

SVEUČILIŠTE JOSIPA JURJA STROSSMAYERA U OSIJEKU
ELEKTROTEHNIČKI FAKULTET

Sveučilišni studij

WEB SUČELJE SAMOPRILAGODLJIVOG ONLINE
TEST GENERATORA

Diplomski rad

Slaven Sakačić

Osijek, 2015.

Obrazac D1: Obrazac za imenovanje Povjerenstva za obranu diplomskog rada

Osijek,

Odboru za završne i diplomske ispite

Imenovanje Povjerenstva za obranu diplomskog rada

Ime i prezime studenta:	Slaven Sakačić
Studij, smjer:	Diplomski studij računarstva, smjer procesno računarstvo
Mat. br. studenta, godina upisa:	D-573R, 2012./2013.
Mentor:	Izv.prof.dr.sc. Ninoslav Slavek
Sumentor:	
Predsjednik Povjerenstva:	
Član Povjerenstva:	
Naslov diplomskog rada:	
Primarna znanstvena grana rada:	
Sekundarna znanstvena grana (ili polje) rada:	
Zadatak diplomskog rada:	
Prijedlog ocjene pismenog dijela ispita (diplomskog rada):	
Kratko obrazloženje ocjene prema Kriterijima za ocjenjivanje završnih i diplomskih radova:	Primjena znanja stečenih na fakultetu: Postignuti rezultati u odnosu na složenost zadatka: Jasnoća pismenog izražavanja: Razina samostalnosti:

Potpis sumentora:

Potpis mentora:

Dostaviti:

1. Studentska služba

U Osijeku, godine

Potpis predsjednika Odbora:



Sveučilište Josipa Jurja Strossmayera u Osijeku

ETFOS

ELEKTROTEHNIČKI FAKULTET OSIJEK



IZJAVA O ORIGINALNOSTI RADA

Osijek, 29.05.2015

Ime i prezime studenta:

Slaven Sakačić

Studij :

Diplomski studij računarstva

Mat. br. studenta, godina upisa:

D-573R, 2012./2013.

Ovom izjavom izjavljujem da je rad pod nazivom: **WEB SUČELJE SAMOPRILAGODLJIVOG ONLINE TEST GENERATORA**

izrađen pod vodstvom mentora Izv.prof.dr.sc. Ninoslav Slavek

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. TEORIJSKE OSNOVE PROBLEMA PROVOĐENJA TESTOVA	2
2.1. Primjena internet aplikacija za provođenje testova	2
2.2. Odabir programerskog okvira	4
3. TEORIJSKE OSNOVE TEHNOLOGIJA KORIŠTENIH	6
3.1. PHP 5.5.	6
3.2. PSR(PHP preporučeni standardi)	7
3.3. Composer	8
3.4. Laravel 5	11
3.5.1. Laravel MVC arhitektura	15
3.5.2. Laravel artisan CLI	17
3.5.3. Životni ciklus zahtjeva unutar Laravel 5	20
3.5.4. Laravel Homestead	21
4. OPIS KREIRANE INTERNET APLIKACIJE	22
4.1. Kreiranje baze podataka	22
4.2. Opis važnijeg koda internet aplikacije	28
4.3. Prikaz i opis izrađene internet aplikacije	34
4.4. Kreiranje i ažuriranje testova	41
4.5. Provođenje testova	47
5. ZAKLJUČAK	52
LITERATURA	53
SAŽETAK	54
ABSTRACT	55
ŽIVOTOPIS	56

1. UVOD

Računala su danas postala integralni dio naše okoline, većina ljudi koristi računala svaki dan za vlastito unapređivanje svojih vještina, za specijalizaciju, edukaciju ili zabavu. Cilj ovog diplomskog rada je bio izraditi internet aplikaciju koja pojednostavljuje način na koji se danas provode ispiti, odnosno kreirati platformu gdje ljudi mogu dijeliti svoja znanja sa ostalim korisnicima putem rješavanja mini testova, kvizova, anketa za istraživanja i sveobuhvatnih testova za potrebe škola, sveučilišta i raznih drugih ustanova. Posebna pozornost je provedena tijekom implementacije učiteljske korisničke ploče. Platforma ima mogućnost polaganja testova za studente koji brzo mogu pronaći potreban test i odmah dobiti povratnu informaciju koja im tada služi kao evidencija njihovog razumjevanja teme obrađene danim testom i sličnim tipovima testova. Platforma je u cjelini sigurna i dovoljno sveobuhvatna za različite vrste računalnih uređaja i različite potrebe učitelja i studenata.

U početnom dijelu su opisani problemi koji se javljaju putem provođenja testova pomoću običnih papirnatih načina u usporedbi s novim modernim načinom, gdje se temeljem internet sučelja preko *clouda* infrastruktura testovi automatski samogeneriraju za više studenata. U trećem poglavlju opisane su teorijske podloge izabranih tehnologija koje su smatrane najboljim za izradu rada. Četvrto poglavlje detaljno ulazi u objašnjavanje kako je kreirana baza podataka, te su prikazani važniji dijelovi koda koji su bitni za razumjevanje kreirane internet aplikacije. Na kraju četvrtog poglavlja prikazan je detaljan opis sučelja internet aplikacije sa objašnjenjem ključnih dijelova korisničkog sučelja.

2. TEORIJSKE OSNOVE PROBLEMA PROVOĐENJA TESTOVA

Sposobnost da se računalo učinkovito koristi je esencijalna vještina u današnjem modernom dobu. Računalne vještine omogućuju ljudima svih dobi razumjevanje i korištenje računalnih tehnologija za poboljšanje kvalitete njihovog osobnog i profesionalnog života. Upotreba računala u svakodnevnim aktivnostima povećava računalnu pismenost i omogućuje veću raspostranjenost usvajanja računala većeg broja ljudi i time djeluje pozitivno na cijeli populacijski razvoj. Certificirane računalne vještine mogu pomoći osobama da napreduju u profesionalnom životu. Uključivanje računala kao edukacijskog mehanizma danas postaje prioritet u svim razvijenim zemljama, a posebno u zemljama u razvoju. Kreiranjem internacionalnih inicijativa sa ciljem proširenja računalnog obrazovanja u nerazvijenim državama zahtjevu nove metode učenja u usporedbi sa „starim“ metodama utemeljenim na papiru i olovci, što postaje neefikasno s obzirom na količinu znanja kojim studenti moraju ovladati, opterećenjima za edukatore i broju potencijalnih studenata.

Kako stanovništvo evoluiralo u svom razvoju, a pogotovo u pristupu i upoznavanju s računalima, učinci uvođenja računalnih ispita u nastavi su znatno porasli. To je odraz sve većeg porasta dostupnosti računala i računalnih softvera svim grupama stanovništva. Ova računalna pismenost građana stvara potrebu za kreiranjem modernijih načina provođenja testova, anketa, kvizova i sličnog. Upotreba računala u svakodnevnom radu učitelja, nastavnika, profesora i odgajatelja danas znači poznavanje najnovijeg softvera koji oni mogu primjeniti u predavanjima. Ovaj softver primarno mora omogućiti brži i lakši način izvođenja određenog zadatka. Potreban je računalni softver koji će modernizirati ovu potrebu i developeri ovog softvera moraju uvijek imati u vidu buduće potrebe koje predavači mogu imati u budućoj implementaciji ili verziji softvera.

2.1. Primjena internet aplikacija za provođenje testova

S porastom internet korisnika u svijetu, potreba za internet aplikacijama koje služe za edukaciju i užu specijalizaciju postaje sve rasprostranjenija. Internet aplikacije koje provode neku vrstu edukacije postaje integralni dio ne samo edukacijskih ustanova nego i kompanija za edukaciju zaposlenika za razne vrste uže specijaliziranih smjerova u svakodnevnom poslu. U novije vrijeme rapidno raste broj edukacijskih kompanija koje istražuju i implementiraju internetsku provedbu ocjenjivanja i polaganja testova kao dio programa procjene znanja svojih korisnika, odnosno studenata. Provođenje testova se brzo prebacuje na internet gdje takav oblik stvaranja i

isporuke testova, kvizova i anketa nudi niz prednosti u odnosu na papirne verzije istih testova i kvizova. Provođenje testova u institucijama, sveučilištima, radnim mjestima i sličnim asocijacijama, danas se prepoznaje kao korisna prednost koju pruža internet aplikacija dizajnirana za automatsko generiranje takvih oblika provjere znanja. Internet aplikacije pružaju idealan temelj za stvaranje računalne okoline koja će omogućiti predavačima i ostalima kreiranje i distribuciju znanja i ideja preko svih kontinenata i time pridonijeti kolektivnom znanju. Kreiranje testova pomoću internet aplikacija omogućuje učiteljima da se fokusiraju na isticanje najboljeg načina kako da razrade i prenesu neku temu do studenta kroz mini-testove i kvizove. Kreiranje testova gdje se omogućuje korištenje više formata tipova odgovora u mini-testovima i kvizovima zapravo ima stvarni potencijal za poboljšanje studentskog učenja.

Mnoge internet aplikacije koje omogućuju uslugu rješavanja testova pružaju studentima izravnu povratnu informaciju, i gdje je moguće uključivanje slika i video zapisa što proces polaganja testa čini zanimljivijim. Mini-testovi tj. testovi koji su manje formalni mogu biti kao nisko opasne (engl. low-stakes) formativni načini procjene sa svrhom poticanja studenata da uče i zadrže stečeno znanje, i gdje procjene omogućuju nastavnicima da bolje strukturiraju svoju pouku. Ovi mini-testovi za provjeru mogu biti i složeniji, kao vrsta provjere znanja dizajnirani za testiranje manjih skupova specijaliziranih vještina. Ovaj način provođenja testova omogućuje mnoge prednosti, a najvažnija prednost je ta da predavačima uštedi vrijeme provjere znanja. Pružanje profesionalno dizajniranih, multimedijalnih tečajeva može biti mnogo privlačnije od testova između četiri zida učionice sa rješavanjem testova na papiru. Provođenje testova u učionici čini studente nervoznima i time postižu lošije rezultate (engl. underperform), dok testovi na računalu mogu imati neograničeni broj pokušaja na pojedinom mini-testu i postoji mnoštvo koristi preko kojih studenti postižu bolje rezultate. Takav vid ispita preko mini-testova koji je stvoren s namjerom da bude osobna provjera znanja može eliminirati potrebu intenzivnog apsorpiranja velike količine informativnih činjenica u kratkom vremenu i na taj način poticati studenta na redovno učenje i usvajanje stečenog znanja. Danas studenti mogu pomoću kvizova i mini-testova učiti bez vremenskog ograničenja, što im dopušta da slobodno vrijeme na poslu ili kod kuće koriste na produktivniji način.

Ključne prednosti za učitelje koje su identificirane kod polaganja mini-testova pomoću računala su :

1. Računalom temeljeno ispitivanje predstavlja zeleno računalstvo (engl. green IT) strategiju koja se koristi za smanjenje potrošnje papira

2. Može se distribuirati u više verzija test, bez potrebe uvida i praćenja koji studenti su primili koju verziju testa
3. Smanjuje stres jer ušteduje vrijeme ispisivanja, odnosno rješavanja
4. Smanjuje vrijeme ocjenjivanja
5. Povećava točnost ocjenjivanja
6. Metoda provođenja je efektivnija i učinkovitija
7. Smanjuje neugodnu situaciju čitanja neurednog rukopisa
8. Povećava transparentnost ocjenjivanja
9. Svi podaci mogu biti pohranjeni na jednom internet poslužitelju
10. Uključivanje grafika čini testove više interaktivnim nego kod papirnih testova
11. Eliminira ljudske pogreške u ocjenjivanju

Ključne prednosti za studente koje su identificirane za polaganje mini-testova pomoću računala su :

1. Smanjenje stresa
2. Sposobnost pružanja povratne informacije, tako da studenti mogu odmah znati što su krivo napravili i zašto
3. Izgrađuje povjerenje o odgovorima
4. Odgovori su vidljivi automatski
5. Rad ispred računala je udobniji
6. Sučelje pruža dodatnu potporu
7. Olakšava promjenu, tj. ispravljanje odgovora
8. Više slobode da se upotrijebe alternative metode rješavanja
9. Omogućuje lagano postizanje maksimalne ocjene

2.2. Odabir programerskog okvira

Vrijedno je razmatrati korištenje programerskog okvira prilikom izrade projekta kada je vrijeme ograničeno i programerske vještine za složeni projekt ne odgovaraju visoko zahtjevnoj razini potrebnoj za izgradnju složenog projekta, tj. složene internet aplikacije. Uporaba robustnog programerskog okvira se preporuča kada je sigurnost internet aplikacije nužan uvjet. To čak postaje nužnost kada programer nema potrebnih znanja i kada ima vremensko ograničenje za izradu aplikacije i kada ima potrebu spriječavanja narušavanja sigurnosnih aspekata aplikacije. Većina modernih programerskih okvira ima ugrađene sigurnosne značajke.

Popularnost programerskih okvira je narasla tijekom posljednjih pet godina. Došlo je do velikog pomaka od ručnog pisanja cijelog koda do iskorištavanja popularnih programerskih okvira s unaprijed izgrađenim komponentama i značajkama. Programerski okviri omogućuju rukovanje svim ponavljajućim osnovnim zadaćama, dopuštajući fokusiranje na poslovnu logiku i opću strukturu projekta kao cjeline. Time postaju idealan alat koje se koristi za brzo izgrađivanje složenih operativnih prototipova u kratkom vremenu, s minimalnim vremenom potrebnim za kodiranje. Ne samo da ovi okviri uvode nove načine pristupa problemima, okviri također izazivaju prethodno znanje o određenom području i pokazuju učinkovitiji način ostvarenja određenih zadataka. Iznad svega, njihov cilj je da programerima pomogne proizvesti bolji kod, te da postanu produktivniji i efikasniji u radu.

Izabran programerski okvir je Laravel baziran na PHP jeziku koji ima mnoštvo prednosti nad konkurentnim PHP okvirima. Glavni i odlučujući faktori za odabir Laravel je iznimno kvalitetna dokumentacija i dostupnost materijala za učenje, iznimno lagan i opisni jezik koda te količina usluga koje pruža. Laravel omogućuje kreiranje agnostičkog koda za programerske okvire. Korištenjem Laravela u izgradnji internet aplikacija programer postaje indirektno više upoznat sa pojmovima koji su prihvatljivi unutar programerske zajednice i prenosivi na bilo koji drugi programerski jezik ili okvir. To uključuje MVC paradigme i objektno orijentirani programerski dizajnerki uzorak, korištenje sadržajnih menadžera ovisnostima, ispitivanje, *dependency injection*, te uključuje snage i ograničenja objektno relacijskog mapiranja, te migracije baze podataka.

Svladavanje novog programerskog okvira, kao što je Laravel, može biti izazovno, ali ujedno i vrlo korisno iskustvo.

3. TEORIJSKE OSNOVE TEHNOLOGIJA KORIŠTENIH

3.1. PHP 5.5

PHP je naširoko korišten serversko tumačen skriptni jezik koji je otvorenog koda i posebno pogodan za programiranje internet aplikacija. Dio je slobodnog LAMP stoga (engl. stack) (Linux, Apache, Mysql, PHP) i na taj način je dostupan na gotovo bilo kojem serveru. PHP je tipično upotrebljen sa internet serverom kao apache i nginx za posluživanje dinamičkog sadržaja. Može se koristiti i za izgradnju snažnih naredbenih aplikacija (CLI aplikacija).

Od siječnja 2013. godine, PHP je bio instaliran na više od 240 milijuna internet stranica (39% onih ispitanih) i 2.1 milion internet poslužitelja. [1]

U prošlosti, uobičajeno je praksa bila iskodirati PHP kod u datoteci sa .php tipom datoteke, postavljanje datoteke na radni poslužitelj pomoću ftp protokola i ftp softvera, i nadati se da radi. Danas je norma postavljanja PHP datoteka i ostalih podataka korištenjem distribuiranih sistema za praćenje, upravljanje izvornim kodom i kontrolom revizija datoteka kao što su *git*, *mercurial*, *bazaar* i slični servisi. [2]

PHP postaje danas moderni skriptni jezik s novim značajkama dodanim nakon nadogradnje na verziju PHP 5.3., nadogradnje poput *namespaces* koje služe kao način inkapsuliranja PHP klasa, *closures* koje omogućuju stvaranje funkcija koje nemaju naznačeno ime (najkorisnije kao vrijednosti povratnih parametara, ali imaju i mnogo drugih koristi).

Neke značajke unutar PHP jezika su neophodne za danas, *namespaces*, na primjer su temelj modernog PHP standarda koji omogućuju nove i moderne razvojne prakse. *Namespaces* omogućuju organizaciju PHP koda u virtualnu hijerarhiju, usporedivo s strukturom datotičnog sustava operacijskih sustava. Danas svaka moderna PHP komponenta i moderni PHP programerski okviri (engl. frameworks) organiziraju svoj kod pod vlastitim globalno jedinstvenim *vendor namespaceom*, tako da nisu u konfliktu sa uobičajenim klasnim imenima korištenim od drugih proizvođača.

Prema [2, str. 21] *namespaces* su praktički zapisi unutar PHP jezika koji su referencirani od strane PHP interpretera da primjeni zajednički prefiks na skupu klasa, sučelja, funkcija ili konstanta.

Nakon 5.4.x verzije postoji mogućnost korištenja *traita*, mehanizama ponovne upotrebe koda tako da se smanje neka ograničenja singularnog nasljeđivanja na način da se omogući programeru ponovna upotreba skupa metoda u nekoliko neovisnih klasa koje žive u klasnoj hijerarhiji.

Još jedna važna značajka unutar 5.4.x verzije je ugrađeni internet server dizajniran posebno za razvoj i testiranje. Ovaj ugrađeni internet server omogućuje testiranje koda bez potrebe za punopravnom *LAMP* stogom s konfiguracijom na lokalnom razvojnom sustavu.

Podrška za generatore je dodana verzijom 5.5.x. Prema [3] opisano je da generatori pružaju jednostavan način za implementaciju jednostavnih iteratora bez dodatne složenosti od implementacije klasa koje implementiraju sučelje iteratora.

Izvorni PHP skriptni *engine* je *zend engine* (program koji parsira, interpretira, i izvršava PHP kod) koji transformira izvorni PHP kod u optkod kao oblik međukoda, te izvršava taj optkod izravno na *zend engine* virtualnom procesoru.

Zend engine je otvoreni kod napisan u C-u, koji omogućava upravljanje resursima, memorijom i drugim standardnim uslugama PHP jezika. *Zend engine* se unapređuje brzim tempom s novim značajkama dodavanim i sa poboljšanim performansama.

Međutim, sada postoje drugi veliki PHP *engini* kao što je HipHop Virtual Machine (HHVM) kreiran od strane kompanije Facebook namijenjen za postizanje superiornijih performansi uz zadržavanje fleksibilnosti razvoja koju PHP pruža. Kompanija Facebook koristi HHVM za implemenatiju svog PHP koda koji je potreban da bude učinkovit i optimiziran za rukovanje vrlo velikim, i vrlo mnogo korištenim PHP kodnim bazama.

HHVM koristi načelo JIT (*just in time*) kompilacije gdje omogućuje virtualnom stroju da napravi pametnije odluke za vrijeme izvršenja koda. Izvršeni PHP kod je prvo transformiran u posredni „hiphop“ bytekod, koji je potom dinamički preveden na x86-64 strojni kod, optimiziran i nativno izvršen. HHVM je optimiziran za rukovanje vrlo velikim i intezivno korištenim PHP kodnim bazama.

PHP jezik se brzo razvija i održavan je od strane desetke vodećih programera iz cijelog svijeta. Budućnost PHP je obećavajuća sa novim projektima poput PHP 7, HHVM, Hack, PHP-FIG.

3.2. PSR (PHP preporučeni standardi)

Umjesto da PHP okviri neprestano iznova rješavaju iste probleme, PHP programerski okviri bi trebali usvojiti neke preporuke te graditi na zajedničkim rješenjima. Time PHP programerski okviri moraju govoriti zajednički jezik koji im omogućuje da komuniciraju i dijele izvorni kod. Potrebni su neki standardi. Na PHP konferenciji 2009. predstavnici različitih PHP projekata raspravljali su o mogućnostima povezanog rada između njihovih programerski okvira i projekata. Ovi predstavnici su se nazvali PHP grupa standarda (engl. standards group), danas, oni djeluju pod okriljem *Framework Interoperability Group* (FIG). Cilj ove grupe je stvaranje dijaloga između predstavnika ovih projekata, sa svrhom pronalaženja načina da rade zajedno (međusobna razmjena novih ideja i međusobna komunikacija).

Umjesto da programeri radeći na projektu od početka kreiraju izvorni kod koji rješava neki potrebiti problem mogu integrirati već izrađenu klasu iz nekog drugog PHP okvira ili projekta koji već rješava dani problem prema nekim standardima koji tada omogućuju laganu implementaciju u projekt.

PSR je skraćenica za PHP standardne preporuke. PSR su standardi koji su preporučeni od strane FIG grupe. Njihova imena počinju sa PSR i završavaju s brojem. PSR-1, PSR-2, PSR-3, PSR-4 su dosad preporučeni standardi. Prvi standard implementiran od strane FIG grupe PSR-0 je danas zastario. Ovi standardi ili preporuke su besplatne i mogu biti usvojene od strane bilo koje osobe ali nitko nije dužan to učiniti. No, većina današnjih PHP okvira poput laravel, zend, codeigniter, symfony su usvojili sve ili neke od ovih standarda i sastavni su dio njihovih jezgrovnih servisa i klasa. Također svi današnji moderni PHP paketi na internetu usvajaju ove standarde.

U Laravel 5 okviru integriran je novi PSR-4 standard za automatsko učitavanje klasa koji je prvi PHP programerski okvir koji je integrirao ovaj standard.

3.3. Composer

Composer je multi platformni sustav za upravljanje ovisnostima za najsuvremenije PHP pakete. Koristi se na komandnoj liniji i gdje automatski preuzima sve potrebne ovisnosti specifične biblioteke ili paketa.

Composer je revolucionizirao način na koji se izrađuju PHP aplikacije, danas je integralni alat za upotrebu u razvoju internet aplikacija u PHP programerskom jeziku. Composer oslobađa programera od potrebe konfiguriranja i instaliranja već postojećih rješenja za najčešće zadatke i

probleme za koje većina projekata i biblioteka je besplatno dostupno na internetu i sa otvorenom kodom (engl. open source). Time omogućava programeru fokusiranje na izradu funkcionalnosti svoje internet aplikacije. Najčešće ovi projekti i biblioteke su napravljene sa standardnim specifikacijama preopručenim od strane FIG grupe na umu, što omogućuje tada lakše razumijevanje ishodišnog koda i jednostavnije integriranje biblioteke u internet aplikaciju.

Uključivanjem paketa u projekt, *composer* automatski kreira *vendor* direktorij u korijenskom direktoriju projekta, gdje onda se pohranjuju ne samo tražene biblioteke, ali i sve druge ovisnosti tj. ovisne biblioteke koje dani paket zahtjeva da normalno radi, sve ovo se izvodi rekurzivno i automatski.

Konfiguracija *composer*a za dani projekt, sa njegovim ovisnostima definira se validnom JSON datotekom, nazvanom *composer.json*. Ova datoteka obuhvaća podatke koje *composer* koristi za pronalaženje, instaliranje i automatsko mapiranje komponenata PSR-4 *namespacea* na zadani direktorij.

Composer pročitava sadržaj *composer.json* datoteke i pročita imena svih PHP komponenti te povezuje se na online repozitorij na internet stranici <https://packagist.org>, gdje tada razriješuje sve ovisnosti pročitanih PHP paketa i njihovih ovisnosti rekurzivno i te sve ovisnosti su tada preuzete na lokalni datotični sustav, u već kreirani ili novo kreirani *vendor* direktorij. Nakon što su ovisnosti preuzete i konfigurirane automatski, ova konfiguracyjska stanja su spremljena u datoteku nazvanom *composer.lock*. Ova datoteka tada sadrži popis svih PHP komponenata koje se koriste u projektu i svi egzaktni brojevi verzija (uključujući *major*, *minor* i *patch* verzije) tih PHP komponenata.

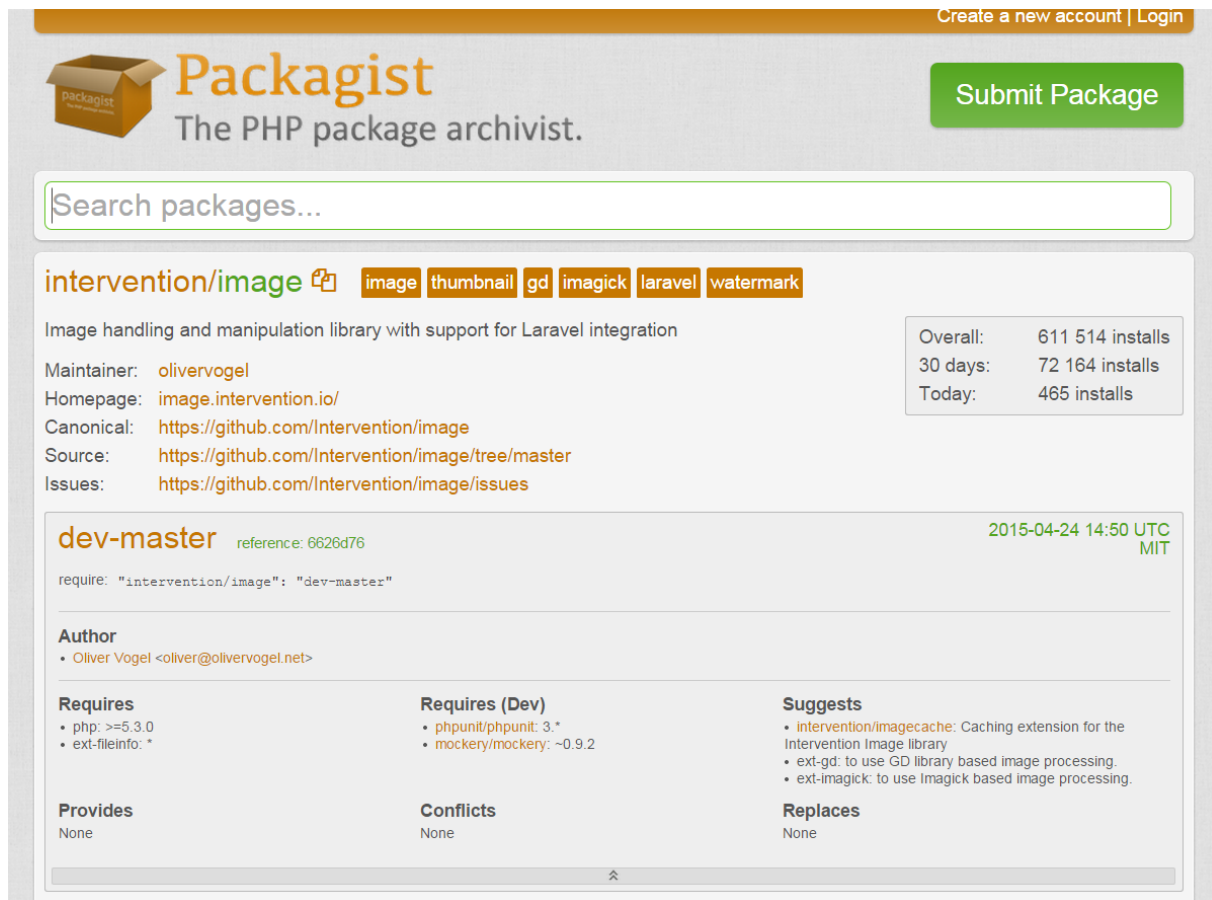
Composer također generira *autoload.php* datoteku u korijenskom *vendor* direktoriju koji se tada uključuje u projekt unosom priloženog koda na vrh klase:

```
require vendor/autoload.php;
```

Time se automatski učitaju sve potrebne klase skinutog paketa.

Na slici 3.1. vidimo službenu internet stranicu za objavljivanje PHP paketa za *composer* i također se vidi biblioteka koja je potrebna za integriranje u internet aplikaciju u ovom radu. Ime paketa je prepoznatljivo po osnovnom imenu paketa i nadnimkom, odvojeno kosom crtom (/). Biblioteka koja je izabrana nazvana je *intervention/image*, koja služi za rukovanje i

manipuliranje digitalnim fotografijama, i sa izrađenim mehanizmima za laganu integraciju sa laravel programerskim okvirom.



Slika 3.1. Izgled stranice i pronalaženje paketa *intervention* za potrebe izrade aplikacije

Nakon instalacije *composer* i pronalaženjem paketa koji želimo koristiti u projektu unosom sljedeće komande u komandnu liniju dohvaća se ova biblioteku u projekt:

```
composer require intervention/image
```

Nakon što *composer* instalira sve ovisnosti, *composer* unese sve točne verzije u *composer.lock* datoteku. U ovoj datoteci se ažuriraju verzije paketa i njihove ovisnosti na specifične verzije.

Time ne moramo osobno konfigurirati verzije ako poslije mislimo uključiti ostale programere na projekt. Oni moraju samo unijeti sljedeću komandu:

```
composer install
```

Time njihovo razvojno okruženje na računalu za njihovu kopiju internet aplikacije postaje konfigurirano sa istim verzijama i postavkama unesenih paketa i biblioteka.

Želi se poslije ažurirat ovisnosti na najnovije, unesu se sljedeće komande:

composer update

i time ažuriramo trenutne ovisnosti biblioteke na najnovije verzije koje smo postavili u composer.json datoteci.

Composer nam omogućuje ažuriranje jezgrenih modula. Jednostavnost dodavanje i održavanje biblioteka drugih proizvođača sa samo nekoliko naredbi. Laravel programerski okvir je besprijekorno integriran s composerom, gdje svaki laravel modul je objavljen kao composer paket.

3.4. Laravel 5

Laravel 5 je moderni PHP okvir otvorenog koda objavljen pod MIT licencom. Zahtijeva relativno noviju verziju PHPa, 5.3.7., objavljen je u kolovozu 2011. Konstruiran je za rapidni razvoj kompletnih internet aplikacija. Laravel je postao jedna od najpopularnijih i nadaleko najkorištenijih PHP programerskih okvira u relativno kratkom rasponu vremena. Laravel olakšava razvojni proces pojednostavljenjem ponavljajućih zadataka koji se koriste u većini današnjih internet aplikacija, uključujući, ali ne ograničavajući se samo na usmjeravanje, autentifikaciju, predmemoriranje (engl. caching) i sjedinavanje (engl. sessions). Laravel treba samo nekoliko redaka konfiguracije PHP koda i postaje spreman za korištenje. Najčešće će to biti konfiguracije ovjere autentičnosti prema bazi podataka koja je korištena u internet aplikaciji (MySQL, Postgres, SQLite i SQL Server), ovjere elektronske pošte i ovjere raznih usluga pružanih od strane drugih servisa.

Svaki novi laravel projekt opremljen je već zadanom strukturom direktorija i većina datoteka već je konfigurirana unutar zadane strukture, time omogućavajući brzi početak razvoja internet aplikacija.

Laravel kombiniran s snagom *composera* daje programeru više slobode u odabiru željenih paketa i biblioteka koje planira koristiti u svojem projektu. Kao zadana komponenta električne pošte koja dolazi sa laravelom, *swiftmailer*, ako se želi zamijenit s nekim poželjnijim paketom kao što je npr. *phpmailer* paketom, zamijena ova dva paketa će biti vrlo jednostavan zadatak.

Ažuriranjem na laravel 5.0. verziju uvodi se nova struktura direktorija u usporedbi sa starijim verzijama. Ova nova struktura služi kao bolji temelj za izgradnju snažnijih internet aplikacija unutar *laravela*, i obuhvaća novi FIG standard PSR-4 automatskog učitavanja datoteka unutar cijele aplikacije.

Zadana datotična struktura u *laravelu* namjerava pružiti laganu polaznu točku za velike i male aplikacije. Imenovanje mapa i datoteka je izražajna, i time je lako odgonetnuti za što svaka mapa i datoteka je zapravo korištena. Na slici 3.2 ovaj vidimo početni strukturni izgled važnijih direktorija. Laravel nameće gotovo nikakva ograničenja na to gdje se nove klase, direktoriji i datoteke trebaju nalaziti. Iako postoje najbolje predložene konvencije na internetu i knjigama koje su i korištene u ovom radu, razumjevanje za postojanje ove slobode postavljanja klasa i direktorija je važna za upotpunjavanje znanja ovog programerskog okvira.



Slika 3.2. Izgled početne strukture direktorija laravel aplikacije

Sav kod internet poslužitelja (jezgreni kod aplikacije) se nalazi unutar *app* direktorija. *Config* direktorij, za koji samo ime implicira, sadrži sve konfiguracijske datoteke internet aplikacije. *Database* direktorij sadrži sve migracije (engl. migrations) potrebne za kreiranje dizajna i sheme baze podataka i sve podatke sa kojima planiramo popuniti bazu podataka (engl. *seeds*). *Public* direktorij sadrži korijensku datoteku *index.php*, moguće slike, javascript, css i ostale javno dostupne datoteke koje planiramo uključiti i prikazati na korisničkom sučelju. *Resource* direktorij

sadrži pogleda (engl. *views*), lokalizacijske datoteke, *less*, *sass* i ostale stilske datoteke. *Vendor* direktorij sadrži sve *composer* ovisnosti i sve pakete uvezene uključujući komponente koje čine laravel programerski okvir.

Laravel predstavlja robustan skup alata i arhitekturu koja uključuje najbolje značajke od nekih najpopularnijih okvira kao što su codeigniter, yii, asp.net mvc, ruby on rails i sinatra. Laravel inkorporira već postojeće komponente drugih projekata da pruži jedan kohezivan sloj na kojem se tada izgrađuju internet aplikacije na mnogo strukturniji, kohezivniji i pragmatičniji način.

Značajne promjene uvedene sa Laravel 5.0. verzijom su sljedeće:

- **Nova struktura direktorija** – struktura direktorija je izmjenjena da bolje reflektira preporučljiv način kodiranja većine laravel programera i da ukloni muke nametnute starom strukturom direktorija potrebne za razumjevanje najboljih razvojnih praksa.
- **HTTP *middleware*** – koje nudi jedno, dosljedno sučelje koje tada zamjenjuje sve vrste filtera, time omogućavajući laganu provjeru i odbijanje http zahtjeva prije nego zahtjev uđe u aplikaciju.
- ***Type-hinting*** – pomoću PHP sadržajnih reflektirajućih (engl. *reflection facilities*) klasa i biblioteka očitavaju se i provjeravaju metode klasa u *laravel* projektu i ustanovljava se koje klase unijet (engl. *import*) u metodu.
- ***Flysystem*** – biblioteka koja apstraktira datotični sustav, pruža bezbolnu integraciju sa lokalnim datotičnim sustavom, Amazon S3 i Rackspace datotičnim sustavima. Sve preko jednostavnog i elegantnog programskog sučelja za aplikaciju (API).
- ***Form requests*** – mogu biti kombinirani sa *type-hinting* tipom uključivanja u metode kontrolera da pruže jednostavni način validacije unesenih korisnikovih podataka.
- **Ugovori** (engl. *contracts*) – centralno smješten set sučelja koje se mogu koristiti za *decoupling* i *dependency injection*.
- **Autentifikacija korisnika** – Zaštita internet aplikacije autentifikacijom korisnika, mogućnost registracije i prijave. Mogućnost provjere autentičnosti korisnika jednostavnim pozivom Auth statičkog sučelja unutar laravel programerskog okvira. Odgovarajući pogledi za ovu autentifikaciju korisnika su već kreirani.

Glavne značajke koje pruža laravel programerski okvir:

1. **Modularnost:** Laravel je sagrađen na vrhu 20 različitih biblioteka i čvrsto integriran sa *composerom*, tako da se može s lakoćom ažurirati i dijeliti.

2. **Usmjeravanje:** Laravel daje puno fleksibilnosti prilikom definiranja ruta. Postoje dvije vrste kontrolera za usmjeravanje, standardni kontroleri i resursni kontroleri. Posao usmjeravanja je da raspozna dolazne zahtjeve i pošalje odgovarajući odgovor. Obje vrste kontrolera se pridržavaju malo različitih konvencija. Standardni kontroleri prime *uri* i *closure* i vrate odgovor. Resursni kontroleri omogućuju definiranje *RESTful* kontrolera koji reagiraju na različite http glagole poput get, post, put i delete. Moguće je potpuno zaobići kontrolere i napisati cijelu logiku internet aplikacije u rutama.
3. **Query builder i Eloquent ORM:** Laravel je isporučen sa izražajnim graditeljem upita prema bazi podataka, koji omogućuje zadavanje upita bazi podataka s PHP sintaksom gdje metode se jednostavno lančaju umjesto da ručno se pišu SQL upiti. Laravel dolazi isporučen i sa ORM (engl. object relational mapperom) i ActiveRecord implementacijom, zvan Eloquent, koji je sličan onom koji se može naći kod Ruby On Rails programerskog okvira. Eloquent pomaže da se definira međusobna povezanost modela. I *query builder* i ORM su kompatibilni s raznim bazama podataka, kao što su PostgreSQL, SQLite, SQL Server i Mysql.
4. **Schema builder, migrations i seeding:** Također inspirirana od Ruby On Rails programerskog okvira, ove značajke omogućuju da definiramo shemu za bazu podataka s PHP kodom i pratimo sve promjene uz pomoć migracija. Migracija je jednostavan način opisa sheme kojom tada mijenjamo bazu podataka i omogućuje vraćanje koraka unazad u slučaju potrebe korekcije. *Seeds* koristimo za popunjavanje baze podataka sa podacima za potrebe testiranja i razvoja aplikacije.
5. **Električna pošta:** Sa svojom klasom, popularnom SwiftMailer bibliotekom, *laravel* čini slanje električne pošte vrlo lakom, čak i sa bogatim sadržajem i prilogima.
6. **Redovi:** Laravel je integriran sa nekoliko usluga reda čekanja, kao što su Amazon SQS i IronMQ, kako bi se omogućilo da se odgode intenzivni zadaci nad resursima, kao što je slanje električne pošte većem broju korisniku, koji se radije izvode u pozadini, nego da se korisnika zadržava dok zadatak ne završi.
7. **CSRF zaštita:** *Cross-Site Request Forgery* je napad koji prisiljava krajnjeg korisnika da izvrši neželjene akcije na internet aplikaciji u kojim je taj isti korisnik trenutno autoriziran. Također poznat kao jedan klik napad ili *session riding*, laravel okvir čini zaštitu od ovakvog napada jednostavnom, tako što generira CSRF token za svaku aktivno korisnikovu sjednicu unutar internet aplikacije. Ovaj token je upotrebljen kako bi se provjerilo da autoriziran korisnik je doista onaj koji stvara zahtjeve prema aplikaciji. Laravel također pohranjuje ove CSRF tokene u CSRF-token kolačiću. Također prilikom

stvaranja obrazaca, laravel automatski umeće skriveno *input* polje sa tokenom u html obliku koji pruža zaštitu od CSRF tipa napada.

3.5.1. Laravel MVC arhitektura

Model-Pogled-Kontroler (engl. model-view-controller) je arhitektonski uzorak (engl. pattern) za implementaciju korisničkih sučelja. MVC dijeli softversku aplikaciju u tri međusobno povezana dijela, kako bi se odvojila unutarnja reprezentacija informacije od načina na koji je prezentirana ta ista informacija ili prihvaćena od strane korisnika. [4]

MVC potječe iz rada Trygve Reenskaug tokom njegovog boravka u tvrtki Xerox PARC gdje je formulirao MVC uzorak za sučeljni dizajn grafičkog korisničkog softvera 1979. godine. [5]

MVC je brzo postao industrijska standardna praksa koja je korištena u svakoj modernoj razvoj okolini. Široko korišten i prihvaćen arhitektonski uzorak znači da su mnoge od najpopularnijih internet programerskih okvira izgrađeni oko ove arhitekture.

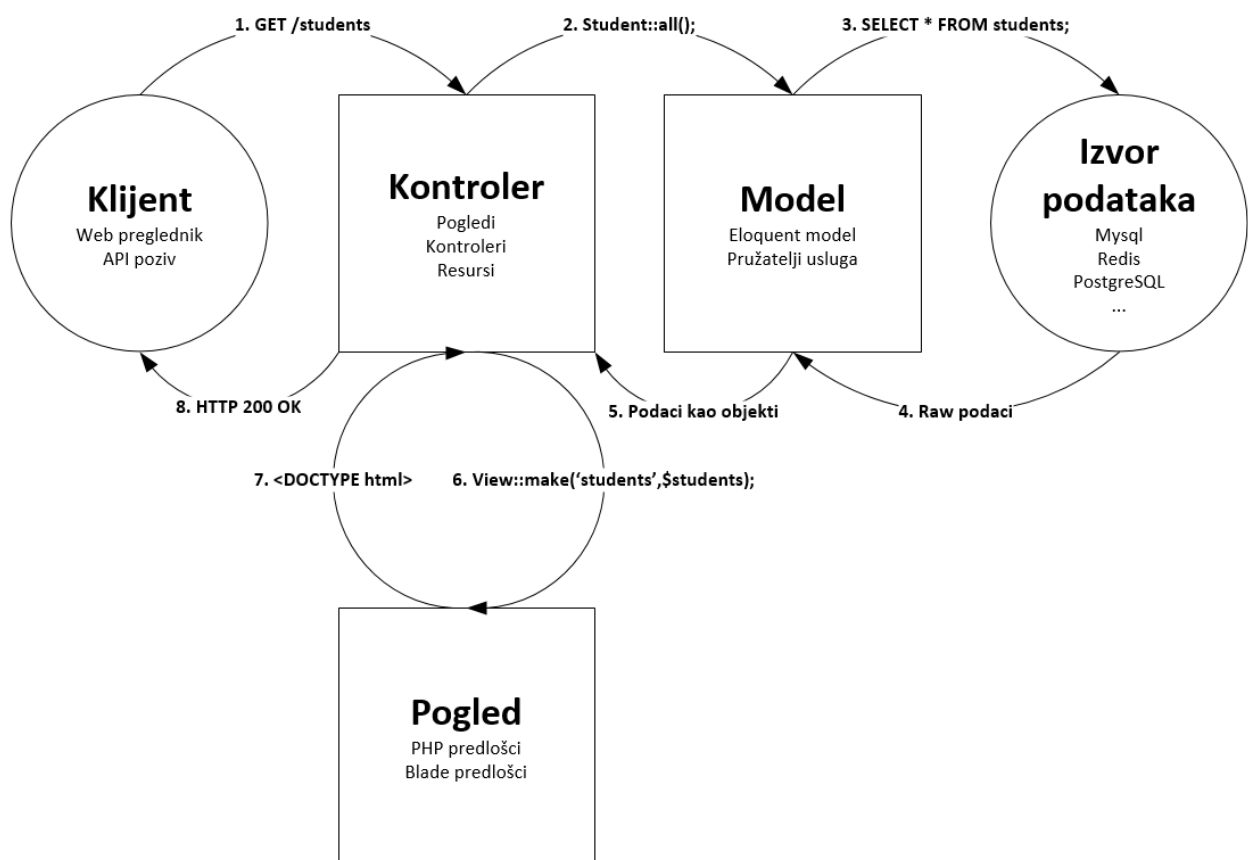
Laravel nije iznimka što se tiče ovog arhitektonskog uzorka. Laravel slijedi MVC arhitektonski uzorak, koji nameće razdvajanje poslovne logike od ulazne i prezentacijske logike koja je inače asocirana sa grafičkim korisničkim sučeljem (GUI).

Iako je i dalje moguće pisati nestrukturirani kod i ići protiv MVC paradigme i okvirnih konvencija, to često uključuje više napora od strane programera.

Postoje tri komponente MVC dizajnerskog uzorka:

1. **Modeli:** su entiteti aplikacije. Model treba sadržavati svu poslovnu logiku. U osnovi svaki važni entitet u aplikaciji, vjerojatno će trebati svoj model. Sve što je potrebno za definirati model u *laravelu* je napraviti novu klasu koja proširuje (engl. extends) *Eloquent* klasu. Dok sam naziv klase definiramo s jedninom imenice u *camelcase*-u, odgovarajuća tablica na razini baze podataka treba biti *pluralsnakecase* inačica imena klase. *Eloquent* klasa će znati da se model nazvan Question odgovara tablici questions u bazi podataka. *Laravel* također očekuje primarni ključ da bude nazvan id, te će tražiti *created_at* i *updated_at* polja da nadopuni automatski. Ako ovo zadano ponašanje nije prigodno za dani problem, može se poništiti i konfigurirati po potrebi. Modeli također sadrže podatke o tome kako se oni odnose sa drugim modelima. Korištenje Active Record terminologije, moguće je definirati *hasOne*, *belongsTo*, *hasMany*, *belongsToMany*, *hasManyThrough*, *morphTo*, *morphMany* relacije između tablica.

2. **Kontroleri i rute:** su odgovorni za obradu ulaznih podataka, djelovanje na modele i odlučivanje o tome koje akcije se trebaju obaviti, kao što je rendiranje pogleda ili preusmjeravanje na neku drugu rutu. Kontroleri se trebaju koristiti samo za prenošenje poruka između modela i pogleda.
3. **Pogledi i predlošci:** su odgovorni za prikazivanje podataka koje kontroleri dobiju od modela. Mogu se izgraditi korištenjem jezičnog predloška Blade ili jednostavno pomoću standardnog PHP jezika.



Slika 3.3. Ilustracija interakcije MVC komponenata u Laravel arhitekturi [6]

Prema dijagramu na slici 3.3. vidi se kako djeluje interakcija između svih sastavnih dijelova u MVC arhitektonskom uzorku.

Slijedi detaljni opis MVC arhitekture u Laravel programerskom okviru prema slici 3.3.:

Kada klijent pošalje GET zahtjev za prikaz svih studenata, definirana ruta /students prepoznaje se iz zahtjeva i pozove se ispravna klasa kontrolera i akcija unutar klase. Ova akcija sadrži potrebnu logiku koja tada preko koda poziva pružatelja usluge npr. *eloquent* model i *eloquent*

metodu `all()` koja izvlači iz izvora podataka sve studente kao sirov (engl. raw) podatak, neovisno o kojoj je bazi podataka riječ, ovaj izgeneriran sirov podatak je primljen nazad do model komponente koja pretvara taj sirov podatak u *eloquent* objekte. Dalje klasa kontrolera prima ove *eloquent* objekte i vrši određenu logiku nad njima, kao na primjer pretvaranje danih objekata u kolekciju (ona omogućava praktično manipuliranje dohvaćenim objektima u kontekstu laravel programerskog okvira kreirano pomoću *Collection* klase). Kontroler komponenta šalje ovu kolekciju podataka (`$students`) pogled komponenti *blade* predlošku koji tada rendira validan html kod na temelju poslane kolekcije. Na kraju kontroler šalje statusni kod 200 i taj validan html kod nazad klijentu.

3.5.2. Laravel artisan CLI

Artisan je ime komandno linijskog sučelja uključenog sa *laravel* programerskim okvirom. Izgrađen je na Symfony konzolnoj komponenti, omogućuje da se većina ponavljajućih i zamornih zadataka obave automatski unosom određene komande na komandnoj liniji, time automatizirajući sve vrste zadataka unutar laravel okruženja.

Pruž a niz korisnih naredbi za upravljanje, inspekciju i interakciju sa aplikacijom tokom razvoja kao što je pokretanje ugrađenog razvojnog servera, izvršavanje i upravljanje migracijama na bazi podataka, generiranje predložaka više vrsta klasa, pokretanje testova, konfiguracija internet aplikacije, punjenje baze podataka sa podacima za svrhu testiranja i drugo.

Sa *route* komandom mogu se pregledati koje rute su definirane, na koje http glagole reagira internet aplikacija kada http zahtjev zaprimi aplikacija, metode koje ih prihvataju, koja su imena tih ruta (*uri*), i ako su bilo koji filtri (*middleware*) primjenjivani prije i poslije svakog prijema zahtjeva.

Za prikaz ove tablice sa svim rutama, pozicioniranjem na korijenski direktorij projekta unosimo komandu u komandnu liniju:

php artisan route:list

Domain	Method	URI
	GET HEAD	users
	GET HEAD	users/create
	POST	users
	GET HEAD	users/{users}
	GET HEAD	users/{users}/edit
	PUT	users/{users}
	PATCH	users/{users}
	DELETE	users/{users}
	GET HEAD	tests/create
	POST	tests
	GET HEAD	tests/{tests}
	GET HEAD	tests/{tests}/edit
	PUT	tests/{tests}
	PATCH	tests/{tests}
	DELETE	tests/{tests}
	GET HEAD	answers/create
	POST	answers
	GET HEAD	answers/{answers}
	GET HEAD	answers/{answers}/edit
	PUT	answers/{answers}
	PATCH	answers/{answers}
	DELETE	answers/{answers}

Slika 3.4. Prikaz ispisa nekoliko ruta, njihovih metoda i uria

Na slici 3.4. i slici 3.5. vidimo ispise nekoliko registriranih ruta unutar internet aplikacije. Za svaku rutu koja je definirana sa *uri-jem* u tablici je prikazan kontroler sa akcijom koji ju prihvaćaju, i ime rute. Još se vide koji http glagoli su prihvaćeni za danu rutu i koji *middleware* filteri su implementirani.

Name	Action	Middleware
users.index	Dipl\Http\Controllers\UserController@index	auth
users.create	Dipl\Http\Controllers\UserController@create	auth
users.store	Dipl\Http\Controllers\UserController@store	auth
users.show	Dipl\Http\Controllers\UserController@show	auth
users.edit	Dipl\Http\Controllers\UserController@edit	auth
users.update	Dipl\Http\Controllers\UserController@update	auth
	Dipl\Http\Controllers\UserController@update	auth
users.destroy	Dipl\Http\Controllers\UserController@destroy	auth
tests.create	Dipl\Http\Controllers\TestController@create	auth
tests.store	Dipl\Http\Controllers\TestController@store	auth
tests.show	Dipl\Http\Controllers\TestController@show	auth
tests.edit	Dipl\Http\Controllers\TestController@edit	auth
tests.update	Dipl\Http\Controllers\TestController@update	auth
	Dipl\Http\Controllers\TestController@update	auth
tests.destroy	Dipl\Http\Controllers\TestController@destroy	auth
answers.create	Dipl\Http\Controllers\AnswerController@create	auth
answers.store	Dipl\Http\Controllers\AnswerController@store	auth
answers.show	Dipl\Http\Controllers\AnswerController@show	auth
answers.edit	Dipl\Http\Controllers\AnswerController@edit	auth
answers.update	Dipl\Http\Controllers\AnswerController@update	auth
	Dipl\Http\Controllers\AnswerController@update	auth
answers.destroy	Dipl\Http\Controllers\AnswerController@destroy	auth

Slika 3.5. Nastavak prikaza imena, akcija i middlewarea za rute sa slike 3.4

Tokom razvoja internet aplikacije nekad je potrebno pokrenuti kratke, jednokratne naredbe za inspekciju sadržaja baze podataka, umetanje nekih podataka u bazu podataka ili provjere sintakse

i rezultata *eloquent* upita. Stvaranje privremenih *closures* ruta koji će tada biti pokrenute za potrebe testiranja bi oduzimalo mnogo vremena i zahtjeva često prebacivanje između editora i internet preglednika što postaje dugotrajno za implementirati. Ove privremene rute stvaraju velike količine nepotrebnog koda unutar aplikacije koje poslije mogu postati sigurnosni rizik ako se zaborave ukloniti. Da bi male naredbe i promjene bile lakše provedene, *artisan* ima naredbu zvanu *tinker*, koja inicijalizira internet aplikaciju i omogućuje interakciju s njom preko komandne linije.

Sljedeća komanda unesena na komandnoj liniji:

```
php artisan tinker
```

Pokreće REPL (*Read-Eval-Print Loop*) jednostavno interaktivno programersko okruženje gdje se tada unose PHP komande s kojima direktno se komunicira sa internet aplikacijom u kontekstu i okruženju te internet aplikacije. Ishod ove komunikacije je direktni ispis rezultata na zaslonu.

Želimo li provjeriti da *eloquent* relacija jedan prema više (engl. one to many) implementirana između tablica *questions* i *answers* je ispravna za pitanje Examples of databases are. pod rednim brojem 8 tj. identifikacijskim brojem u tablici baze podataka, unosom komandi prema slici 3.6. dobijemo ispis rezultata iz kojeg se može zaključiti korektnost koja je definirana *eloquent* relacijom. Ustanovljujemo da je ispis ispravan te smo relaciju između tablica ispravno implementirali.

```
C:\wamp\www\diplomski>php artisan tinker
Psy Shell v0.4.2 (PHP 5.5.12 64bit cli) by Justin Hileman
>>> $answers = Dipl\Question::find(8)->answers;
=> <Illuminate\Database\Eloquent\Collection #00000000006f54a200000000bdea279a> [4]

    <Dipl\Answer #00000000006f54ad00000000bdea279a> <
        id: 16,
        answer: "a library catalogue",
        correct: 1,
        question_id: 8
    >,
    <Dipl\Answer #00000000006f54ae00000000bdea279a> <
        id: 17,
        answer: "an article index",
        correct: 1,
        question_id: 8
    >,
    <Dipl\Answer #00000000006f54af00000000bdea279a> <
        id: 18,
        answer: "a computerized warehouse inventory",
        correct: 1,
        question_id: 8
    >,
    <Dipl\Answer #00000000006f54a800000000bdea279a> <
        id: 19,
        answer: "the Internet",
        correct: 0,
        question_id: 8
    >
>>>
```

Slika 3.6. Testiranje modela kontrolera pomoću komandne linije

Par važnih značajki *artisansa* je da omogućuje stvaranje vlastitih naredbi, testiranje internet aplikacija, stvaranje i upravljanje poslovima koji se postavljaju u redove čekanja koji se onda izvode nakon određenog vremena (engl. queues). *Artisan* ima niz drugih komandi kao što su mogućnost migracije baza podataka i popunjavanje baze podataka sa sadržajem.

3.5.3. Životni ciklus zahtjeva unutar Laravel 5

Razumjevanje kako dolazeći http zahtjev komunicira sa laravel programerskim okvirom je jedno od važnijih koraka za razumjevanje i ovladanje danim okvirom. Životni ciklus http zahtjeva u *laravelu* je značajno različit nego kod običnih kreiranih PHP skripti koje su dohvaćene direktno preko njihovih *url* adresa (primjer GET <http://example.com/student.php>).

Polazna točka za sve http zahtjeve prema *laravel* internet aplikaciji je `public/index.php` datoteka. *Public* direktorij djeluje kao direktorij u kojem web poslužitelj počinje potragu za svim mogućim ulazećim zahtjevima. Svaki zahtjev koji ne odgovara postojećoj datoteci ili direktoriju prolazi kroz `public/index.php` datoteku. Nakon što je *url rewriting* pravilno pročitano i postavljen, posao `index.php` datoteke je registrirati *composer* automatski klasni *loader*, što govori PHP-u gdje da traži bilo koje klase koje su tada zvane. A zatim se dohvaća instanca *laravel* aplikacije preko `bootstrap/app.php` skripte kojoj se tada postavlja okolina bazirana na imenu hosta i povezuju se različiti putevi prema danom zahtjevu. Zatim, dolazni zahtjev je poslan na http kernel ili konzolni kernel, ovisno o vrsti zahtjeva koji ulazi u aplikaciju. Ako je zahtjev poslan na http kernel tada se definira niz *loadera* koji će biti pokrenuti prije nego je zahtjev izvršen. Ovi *loaderi* konfiguriraju rukovanje pogreškama, konfiguriraju autentifikaciju, otkrivaju aplikacijski okoliš preko *ENV* varijabla, te obavljaju druge poslove koji se trebaju obaviti prije nego zahtjev se obradi. Http *kernel* također definira http *middleware* kroz koje ulazni zahtjevi moraju proći da bi bili obrađeni od strane aplikacije. Nakon svega toga, poziva se *router* objekt koji provjerava koja ruta ili kontroler će se koristiti za dani zahtjev. Upotrebom http glagola i *url* primljenog od zahtjeva (na primjer, POST `/search`) zahtjev se mapira prema ispravnoj kontroler metodi ili ruti. Dalje metoda ili ruta provode određenu logiku u skladu sa zahtjevom i dohvaćaju podatke iz baze podataka i vraća određeni pogled.

Postoje mnogo izuzetaka gornjem životnom ciklusu zahtjeva. Gornji opis pokazuje „standardni“ životni ciklus, ali opet, može postojati cijeli drugačiji ciklus gdje je zaobijen ovaj „standardni“ ciklus. Moć laravel programerskog okvira je njegova fleksibilnost u konfiguriranju okvira prema vlastitoj potrebi.

3.5.4. Laravel Homestead

Laravel *homestead* je službeni, multi platformni, već zapakirani *vagrant* virtualni uređaj (softver za stvaranje i konfiguriranje virtualnog razvojnog okruženja) za laravel programerski okvir koji pruža razvojno okruženje bez potrebe za instaliranjem PHP, HHVM, internet poslužitelja, i bilo kojeg poslužiteljskog softvera na lokalnom računalu. [7]

Homestead omogućuje oponašanje proizvodnog internet poslužitelja sa postavkama na lokalnom računalu. Za konfiguriranje i mapiranje direktorija iz datotičnog sustava operacijskog sustava u datotični sustav *homestead* virtualnog uređaja konfigurira se skripta nazvana Homestead.yaml.

Nakon instalacije *homestead* virtualnog uređaja uključen softver je Ubuntu 14.04, PHP 5.6, HHVM, Nginx, MySQL, Postgres, Node.js, Redis, Memcached, Beanstalkd, Laravel Envoy, Fabric koji pružaju sustav koji iznimno brzo se pokreće i konfiguriran za optimalnu upotrebu sa laravel programerskim okvirom. U slučaju da se neka greška pojavi unutar *homestead* virtualnom uređaju, lagano je srušiti virtualni uređaj i ponovno stvoriti novi isti virtualni uređaj bez ikakvog utjecanja na lokalno okruženje računalnog sustava.

4. OPIS KREIRANE INTERNET APLIKACIJE

Prvi korak prilikom implementiranja zahtjevne internet aplikacije je odlučiti o tehnologiji i softveru koji će se koristiti tokom razvoja. Za potrebe ovog rada izabran je Sublime Text 3 tekstualni editor koji je robustan, funkcionalan i brz. Laravel *homestead* je izabran kao razvojno okruženje unutar kojeg je već konfiguriran softver koji je sličan raspoložljivom okruženju izabranom na *cloud* infrastrukturi nazvanoj *digitalocean* na koji će se internet aplikacija postaviti. Git je korišten kao način kontrole verzije izvornog koda i kao dostavni izvor datoteka i preko dploy.io servisa koji služi kao most između osobnog homestead razvojnog okruženja prema git repozitoriju do razvojnog okruženja ponuđen *cloud* infrastrukturom.

4.1. Kreiranje baze podataka

Za svaku složeniju internet aplikaciju danas podrazumjeva se potreba za bazom podataka sa mogućnošću skladištenje raznih postavki i korisnikovih podataka. U ovom radu je dizajnirana i implementirana baza podataka za potrebe skladištenja korisnikovih i studentovih unesenih podataka i postavki preko obrazaca.

Nakon kreiranja baze podataka, tablice i njihove relacije su izrađene pomoću *eloquent* objektno relacijskog mapiranja koji je uključen sa laravel programerskim okvirom.

Korištenjem *eloquent* objektno relacijskog mapiranja omogućena je jednostavna izrada dizajna baze podataka, gdje svaka tablica u bazi podataka ima odgovarajući model koji se koristi za interackiju sa tom tablicom.

Na slici 4.1. se vidi kreirani model nazvan Question.php koja će služiti kao referenca koja omogućuje interakciju unutar baze podataka sa tablicom questions.

```
class Question extends Model {  
    public $timestamps = false;  
    protected $fillable= array('question', 'points', 'shuffle_question', 'type', 'test_id',  
                               'question_image');  
    public function answers(){  
        return $this->hasMany('Dipl\Answer');  
    }  
    public function test(){
```

```

        return $this->belongsTo('Dipl\Test');
    }
}

```

Slika 4.1. Kod model kontrolera za tablicu questions

Za *eloquent* mapiranje nije potrebno implicitno definirati koja tablica će se koristiti za interakciju. Množina imena klase modela se koristi kao naziv tablice unutar baze podataka. Dakle, za ime klase modela Question množinsko ime questions će *eloquent* mapiranje postaviti kao zadano i smatrati da je odgovarajuća tablica unutar baze podataka. Naravno ovo zadano ponašanje se može poništiti eksplicitnim postavljanjem u skripti modela. Nakon što je definiran model, moguće je dohvaćati i stvarati nove zapise u tablici sa odgovarajućim laravel metodama. Korištenjem *eloquent* laravel programski okvir automatski održava polja unutar baze podataka nazvana *created_at*, *remember_token*, *deleted_at* i *updated_at*. Osim ako implicitno definiramo da ne želimo ova polja. Kada se koristi *query builder* sučelje koji je dio laravel-a, onda sami moramo iskodirati ova polja.

Na slici 4.1. i slici 4.2. vide se kako su relacije definirane između modela Question i Answer.

```

class Answer extends Model {
    protected $table = 'answers';
    protected $fillable = array('answer','correct');
    public $timestamps = false;
    public function question(){
        return $this->belongsTo('Dipl\Question');
    }
}

```

Slika 4.2. Kod model kontrolera za tablicu answers

Relacije između pitanja i odgovora je jedan na mnogo (engl. one to many). Što je definirano sa metodama answers unutar Question i question unutar Answer. Time govorimo da za svako pitanje u bazi podataka može postojati niti jedan ili može postojati više odgovora, i obrnuto govorimo da za svaki odgovor može postojati samo jedno pitanje. Ovom relacijom omogućujemo lakše dohvaćanje svih odgovora za dano pitanje pomoću eloquent metoda.

Nakon što su dizajnirani modeli za tablice, imena modela i tablica definiramo, pa kreiramo relacije pomoću *eloquent*, slijedeće je implementiranje tablica u bazi podataka. Ovo je moguće kreirati na više način, kao što je pomoću administratorskih softvera kao *phpmyadmin*. Laravel omogućava mnogo jednostavniji i zaista brži način, pomoću *laravel* shematskih (engl. *scheme*) klasa. Pomoću ovih *scheme* klasa kreiraju se migracijske klase, definiraju polja za tablice i *artisan* komandom kreiraju dane tablice. Shema klasa pruža agnostički način manipuliranja tablicama baza podataka, što znači da funkcionira unutar više vrsta baza podataka. Nakon kreiranja migracije poslije je lagano modificirati tablice promjenom polja u migracijskoj klasi i pokretanjem određene *artisan* komande.

```
class CreateQuestionsTable extends Migration {  
    public function up(){  
        Schema::create('questions', function(Blueprint $table) {  
            $table->increments('id');  
            $table->text('question');  
            $table->integer('points');  
            $table->string('question_image',400)->default(' ');  
            $table->boolean('shuffle_question')->default('0');  
            $table->enum('type', array('true_false', 'multiple_choice',  
                                     'multiple_response', 'fill_in'));  
            $table->unsignedInteger('test_id');  
            $table->timestamps();  
        });  
    }  
    public function down(){  
        Schema::drop('questions');  
    }  
}
```

Slika 4.3. Kreirana migracija za tablicu *questions*

Prema slici 4.3. vidimo migraciju potrebnu za tablicu *questions*, gdje su definirana polja koja će biti kreirana unutar tablice. Dvije funkcije su potrebne. *Up* metoda je pokrenuta kada je migracija izvršena sa namjerom promjene baze podataka kao što je kreiranje nove tablice,

ažuriranje polja tablice ili brisanje polja/cijele tablice. Down metoda služi za povrat promjena izvršenih unutar up metodi u prošlom koraku.

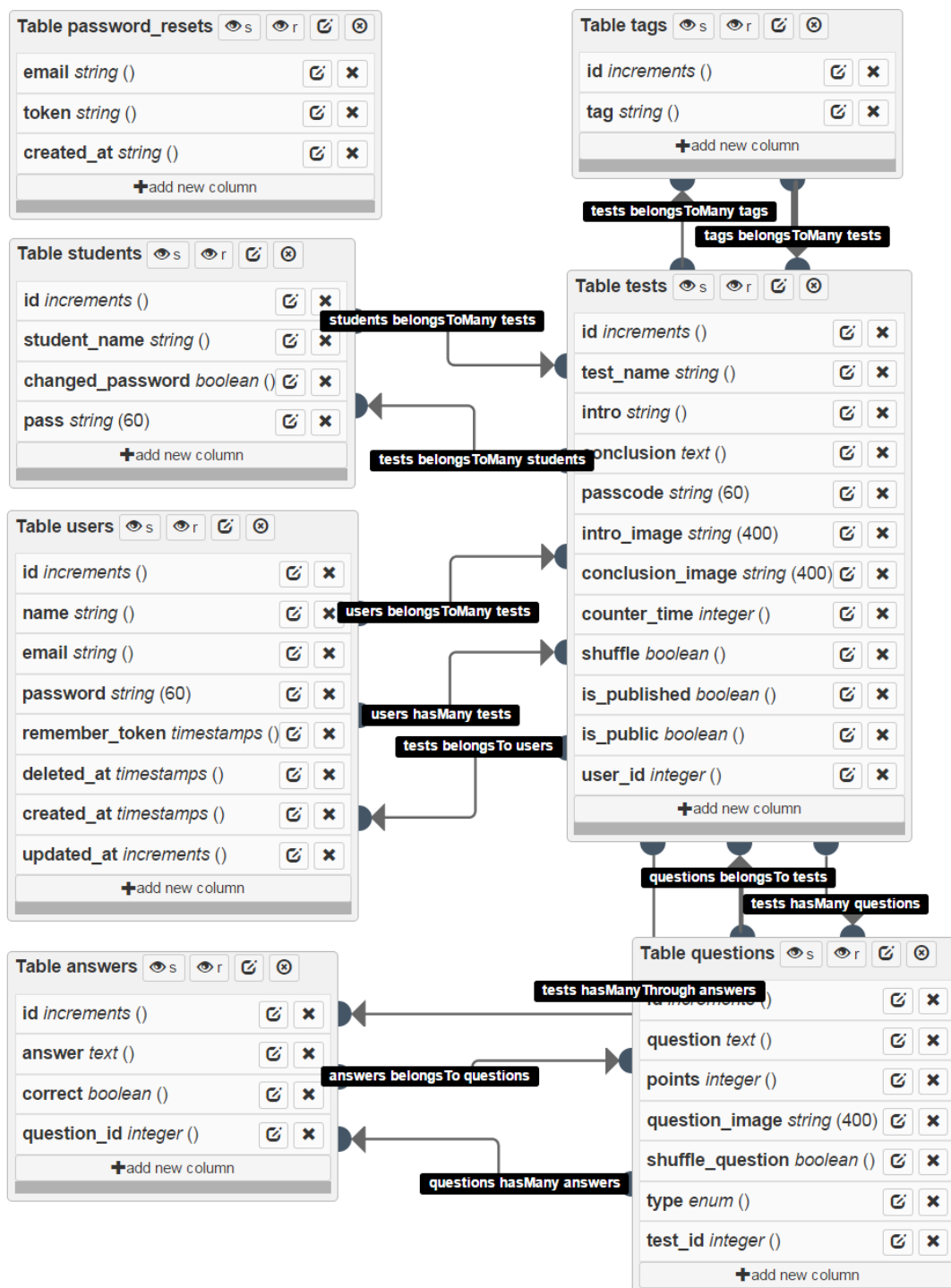
Konačni dizajn baze podataka se može vidjeti na (Sl. 4.4.). Vide se relacije kreirane između tablica.

Kreirane su sljedeće tablice za potrebe internet aplikacije:

- **users tablica** – sadrži imena korisnika, adresu njihove električne pošte, šifru potrebnu za autentifikaciju i vremenske oznake (engl. timestamps) zvani *remember_token*, *deleted_at*, *created_at* i *updated_at*.
- **students tablica** – sadrži studentovo ime, šifru registriranog studenta za autentifikaciju i *boolean* vrijednost tj. zastavica za indikaciju jer promjenjena šifra.
- **tests tablica** – tablica tests služi za pohranjivanje testova i postavki tih testova kreirani od korisnika. Sadrži ime za kreirani test, uvodnu sliku koja će dati vizulani identitet cijelom testu, završnu sliku za potrebe vizulanog opisa konačnih napomena nakon polaganja testa, uvodni tekst opisom namjene testa za studenta, završni tekst prikazan kada student obavi test (završne riječi), moguću šifru ako se odluči test napraviti privatnim, vrijeme potrebno za polaganje testa, strani ključ povezanosti sa users tablicom (*user_id*) i zastavice za objavu/odjavu testa, za privatni/javni test i za odabir slučajnog generiranja redoslijeda pitanja odgovarajućeg testa.
- **questions tablica** – tablica questions služi za pohranjivanje pitanja kreiranih za određeni test definiran *test_id* poljem u tablici. Sadrži opisni tekst pitanja, bodove koji predstavljaju cijelobrojnu težinu donesenog točnog odgovor na pitanje, opcionalna slika koja dolazi uz pitanje (kao opisna slika pitanja ili slika sa dodatnim informacijama potrebnim za razumijevanje danog pitanja), zastavicu za odabir da se slučajno generira redoslijed svih odgovora danog pitanja, strani ključ povezanosti sa tests tablicom nazvan *test_id* i sadrži još odabir tipa pitanja koji su implemetirani u internet aplikacija. Četiri tipa pitanja su implementirana *fill-in*, *multiple choice*, *multiple response* i *true false*.
- **answers tablica** – tablica answers služi za pohranjivanje svih odgovora za svako pitanje. Sadrži polje *question_id* što je strani ključ povezanosti sa tablicom *questions*, zastavicu za indikaciju jer odgovarajući odgovor točan ili netočan i polje teksta za opis odgovora.
- **tags tablica** – tablica tags služi za pohranjivanje novo kreiranih oznaka (engl. tags) za testove. Test može imati više tagova i gdje tag može imati više testova, dakle ova relacija između oznaka i testova je više prema više (engl. many to many). Ova tablica služi za

moгуćnost bržeg i jednostavnije pretraživanje testova na studentovoj kontrolnoj ploči koja može imat i druge primjene kao što je kategoriziranje testova prema priloženoj oznaci.

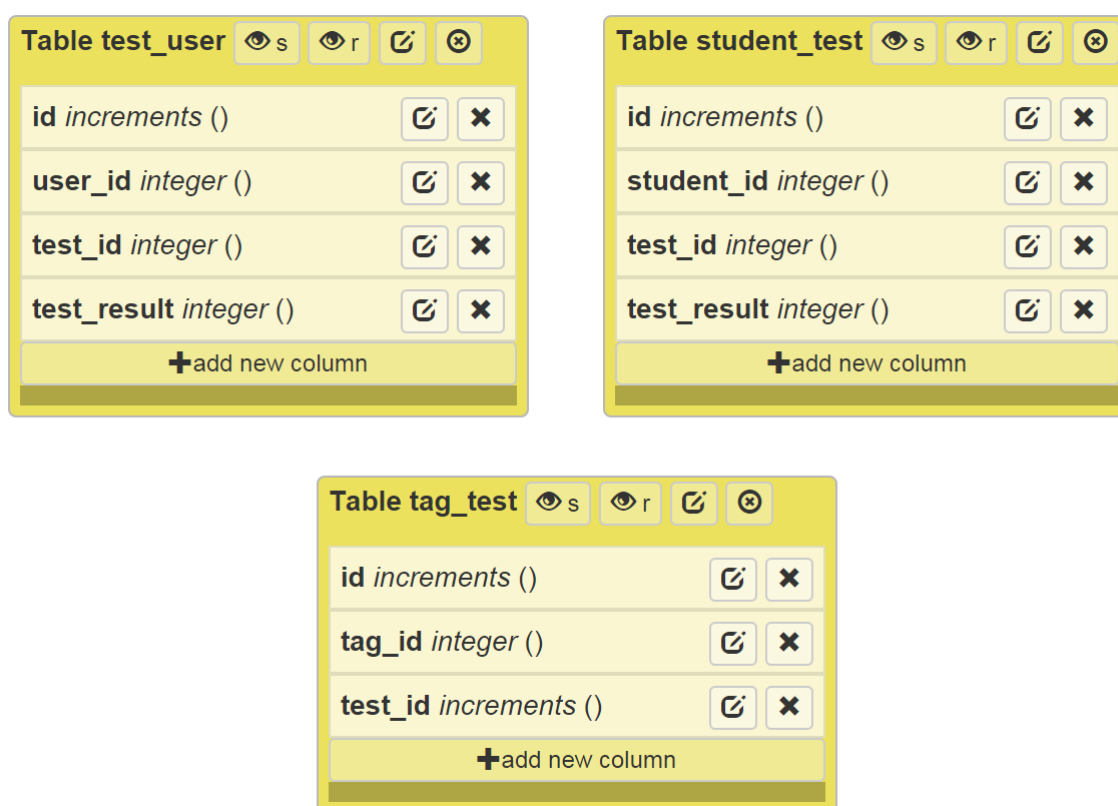
- **password_resets** tablica – tablica služi za moguću promjenu korisnikove šifre. Nakon što korisnik pošalje zahtjev za promjenu šifre, poslana je korisniku električna pošta na adresu koju je definirana u tablici users sa informacijom za promjenu njihove šifre.



Slika 4.4. Model baze podataka samogenerirajućeg test generatora.

Za definiranje relacije jedan prema više potrebno je imati jednu dodatnu tablicu nazvanu *pivot* tablica koja služi kao intermedijarna tablica koja povezuje dva modela definirane s relacijom jedan prema više i time njihove dvije tablice su povezane zajedno preko dva strana ključa koji su kreirani unutar *pivot* tablice, gdje ovi strani ključevi zapravo referenciraju id polje nad ovim povezivajućim tablicama tj. njihovih modela.

Na slici 4.5. se vidi kreirane *pivot* tablice i njihovi strani ključevi. Za imenovanje ovih tablica korišteno je sljedeće pravilo: imena *pivot* tablica su izvedena iz abecednog redoslijeda srodnih imena njihovih modela izražena u jednini tih imena i povezana sa *underscore* ključem (Tab. 4.1.).



Slika 4.5. *Pivot tablice za relaciju „one to many“*

Tablica 4.1. *Pravilo imenovana diskutiranih pivot tablica*

Povezivajuće tablice	Jednina imena	Abecedni redoslijed	Imena pivot tablica
tests, users	test, user	test, user	test_user
tests, students	rest, student	student, test	student_test
tests, tags	test, tag	tag, test	tag_test

Kreirane su tri *pivot* tablice. Unutar pivot tablica mogu se kreirati polja koje su tada dijeljena između povezivajućih tablica. Kreirano je polje nazvano `test_result` unutar tablica `test_user` i `student_test` koje služe za povezivanje konačnog rezultata na testu sa korisnikom i sa studentom koji je polagao pripadajući test.

4.2. Opis važnijeg koda internet aplikacije

Ispis i objašnjenje važnijih dijelova koda internet aplikacije omogućuje lakše razumjevanje funkcioniranja internet aplikacije i time razumjevanje programerskog okvira kao što je *laravel* u ovom slučaju. Dani kod je detaljno komentiran i objašnjen.

Prema slici 4.6. vidi se isječak koda od klase nazvane `PublishController`. Ova klasa sadrži dio logike koji najviše predstavlja danu internet aplikaciju. Vidi se kako je omogućeno da se pitanja od testa automatski slučajno generiraju unutar obrazca pozivom `shuffle` metode.

```
public function take_test($test) {
...
// Ako je zastavica za slučajno postavljanje pitanja postavljena
// upotrebljava se laravel shuffle metoda nad svim pitanjima
if($the_test->shuffle){
    $questions = $questions->shuffle();
}
...
// Ako je korisnik ulogiran, upotrijebi korisnikovo ime kao polagač testa
if(Auth::check()){
    $student_name = Auth::user()->name;
// Inače je krajnji korisnik za kojeg onda generiramo random ime za prijavu kao student
} else {
    $numberrand = rand(1, 1000);
    $student_name = "User$numberrand";
}
// Dalje u kodu je logika ako student je ulogiran, upotrijebi studentovo ime kao polagač
testa
...
}
```

Slika 4.6. Isječak koda klase `PublishController`

Dalje u kodu je prikazano kako laravel autentifikacija djeluje, prvo se provjera jer korisnik prijavljen, ako je dohvaćamo njegovo ime koje se koristi kao studentsko ime tokom prolaska kroz aplikaciju. Inače, krajnji korisnik je otvorio test za kojeg onda generiramo ime koje ćemo koristiti kao studentsko ime.

```
public function testing1($id){
    $points = [], $correct_points = [], $points_decrease = [];
    $points_count=0;
    $test =Test::find($id);
    $input = Input::all();
    ...
    // Za svako multiple_response pitanje unutar testa pronalazimo odgovore
    $multiple_response = Question::where('type', '=', 'multiple_response')
        ->where('test_id', "=", $id)->get();
    $input = array_except($input, ['_token']); //uklananje laravel tokena
    // Metodom map dohvaćamo sve točne odgovore za pitanje
    $answers = $multiple_response->map(function($question) {
        return $answers = DB::table('answers')
            ->where('question_id', '=', $question->id)
            ->where("correct","=", "1")->get();
    });
    // $answers je u formi laravel collection koji pretvaramo u array
    $answers = $answers->toArray();
    //array_flatten pretvara višedimenzionalni array u jednodimenzionalni
    $answers = array_flatten($answers);
    //Za svaki točan odgovor dodaj u array bodove točnog odgovora
    foreach($answers as $answer) {
        if(in_array($answer->answer,$input)) {
            $quest = Answer::find($answer->id)->question;
            $points[] = $quest;
            $answer_correct = Answer::find($answer->id);
            $correct_points[] =$answer_correct;
        }
    }
}
```

```

    }
...
//Sumiranje svih točnih odgovora, pa oduzimanje netočne odgovore
//samo ako je tip pitanja multiple response
    for($i=0;$i<count($points);$i++){
        $points_count += $points[$i]["points"];
    }
    for($i=0;$i<count($points_decrease);$i++){
        $points_count -= $points_decrease[$i]["points"];
    }
}

```

Slika 4.7. *Isječak metode unutar PublishController klase koja zbraja rezultate testa*

Slikom 4.7. prikazan je isječak za *multiple_response* tip pitanja koji je najbolji kandidat za razmatranje jer za razliku od drugih tipova pitanja koji imaju isti kod kao za ovaj tip pitanja za zbrajanje, kod ovog tipa pitanja se mora uzeti još u obzir i potreba za oduzimanjem bodova za neispravno izabrane odgovore.

Prvo se pronalaze sva pitanja ovog tipa za dani test, nakon čega se mapira tj. dohvaćaju se svi asocirani odgovori koji su točni za pitanja, nakon čega se *foreach* petljom uspoređuju dohvaćeni točni odgovori sa ulaznim podacima koje je student priložio u obrazcu testa.

Za svaki ispravno odgovorenog odgovora dodano je u polje (engl. array) korektan broj bodova i sumirano je sa ostalim ispravnim odgovorima. Odmah nakon toga oduzima se korektan broj bodova za svako neispravno odgovoren odgovor.

```

public function copy_public_test($test_id) {
    $test = Test::find($test_id); //Pronalaženje testa preko ida
    $new_test = new Test;
    $new_test->test_name = $test->test_name;
    $new_test->intro = $test->intro;
    $new_test->conclusion = $test->conclusion;
    $new_test->intro_image = $test->intro_image;
}

```

```

$new_test->conclusion_image = $test->conclusion_image;
$new_test->shuffle = $test->shuffle;
$new_test->is_public = $test->is_public;
$new_test->user_id = Auth::user()->id;
$new_test->save(); // Spremanje test postavki na korisnikov id
//dohvaćanje zadnjeg ida kreiran gornjim save metodom
$last_id=DB::getPdo()->lastInsertId();
//Pronalažene svakog pitanja za dani test
$questions = Test::find($test_id)->questions;
foreach($questions as $question){
    $new_question = new Question;
    $new_question->question = $question->question;
    $new_question->points = $question->points;
    $new_question->question_image = $question->question_image;
    $new_question->shuffle_question = $question->shuffle_question;
    $new_question->type = $question->type;
    $new_question->created_at = Carbon::now();
    $new_question->updated_at = Carbon::now()->addMinutes(2);
    $new_question->test_id = $last_id;
    //za svako pitanje spremamo pitanja u tablicu questions
    $new_question->save();
}
//Za svako pitanje dohvaćamo odgovore
foreach($questions as $key => $question){
    $questions_last = Test::find($last_id)->questions;
    $answers = Question::find($question->id)->answers;
    //tada dohvaćeni odgovori se spremaju u tablicu answers
    foreach($answers as $answer){
        $new_answer = new Answer;
        $new_answer->answer = $answer->answer;
        $new_answer->correct = $answer->correct;
        $new_answer->question_id = $questions_last[$key]["id"];
        $new_answer->save();
    }
}

```

```

    }

    }

    //Preusmjeri na rute tests sa porukom uspjeha kopiranja testa
    return Redirect::route('tests')
    ->with('message',"The public test \"{$test->test_name}\" has been copied to
your panel.");
}

```

Slika 4.8. Metoda za automatsko samogeneriranje testa unutar PublishController klase

Metoda *copy_public_test* prikazana na slici 4.8. služi za mogućnost automatskog samogeneriranja testova sa svim postavkama, pitanjima i odgovorima kreiranih od drugih korisnika koji su označili neki od svojih testova kao objavljeni (engl. publish) i postavili opciju javno (engl. public).

Vidimo na slici 4.8. ovu metodu, gdje se prvo pronalazi oznaka (engl. id) od testa koji se želi samogenerirati, kreira se novi test pomoću *new* operatora te postavljaju sve postavke danog testa u varijablu tog novog testa i pomoću metode *save* spremaju sve postavke u bazu podataka pod oznakom novog korisnika koji je inicijalizirao samogeneriranje testa.

Nakon ovog kopiranja testnih postavki, dohvaća se zadnja oznaka unesenog stupca u bazi podataka što je uvijek oznaka novo kreiranog testa. Dalje pronalaze se svi odgovori danog testa i ponovno spremaju sva pitanja u bazu podataka sa oznakom novog korisnika testa. Kada su spremljena sva pitanja potrebno je dohvatiti i sve pripadajuće odgovore za koje pomoću dvije *foreach* petlje spremamo u bazu podataka u odgovarajuću tablicu. Na kraju preusmjeravamo korisnika na rutu *tests* sa porukom uspjeha.

```

//Za svako pitanje pomoću laravel each metode za kolekcije
$answer = $questions->each(function($question) use($test, $student_name){
//Ispiši pitanje u strong formatu
    echo '<p><strong>'.($question["question"]).'</strong></p>';
    if($question["question_image"]){ //Ako slika ide uz pitanje, prikaži sliku
        echo Html::image("question_uploads/".$question["question_image"]);
    }
//Pronađi sve odgovore za dano pitanje

```

```

$answers = Question::find($question->id)->answers;
//Ako je zastavica za slučajno postavljanje odgovora postavljena
//upotrijebi shuffle metodu nad svim odgovorima
    if($question->shuffle_question){
        $answers->shuffle();
    }
//Unutar question->each() petlje, imamo ovu petlju $answers->each
//koja za svaki odgovor od danog pitanja kreira obrazac
    $answers->each(function($answer) use ($question,$answers,$test, $student_name){
        echo Form::open(array('route' => array('testing1',$test->id),'method' =>
'post','id'=>"test_form"));
        if($question->type === 'multiple_choice' || $question->type === 'true_false') { ?>
            <ul><li>
                <?php echo Form::label($answer["answer"],$answer["answer"]); ?>
                <?php echo Form::radio($answer->question_id, $answer["answer"]); ?>
                <?php echo Form::hidden('student_name', $student_name); ?>
            </li></ul>
            <?php } elseif($question->type === 'multiple_response') {
                echo Form::label($answer["answer"], $answer["answer"]);
                echo Form::hidden('student_name', $student_name);
                echo Form::checkbox($answer["id"],$answer["answer"]);
            } else {
                echo Form::text($answer->question_id);
            }
        });
    });
?>
<?php echo Form::submit('Send', array('class' => 'btn btn-info updated_answers')); ?>
{{ -- <a href="{{ URL::previous() }}" class="btn btn-danger">Go Back</a> -- }}
<?php echo Form::close(); ?>

```

Slika 4.9. Pogled odgovoran za generiranje testa za polaganje

Logika potrebna prilikom polaganja testa priložena je unutar PublishControler klase. Nakon što ova klasa konfigurira i dohvati sve postavke i podatke iz baze podataka, tada pošalje dane postavke i podatke skripti nazvanoj *testing1.blade.php*. Ova skripta je prateći pogled (engl. view) koji generira obrazac sa svim pitanjima i odgovorima. Za kreiranje ovog obrazca korišten je neslužbeni *blade* jezični predložak koji od *laravel* 5.0. verzije je uklonjen iz jezgrenog izvornog koda *laravela* i priložen *laravel* zajednici kao otvoreni paket koji tada ostali pridonositelji trebaju održavati. Blade jezični predložak uvodi pogodnosti u kreiranju obrazaca usporedbi s običnim html programskim jezikom.



Prema slici 4.9. vidi se isječak *blade* pogleda koji generira obrazac sa pitanjima i odgovorima. Za pitanja koja su primljena kao *laravel* kolekcija od kontroler klase upotrebljava se *each* funkcija nad pitanjima gdje se ispisuju pitanja, nakon čega je dinamički slučajno generiran redoslijed odgovora ako dano pitanje ima postavljenu opciju *shuffle_question*. Sa sljedećom *each* funkcijom generiraju se sva pitanja ovisno o tipu pitanja.

4.3. Prikaz i opis izrađene internet aplikacije


Za dizajn korisničkog sučelja izrađene internet aplikacije korišten je bootstrap koji je html, css i javascript programerski okvir otvorenog koda. Služi za razvoj responsivnih internet aplikacija, sa fokusom na mobilne internet aplikacije. Dolazi prekonfiguriran sa *laravel* programerskim okvirom. Korištenjem bootstrap okvira kreiran je osnovni dizajn internet aplikacije za koji ima puno potrebe za unapređenjem. Tablice i gumbi (engl. buttons) su stilizirani pomoću bootstrap klase koje se lagano uključe u skripti.

Početna stranica za krajnjeg korisnika se može vidjeti na slici 4.10. Sa *laravel* programerskim okvirom internet aplikacija je već prilagođena za različite rezolucije ekrana uređaja. Time je omogućen adaptivan dizajn za različite vrste ekrana kao što je mobilni uređaj gdje se aplikacija repositionira da stane na cijeli zaslon mobilnog uređaja.

Take public tests:

Created By User	Test name	Questions Shuffled	Password	
Slaven	 Math test	Yes	No	Take This Test
Slaven	 Kviz o studentskom životu	Yes	No	Take This Test
Marin	 Baza podataka	No	No	Take This Test
Ivor	 Test o Laravelu	Yes	No	Take This Test

Take private tests:

Created By User	Test name	Questions shuffled	Password	
Ivor	 Test o društvenim mrežama	Yes	Yes	Take This Test
Ivor	 Kviz o popularnim filmovima	Yes	Yes	Take This Test

Slika 4.10. Izgled početne stranice za krajnjeg korisnika

The image displays four distinct web forms arranged vertically, each within a light gray header bar. The first form, 'Student Login', has a title 'Student Login Form' and fields for 'Username' (placeholder: 'Type username') and 'Password' (placeholder: 'Type Password'), with a blue 'Login' button. The second form, 'Student Register', has a title 'Student Register Form' and similar fields for 'Username' and 'Password', with a blue 'Student Register' button. The third form, 'User Login', has a title 'User Login Form' and fields for 'E-Mail Address' (placeholder: 'Type email') and 'Password' (placeholder: 'Type password'), including a 'Remember Me' checkbox, a blue 'Login' button, and a 'Forgot Your Password?' link. The fourth form, 'User Register', has a title 'User Register Form' and fields for 'Name' (placeholder: 'Type username'), 'E-Mail Address' (placeholder: 'Type email'), 'Password' (placeholder: 'Type password'), and 'Confirm Password' (placeholder: 'Type password again'), with a blue 'Register' button.

Slika 4.11. *Prijavni i registracijski obrazci za studente i za korisnike*

Na slici 4.11. se vide prijavni i registracijski obrazci za studente i za korisnike.

Za studente prijava ili registracija je omogućena preko jedinstvenog imena koji je priložen tokom registracije. Dok korisnici se prijavljuju u sustav i registriraju preko priložene adrese električne pošte tokom registracije.

Studentov odjeljak nakon registracije ili prijave sadrži kontrolni ploču gdje se brzo može doći do novog testa koji se želi polagat preko obrazca za pretraživanje. Moguća je pretraga testova po imenu testa tako da unosom ključnih riječi imena testa su prikazani svi testovi sa tim ključnim riječima, implementirano pomoću *MySQL* dodatnom podrškom *full-text* koja služi za indeksiranje i pretraživanje.

MySQL kod potreban za *full-text* pretraživanje nad tablicom tests unutar baze podataka:

```
ALTER TABLE tests ADD FULLTEXT search(test_name);
```

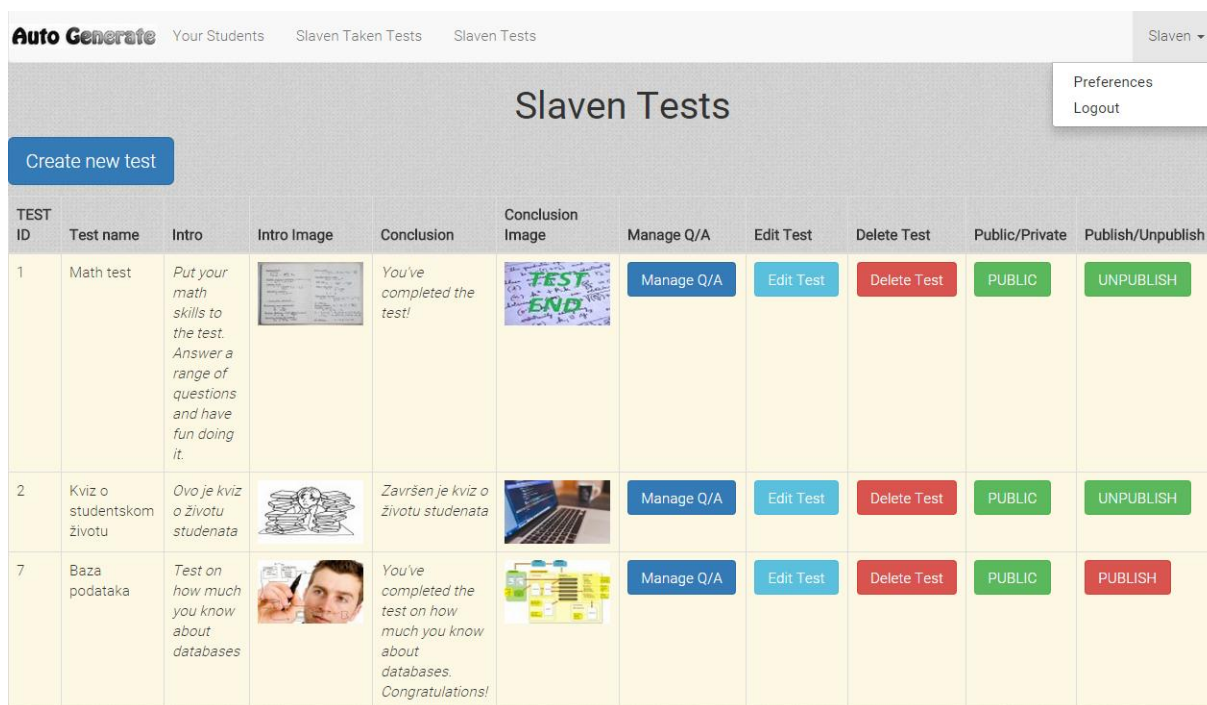
Sljedeći način za pretraživanje je pretraživanje testova pomoću tagova, gdje su dinamički izlistani traženi tagovi dok student unosi upit u obrazac (Sl. 4.12.).

Također na studentovom odjeljku se može vidjeti i povezanica (engl. link) koja pruža mogućnost prikaza svih testova koje je dotad student polagao sa svim postignutim rezultatima i prikaz vremena kada je test proveden i dugme za mogućnost prikaza testa za studentovu evidenciju prikazano na (Sl. 4.14.).

The screenshot shows a web interface titled 'Auto Generate' with two tabs: 'Students Taken Tests' and 'Control Panel'. The 'Control Panel' tab is active. It contains two search sections. The first section, 'Search for tests by test name:', has a text input field with the value 'test' and a 'Search' button. Below this, two test names are listed: 'Matematika test' and 'Test o društvenim mrežama'. The second section, 'Search for tests by there tag:', has a text input field with the placeholder 'Search query...' and a 'Search' button. Below this, a list of tags is displayed: 'baza', 'database', 'kviz', 'laravel', 'life', 'matematika', 'math', 'studenti', and 'test'.

Slika 4.12. Kontrolna ploča studenta sa mogućnošću pretraživanja svih testova



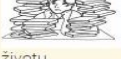


Korisnikov odjeljak nakon registracije ili prijave sadrži kontrolnu ploču sastavljenu od tablice svih testova koje je korisnik dosad kreirao (Sl. 4.13.). Korisnik ima mogućnost kreiranja neograničenog broja novih testova, ažurirat već kreirane testove ili brisat testove. Korisnik još ima mogućnost polaganje ostalih testova gdje je autentifikacija tokom polaganja preko korisnikovog imena.



TEST ID	Test name	Intro	Intro Image	Conclusion	Conclusion Image	Manage Q/A	Edit Test	Delete Test	Public/Private	Publish/Unpublish
1	Math test	Put your math skills to the test. Answer a range of questions and have fun doing it.		You've completed the test!		Manage Q/A	Edit Test	Delete Test	PUBLIC	UNPUBLISH
2	Kviz o studentskom životu	Ovo je kviz o životu studenata		Završen je kviz o životu studenata		Manage Q/A	Edit Test	Delete Test	PUBLIC	UNPUBLISH
7	Baza podataka	Test on how much you know about databases		You've completed the test on how much you know about databases. Congratulations!		Manage Q/A	Edit Test	Delete Test	PUBLIC	PUBLISH

Slika 4.13. Kontrolna ploča korisnika sa tablicom kreiranih testova

Korisnikov odjeljak također sadrži mogućnost prikaza svih testova koje je dotad korisnik polagao i dostupno je preko povezanice (Sl. 4.14.). Nakon otvaranja povezanice prikazani su postignuti rezultati, identifikacija testa i vrijeme kada je test polagan. Postoji još mogućnost prikaza danog testa za potrebe evidencije. Sljedeća povezanica je za prikaz svih studenata i/ili korisnika koji su polagali neki od testova koji su kreirani od autoriziranog korisnika (Sl. 4.15.). Na ovoj povezanici prikazano je u tablici koji je test polagan, vrijeme kada je test polagan, ime studenta koji je polagao dani test i postignuti rezultat na testu.











<div>Auto Generate</div> <div>Your Students Slaven Taken Tests Slaven Tests</div>					
Tests you have taken:					
<div> <div><</div> <div>1</div> <div>2</div> <div>3</div> <div>></div> </div>					
Test Id/Name	Student Name	Your Test Result Was	Test Taken At	Show Test	
 <div>TEST ID: Kviz o studentskom životu</div> <div>2</div>	Slaven	12	06:16 22.04.2015	Show	
 <div>TEST ID: Baza podataka</div> <div>3</div>	Slaven	3	06:16 22.04.2015	Show	
 <div>TEST ID: Kviz o studentskom životu</div> <div>2</div>	Slaven	10	06:16 22.04.2015	Show	
 <div>TEST ID: Test o društvenim mrežama</div> <div>4</div>	Slaven	9	06:16 22.04.2015	Show	
 <div>TEST ID: Test o Laravelu</div> <div>5</div>	Slaven	0	06:16 22.04.2015	Show	

Slika 4.14. Ispis svih polaganih testova i njihovih rezultata koje je korisnik polagao

Kreirana je opcija da korisnik ima mogućnost polaganja testova. Time korisnik nakon polaganja testa postaje student preko logike kreirane unutar internet aplikacije i prikazan je kao student nakon polaganja na kontrolnoj ploči evidencije studentata.

Autorizirano ime korisnika je korišteno kao studentsko ime unutar internet aplikacije koje se propagira unutar internet aplikacije pomoću sesije (engl. session) varijable i služi kao identifikacijska varijabla. Ovo ime je jedinstveno u bazi podataka gdje dvije tablice users i students ne mogu imati ista imena.

Kada novi student se registrira preko korektnog obrazca provjerava se da ne postoji priloženo ime unutar tablice *students* i unutar tablice *users*. Ako je priloženo ime već registrirano unutar tablice *students* ilil *users* kontroler klasa preusmjerava nazad na registracijski obrazac sa porukom zauzetosti priloženog imena.





Auto Generate Your Students Slaven Taken Tests Slaven Tests					
This are your students who took your tests					
Test Name	Test ID	Test Taken At	Students Name	Students Test Result	
 Math test	1	06:16 22.04.2015	Marin	10	
 Math test	1	06:16 22.04.2015	Davor	23	
 Math test	1	06:16 22.04.2015	Slavica	32	
 Math test	1	06:16 22.04.2015	Novi	23	
 Math test	1	06:16 22.04.2015	Neko_Novi	32	
 Math test	1	11:26 23.04.2015	User155	4	
 Math test	1	11:27 23.04.2015	efefae fw	0	
 Kviz o studentskom životu	2	06:16 22.04.2015	Slaven	10	
 Kviz o studentskom životu	2	06:16 22.04.2015	Loki	23	
 Kviz o studentskom životu	2	22:34 25.04.2015	Slaven	0	

Slika 4.15. Prikaz svih studenata koji su polagali testove od danog korisnika sa njihovim rezultatima

Korisnik nakon prijave ili registracije u sustav dobiva novu mogućnost gdje povratkom na početnu stranicu je prikazan novi gumb nad testom kreiranim od ostalih korisnika. Na slici 4.16. vidi se ova gumb nazvan *copy this test* koji nakon odabiranja za izabrani test se **automatski samogenerira** dani test sa svim postavkama, pitanjima i odgovorima na kontrolnu ploču autoriziranog korisnika gdje ovaj korisnik onda može vrši sve iste operacije ažuriranja testa kao sa ostalim njegovim kreiranim testovima. Ovo kopiranje testova korisnik može izvršiti jedino ako je test objavljeni i postavljen kao javni. Testovi sa šifrom i postavljeni kao privatni nemaju ovu mogućnost kopiranja.

Auto Generate
Your Students
Slaven Taken Tests
Slaven Tests
Slaven

Take public tests:

Created By User	Test name	Questions Shuffled	Password		
Slaven	 Math test	Yes	No	Take This Test	
Slaven	 Kviz o studentskom životu	Yes	No	Take This Test	
Marin	 Baza podataka	No	No	Take This Test	Copy this Test
Ivor	 Test o Laravelu	Yes	No	Take This Test	Copy this Test

Slika 4.16. Automatsko samogeneriranje novog testa

4.4. Kreiranje i ažuriranje testova

Za potrebe kreiranja i ažuriranja testova, pitanja i odgovora nužno je imat više vrsta obrazaca koji mogu postat veliki s puno polja koja se mogu ispunjavati, gdje korektno ispunjavanje cijelog obrazca postaje teško prilično brzo. Dizajn potreban za ove obrazce mora biti jednostavan bez nepotrebnih detalja i dovoljno intuitivan da se može brzo ispuniti. Uzimajući ovo u obzir kreirani su obrazci sa što egzaktnijim naslovima i oznakama za polja i sa opisnim *placeholder* poljima.

Create a New Test

Test name:

Type the test name

Intro message:

Type the intro message

Intro image:

Choose File No file chosen

Conclusion image:

Choose File No file chosen

Test conclusion message:

Type the conclusion message

Time of test in minutes: no time ▼

Shuffle test questions: ☐

Passcode: Enter passcode

Tags: Separate by commas

Is test public: ☐

Public test doesn't need a passcode, but you can still enter it, if later you decide to make the test private.

Create Test Go Back

Slika 4.17. Obrazac sa svim opcijama za generiranje novog testa

Na slici 4.17. vidi se obrazac koji služi za kreiranje novog testa. Za lakše korištenje internet aplikacijom, napravljeno je da za kreiranje testa je jedino obvezno polje potrebno da bude ispunjeno je ime testa koje mora biti najmanje tri karaktera, a najviše do osamdeset karaktera dugačka.

Ostala polja unutar obrazca korisnik može ostaviti prazna koje naknadno unosi po potrebi. Ako se i unesu neki podaci u ostalim polja, onda se treba držati validacijskih pravila koja se mogu vidjeti na slici 4.18. koja su kreirana unutar klase od modela nazvana *Test.php*.

```
public static $test_upload_rules = array(  
    'test_name'=> 'required|min:3|max:80',  
    'intro' => 'min:3', // Minimalno tri karaktera  
    'conclusion' => 'min:3',  
    'passcode' => 'min:6',
```

```
'counter_time' => 'integer', // Samo cijeli brojevi  
'intro_image' => 'image', // Mora biti slika (jpeg, png, bmp, gif ili svg tipovi)  
'conclusion_image' => 'image',  
'tags' => 'min:3|max:80'  
);
```

Slika 4.18. Validacijska pravila za obrazac za kreiranje novog testa

Za dodatnu pomoć s ispunjavanjem obrazaca kreirani su indikacijski prozori (engl. pop-up windows) pomoću javascript programerskog jezika. Postavljanjem ikone miša nad *input* poljem obrazca pojavljuje se indikacijski prozor sa pomoćnim tekstom (Sl. 4.19.).



Slika 4.19. Indikacijski prozor sa pomoćnim tekstom

Prema slici 4.20. se vidi obrazac za ažuriranje testa. *Laravel* omogućuje jednostavan mehanizam dohvaćanja podataka iz baze podataka za potrebe prikaza podataka unutar *input* polja obrazca pomoću „statičkih“ sučelja zvanih fasade (engl. facades). Fasada korištena se naziva Input i metoda dohvaćanja ovih podataka se zove *old*.

Edit a Test


Test name:

Math test


Test Intro message:

Put your math skills to the test.
Answer a range of questions and
have fun doing it.

Intro image:

 DELETE

Conclusion image:

 DELETE

Test conclusion message:

You've completed the test!

Time of test in minutes: no change ▾

Shuffle test questions: ☒

Update passcode:

Added tags: DELETE ALL TAGS

1. math
2. mathematics

To remove the passcode entirely, consider making the test PUBLIC on your [control panel](#)

Update Go Back

Slika 4.20. *Mogućnost editiranja svih postavki danog testa*

Go Back To Tests		Add new question				
ID	Question Image	Question	Points	Shuffle_question	Type	Test_Id
15		Koji je rezultat	4	1	multiple_choice	1
<div>Add new Answer</div> <div>Edit Q/A</div> <div>Delete Question</div>						
ID	Answer	Correct	Question ID	Delete Answer		
40	3	1	15	Delete Answer		
41	4	0	15	Delete Answer		
42	5	0	15	Delete Answer		
43	6	0	15	Delete Answer		
ID	Question Image	Question	Points	Shuffle_question	Type	Test_Id
17	X	googol is bigger than a billion.	3	1	true_false	1
<div>Add new Answer</div> <div>Edit Q/A</div> <div>Delete Question</div>						
ID	Answer	Correct	Question ID	Delete Answer		
46	True	1	17	Delete Answer		
47	False	0	17	Delete Answer		
ID	Question Image	Question	Points	Shuffle_question	Type	Test_Id
18	X	Check all the answers that represent the same level of precision.	5	1	multiple_response	1
<div>Add new Answer</div> <div>Edit Q/A</div> <div>Delete Question</div>						
ID	Answer	Correct	Question ID	Delete Answer		
48	A. 1.045	0	18	Delete Answer		
49	B. 0.59	1	18	Delete Answer		
50	C. 98.42	1	18	Delete Answer		
51	D. 100.6	0	18	Delete Answer		
ID	Question Image	Question	Points	Shuffle_question	Type	Test_Id
19	X	$9 \times 8 = ?$	5	1	fill_in	1
<div>Add new Answer</div> <div>Edit Q/A</div> <div>Delete Question</div>						
ID	Answer	Correct	Question ID	Delete Answer		
52	72	1	19	Delete Answer		

Slika 4.21. Sučelje za kreiranje, editiranje i brisanje pitanja i odgovora

Sučelje za kreiranje novih pitanja i pripadajućih odgovora prikazano je na (Sl. 4.21.). Korisnik može kreirati, ažurirati, korigirati i brisati pitanja i odgovore. Implementirane su određene provjere valjanosti ovisno o izabranom tipu pitanja. Odabirom tipa pitanja između četiri implementiranih logika nalaže nad odgovorima da se korektno kreiraju pitanja i odgovori.

Logika ovisno o tipu pitanja:

1. **Multiple choice** – ovaj tip pitanja može imati više odgovora definiranih, ali samo jedan može biti točan. Može se izabrati samo jedan odgovor na ovo pitanje.
2. **Multiple response** – može imati više odgovora definiranih i više točnih odgovora postavljenih. Izabire se jedan ili više kao točnih odgovora.

3. **Fill-in** – može imat samo jedan točan odgovor koji se unosi u tekstualno polje. Neovisno o velikim ili malim slovima unesenim.
4. **True-false** – Samo dva odgovora su definirana. Jedno od dva mora biti točan, drugo je netočno.

The screenshot displays two sections of a web interface for editing a question.

Edit The Question

- Question:
- Points:
- Question Image:
- shuffle_question ☒
-

Edit The Answer For Question

- Answer: Correct: ☒
- Answer: Correct: ☐
- Answer: Correct: ☐
- Answer: Correct: ☐
-


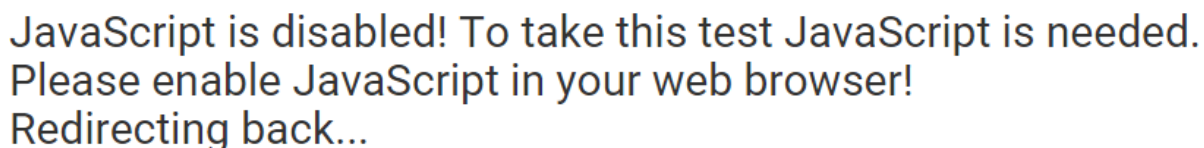
Slika 4.22. Sučelje za editiranje pitanja i povezanih odgovora

Uređivanje pitanja i odgovara je prikazano (Sl. 4.22.) za *multiple_choice* tip pitanja.

Ovisno o tipu pitanja rendira se ispravan pogled koji dolazi sa drugačijom logikom za ažuiranje pitanja i odgovora.

4.5. Provođenje testova

Za potrebe provođenja testova potrebno je imati određene mjere opreza koje pomažu da polaganje testova bude korektno bez mogućnosti prevare sustava. Najvažnija zaštita od moguće prevare tokom polaganja je otvaranje testa više puta. Ovaj problem je riješen pomoću javascript programerskog jezika. Za svako polaganje testa potrebno je da student ima uključen javascript u svom internet pregledniku, inače internet aplikacija obaviještava studenta da je javascript isključen i da je potrebno omogućiti javascript da se polaže dani test (Sl. 4.23.). Sa uključenim javascriptom moguće je polaganje testa. U slučaju da student pokuša odustat od polaganja testa tada javascript skripta prepoznaje kada korisnik pokuša izaći iz internet preglednika ili pokuša kliknut na dugme za povratak nazad ili izaći iz *taba*. Ako jedna od ovih mogućnosti je izvedena, javascript skripta prepoznaje ovo i pomoću *ajax* poziva se sprema dani pokušaj polaganja testa od studenta u bazu podataka za potrebe evidencije korisnika. Ovim spremanjem u bazu podataka poslije se prikazuje na korisnikovom odjeljku na *tabu* „Your students“ gdje je zabilježeno koji test je bio polagan, kada je polagan i ime studenta ili korisnika koji je pokušao polagati test.

A light gray rectangular box with a thin border. Inside, the text "Test Started" is written in a dark blue, sans-serif font.A screenshot of a browser's JavaScript error message. The text is in a dark blue, sans-serif font and reads: "JavaScript is disabled! To take this test JavaScript is needed. Please enable JavaScript in your web browser! Redirecting back...".

Slika 4.23. Prikazana poruka ako je isključen javascript u internet pregledniku

Kreiranje obrazca za test se obavlja automatski sa svim pitanjima prikazanim na toj jednoj internet stranici. Uz svako pitanje može postojati pripadajuća slika. Primjer testa se može vidjeti na (Sl. 4.24.). Slika prikazuje primjer četiri tipa od mogućih tipova pitanja.

Uz svaki test može postojati vrijeme u kojem je potrebno odraditi test ili inače ovog vremenskog ograničenja nema. Korisnik koji je kreirao test odlučuje tokom kreiranja testa da postavi ovaj vremenski limit na test. Kod testova sa ovim vremenskim ograničenjem nakon isteka vremena obrazac se sam pošalje pomoću javascripta.

googol is bigger than a billion.

- False ☐
- True ☒

Check all the answers that represent the same level of precision.

C. 98.42 ☒

B. 0.59 ☒

D. 100.6 ☐

A. 1.045 ☐

Koji je rezultat

$10 - 9 + 2 = ?$

3 ☒ 4 ☐ 5 ☐ 6 ☐

9 x 8 = ?

72

Send

Slika 4.24. Samogenerirani obrazac za polaganje testa

Kada je obrazac sa riješenim odgovorima poslan logika unutar kontroler klase i prateće akcije tj. metode izračunava korektan broj ispravnih bodova postignutih, prema sve podatke u bazu podataka i poziva pogled za prikaz konačnog rezultata postignutog (Sl. 4.26.).

Trenutan je ispis tablice sa prikazom svih pitanja i pripadajućih točnih odgovora i ispravno odgovorenim odgovorima. Na slici 4.25. vidimo ovu tablicu i sa bodovima koji su mogući postići na točno odgovorenim pitanjima.

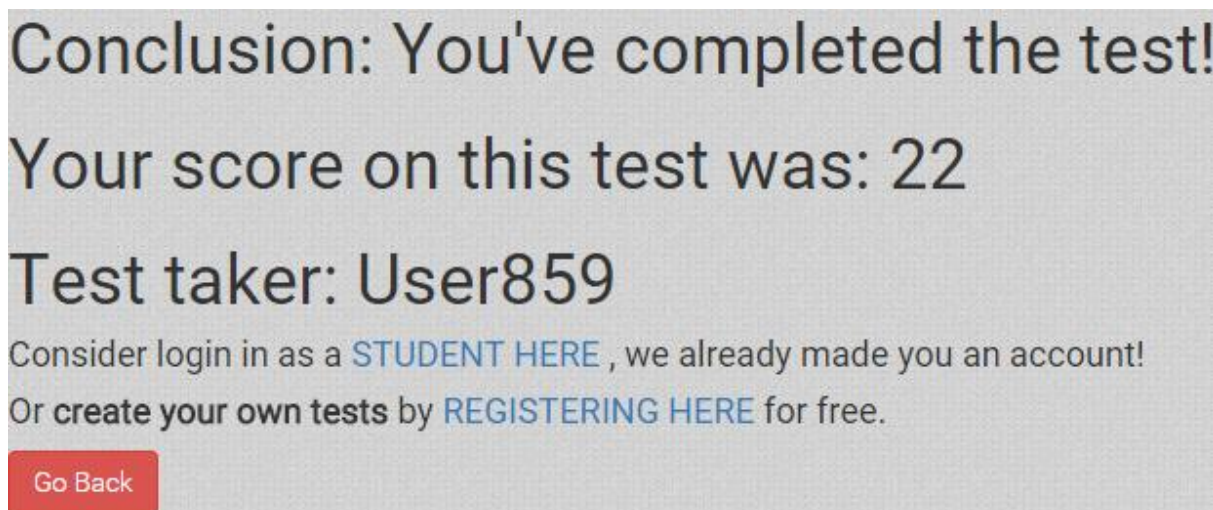
Question Image	ID	Question	Points	Shuffle_question	Type	Test_id
	15	Koji je rezultat	4	1	multiple_choice	1
ID	Answer	Correct	Question ID			
You answered Correctly	3					
40	3	1	15			
41	4	0	15			
42	5	0	15			
43	6	0	15			
	17	googol is bigger than a billion.	3	1	true_false	1
ID	Answer	Correct	Question ID			
You answered Correctly	True					
46	True	1	17			
47	False	0	17			
	18	Check all the answers that represent the same level of precision.	5	1	multiple_response	1
ID	Answer	Correct	Question ID			
48	A. 1.045	0	18			
You answered Correctly	B. 0.59					
49	B. 0.59	1	18			
You answered Correctly	C. 98.42					
50	C. 98.42	1	18			
51	D. 100.6	0	18			
	19	9 x 8 = ?	5	1	fill_in	1
ID	Answer	Correct	Question ID			
You answered Correctly	72					
52	72	1	19			

Slika 4.25. Tablica sa prikazom odgovora i ispravno odgovorenim pitanjima

Za polaganje testova postoje tri moguće uloge:

1. Krajnji korisnik koji nije registriran i prvi put posjećuje stranicu.
2. Studenti registrirani ili prijavljeni sa studentskim imenom.
3. Korisnik registriran i prijavljen kao učitelj sa korisničkim imenom.

Krajnji korisnik koji nije registriran ima mogućnost polaganja testova. Neregistrirani krajnji korisnik nakon polaganja jednog od javnih testova, za njega je automatski generira studentsko ime, šifra i *account* kao student i spremljen je položeni test sa konačnim rezultatom u bazu podataka (Sl. 4.26 i 4.27.).



Slika 4.26. Ispis konačnog rezultat ostvarenog na testu

Novo kreirani student nakon prijave ima mogućnost promjene šifre koja je generirana slučajno ili može nastaviti koristiti automatsko generiranu šifru. Automatsko generirane šifre su relativno sigurne.

A screenshot of a "Student Login" form. The form is titled "Student Login Form". It has two input fields: "Username" with the value "User859" and "Password" with the value "vRHxKGp16c". Below the password field, there is a message: "We already made you an account" followed by "Enjoy". At the bottom of the form is a blue button labeled "Login".

Slika 4.27. Obrazac za prijavu novo stvorenog studenta

Za studente prijavljene u sustav nakon polaganja ispita autorizira se sa njihovim jedinstvenim imenom. Slična kao kod studenata je autorizacija i korisnika koji su prijavljeni u sustav.

6. ZAKLJUČAK

Cilj ovog diplomskog rada je bio napraviti internet aplikaciju koja bi služila kao sučelje za lagano kreiranje, ažuriranje i provođenje testova, kvizova, anketa i drugih preko interneta. Ova aplikacija pruža jednostavno korisničko sučelje sa intuitivnom obrazcima za kreiranje i ažuriranje testova i parsiranim tablicama za prikaz relevantnih informacija o položenim testovima i evidenciji studenata. Implementirana je studentska kontrolna ploča nakon obvezne autorizacije imenom i lozinkom. Nakon autorizacije omogućeno je lagano pretraživanje prema ključnim riječima ili prema *tagovima*. Aplikacija sadrži određene mehanizme za zaštitu od raznih mogućih načina prijevara tijekom polaganja testova, gdje javascript jezik pruža sigurnosne značajke i koji mora biti postavljen u internet pregledniku za mogućnost polaganja nekog od testova. Implementirana su četiri tipa pitanja koji onda mogu biti slučajno poredani tijekom generiranja obrazca za polaganje testa. Kreirana je mogućnost samogeneriranja već kreiranih testova sa jednim klikom od drugih korisnika na korisničku ploču autoriziranog korisnika. Nužan uvjet internet aplikacije je bila potreba za određenim sigurnosnim mehanizmima za koji je izabran *laravel* robustan programerski okvir koji ima ugrađene sigurnosne mehanizme u rasponu od vrednovanja unesenih korisničkih podataka do identifikacije korisnika i studenata. Korištena je MVC paradigma integrirana unutar *laravel* okvira. Internet aplikacija je prenijeta (engl. deployed) na *cloud* infrastrukturu nazvanu *digitalocean* pomoću suvremenih tehnologija.

Daljnja nadogradnja sučelja je potreba implementiranja složenijih tipova pitanja kao što su pitanja podudaranja odgovora, pitanja poredka odgovora, *drag and drop* pitanja i drugih. Isto tako, bila bi prikladna i razrada na kontrolnim pločama korisnika i studenata gdje se treba implementirati ažuriranje postavki računa, kao što je mogućnost brisanja studentskog računa i slično.

LITERATURA

- [1] PHP [online] URL: <http://en.wikipedia.org/wiki/PHP> Dan pristupa: 9. svibanj 2015.
- [2] J., Lockhart, Modern PHP. O'Reilly Media, Sebastopol CA, 2015.
- [3] Generators [online] URL <http://php.net/manual/en/language.generators.overview.php> Dan pristupa: 19. svibanj. 2015.
- [4] Model-View-Controller [online] URL <http://en.wikipedia.org/wiki/Model-view-controller> Dan pristupa: 10. svibanj 2015.
- [5] Trygve Reenskaug [online] URL http://en.wikipedia.org/wiki/Trygve_Reenskaug Dan pristupa: 10 svibanje 2015.
- [6] Raphaël S. Getting Started with Laravel 4. Packt Publishing Limited, Birmingham 2014.
- [7] Laravel homestead [online] URL <http://laravel.com/docs/4.2/homestead> Dan pristupa: 9. svibanj 2015.

SAŽETAK

Ovim radom opisane su mogućnosti predstavljenje upotrebom programerskih okvira u projektima i prednosti dostupne korištenjem internet aplikacija za kreiranje testova, kvizova i anketa. Teorijske osnove korištenih tehnologija i opis glavnih usluga iskazanih internet aplikacijom u radu su također prikazane. U radu je razrađen dizajn baze podataka potreban za kreiranje testova sa pripadajućim pitanjima i odgovorima. Detaljno su opisani glavni isječci koda koji su važni za razumjevanje rada aplikacije. Prikazana i objašnjena je implementirana funkcionalnost tijekom korištenja internet aplikacije, kako se kreiraju testovi i kako se provode testovi. Objašnjen je cijeli proces polaganja testa za autorizirane studente i velika pažnja je dana za sprječavanje mogućeg varanja tokom polagana mogućih testova. Implementirano je svojstvo samo generiranja već kreiranih testova na korisničku kontrolnu ploču autoriziranog korisnika.

Ključne riječi: PHP programerski okviri, Laravel 5, testovi, MVC, PSR, Homestead, polaganje testova, Composer, Javascript, PHP 5.5

ABSTRACT

This paper describes the opportunities represented by the use of development frameworks in projects and the benefits available by using Internet applications for the creation of tests, quizzes and surveys. Theoretical basics of technologies used and description of the main services shown in the internet application in the work are also presented. Paper elaborates the database design required for creating tests with accompanying questions and answers. The main code snippets are detailed that are important for understanding the operation of the application. Displayed and explained are the implemented functionalities while using the internet application, how to create tests and how to take tests. Whole process of taking a test for authorized students is explained and great care is given to preventing possible cheating during possible test taking. Implemented a feature for auto-generating already created tests on to the users control panel for authorized users only.

Keywords: PHP frameworks, Laravel 5, tests, MVC, PSR, Homestead, test taking, Composer, Javascript, PHP 5.5

ŽIVOTOPIS

Slaven Sakačić rođen 1986. u Osijeku, Hrvatska. Osnovnu školu završava 2001. godine u Osijeku. Iste godine upisuje se u Elektrotehničku i prometnu srednju školu u Osijeku, smjer elektromehaničar. Srednju školu u trajanju od tri godine završava 2004., nakon čega upisuje razlikovnu godinu u istoj školi za prekvalifikaciju na zvanje tehničara za elektroniku, koju uspješno polaže. 2007. godine završava srednju školu i iste godine upisuje Građevinski fakultet u Osijeku. Godine 2008. odlučuje upisat stručni studij Elektrotehničkog fakulteta u Osijeku, smjer informatika. 2011. godine završava stručni studij. Nakon upisuje razlikovnu godinu, smjer računarstvo te ga uspješno završava 2012. godine. Te upisuje diplomski studij na istom fakultetu, smjer procesno računarstvo.

Slaven Sakačić