Datum prijedloga ocjene mentora:** | 20.02.2020. |

| | |
|---|---|
| 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, 25.03.2020.**

| | |
|---|---|
| **Ime i prezime studenta:** | Marko Krajina |
| **Studij**: | Diplomski sveučilišni studij Računarstvo |
| **Mat. br. studenta, godina upisa:** | D-917R, 03.10.2019. |
| **Ephorus podudaranje [%]:** | 4% |

Ovom izjavom izjavljujem da je rad pod nazivom**: Web aplikacija za obavještavanje korisnika o važnim terminima**

izrađen pod vodstvom mentora Prof. dr. sc. Dominika Crnjac Milić

i sumentora Izv. prof. dr. sc. Krešimir Nenadić

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

Sve brži i dinamičniji način života uzrokuje povećavanje broja događaj o kojima jedna osoba mora voditi računa. Oslanjanje isključivo na vlastito pamćenje može vrlo lako dovesti do previda i zaboravljanja dogovorenih događaja. Paralelno s ubrzanjem ritma života razvija se i tehnologija. Moderni problemi su rješivi modernim rješenjima, kao što su web aplikacije. Web aplikacije su programska rješenja kojima se može pristupiti korištenjem web preglednika. Svoj rast i razvoj mogu zahvaliti svojoj dostupnosti. Web aplikacijama se može pristupiti s bilo kojeg mjesta, mobitela, računala i u bilo koje vrijeme.

Aplikacija za obavještavanje korisnika o važnim terminima pruža mogućnost za točnu i preciznu kontrolu dogovorenih događaja, kao i dogovaranje novih događaja s kontaktima. Također, aplikacija pravovremeno obavještava korisnike o nadolazećim događajima. Redovito praćenje dogovorenih događaja može potaknuti razvoj boljih organizacijskih navika.

Nakon uvoda, u drugom poglavlju napravljen je osvrt na alate i tehnologije koje su korištene prilikom izrade aplikacije. Nastavno na teorijski osvrt, napravljena je analiza aplikacija na tržištu koje nude slične funkcionalnosti te su navedene prednosti korištenja aplikacije koja je napravljena u ovom diplomskom radu. Četvrto poglavlje opisuje način dizajniranja aplikacije. Nakon osvrta na dizajn, u petom poglavlju je opisan nastanak same aplikacije kao i način njenog korištenja. Šesto poglavlje opisuje načine i tehnike čijom provedbom bi se napravljena aplikacija mogla jednog dana i monetizirati.

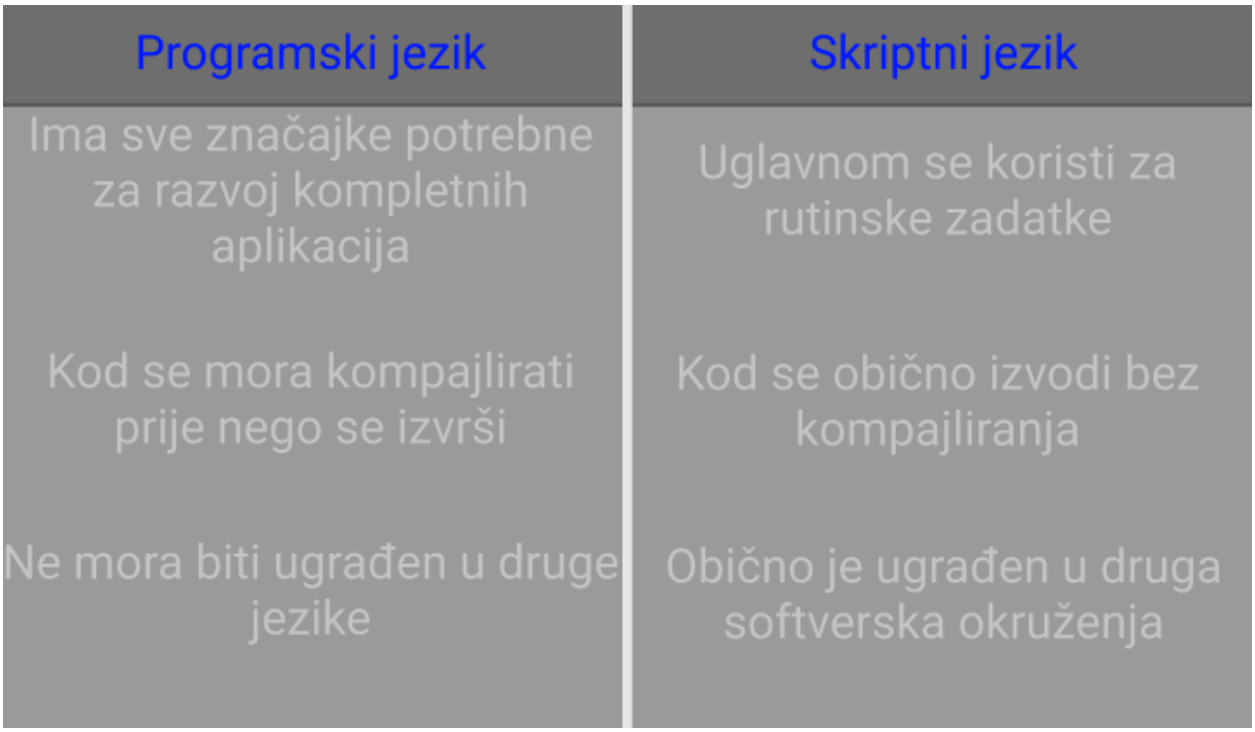
## 1.1. Zadatak diplomskog rada

Zadatak ovog diplomskog rada je izraditi web aplikaciju pomoću koje bi registrirani korisnici bili pravovremeno obavješteni o važnim terminima povezanim s kontaktima iz grupe ili imenika. Napraviti analizu postojećih aplikacija na tržištu, istaknuti prednosti koje omogućava novo izrađena web aplikacija te opisati tehnologije potrebne za njenu izradu.

# 2. KORIŠTENI ALATI I TEHNOLOGIJE

## 2.1. Skriptni jezici

Skripta je skup programskih uputa koje se interpretiraju tijekom izvođenja. Skriptni jezik je jezik koji interpretira skripte tijekom izvođenja. Skripte su obično ugrađena u druga softverska okruženja. Svrha skripti je obično poboljšati izvedbu ili izvesti rutinske zadatke za aplikaciju.




**Sl. 2.1.** Usporedba skriptnih i programskih jezika

## 2.2. PHP

PHP je skriptni jezik na strani poslužitelja koji se koristi za razvoj statičkih stranica, dinamičkih stranica i web aplikacija. PHP je akronim za *Hypertext Pre-processor,* dok je prije PHP akronimom stajao za *Personal Home Pages*. PHP skripte se mogu interpretirati samo na serveru koji ima instaliran PHP interpreter.

### 2.2.1. Sintaksa PHP-a

Umjesto velikog broja dodatnih naredbi, PHP skripte u sebi imaju HTML s ugrađenim kodom koji obavlja određene funkcije (na slici 2.2. vidljiv je primjer korištenja PHP sintakse pomoću koje se ispisuje „Markov diplomski rad“ ) Početak i kraj izvođenja PHP koda označen je posebnim oznakama. Oznaka „<?php” označava početak, a oznaka „?>” kraj skriptnog PHP koda.

```
1   <!DOCTYPE html>
2   <html>
3   <body>
4
5
6
7   <?php
8   echo "Markov diplomski rad";
9   ?>
10
11  </body>
12  </html>
```

**Sl. 2.2.** Primjer korištenja PHP sintakse

Znakom dolara u PHP-u se predstavljaju varijable. PHP razlikuje velika i mala slova. Varijable mogu započeti bilo kojim slovom ili znakom podcrtavanja, iza varijable može se koristiti bilo koji slijed slova, brojeva ili znakova. Na slici 2.3. vidljiv je primjer korištenja varijabli..

```php
<?php //php 7.0.8

$var = 'Marko';
$Var = 'Krajina';

echo "$var, $Var"; //ispisuje "Marko, Krajina"

$4stranica = 'jos ne'; //greška: varijabla počinje brojem
$_4stranica ='jos ne'; //nema greške: varijabla počinje '_'
?>
```

**Sl. 2.3.** Primjer upotrebe varijabli

Na slici 2.4. vidljiv je primjer stvaranja polja u PHP-u. Mape su tipovi koji povezuju vrijednosti s ključevima. Takvi tipovi su optimizirani za više različitih primjena. Može ih se tretirati kao polja, liste (vektore), *hash* tablice, kolekcije, redove… S obzirom na to da vrijednosti polja mogu biti i druga polja, moguće je stvarati i stabla.

```php
$array = [
    "foo" => "bar",
    "bar" => "foo",
];
?>
```

**Sl. 2.4.** Primjer polja

Sve PHP skripte su sastavljene od skupa izjava. Izjave mogu biti dodjele, funkcijski pozivi, petlje, uvjetne izjave pa čak mogu biti i prazne (izjave koje ne čine ništa).

Neke od najčešće korištenih naredbi za kontroliranje toka su*: if, else, switch case* (na slici 2.5. vidljiv je primjer korištenja switch case naredbe) , *while, for i foreach.* [5]

```php
<?php //php 7.0.8

switch ($i) {
    case 0:
        echo "i je = 0";
        break;
    case 1:
        echo "i je = 1";
        break;
    case 2:
        echo "i je = 2";
        break;
}


?>
```

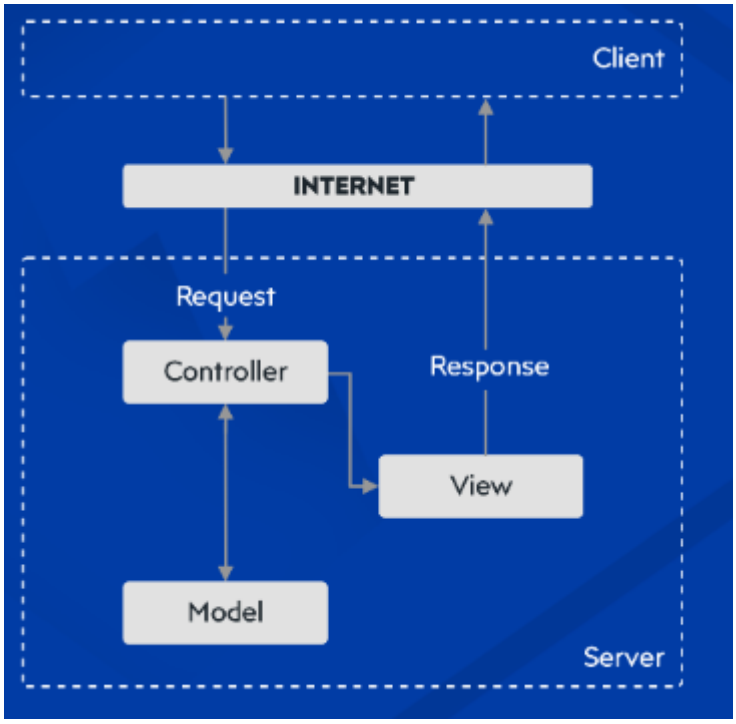
**Sl. 2.5.** Switch case naredba

## 2.3. Laravel

Laravel je besplatan PHP okvir (engl. *Framework*) koji se upotrebljava za izradu internet aplikacija temeljenih na *Model-View-Controller (MVC)* obrascu softverske arhitekture. PHP okvir se sastoji od skupa gotovih modula napravljenih u programskom jeziku PHP koji zajedno predstavljaju platformu za razvoj aplikacija. Skup već stvorenih modula pruža programeru mogućnost korištenje istih u svojim projektima. Efekt takvog pristupa je znatno ubrzavanje rada. Uz povećanje brzine izrade aplikacija, PHP okvir osigurava i izgradnju stabilnijih aplikacija. Laravel je razvijen 2011. godine a njegov tvorac je Taylow Otwell. Ubrzo nakon njegovog nastanka Laravel postaje jedan od najkorištenijih PHP okvira, čiji se cijeli kod nalazi na GitHub-u[1]. Prije korištenja Laravela na računalo je potrebno instalirati Composer. Composer je alat za upravljanje ovisnim paketima koji omogućuje upravljanje komponentama od kojih je sastavljen

[1] Github je servis za Git repozitorije koji korisnicima omogućava upotrebu svih funkcionalnosti Gita u kombinaciji sa setom različitih mogućnosti koje su specifične za GitHub. Odnosno, GitHub je mjesto gdje svaki programer može podijeliti svoj kod s ostatkom svijeta.

Laravel. Prednosti Laravela su njegova detaljna dokumentacija, mnoštvo materijala na internetu, pregledan kod i dosljednost MVC arhitekturi. MVC arhitektura se koristi kako bi se odvojili pojedini dijelovi aplikacije u komponente ovisno o njihovoj namjeni. Prvi dio je model koji se sastoji od podataka, logike, pravila i funkcija. Drugi dio je pogled. Pogled je prikaz podataka. Zadnji dio je upravitelj. Upravitelj prima inpute nad njima izvršava PHP kod te ih povezuje s modelom ili pogledom. Koncept MVC arhitekture vidljiv je na slici 2.6. [1]




**Sl. 2.6.** MVC arhitektura [1]

## 2.4. HTML

HTML je kratica za *HyperText Markup Language*. HTML dokument nastaje pomoću HTML opisnog jezika. HTML opisni jezik se koristi za prikazivanje i oblikovanje sadržaja. Jednostavnost korištenja HTML-a je jedan od razloga njegove popularnosti. Svi današnji internet preglednici mogu prikazivati HTML dokument dok HTML upućuje internet preglednik na koji

način krajnjem korisniku prikazati hipertekst dokument. Uloga HTML-a je da u svim preglednicima jednako prikaže hipertekst dokument. HTML nije programski nego opisni jezik, te ne može izvršiti jednostavne zadatke zbrajanja ili oduzimanja. HTML datoteke završavaju ekstenzijom html i sastoje se od elemenata podijeljenih na oznake (*tags*). [2]

## 2.4. CSS

CSS (*Cascading Style Sheets*) je skup pravila koja govore internet pregledniku na koji način prikazati neku HTML oznaku. Postoji više inačica CSS-a dok je CSS3 najnovija. CSS atributi omogućavaju pomicanje, promjenu veličine, širine, boje i mnoštva drugih efekata nad HTML elementima. Prije nego što je izašao CSS3 animacije su bile moguće korištenjem JavaScript-a, a nakon njegovog izlaska animacije su postale moguće i bez JavaScript-a. Najveći nedostatak animacija u CSS3-u je to što se u različitim preglednicima treba koristiti specifičan prefiks, ovisno o atributu koji se upotrebljava. [3] Neki od CSS3 modula
su:

- selektori
- pozadine i granice
- efekti nad tekstom
- animacije
- govorni moduli
- hiperlink prezentacije
- transformacije

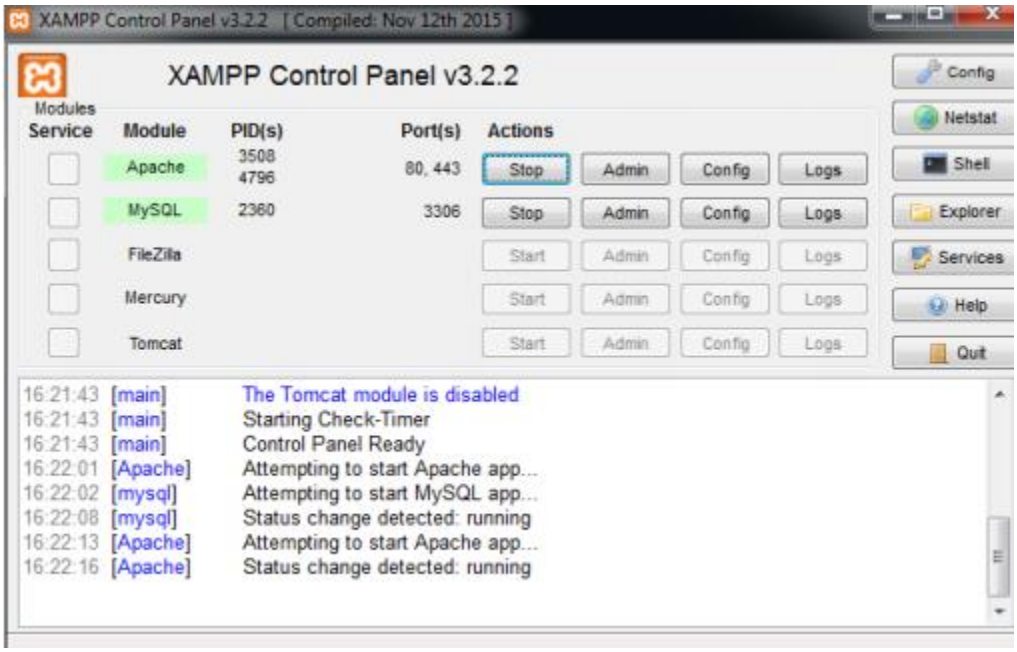
## 2.5. Bootstrap

Bootstrap je biblioteka koja sadrži gotove predloške iz HTML-a, CSS-a i JavaScript-a. Prva inačica Bootstrap-a je izdana 2006. godine , a do danas su izašle četiri inačice. Biblioteka se sastoji od velikog broja gotovih CSS stilova i JavaScript funkcija koje se mogu iskoristiti na vrlo jednostavan način, tako da se potrebne klase uključe u HTML elemente. Najpopularnija

karakteristika Bootstrap-a je *Grid System* koji omogućuje dijeljenje stranice na više dijelova i tako stranicu učiniti privlačnijom  i responzivnijom. *Grid* sustav koristi niz spremnika, redaka i stupaca za raspored i usklađivanje sadržaja. [7]

## 2.6. XAMPP

XAMPP je besplatan servis odnosno platforma pomoću koje je moguće pokrenuti servise kao što su Apache i MySQL. Izgled kontrolne ploče i sve mogućnosti XAMPP platforme vidljive su na slici 2.7. [4]

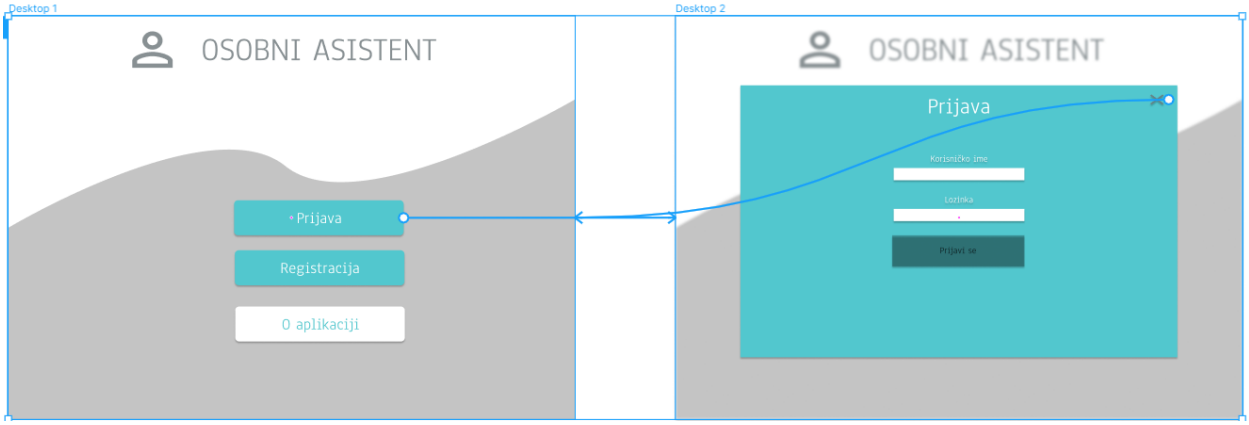



**Sl. 2.7.** Upravljačka ploča XAMPP okruženja

## 2.7. Figma

Figma je prvi alat za dizajn korisničkog sučelja koji se temelji na internet pregledniku, što ga čini vrlo pristupačnim. Osim što za Figmu nije potrebna nikakva instalacija, Figma je i

besplatna za nekomercijalnu upotrebu. Figma dolazi sa svim alatima koji su potrebni za dizajn web aplikacije. Među tim alatima su i vektorski alati koji su sposobni stvarati potpune animacije. Za prototipiranje, mogu se stvoriti poveznice na dizajnu te se simulirati kako bi se korisnik kretao kroz dizajnirano sučelje (na slici 2.8. vidljiv je primjer korištenja prototipiranja u Figmi).

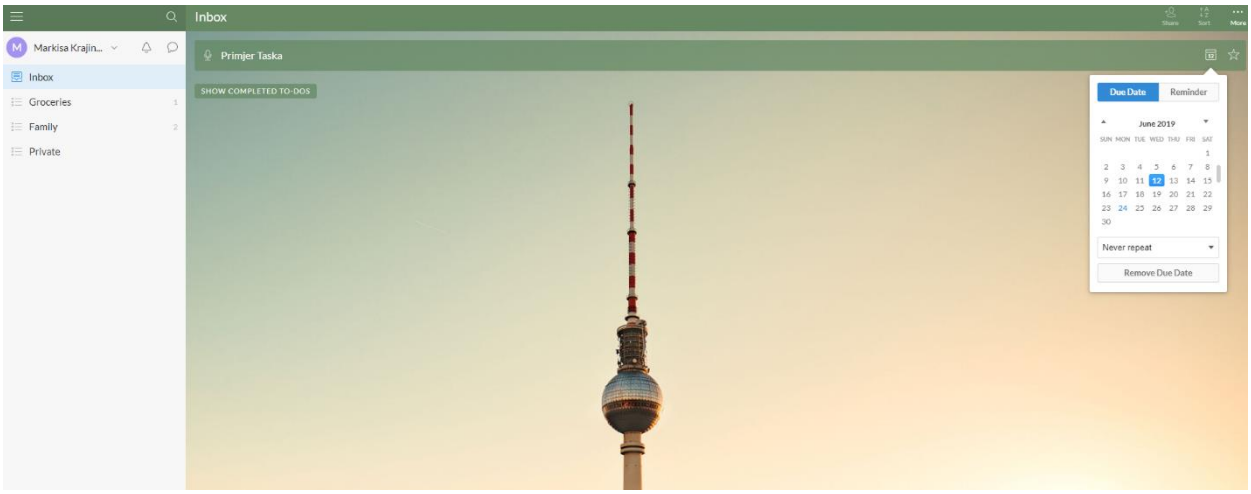



**Sl. 2.8.** Primjer prototipiranja u Figmi

# 3. Konkurentne aplikacije

Pregled sličnih rješenja, koja pružaju uslugu obavještavanja o bitnim terminima, temelji se na web aplikacijama napravljenim sa sličnim filozofijama korištenja.

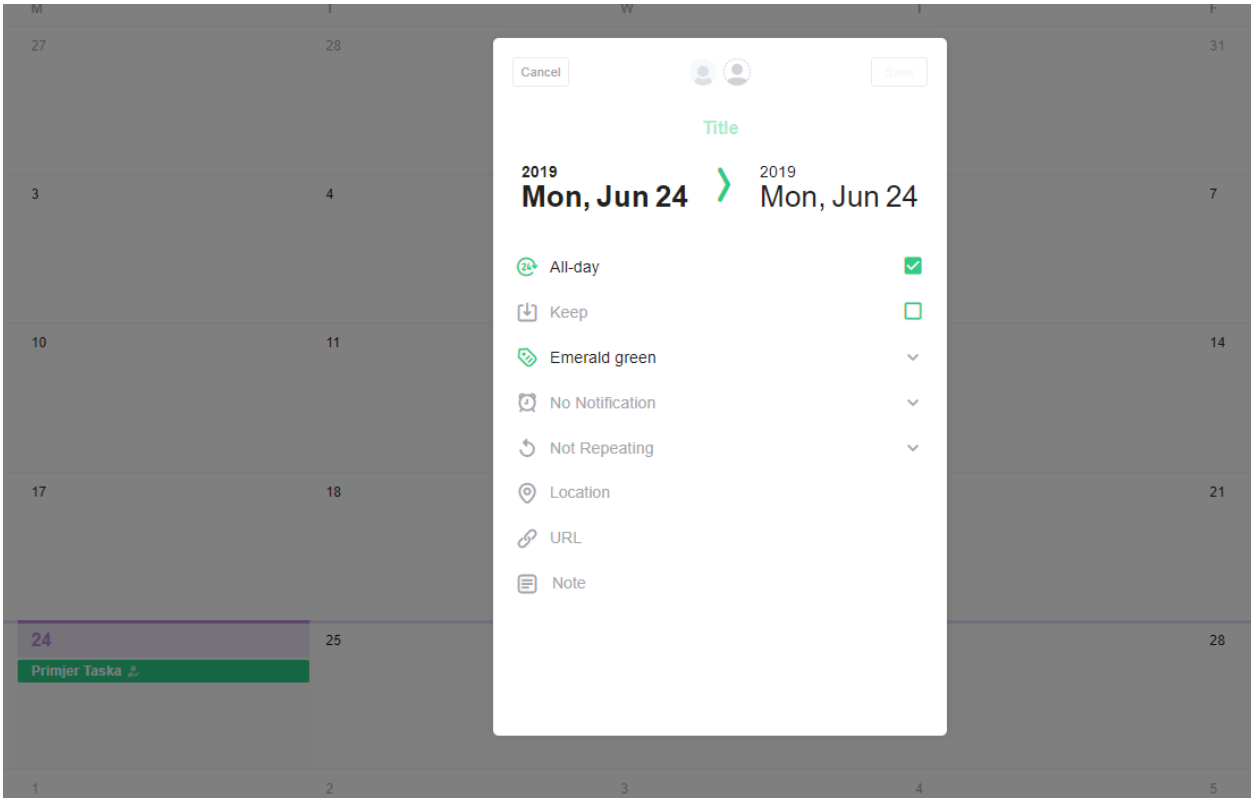
Na slici 3.1. je vidljivo sučelje web aplikacije Wunderlist koja pruža mogućnost stvaranja grupa, dodavanja ljudi u te grupe i stvaranje događaja za sve članove te grupe.




**Sl. 3.1.** Web aplikacija Wunderlist

Korisnicima ove web aplikacije, također, je omogućeno da označe neki zadatak kao izvršen. Unutar "*SHOW COMPLETED TO-DOS"* liste moguće je vidjeti sve uspješno odrađene termine do sada.

Slično tome, ali omogućeno samo za autentificiranog korisnika, web aplikacija TimeTree pruža mogućnosti stvaranja termina. Na slici 3.2. vidljiv je primjer korištenja TimeTree aplikacije za stvaranje novog termina. Za razliku od Wunderlista, TimeTree nudi više mogućnosti tijekom stvaranja termina kao što su dodavanje lokacije i linkova. Uz glavnu funkciju, ova aplikacija pruža i mogućnost kopiranja lokacije sa Google-ovih mapa.




**Sl. 3.2.** Web aplikacija TimeTree

## 3.1. Prednosti aplikacije u odnosu na konkurente

Prednost web aplikacije za obavještavanje korisnika o važnim terminima je u tome što ona pruža korisnicima mogućnost stvaranja liste kontakata i dodavanja svojih poslovnih partnera ili prijatelja. Pri stvaranju novog termina korisnici će moći dodati osobe iz imenika te će obavijesti o novom terminu biti vidljive kako prvobitnom tako i dodanom korisniku. Dodatan plus je i činjenica da nijedna od analiziranih konkurentni aplikacija nema podršku za hrvatski jezik.

# 4. Dizajn web aplikacije

Pri izradi dizajna web aplikacije potrebno je voditi računa o opširnosti ponuđenog sadržaja. Preopširan izbor sadržaja i prevelik broj ponuđenih opcija korištenja može utjecati na funkcionalnost aplikacije.

Iz toga razloga, dobro je put izvršavanja akcije unutar aplikacije podijeliti na veći broj smislenih cjelina s manjim količinama podataka, a maknuti one podatke koji trenutno za korisnika nemaju važnost kako bi se izbjegao osjećaj dezorijentacije i izgubljenosti tokom korištenja aplikacije.

## 4.1. Boja

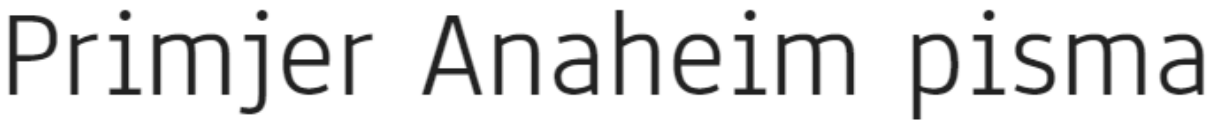
Prvi element vizualnog dizajna aplikacije koji se primijeti je boja. Izbor službenih boja bi trebao iza sebe imati promišljenu ideju, trebaju se znati razlozi zbog kojih je odabrana baš ta boja. Aplikacije koje su idejno slične mogu imati skroz različite palete boja što ne mora biti krivo. Isto tako, kod odabira boje može se slijediti i analiza dizajna nekih konkurentnih aplikacija.

Korisnici aplikacije za obavještavanje o bitnim terminima obično očekuju osjećaj smirenosti, preglednosti, kao i minimalističku obojanost sučelja kako bi se izbjegle nepotrebne ili zbunjujuće situacije. No, ako se kreira aplikacija kojoj je glavna aktivnost obavještavanje korisnika o bitnim terminima, treba se voditi briga da aplikacije ne bude premonotona ili prezamorna za svakodnevnu upotrebu.

## 4.2. Tipografija

Kada se radi odabir tipografije za web aplikaciju veliku ulogu igra optimiziranost, odnosno prilagođenost prikazivanju u digitalnom obliku. Sva tipografska pisma nemaju istu namjenu i nisu ista. Neka pisma su više namijenjena tiskovnom mediju, dok su pak druga prilagođenija digitalnom mediju. Pismo poput Anaheima, razvijeno je s ciljem korištenja u digitalnom obliku, te se upravo u digitalnom svijetu najbolje i ponaša. Anaheim pismo je odabrano iz razloga kako

bi se što je moguće više izbjeglo nepotrebno prilagođavanje pisma aplikaciji. Na slici 4.1. vidljiv je primjer teksta koji je pisan Anaheim pismom.




**Sl. 4.1.** Tipografsko pismo Anaheim

## 4.3. Tekst

Kada je korišteni tekst u aplikaciji nečitljiv, aplikacija kao rezultat gubi korisnike. Čitljivost je vrlo važan element svih digitalnih proizvoda, te je vrlo važan sastojak cjelokupnog korisničkog iskustva. Određeni tekstualni dijelovi mogu se istaknuti ili sakriti upotrebom boje. Dobra praksa je da je tekst s kojim aplikacija vrši komunikaciju s korisnikom u obliku kratkih fraza i jednostavnog vokabulara. Tekst nikad ne bi trebao biti pisan u velikim odlomcima, već u što jasnijim rečenicama koje korisnika informiraju na što jednostavniji način. Prezasićenost tekstom može implicirati negativnim efektom u očima korisnika, kao i pretjerano korištenje detalja u svrhu ukrašavanja ili zasićenja bojama.

## 4.4. Sažetak cjelokupnog vizualnog dizajna

Cjelokupni vizualni dizajn web aplikacije dane ovim radom je minimalan kako bi se korisnik mogao fokusirati na osnovne akcije koje aplikacija nudi. U većem dijelu ove aplikacije korištena je monokromatska paleta boja, s pojedinim iznimkama. Tipografski dizajn korišten u aplikaciji je napravljen s ciljem što čišćeg prikaza u digitalnom obliku. Tekstualni dio je napravljen sa fokusom na kratkoću i jasnoću.

# 5. Realizacija sustava

Ideja je napraviti web aplikaciju koja bi omogućila pravovremeno obavještavanje putem maila o važnim terminima koji su povezani s kontaktima iz liste prijatelja. Nakon što korisnik kreira novi događaj, svi korisnici na koje se odnosi taj događaj dobivaju obavijest.

## 5.1. Baza podataka

Shema baze podataka ove aplikacije stvorena je u MySQL sustavu za upravljanje bazom podataka. Ukupna shema baze podataka je osmišljena tako da se sastoji od sedam relacijskih tablica. Tih sedam tablica su redom: *users, profiles, events, event_users, friend_users, password_resets, migrations*. Tablica *users* (korisnici) sadrži podatke o korisnicima: *name, surname, email, password*. Na slici 5.1. vidljiv je programski kod migracije za tablicu *Users*. Svi atributi iz tablice *users* su tipa string. Atribut *email* je unikatan, što znači da se u bazi podataka može ponoviti samo jedan zapis, dok su svi atributi obavezni kod svakog novog zapisa u tablicu users. Tablica *profiles* sadrži podatke o profilu koji pripada korisniku. Relacija tablica *users* i *profiles* je jedan na prema jedan. Relacija jedan na prema jedan znači da svaki korisnik ima točno jedan pripadajući profil, odnosno, da profil pripada samo jednom korisniku. Tablica *profiles* sadrži atribute*: ID, user_id, image, description*. Atribut *user_id* je vanjski ključ prema tablici *users*. Atributi *image* i *description* nisu obavezni zato što se korisnicima opcija dodavanja slike i opisa želi omogućiti ali ne i nametnuti. Tablica *events* služi za skladištenje događaja te sadrži sve informacije o događaju. Atributi koji čine tablicu events su: *ID, description, date, time. Events* tablica sadrži relaciju više prema više u odnosu na tablicu *users*. Kod relacije više prema više potrebna je pivot tablica. Pivot tablica u ovom slučaju je *event_users* i u njoj se nalaze podatci o tome koji se događaji odnose na koje korisnike. Atribut *description* je opis stvorenog događaja koji je tipa *string*. *Date* i *time* su atributi kojima se određuje datum i vrijeme događaju. Druga pivot tablica je *friend_users*. Ona je produkt relacije više prema više i u njoj se nalaze informacije o tome koji je korisnik prijatelj s kojim korisnikom. Tablica *password_resets* je produkt funkcije aplikacije koja omogućava povrat zaboravljenih zaporki. Sadrži podatke o korisniku koji je zatražio povrat lozinke.

```php
public function up()
{
    Schema::create('users', function (Blueprint $table) {
        $table->bigIncrements('id');
        $table->string('name');
        $table->string('surname');
        $table->string('email')->unique();
        $table->timestamp('email_verified_at')->nullable();
        $table->string('password');
        $table->rememberToken();
        $table->timestamps();
    });
}

/**
 * Reverse the migrations.
 *
 * @return void
 */
public function down()
{
    Schema::dropIfExists('users');
}
```

**Sl. 5.1.** Izgled migracije za tablicu *Users*

Baza je stvarana korištenjem Laravel migracija. Migracije se koriste kao sheme za jednostavno stvaranje i uređivanje stvarnih baza podataka. Laravel migracije su korištene iz više razloga od kojih je najvažniji bio taj što Laravel omogućuje „*rollback*“, odnosno povratak prijašnjih migracija. Rezultat ovog pristupa je i tablica migrations u kojoj se nalaze detalji pojedinih migracija.

| # | Name | Type | Collation | Attributes | Null | Default | Comments | Extra | Action |
|---|---|---|---|---|---|---|---|---|---|
| 1 | id | bigint(20) | | UNSIGNED | No | None | | AUTO_INCREMENT | Change Drop More |
| 2 | name | varchar(255) | utf8mb4_unicode_ci | | No | None | | | Change Drop More |
| 3 | surname | varchar(255) | utf8mb4_unicode_ci | | No | None | | | Change Drop More |
| 4 | email | varchar(255) | utf8mb4_unicode_ci | | No | None | | | Change Drop More |
| 5 | email_verified_at | timestamp | | | Yes | NULL | | | Change Drop More |
| 6 | password | varchar(255) | utf8mb4_unicode_ci | | No | None | | | Change Drop More |
| 7 | remember_token | varchar(100) | utf8mb4_unicode_ci | | Yes | NULL | | | Change Drop More |
| 8 | created_at | timestamp | | | Yes | NULL | | | Change Drop More |
| 9 | updated_at | timestamp | | | Yes | NULL | | | Change Drop More |

**Sl. 5.2.** Tablica *users* u bazi podataka nakon migracije

Za svaku tablicu iz baze podataka postoji točno jedan odgovarajući model pomoću kojeg se vrši interakcija s tablicama. Model pruža mogućnost pretraživanja podataka iz tablica, kao i zapis novih podataka u tablice. Stoga je neophodno stvoriti i modificirati modele kako bi se uspostavile relacije.

Novi model se stvara pomoću unošenja naredbe `php artisan make:model` u upravljački prozor. Na primjer, za stvaranje modela Profil naredba bi bila `php artisan make:model Profil`. Na isti način su kreirani modeli za sve ostale tablice.

Sljedeći korak je određivanje koji se parametri mogu odnosno ne mogu popunjavati. Svojstvo popunjivosti je moguće odrediti na dva načina: koristeći varijablu *fillable* ili varijablu *guraded*. Kod varijable fillable navodimo za koje atribute želimo da imaju svojstvo popunjivosti, odnosno, za koje atribute želimo da imaju svojstvo nepopunjivosti u slučaju varijable guarded. Kod *User* modela svojstvo popunjivosti dodijeljeno je atributima name, surname, email i password.

```php
/**
 * The attributes that are mass assignable.
 *
 * @var array
 */

protected $fillable = [
    'name','surname', 'email', 'password'
];
```

**Sl. 5.3.** Svojstvo popunjivosti kod modela *User*

Nakon određivanja svojstva popunjivosti, na red dolazi postavljanje relacija.

```php
public function profile()
{
    return $this->hasOne('App\Profile');
}


public function events()
{
    return $this->belongsToMany('App\Event');
}
```

**Sl. 5.4.** Relacije modela *User*

Relaciju *one-to-one* predstavlja metoda *hasOne*, iz koje se može vidjeti da *User* ima jedan profil. Relacija *many-to-many* je predstavljena metodom *belongsToMany*, iz koje se može vidjeti da jedan *user* može imati više događaja.

## 5.2. Izrada aplikacije

Prvi prioritet kod izrade aplikacije u Laravel okviru je pravilna podjela aplikacije u dijelove i komponente u skladu s njihovim namjenama. Pristup kod kojeg se cjelokupni kod dijeli na: model, pogled i upravitelja omogućuje superiorniju preglednost kao i nezavisan razvoj aplikacije, što rezultira olakšavanjem posla održavanja aplikacije. Model kao jedinka nije funkcionalan bez pogleda i upravitelja. Model, pogled i upravitelj zajedno čine MVC (*Model-View-Controller*) arhitekturu na kojoj se temelji Laravel okvir. Pogled služi za interakciju između aplikacije i korisnika. Također, pogled razdvaja prezentacijsku logiku od aplikacijske logike koja se nalazi u upravitelju. Dobra praksa je da se prije izrađivanja upravitelja definiraju i izrade rute (engl. *Routes*) u routes/web.php datoteci. Rute koje se definiraju služe kao način povezivanja upravitelja i URL-a.

Na slici 5.5. vidljiv je dio programskog koda web.php datoteke te se iz njega može vidjeti način definiranja toka u aplikaciji. Ključna riječ *Route* je uvijek na početku te nakon nje se

navodi vrsta zahtjeva koja se koristi. Unutar specifikacije svakog toka piše se naziv toka (npr. friend/{user}). Viličaste zagrade se koriste kada se u točno tom dijelu definiranog toka nalazi parametar, koji je u ovom slučaju user. Nakon toga se određuje koji će točno upravitelj koristiti novi tok ali također i metodu koja je definirana iza „@“ znaka. Na kraju je moguće i postaviti ime rute. Ime rute se ne mora postaviti ali njeno postavljanje omogućuje jednostavnije i kraće pozivanje toka svaki put kada se na njega šalje zahtjev.

```
Route::get('/newevent','EventController@index')->name('event');
Route::get('/friend/{user}','FriendController@store')->name('friend');

Route::post('/p','ProfileController@store')->name('store');
Route::post('newevent/create','EventController@create')->name('newevent');
```

**Sl. 5.5.** Dio programskog koda web.php datoteke

Kako bi se postigao određeni stupanj sigurnosti, korišteni su Laravel middleware-i odnosno međusloj sloj web aplikacije kojem je zadaća kontrola nad korisničkim zahtjevima. Svaka Laravel aplikacija dolazi s međuslojem naziva „auth“, koji se koristi kako bi određenim tokovima omogućio pristup samo u slučaju kada je korisnik prijavljen s korisničkim imenom i zaporkom za korištenje aplikacije. Međusloj se može koristiti i u metodi upravitelja kao i unutar metode web.php datoteke (na slici 5.6. je prikazan primjer korištenja međusloja unutar web.php datoteke) .

```
Route::group(['middleware' => 'auth'], function(){

    Route::get('profile/{username}',['uses' =>
    'ProfileController@index','as' => 'profile'
    ]);
```

**Sl. 5.6.** Metoda web.php datoteke

Nakon što su definirane rute na red dolazi stvaranje upravitelja. Za potrebe ove web aplikacije kreirani su upravitelji: *AddContactController, Controller, EventController, FriendController, HomeController i ProfileController*. Glavna zadaća upravitelja je upravljanje korisničkim zahtjevima prema web aplikaciji. Za svaki pojedini zahtjev koji napravi korisnik postoji određena metoda upravitelja. Upravitelj najčešće upravlja modelom aplikacije ali također može upravljati i s drugim dijelovima aplikacije.

Na slici 5.7. vidljiva je jedna od metoda *FriendController* upravitelja. Metoda store služi za dodavanje odnosno brisanje korisnika iz liste prijatelja. Svaki puta kada se na stranici profila odabere opcija dodavanja ili opcija brisanja prijatelja poziva se funkcija store. Kao argument ovoj metodi šalje se redni broj korisnika u tablici *Users* što omogućava pronalazak toga korisnika i njegovo spremanje u $*user* varijablu. Pomoću metoda *auth*() i *user*() pronalazi se trenutno autentificirani korisnik nakon čega se na njega poziva metoda *following*(). Metoda *following*() vraća povratnu informaciju o tome da li je korisnik prijatelj s korisnikom na čijem se profilu poziva funkcija dodavanja odnosno brisanja prijatelja. Naposljetku, metoda *toggle*() dodaje korisnika u listu prijatelja ako to prije nije bio , odnosno, briše ga iz liste prijatelja u suprotnom slučaju. Korisnik se nakon svega prosljeđuje na profil pozivom metode *redirect*().

```
class FriendController extends Controller
{
    public function store(\App\User $user)
    {
        auth()->user()->following()->toggle($user->profile);

        return redirect('profile/'.$user->id);
    }
}
```

**Sl. 5.7.** Metoda iz *FriendController* upravitelja

Kako bi modeli i upravitelji bili u potpunosti funkcionalni potrebno je stvoriti i poglede koji omogućuju interakciju korisnika i aplikacije. Svi pogledi ove aplikacije se nalaze unutar *Views* poddirektorija. Razvrstani su u nekoliko direktorija ovisno o tome kojem dijelu aplikacije ti pogledi pripadaju. Unutar *Layouts* direktorija se nalazi glavni predložak koji sadrži navigaciju, *header* i *footer* aplikacije.

```
@extends('layouts.app')

@section('content')
<div class="container">
    <div class="row">

        <div class="col-4">
        <img src="/storage/{{$profile->image}}" class="rounded-circle p-4" height="270px" alt="">

        @if($user->id == Auth::user()->id)

        <a href="{{ route('edit') }}">Uredi profil</a>
        @endif
        </div>

        <div class="col-8">
        <div class="d-inline-flex"><h1 class="pt-5 pl-5 pr-3 ">{{$user->name}}</h1><h1 class="pt-5 pl-1 pr-

        @if($user->id != Auth::user()->id)
            @if($follows != true)

            <div class="pt-5 pl-4">
            <a href="{{route('friend',$user->id)}}">
            <button class="btn btn-primary">Dodaj prijatelja</button>
            </a>
            </div>
```

**Sl. 5.8.** Dio koda profile.blade.php datoteke

Glavni predložak koristi svaki pogled ove aplikacije koji ga proširuje ovisno o tome što želi pokazati. Pomoću *@yield* oznake označuju se linije koda koje će se proširivati u novom pogledu. Na slici 5.8. vidljiv je primjer programskog koda jednog od pogleda. Prikazani pogled predstavlja stranicu profila korisnika. Pomoću *blade* oznake *@extends* definirana je poveznica s glavnim predloškom. Dijelovi koda glavnog predloška koji su bili označeni s *@yield* oznakom, prilikom proširivanju su zamjenjeni oznakama *@section*('ime odjeljka') i *@endsection*.

Kreiranje ugrađenih poruka elektroničke pošte se izvodi pomoću „Mailtrap“ servisa. Mailtrap radi tako da on oponaša rad stvarnog SMTP (*Simple Mail Transfer Protocol*) poslužitelja, odnosno sprječava probne mailove da dospiju u stvarni poštanski sandučić.

```
$users = User::where([
    ['name','=',$name],
    ['surname','=',$surname]
    ])->first();

foreach($users->following as $user)
{
    if($user->id == $auth_user->id)
    {
        $event_user = \App\event_user::create([
            'user_id' => $users->id,
             'event_id' => $event->id,
        ]);
        $event_user->save();

        $event_user = \App\event_user::create([
            'user_id' => $auth_user->id,
             'event_id' => $event->id,
        ]);
        $event_user->save();
        \Mail::to($users)->send(new Notifikacija($auth_user));
        return view('success');
```

**Sl. 5.9.** Dio koda u EventControlleru za slanje elektroničke pošte

Naredba \Mail::to() definira na koju će se adresu *email* isporučiti. Klasa Notifikacija stvara se pomoću php artisan naredbe „`php artisan make:mail Notifikacija`“ u naredbenom prozoru. Stvorena klasa se sastoji od dvije metode a to su *__construct*() i *build*(). Kod metode *__construct*() vrši se inicijalizacija objekta koji se uporabljuju u pogledu za uređivanje izgleda *e-maila*. Unutar *build*() se prosljeđuje inicijalizirani objekt pogledu koristeći *compact*() metodu, te se određuju detalji *emaila*.

```php
/**
 * Create a new message instance.
 *
 * @return void
 */
public function __construct(\App\User $auth_user)
{
    $this->auth_user = $auth_user;
}

/**
 * Build the message.
 *
 * @return $this
 */
public function build()
{
    return $this->markdown('emails.notifikacija');
}
```



**Sl. 5.10.** Metode *__construct*() i *build*() unutar klase Notifikacija

Unutar pogleda se koristeći HTML uređuje izgled emaila koji će biti poslan korisniku.

```blade
@component('mail::message')


<h1 class="text-center">Vaš prijatelj {{$auth_user->name}} {{$auth_user->surname}}
Vas poziva na novi događaj</h1>

@component('mail::button', ['url' => 'http://diplomskirad.test/upcomming/events'])
Poveznica
@endcomponent

Hvala što koristite našu uslugu,<br>
Osobni Asistent
@endcomponent
```

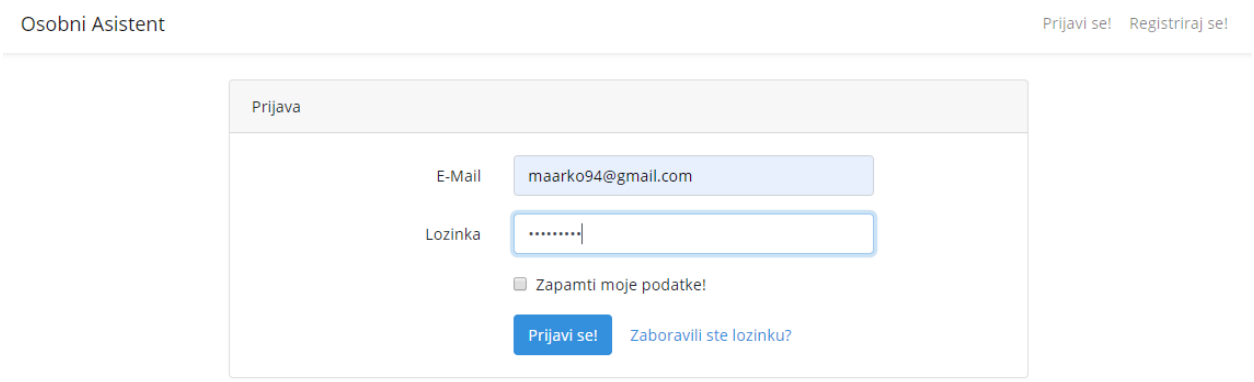
**Sl. 5.11.** Pogled notifikacija.blade.php

## 5.3. Analiza rada aplikacije

Pristupom početnoj stranici aplikacije korisniku se prikazuje polje za prijavu i polje za registraciju. Postojeći korisnici se mogu prijaviti s valjanom adresom elektroničke pošte i pripadajućom zaporkom dok novi korisnici moraju prvo napraviti novi račun i oni biraju polje za registraciju.

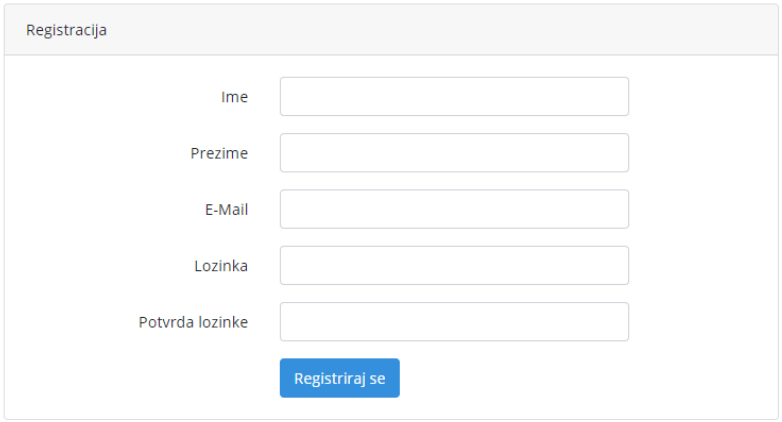

**Sl. 5.12.** Početna stranica

U prozoru prijave nalaze se i dodatne opcije kojima korisnici mogu pristupati. Te dodatne opcije su povrat zaboravljene zaporke i pamćenje korisničkih podataka. Prozor za povrat izgubljene zaporke traži od korisnika valjanu adresu elektroničke pošte te na nju šalje novu zaporku.



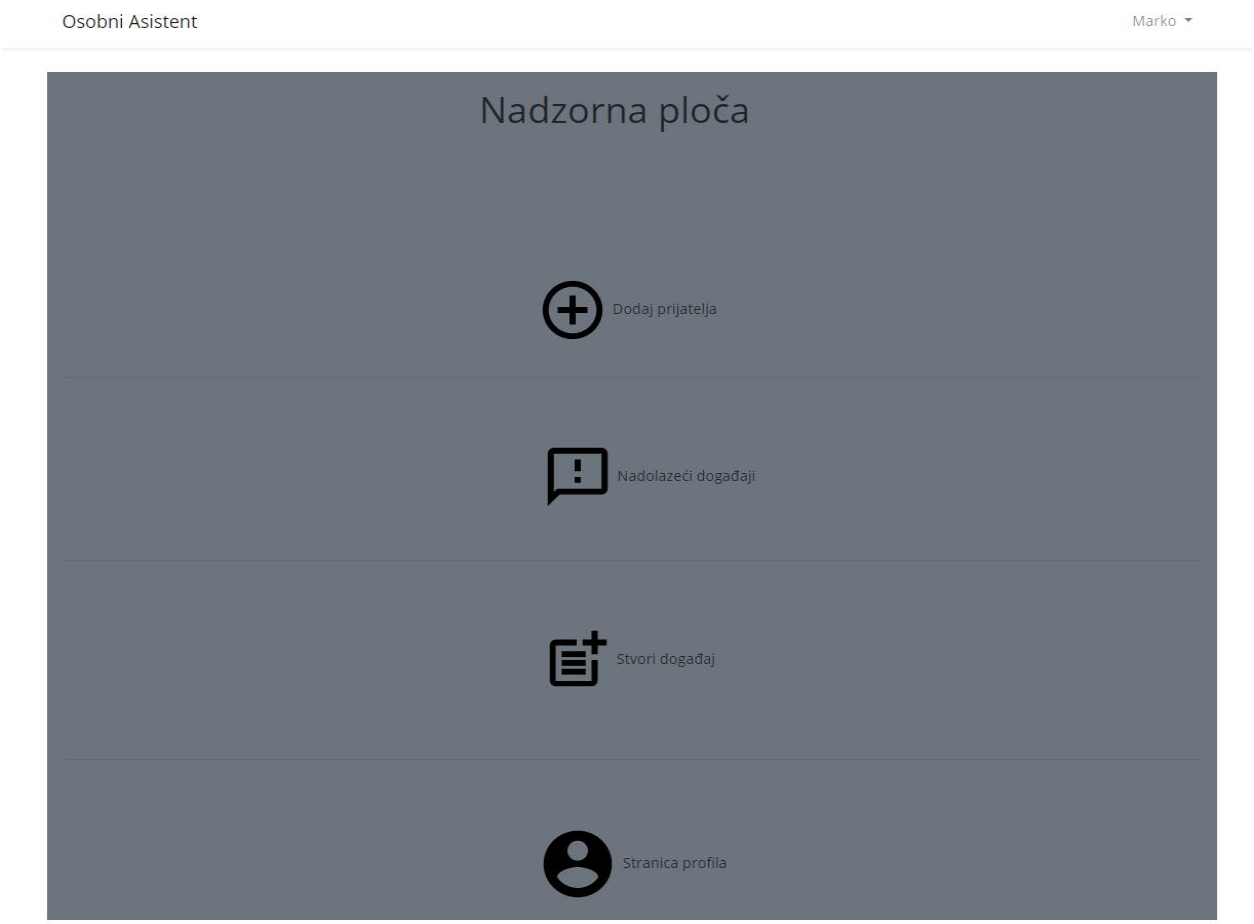

**Sl. 5.13.** Stranica prijave korisnika

Stranica registracije od korisnika traži: ime, prezime, adresu elektroničke pošte koja mora biti postojeća kao i jedinstvena u bazi podataka, zaporku i potvrdnu zaporku. Na slici 5.14. je prikazana stranica za registraciju korisnika.



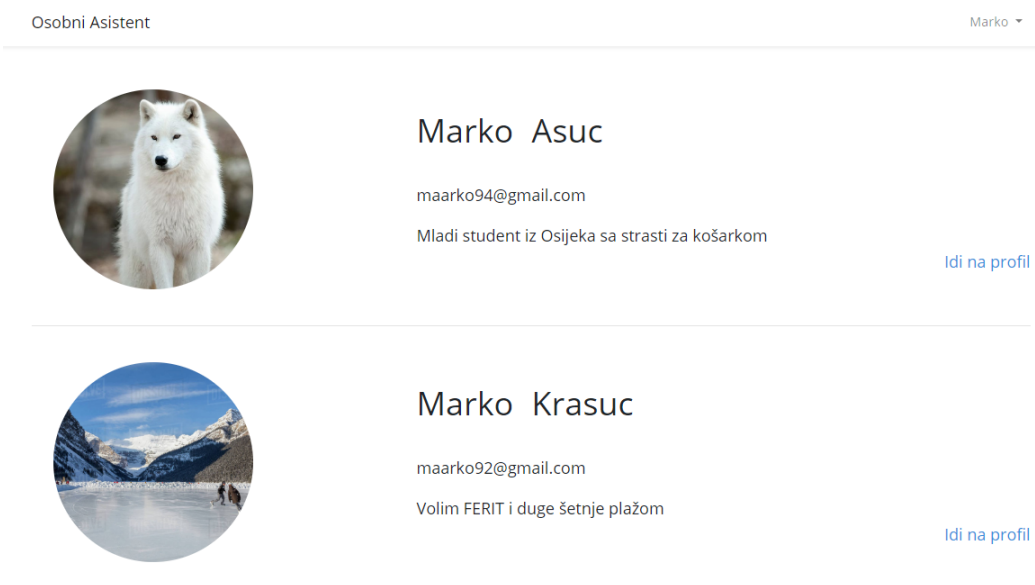

**Sl. 5.14.** Stranica registracije korisnika

U slučaju uspješnog prolaska kroz proces prijave korisnik se preusmjerava na nadzornu ploču. Na nadzornoj ploči se nalaze najvažnije funkcionalnosti kojima korisnik može pristupiti. Korisnik može birati između: dodavanja novih prijatelja, pregleda nadolazećih događaja koji se odnose na njega, stvaranje novih događaja za sebe i svoje prijatelje i odlaska na stranicu svoga profila.



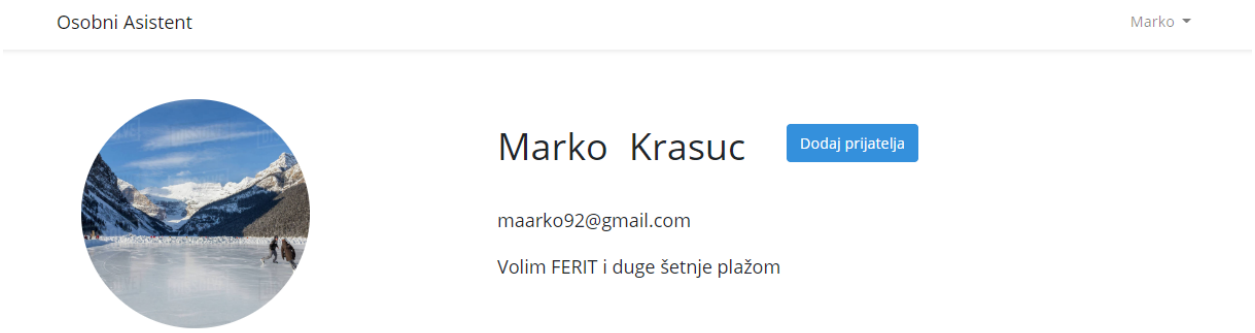

**Sl. 5.15.** Nadzorna ploča

Funkcija „dodaj prijatelja“ preusmjerava korisnika na tražilicu. Tražilica pretražuje bazu podataka u potrazi za željenim korisnikom. Pretraživanje se može izvršavati po punom imenu i prezimenu kao i samo po imenu ili samo po prezimenu.

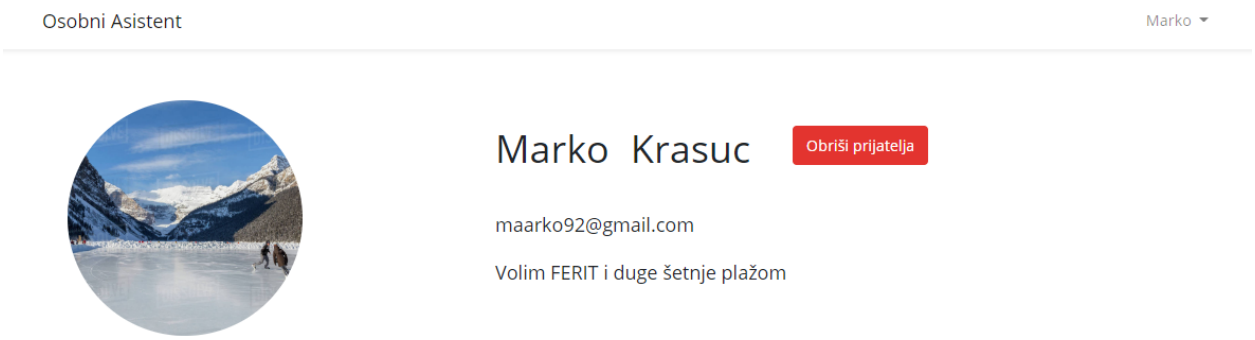


**Sl. 5.16.** Pretraživanje osoba sa imenom Marko

Nakon što korisnik pronađe željeni profil njemu lako pristupa preko poveznice „idi na profil“. Na profilu su vidljivi osnovni podatci od tražene osobe kao što su ime, prezime , adresa elektroničke pošte i profilna slika. Ako autentificirani korisnik nije prijatelj s korisnikom kojem pripada trenutno promatrani profil, tada mu se pojavljuje gumb „dodaj prijatelja“ dok se u suprotnom slučaju pojavljuje „obriši prijatelja“ gumb.



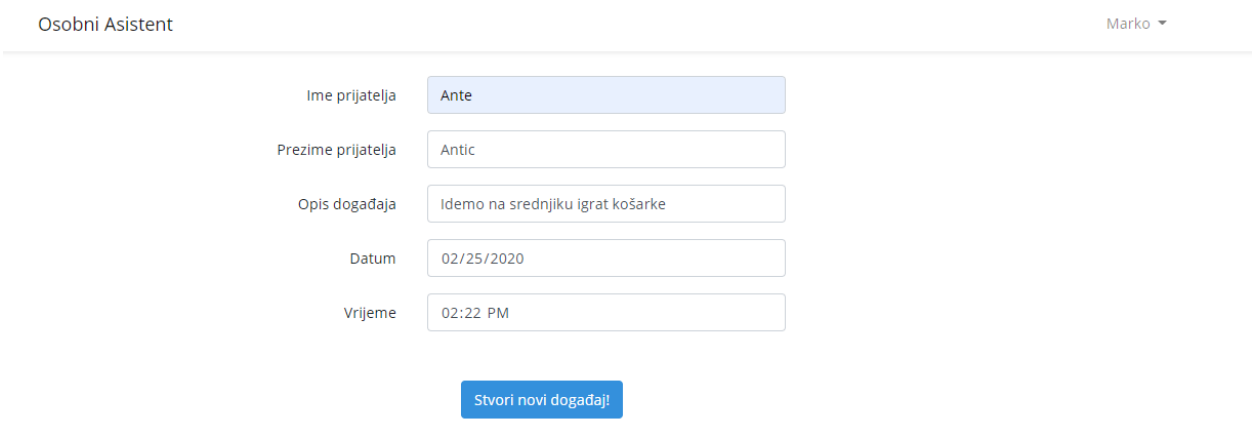

**Sl. 5.17.** Profil korisnika koja nije prijatelj s autentificiranim korisnikom




**Sl. 5.18.** Profil korisnika koja je prijatelj s autentificiranim korisnikom

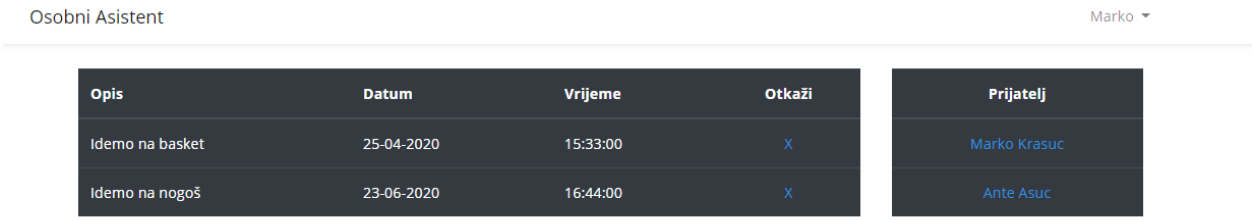
Za stvaranje novih događaja se koristi „stvori događaj“ funkcionalnost s nadzorne ploče. Za stvaranje novog događaja potrebno je unijeti ispravno ime i prezime prijatelja kojima se želi poslati događaj. Događaj je potrebno i opisati te aplikacija traži od korisnika određene atribute koji opisuju događaj. Atributi koje korisnik mora unijeti su: opis događaja, datum i vrijeme. Korisnik novi događaj može poslati drugom korisniku samo ako ga taj drugi korisnik ima u listi svojih prijatelja. Ako su svi uvjeti ispunjeni, kreira se novi događaj te svi korisnici na koje se

odnosi događaj dobivaju elektroničku poštu sa svim informacijama o stvorenom događaju kao i poveznicom na adresu na kojoj taj događaj mogu otkazati.



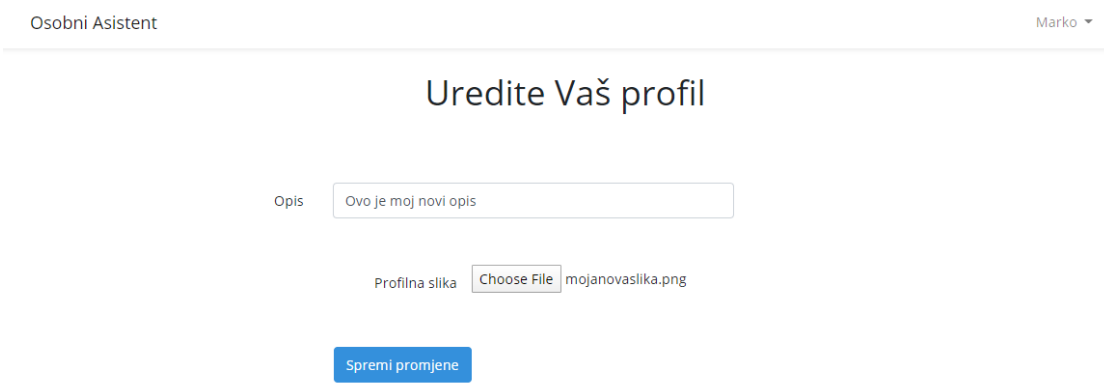

**Sl. 5.19.** Stvaranje novog događaja

Funkcionalnost „Nadolazeći događaji“ s nadzorne ploče omogućava pregled svih nadolazećih događaja na koje je trenutno prijavljeni korisnik pozvan. Sadrži opis, datum, vrijeme, ime i prezime prijatelja s kojima idemo na događaj kao i opciju otkazivanja. Pritiskom na znak „X“ korisnik otkazuje događaj te se svim ostalim korisnicima koji su bili dio tog istog događaja šalje e-mail koji ih obavještava o otkazivanju.




**Sl. 5.20.** Pregled događaja

Zadnja funkcionalnost koja se nalazi na nadzornoj ploči je „Stranica profila“. Klikom na ovu funkcionalnost korisnik se prosljeđuje na stranicu za uređivanje vlastitog profila. Moguće je neograničen broj puta mijenjati i opis i profilnu sliku.




**Sl. 5.21.** Uređivanje korisničkog profila

# 6. Potencijalna monetizacija aplikacije

Postoji više načina na koji se web aplikacija može monetizirati. Odabir načina ovisi o više čimbenika. Bitni čimbenici su broj dnevnih korisnika, dužina interakcije korisnika s aplikacijom i vrsta aplikacije. Neke od strategija su pasivne i uzimaju vrlo malo vremena dok druge traže veliki trud i ulaganje. Strategije koje bi bile najprimjerenije za ovaj tip aplikacije su partnerski marketing i „*Pay Per Click Advertising*“ (Google adsense).

Partnerski marketing je jedan od najbržih načina za zaradu od web aplikacije. Ova strategija radi tako da se pronađu zanimljivi i dobri proizvodi te se oni promoviraju na aplikaciji. Ako se radi o proizvodu ili usluzi koja se sviđa korisnicima, oni će kliknuti na partnersku poveznicu i kupiti proizvod pri čemu vlasnik aplikacije dobije svoj dio novca. Dio koji dobiva vlasnik ovisi od internetske trgovine, ali on može iznositi i 30% za proizvode kao što su e-knjige. [10]

Google Adsense je napravljen s ciljem pojednostavljivanja procesa monetizacije putem prikazivanja reklama. Umjesto sklapanja dogovora s pojedinim oglašivačima i brige o kompliciranim sustavima koji prate stvari kao što su broj klikova na reklamu sve više ljudi prelazi na Google-ov sustav. Oglašivači daju novac Google-u za prikazivanje njihovih reklama, nakon čega Google prosljeđuje dio te zarade vlasnicima stranica na kojima se te reklame prikazuju. Trenutno Google zadržava 32% ukupnog novca koji plaćaju oglašivači. [11]

# 7. Zaključak

U ovom diplomskom radu izrađena je web aplikacija za obavještavanje korisnika o bitnim terminima. Proučene su i analizirane slične aplikacije Wunderlist i Timetree kako bi se utvrdio način na koje one funkcioniraju i kako bi aplikacija toga tipa trebala izgledati. Nakon pregleda sličnih aplikacija osmišljen je okvirni dizajn aplikacije koristeći alat Figmu. Poslije osmišljenog dizajna počinje razvoj aplikacije u Laravel okviru. Prvi korak razvoja unutar Laravela bio je stvaranje migracija pomoću kojih su se stvorile tablice u bazi podataka. Nakon tablica izrađeni su modeli, to jest atributi kojima je bilo potrebno odrediti svojstvo popunjivosti i odgovarajuće relacije. S obzirom da modeli moraju imati svrhu, potrebni su upravitelji i pogledi. Upravitelji sadrže programski kod koji obavlja logičke zadaće aplikacije. Pogledi se uporabljuju za stvaranje interakcije između korisnika i web aplikacije. Kod pogleda prioritet je bila funkcionalnost i estetika izgleda. U aplikaciji je stvoren sustav prijave i registracije novih korisnika. Korisnici koji se registriraju imaju mogućnosti dodavanja prijatelja, stvaranja događaja, brisanja događaja i uređivanja profila. Također je stvoren i sustav obavještavanja putem elektroničke pošte, koji obavještava korisnike o bitnim događajima kao i o otkazivanju istih. Zahvaljujući držanju načela koja omogućuju vrlo lagano čitanje i nadogradnju programskog koda, aplikaciji se mogu dodati nove funkcionalnosti. Tako bi se u budućnosti mogao implementirati sustav za razmjenu poruka u stvarnom vremenu. Također, aplikaciju bi bilo moguće i monetizirati koristeći strategije kao što su partnerski marketing i Google Adsense.

# LITERATURA

[1] – Laravel, https://laravel.com, pristupljeno siječanj 2020.

[2] – HTML, https://www.w3schools.com/html/html_intro.asp, pristupljeno siječanj 2020.

[3] – CSS, https://www.w3.org/Style/CSS/Overview.en.html, pristupljeno siječanj 2020.

[4] – XAMPP, https://www.apachefriends.org/index.html, pristupljeno siječanj 2020.

[5] – PHP, https://www.php.net/, pristupljeno siječanj 2020.

[6] – phpMyAdmin, https://www.phpmyadmin.net/, pristupljeno siječanj 2020.

[7] – Bootstrap, https://getbootstrap.com/, pristupljeno siječanj 2020.

[8] – Wunderlist, https://www.wunderlist.com/, pristupljeno prosinac 2019.

[9] – TimeTree, https://timetreeapp.com/intl/en/, pristupljeno prosinac 2019.

[10] – Partnerski marketing, https://pisalica.com/sto-je-affiliate-ili-partnerski-marketing/, pristupljeno siječanj 2020.

[11] – Pay-Per-Click Advertising, https://www.wordstream.com/pay-per-click-advertising, pristupljeno siječanj 2020.

# SAŽETAK

**Naslov:** Web aplikacija za obavještavanje korisnika o bitnim terminima

Svrha izrade aplikacije je olakšati korisnicima praćenje i pravovremeno ispunjavanje svakodnevnih obaveza. Kroz rad je napravljen osvrt na korištene programske i skriptne jezike i tehnologije koji su korišteni pri izradi aplikacije. Također, opisane su tehnologije koje su korištene kod realizacije programskog dijela. Korišten je skriptni jezik PHP i okvir Laravel. Izrada aplikacije odrađena je na temelju MVC arhitekture na kojoj se i temelji Laravel. Aplikacija se sastoji od sučelja za: dodavanje prijatelja, stvaranje događaja, pregled nadolazećih događaja i uređivanja profila. Rezultat diplomskog rada je funkcionalna aplikacija s potencijalom za monetizaciju na tržištu.

**Ključne riječi:** PHP, Laravel, web aplikacija, MVC

# ABSTRACT

**Title:** Web application for informing users about important events

This master's thesis goal was to make it easier for users to monitor and fulfill their daily obligations in a timely manner. Throughout the work, a review was made of the programming languages and technologies used to create the application. Also, the technologies used in the implementation of the program part are described. The PHP programming language and the Laravel framework were used. The application was designed based on the MVC architecture on which Laravel is based. The application consists of interfaces to: add friends, create events, view upcoming events and edit profiles. The result of my thesis is a functional application with the potential for monetization in the market.

**Keywords:** PHP, Laravel, web application, MVC

# ŽIVOTOPIS

Marko Krajina rođen je 25. studenog 1994. godine u Tomislavgradu. U Tomislavgradu 2009. završava Osnovnu školu Ivana Mažuranića. Iste godine upisuje Gimnaziju Marka Marulića u Tomislavgradu gdje 2013. godine završava opći smjer. Nakon srednje škole obrazovanje nastavlja na Fakultetu elektrotehnike, računarstva i informacijskih znanosti u Osijeku koji upisuje 2013. godine. Nakon završenog preddiplomskog studija 2017. godine, upisuje diplomski studij Računarstva.

potpis: _____

# PRILOZI (NA CD-U)

Prilog 1. Dokumentacija diplomskog rada
Prilog 2. Programski kod aplikacije