SVEUČILIŠTE U SPLITU

SVEUČILIŠNI ODJEL ZA STRUČNE STUDIJE

Preddiplomski stručni studij Računarstvo

ANAMARIJA PAPIĆ

Z A V R Š N I R A D

IZRADA PLATFORME ZA TIMSKI RAD

Split, lipanj 2023.

SVEUČILIŠTE U SPLITU
SVEUČILIŠNI ODJEL ZA STRUČNE STUDIJE

Preddiplomski stručni studij Računarstvo

Predmet: Odabrani alati i naredbe u Linuxu

Z A V R Š N I R A D

Kandidat: Anamarija Papić

Naslov rada: Izrada platforme za timski rad

Mentor: Nikola Grgić, viši predavač

Split, lipanj 2023.

Sadržaj

Sažetak **1**

1 Uvod **2**

2 Korištene tehnologije **3**

2.1 Laravel . . . 3

2.2 Docker spremnici kao razvojno okruženje . . . 4

2.2.1 Nginx . . . 4

2.2.2 PHP . . . 4

2.2.3 MySQL . . . 5

2.2.4 phpMyAdmin . . . 5

2.3 Upravitelji paketima i ovisnostima . . . 5

2.3.1 npm . . . 5

2.3.2 Composer . . . 6

2.4 Razvojna okruženja za testiranje . . . 6

3 Opis implementacije praktičnog rada **7**

3.1 Instalacija i konfiguracija okruženja . . . 7

3.1.1 Početni komplet Laravel Jetstream . . . 7

3.1.2 `.env` datoteka . . . 8

3.2 Struktura Laravel aplikacije . . . 9

3.2.1 Struktura početnog direktorija s izvornim kôdom aplikacije . . . 9

3.2.2 Struktura `app` direktorija . . . 10

3.3 Struktura relacijske baze podataka . . . 11

3.4 Interakcija s bazom podataka . . . 12

3.4.1 Migracije . . . 12

3.4.2 Populacija podatcima (engl. *seeding*) . . . 15

3.4.3 Eloquent ORM . . . 15

3.4.3.1 Modeli i tvornice modela (engl. *model factories*) . . . 15

3.4.3.2 Relacije . . . 15

3.4.3.3 Query Builder i kolekcije . . . 15

3.5 Usmjerivanje zahtjeva (engl. *routing*) . . . 15

3.6 Middleware . . . 15
3.7 Autentikacija . . . 15
3.8 Autorizacija . . . 15
3.8.1 Politike (engl. *policies*) . . . 15
3.8.2 Vrata (engl. *gates*) . . . 15
3.9 Livewire komponente . . . 15
3.9.1 Akcije . . . 15
3.9.2 Događaji . . . 15
3.10 Pohrana datoteka . . . 15
3.10.1 MinIO - AWS S3 kompatibilan servis za pohranu . . . 15
3.11 Spatie Laravel Media Library paket . . . 15
3.12 Lokalizacija . . . 15
3.13 Testiranje i kvaliteta kôda . . . 15
3.13.1 Korišteni alati . . . 15
3.13.1.1 Laravel Debug Bar . . . 15
3.13.1.2 Laravel Telescope . . . 16
3.13.1.3 Laravel Pint . . . 16
3.13.2 Pretpregled elektroničke pošte - MailHog . . . 16
3.13.3 Pisanje i pokretanje testova . . . 16
3.14 Otvoreni kôd i doprinos zajednice na projektima Laravel ekosustava . . . 16

4 Prezentacija korisničkog sučelja pri korištenju aplikacije 17

5 Zaključak 18

Literatura 19

Dodatci 21

Sažetak

Cilj ovog završnog rada je razviti web aplikaciju koja će služiti kao platforma za timski rad na različitim projektima. Koristeći razvojni okvir Laravel izrađena je aplikacija *Teamstructor*. U navedenoj aplikaciji korisnici mogu formirati timove u kojima članovi tima imaju pristup projektima na kojima tim radi. Unutar svakog projekta članovi tima imaju pristup projektnoj diskusiji - mjestu pogodnom za dijeljenje svih novosti i ažuriranja vezanih uz taj projekt te pristup dijeljenim projektnim resursima - datotekama prenesenim od strane članova tima. Također je implementirano i jednostavno administrativno sučelje. U ovom pisanom radu bit će predstavljene tehnologije koje su korištene i način na koji je aplikacija implementirana.

Ključne riječi: *razvojni okvir Laravel, Livewire, Docker razvojno okruženje, pohrana datoteka, lokalizacija*

Summary

Developing a Teamwork Platform

The goal of this final thesis is to develop a web application that will serve as a platform for teamwork on various projects. The *Teamstructor* application was developed using the Laravel framework. In the mentioned application, users can form teams in which team members have access to the projects the team is working on. Within each project, team members have access to the project discussion - a place suitable for sharing all news and updates related to that project and access to shared project resources - files uploaded by team members. A simple administrative interface is also implemented. This paper will present the technologies that were used and the way in which the application was implemented.

Keywords: *Laravel framework, Livewire, Docker development environment, file storage, localization*

1. Uvod

U današnjem dinamičnom poslovnom svijetu, gotovo je nazamisliv posao koji barem u nekoj mjeri ne uključuje rad unutar tima. Također sve veći dio zaposlenika radi na izdvojenom radnom mjestu tj. *remote* ili hibridno, a sve manji radi isključivo iz ureda tj. *on-site*, pa je potrebno svladati i tu prepreku te nekadašnju komunikaciju isključivo licem u lice sada dijelom zamijeniti novim alatima. Kako bi tim uspješno djelovao i ostvario postavljeni cilj nužno je da su svi članovi tima međusobno usklađeni i mogu neometano surađivati - zato je potrebno osigurati da su članovi tima umreženi te mogu brzo primati novosti vezane uz poslovne zadatake dodijeljene timu te lako međusobno dijeliti resurse.

Kao praktični dio ovog završnog rada izrađena je web aplikacija *Teamstructor*. Registrirani korisnici mogu kreirati timove i pozivati članove u svoj tim. Članovi tima mogu kreirati različite projekte unutar tima. Unutar svakog projekta članovi tima mogu raspravljati i objavljivati novosti relevantne za projekt u formi objava i jednorazinskih komentara. Također, unutar svakog tima članovi tima mogu pregledavati dijeljene resurse te prenositi nove datoteke s osobnog računala među resurse. Prijavljeni korisnici mogu pristupati timovima i timskim projektima (te raspravama i resursima) samo ukoliko su članovi tog tima. Implementirano je i jednostavno administrativno sučelje dostupno korisniku s ulogom administratora u kojemu može lako pregledavati postojeće korisnike, timove i projekte.

GitHub repozitorij u kojem je sadržan cijeli izvorni kôd aplikacije *Teamstructor* dosupan je na poveznici `https://github.com/anamarijapapic/teamstructor`, a repozitorij s LaTeX datotekama korištenima za pisanje ovog rada na poveznici `https://github.com/anamarijapapic/teamstructor-docs`.

U poglavljima koja slijede prvo će se kratko predstaviti tehnologije korištene pri izradi aplikacije, a zatim će se detaljno i pregledno opisati način na koji je sama aplikacija implementirana.

2. Korištene tehnologije

2.1. Laravel

Laravel je PHP (engl. *PHP: Hypertext Preprocessor*) razvojni okvir (engl. *framework*) namijenjen razvoju web aplikacija zasnivanih na MVC (*"Model-View-Controller"*) arhitekturi. Njegove značajke su da je slobodan, besplatan te otvorenog kôda, stoga je cijeli njegov izvorni kôd dostupan na platformi GitHub [1] gdje svatko može dati svoj doprinos pa se naziva i razvojnim okvirom zajednice (engl. *community framework*). Tvorac Laravela je Taylor Otwell, koji je 2011. godine razvio početnu verziju Laravela pokušavajući poboljšati tada popularan razvojni okvir CodeIgniter.

Trenutno je, uz Symfony, Laravel najpopularniji PHP razvojni okvir zahvaljujući svojoj jednostavnoj i izražajnoj sintaksi, detaljnoj dokumentaciji i obilnoj količini (video)vodiča, jasnoj strukturi, iznimnoj skalabilnosti te bogatom Laravel ekostustavu s dostupnom ogromnom bibliotekom pomno održavanih paketa.

Pruža elegantna gotova *"out-of-the-box"* rješenja za česte značajke potrebne web aplikacijama: jednostavno i brzo usmjerivanje zahtjeva, moćno ubrizgavanje ovisnosti, pohranjivanje sesija i predmemorije, autentikaciju, autorizaciju, interakciju s podatcima u bazi, modelima, migracijama i relacijama, validaciju, slanje e-pošte i notifikacija, pohranu datoteka, red čekanja i pozadinsku obradu poslova, zakazivanje zadataka, događaje itd [2].

Dolazi s **Artisan** sučeljem naredbenog retka (engl. *command line interface*) koje pruža mnoge korisne naredbe koje pomažu pri razvoju aplikacije. Artisan naredbe su u formi `php artisan <command>`.
Instalacijom Laravela također se dobiva i **Tinker** (REPL - *"Read—Eval—Print—Loop"*) interaktivna ljuska koja omogućava testne interakcije s modelima, poslovima, događajima itd. Za početak rada u Tinker razvojnom okruženju pokreće se Artisan naredba `php artisan tinker` [3].

Laravel i njegovi ostali paketi prve strane (engl. *first-party*) prate semantičko verzioniranje, pri čemu se *major* verzija razvojnog okvira Laravel sada izdaje svake godine, dok su *minor* i *patch* izdanja češća. Sva izdanja razvojnog okvira Laravel primaju ispravke pogrešaka do 18 mjeseci od izdavanja, a sigurnosne ispravke do 2 godine od izdavanja. Trenutna verzija razvojnog okvira Laravel je 10.x te zahtijeva minimalnu PHP verziju 8.1 [4].

2.2. Docker spremnici kao razvojno okruženje

Kako bi se web aplikacija jednako uspješno izvodila u različitim okruženjima - na različitim operativnim sustavima i arhitekturama računala, korištena je tehnologija Docker **spremnika** (engl. *containers*) te je aplikacija kontejnerizirana.

Svaki stroj koji ima instaliran Docker Engine može pokretati i stvarati Docker spremnike konstruirane iz Docker **slika** (engl. *image*). Za izgraditi sliku spremnika Docker koristi `Dockerfile` datoteku u kojoj se nalazi skripta s uputama za kreiranje iste. Za inicijalizaciju i izvođenje aplikacija koje se sastoje od više spremnika potreban je alat **Docker Compose** te su u tom slučaju aplikacijski servisi konfigurirani u `docker-compose.yml` datoteci. Instalacijom Docker Desktop aplikacije na računalo dobiva se i Docker Engine i Docker Compose V2 te dodatni alati. Upisivanjem naredbe `docker compose up` u terminal iz direktorija projekta pokreće se aplikacija i svi njeni servisi, a naredbom `docker compose down` zaustavljaju se svi pokrenuti servisi te se brišu spremnici [5]. Unutar `docker-compose.yml` datoteke kreirani su imenovani **volumeni** (engl. *volumes*) koji služe za očuvanje podataka koje generiraju i koriste Docker spremnici [6].

U nastavku će kratko biti opisane tehnologije koje su nužne za rad aplikacije, a definirane su kao servisi u `docker-compose.yml` datoteci.

2.2.1. Nginx

Kao lokalni web poslužitelj (engl. *server*) korišten je Nginx [engine x]. Kao početna točka za konfiguraciju web poslužitelja koristi se datoteka `default.conf` [7], u čijem se sadržaju unutar `server` direktive konfigurira virtualni poslužitelj [8].

2.2.2. PHP

PHP (rekurzivni akronim za *PHP: Hypertext Preprocessor*, a prije je kratica označavala *Personal Home Page*) je popularni besplatni skriptni jezik otvorenog kôda (izvorni kôd je dostupan na platformi GitHub [9]) koji se izvršava na poslužitelju (engl. *server-side scripting*). 1994. godine razvio ga je Rasmus Lerdorf, a sintaksa mu je bazirana na C, Java i Perl programskim jezicima. PHP je jezik opće namjene, ali je posebno prikladan za razvijanje web aplikacija [10].

PHP se koristi u najpopularnijem sustavu za upravljanje sadržajem (engl. *CMS - Con-*

tent Management System) – WordPressu, a postoji i nekoliko PHP razvojnih okvira: Laravel, Symfony, CodeIgniter, Zend, Yii 2, CakePHP, Fuel PHP, FATFree, Aura i dr. [10]

Trenutna verzija PHP jezika je 8.2, a trenutno je aktivno podržana i verzija 8.1. Svaka grana izdanja (engl. *release branch*) PHP-a u potpunosti je podržana dvije godine od svog prvog stabilnog izdanja i tijekom tog razdoblja za nju se objavljuju ispravci pogrešaka i sigurnosnih problema. Nakon tog razdoblja aktivne podrške, grana izdanja dobiva još jednu godinu podrške samo za kritične sigurnosne ispravke i to po potrebi. Nakon što isteknu tri godine podrške, grana dolazi do kraja života (engl. *end of life*) i više nije podržana [11].

2.2.3. MySQL

MySQL je besplatni program otvorenog kôda za upravljanje relacijskim bazama podataka (engl. *RDBMS - Relational Database Management System*). Pokrata "SQL" u imenu stoji za *"Structured Query Language"* - najčešći standardizirani jezik korišten za pristup bazama podataka. Najčešća alternativa MySQL-u su također besplatni programi otvorenog kôda MariaDB i PostgreSQL. Kod MySQL-a osnovni stroj baze podataka i ujedno i najčešće korišteni je InnoDB koji koristi transakcijski mehanizam [10].

2.2.4. phpMyAdmin

phpMyAdmin je besplatni softverski alat otvorenog kôda napisan u PHP-u koji podržava širok spektar operacija unutar MySQL i MariaDB relacijskih baza podataka. Radi se o intuitivnom grafičkom korisničkom sučelju (engl. *GUI - Graphical User Interface*) pa akcije mogu biti izvršene unutar korisničkog sučelja izravno u web pregledniku [10].

Može mu se pristupiti unosom adrese `http://localhost:8080` u web preglednik, pri čemu će se pojaviti phpMyAdmin "Welcome to phpMyAdmin" stranica koja traži unos korisničkog imena i loznike.

2.3. Upravitelji paketima i ovisnostima

2.3.1. npm

npm (Node package manager) je upravitelj paketa za Node.js nastao 2009. godine kao projekt otvorenog kôda koji bi JavaScript programerima pomogao da jednostavno dijele zapakirane module kôda.

npm je klijent naredbenog retka koji programerima omogućuje instaliranje i objavljivanje tih paketa.

Frontend paketi o kojima aplikacija ovisi zapisani su u `package.json` datoteku, a te iste ovisnosti (engl. *dependencies*) instaliraju se pozivom naredbe `npm install` iz terminala. Pokretanjem te naredbe kreira se `node_modules` direktorij koji u sebi sadrži poddirektorije - instalirane module tj. biblioteke potrebne za *frontend* dio aplikacije [12].

2.3.2. Composer

Composer je alat za upravljanje ovisnostima u PHP-u te je osnova modernog PHP razvoja. Omogućava deklariranje "paketa" tj. biblioteka o kojima projekt ovisi te će njima upravljati - voditi brigu o instalaciji, ažuriranju i uklanjanju istih na bazi projekta tako da će ih preuzeti unutar projekta u `vendor` direktorij. Unutar `composer.json` datoteke zapisane su željene ovisnosti projekta. Unutar `composer.lock` datoteke zapisani su svi instalirani paketi i njihove točne verzije, tako da se projekt "zaključava" na te konkretne verzije paketa [13].

Određeni paket može se zatražiti naredbom `composer require` gdje se navodi naziv isporučitelja paketa, naziv paketa te ograničenje verzije paketa. Naredbom `composer install` instaliraju se sve Composer ovisnosti aplikacije, naredbom `composer update` ažuriraju se paketi, a naredbom `composer remove` i navođenjem naziva isporučitelja paketa i naziva paketa može se ukloniti paket iz liste ovisnosti [14].

Minimalna Composer verzija koju Laravel 10.x zahtijeva je 2.2.0.

2.4. Razvojna okruženja za testiranje

PHPUnit Sebastiana Bergmanna najpopularnije je PHP razvojno okruženje za testiranje. U Laravelu je podrška za testiranje s PHPUnitom zadano uključena te je `phpunit.xml` datoteka unaprijed postavljena. PHPUnit 10 je trenutna stabilna verzija.

Pest je razvojno okruženje za testiranje izgrađeno na PHPUnitu, ali s uključenim novim dodatnim značajkama. Podržava i pokretanje testova pisanih za PHPUnit. Trenutna verzija Pesta je 2.0.

3. Opis implementacije praktičnog rada

3.1. Instalacija i konfiguracija okruženja

3.1.1. Početni komplet Laravel Jetstream

Kako bi developerima uštedio vrijeme u samome početku razvijanja nove aplikacije, Laravel nudi autentikacijske i aplikacijske početne komplete (engl. *starter kits*) kao što su Laravel Breeze i Laravel Jetstream koji automatski pružaju rute, kontrolere i poglede (engl. *views*) potrebne za registraciju i autentikaciju [15].

Laravel Jetstream dizajniran je koristeći **Tailwind CSS**, *utility-first* CSS razvojni okvir. Datoteke `postcss.config.js` i `tailwind.config.js` koriste se pri *buildanju* kompajliranog CSS-a aplikacije.

Značajke koje početni komplet Laravel Jetstream pruža su autentikacija, registracija, upravljanje korisničkim profilima, ponovno postavljanje lozinke, verifikacija e-adrese, dvostruka provjera autentičnosti (engl. *two-factor authentication*), upravljanje aktivnim sesijama u web preglednicima, API podrška te opcionalne opcije za upravljanje timovima [16].

Instalira se koristeći Composer, a naredbe za instalaciju prikazane su u ispisu 1.

```
composer create-project laravel/laravel teamstructor-app

cd teamstructor-app

composer require laravel/jetstream
```

Ispis 1: Naredbe za instalaciju Jetstream paketa u novi Laravel projekt

Jetstream pruža izbor između korištenja **Livewire** ili Inertia.js *frontend scaffoldinga*. O tom izboru ovisi i odabrani jezik za predloške (engl. *templating language*) jer uz Livewire to je **Blade**, a uz Inertia.js to je Vue.js [16].
Za instaliranje Jetstreama s Livewire *frontend scaffoldingom* i to s uključenom podrškom za timove koristi se naredba `php artisan jetstream:install livewire --teams` [17].

`php artisan vendor:publish --tag=jetstream-views` je naredba či-

jim se izvršavanjem u `app` direktoriju kreira direktorij `resources/view/components` koji sadrži razne generičke Blade komponente čija je svrha da ih se jednostavno može koristiti te pružanje konzistentnog korisničkog sučelja bez potrebe da developer kreira vlastite komponente [18].

Nakon instalacije Jetstreama potrebno je instalirati i pokrenuti *build* NPM ovisnosti pomoću naredbi `npm install` i `npm run build` te migrirati bazu naredbom `php artisan migrate` [17].

3.1.2. `.env` datoteka

`.env` datoteka nalazi se u aplikacijskom root direktoriju i služi kao konfiguracijska datoteka za Laravel web aplikaciju. Pri instalaciji Laravela kreirana je tako da se u nju kopira ogledna konfiguracijska datoteka `.env.example` [19].

Dobra je praksa i zbog sigurnosti i zbog toga što je konfiguracija promjenjiva ovisno o pojedinačnim okruženjima da neenkriptirana `.env` datoteka nije dio kontrole izvornog kôda aplikacije, dok je to u redu za `.env.example` datoteku te ista može poslužiti kao ogledni primjer s *placeholder* vrijednostima za varijable koje je potrebno definirati [19].

Primjer definicije varijable može se vidjeti u ispisu 2.

```
APP_NAME=Teamstructor
```

Ispis 2: Definicija varijable u `.env` datoteci

U Laravel aplikaciji pristup vrijednosti pojedine varijable iz `.env` datoteke moguć je pomoću `$_ENV` PHP superglobalne varijable ili koristeći funkciju `env`, kao u ispisu 3.

```
'name' => env('APP_NAME', 'Laravel'),
```

Ispis 3: Pristup vrijednosti *environment* varijable u Laravelu

3.2. Struktura Laravel aplikacije

3.2.1. Struktura početnog direktorija s izvornim kôdom aplikacije

Osnovna struktura *root* direktorija Laravel aplikacije jest sljedeća [20]:

```
teamstructor-app
├── app
├── bootstrap
├── config
├── database
├── node_modules
├── public
├── resources
├── routes
├── storage
├── tests
└── vendor
```

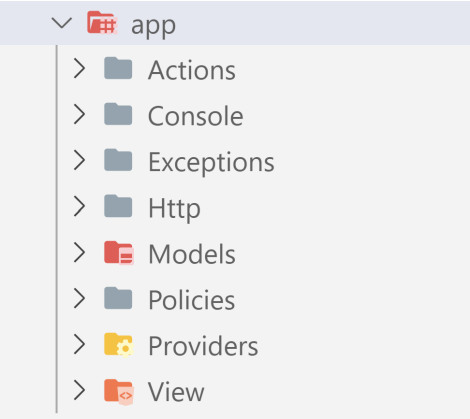
- `app` - Sadrži "jezgru" aplikacije te će biti posebno predstavljen u poglavlju 3.2.2.
- `bootstrap` - Sadrži datoteku `app.php` koja pokreće razvojni okvir te direktorij `cache` koji sadrži datoteke predmemorije za optimizaciju performansi.
- `config` - Sadrži konfiguracijske datoteke.
- `database` - Sadrži migracije, tvornice modela (engl. *model factories*) i *seed*-ove.
- `node_modules` - Nije u osnovnoj strukturi, međutim nastaje instalacijom NPM ovisnosti te sadrži instalirane module tj. biblioteke potrebne za *frontend* dio aplikacije kao poddirektorije.
- `public` - Sadrži datoteku `index.php` koja je polazna točka svih zahtjeva prema aplikaciji te konfigurira automatsko učitavanje (engl. *autoloading*). Također se tu nalaze i javna "imovina" aplikacije (engl. *assets*): slike te JavaScript i CSS datoteke.
- `resources` - Sadrži poglede (engl. *views*) i nekompajlirane JavaScript i CSS datoteke koje su *assets* aplikacije.
- `routes` - Sadrži definicije svih ruta u aplikaciji. Zadano su uključene datoteke: `web.php`, `api.php`, `console.php` i `channels.php`.
- `storage` - Služi kao lokalno spremište podataka aplikacije te je podijeljen na direktorije `app`, `framework` i `logs`. Sadrži `.log` datoteke zapisnika, kompajlirane

Blade predloške, sesije, predmemorije datoteka itd.

- `tests` - Sadrži skripte za automatizirane aplikacijske testove te dolazi s po jednim oglednim primjerom *unit* i *feature* testova.
- `vendor` - Sadrži preuzete Composer ovisnosti aplikacije.

3.2.2. Struktura `app` direktorija

Na slici 1 prikazan je sadržaj `app` direktorija.



Slika 1: Sadržaj `app` direktorija

Možemo vidjeti da se `app` direktorij sastoji od sljedećih poddirektorija [20]:

- `Actions` - Direktorij koji se kreira pri Jetstream instalaciji, a sadrži `Action` klase koje obično izvode samo jednu akciju i odgovaraju jednoj Jetstream ili Fortify značajki kao npr. kreiranje korisnika ili tima, postavljanje nove lozinke, brisanje korisnika ili tima, dodavanje člana u tim itd. [21]
- `Console` - Sadrži datoteku `Kernel.php` u kojoj se registriraju *custom* Artisan naredbe.
- `Exceptions` - Sadrži datoteku `Handler.php` koja upravlja svim iznimkama u aplikaciji te je moguće registrirati nove *custom* iznimke.
- `Http` - Gotovo sva logika aplikacije smještena je u ovaj direktorij. Sadrži kontrolere, klase Livewire komponenti te *middleware*.
- `Models` - Sadrži klase svih Eloquent modela ove web aplikacije.
- `Policies` - Nastaje izvođenjem `make:policy` Artisan naredbe te sadrži klase u

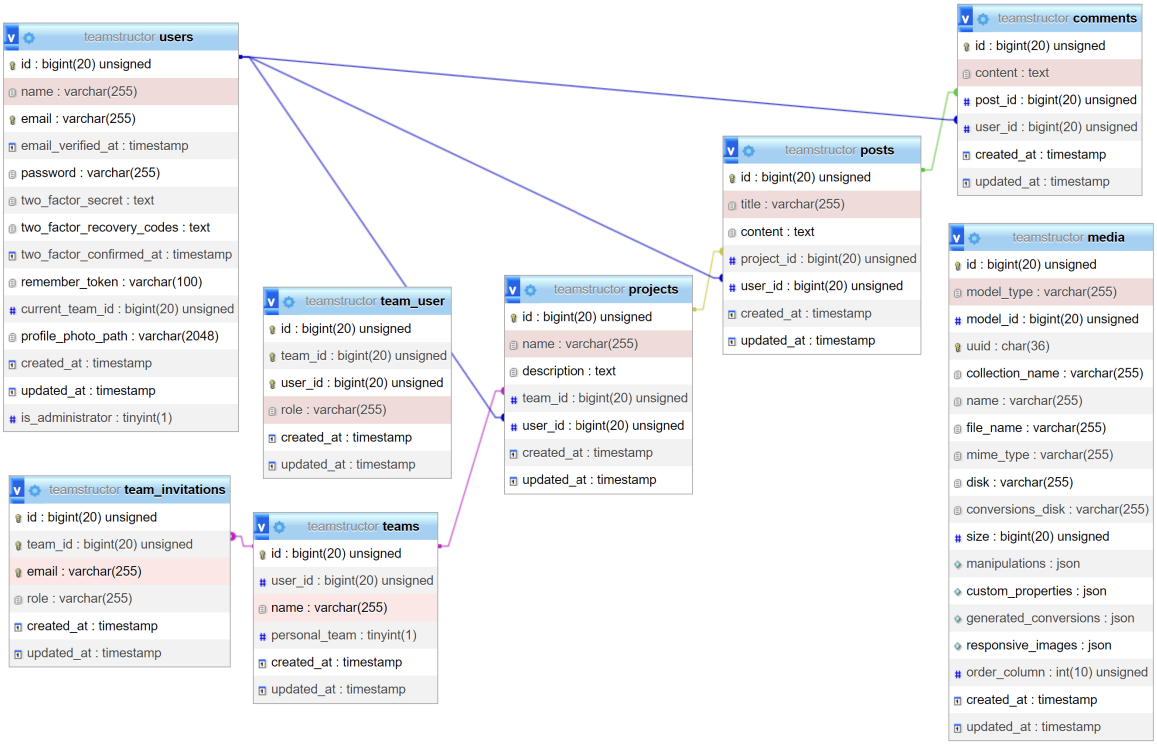
kojima su definirana pravila autorizacije unutar aplikacije.

- `Providers` - Sadrži sve davatelje usluga (engl. *service providers*) aplikacije.
- `View` - Direktorij se kreira pri Jetstream instalaciji, a sadrži klase Livewire *layout* komponenti - "application" *layout* te "guest" *layout* [22].

Još nekolicina direktorija može biti generirana unutar `app` direktorija izvršavanjem Artisan `make` naredbi za generiranje klasa [20].

3.3. Struktura relacijske baze podataka

Na slici 2 prikazan je ER (engl. *Entity Relationship*) dijagram relacijske baze podataka u phpMyAdmin sučelju na kojemu možemo vidjeti tablice u koje su podatci spremani te njihove međusobne relacije.



Slika 2: Dijagram relacijske baze podataka

Radi preglednosti su na dijagramu sa slike 2 prikazane samo tablice koje se odnose na modele, a izostavljene su tablice:

- `failed_jobs` - Automatski prisutna u Laravel aplikacijama, služi za pohranu poslova iz reda čekanja koji se nisu uspjeli izvršiti.

- `migrations` - Automatski prisutna u Laravel aplikacijama, služi za pohranu migracija.
- `password_reset_tokens` - Automatski prisutna u Laravel aplikacijama, služi za pohranu tokena za ponovno postavljanje lozinki u aplikaciji.
- `personal_access_tokens` - Kreira ju Laravel Sanctum pri instalaciji Laravel Jetstreama, a služi za pohranu API tokena u aplikaciji.
- `sessions` - Kada se koristi `database` *driver* za sesije, onda se u ovu tablicu iste pohranjuju.
- `telescope_entries`, `telescope_entries_tags`, `telescope_monitoring` - Nastaju pri instalaciji Laravel Telescopea, a služe za pohranu podataka za Telescope.

U poglavlju 3.4 će biti više rečeno o načinima na koje se ostvaruje interakcija s bazom podataka te o samim modelima i relacijama.

3.4. Interakcija s bazom podataka

3.4.1. Migracije

Migracije služe kao kontrola verzije baze podataka. U Laravelu migracije koriste fasadu (engl. *facade*) `Schema` koja pruža *database agnostic* podršku za sve sustave baza podataka koje Laravel podržava [23].

Migracija se može kreirati pozivom `make:migration` Artisan naredbe te će se ista smjestiti u `database/migrations` direktorij. Nazivu svake migracije dodaje se pripadajući *timestamp* prema kojem Laravel prati redoslijed migracija [23].

Sama migracijska klasa sadrži dvije metode: `up` koja kreira ili modificira tablice, stupce ili indekse u bazi podataka te `down` koja bi trebala poništiti promjene učinjene u `up` metodi inverznim operacijama [23].

U ispisu 4 možemo vidjeti kôd migracije za kreiranje tablice `projects` u kojoj u `up` metodi kreiramo tablicu navodeći njezin naziv i stupce koje sadrži, a u `down` metodi poništavamo operacije izvedene u `up` metodi na način da se odbacuje novokreirana tablica tj. izvodi *drop* tablice.

```php
<?php

use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;

return new class extends Migration
{
    /**
     * Run the migrations.
     */
    public function up(): void
    {
        Schema::create('projects', function (Blueprint $table) {
            $table->id();
            $table->string('name');
            $table->text('description');
            $table->foreignId('team_id')->constrained();
            $table->foreignId('user_id')->constrained();
            $table->timestamps();
        });
    }

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

Ispis 4: Migracija za kreiranje tablice `projects`

U ispisu 5 prikazana je migracija koja modificira tablicu `users` tako da joj dodaje stupac `is_administrator`. U `up` metodi definira se stupac koji će se dodati, a u `down` metodi te promjene će biti poništene odbacivanjem novododanog stupca metodom `dropColumn`.

```php
<?php

use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;

return new class extends Migration
{
    /**
     * Run the migrations.
     */
    public function up(): void
    {
        Schema::table('users', function (Blueprint $table) {
            $table->boolean('is_administrator')->default(false);
        });
    }

    /**
     * Reverse the migrations.
     */
    public function down(): void
    {
        Schema::table('users', function (Blueprint $table) {
            $table->dropColumn('is_administrator');
        });
    }
};
```

Ispis 5: Migracija za dodavanje stupca `is_administrator` u tablicu `users`

Za izvršiti sve migracije koje se još nisu primjenile na bazu podataka pokreće se Artisan naredba `migrate`, a za povratak na prethodnu migraciju tj. poništavanje promjena posljednjih *n* migracija Artisan naredba `migrate:rollback` [23].

3.4.2. Populacija podatcima (engl. *seeding*)

3.4.3. Eloquent ORM

3.4.3.1. Modeli i tvornice modela (engl. *model factories*)

3.4.3.2. Relacije

3.4.3.3. Query Builder i kolekcije

3.5. Usmjerivanje zahtjeva (engl. *routing*)

3.6. Middleware

3.7. Autentikacija

3.8. Autorizacija

3.8.1. Politike (engl. *policies*)

3.8.2. Vrata (engl. *gates*)

3.9. Livewire komponente

3.9.1. Akcije

3.9.2. Događaji

3.10. Pohrana datoteka

3.10.1. MinIO - AWS S3 kompatibilan servis za pohranu

3.11. Spatie Laravel Media Library paket

3.12. Lokalizacija

3.13. Testiranje i kvaliteta kôda

3.13.1. Korišteni alati

3.13.1.1. Laravel Debug Bar

3.13.1.2. Laravel Telescope

3.13.1.3. Laravel Pint

3.13.2. Pretpregled elektroničke pošte - MailHog

3.13.3. Pisanje i pokretanje testova

3.14. Otvoreni kôd i doprinos zajednice na projektima Laravel ekosustava

4. Prezentacija korisničkog sučelja pri korištenju aplikacije

5. Zaključak

U ovom tekstu dan je pregled izrade završnog rada. Na temelju iznesenog, studenti bi trebali izraditi korektno sastavljeni završni rad. Naravno, najteži dio izrade je sadržaj rada, koji studenti moraju sami osmisliti. Ovaj tekst pomoći će im da izbjegnu zamke plagiranja i da uoče vrijednost samostalnog i autorskog rada.

Literatura

[1] https://github.com/laravel/laravel, posjećeno 12.6.2023.

[2] Laravel LLC, "Laravel - The PHP Framework For Web Artisans," https://laravel.com/, posjećeno 12.6.2023.

[3] ——, "Laravel Documentation - Artisan Console," https://laravel.com/docs/10.x/artisan, posjećeno 12.6.2023.

[4] ——, "Laravel Documentation - Release Notes," https://laravel.com/docs/10.x/releases, posjećeno 12.6.2023.

[5] Docker Inc., "Docker Docs - Try Docker Compose," https://docs.docker.com/compose/gettingstarted/, posjećeno 14.6.2023.

[6] ——, "Docker Docs - Volumes," https://docs.docker.com/storage/volumes/, posjećeno 14.6.2023.

[7] Laravel LLC, "Laravel Documentation - Deployment," https://laravel.com/docs/10.x/deployment, posjećeno 14.6.2023.

[8] NGINX - Part of F5, Inc., "NGINX Docs - Configuring NGINX and NGINX Plus as a Web Server," https://docs.nginx.com/nginx/admin-guide/web-server/web-server/, posjećeno 14.6.2023.

[9] https://github.com/php/php-src, posjećeno 14.6.2023.

[10] S. Brekalo, *Uvod u PHP programiranje*. Međimursko veleučilište u Čakovcu, 2018., https://www.mev.hr/wp-content/uploads/2019/01/Uvod_u_PHP_programiranje.pdf, posjećeno 14.6.2023.

[11] The PHP Group, "php.net - Supported Versions," https://www.php.net/supported-versions.php, posjećeno 14.6.2023.

[12] npm, Inc., "npm Docs - About npm," https://docs.npmjs.com/about-npm, posjećeno 14.6.2023.

[13] Composer, "Composer Documentation - Getting Started," https://getcomposer.org/doc/00-intro.md, posjećeno 14.6.2023.

[14] ——, “Composer Documentation - Basic Usage,” https://getcomposer.org/doc/01-basic-usage.md, posjećeno 14.6.2023.

[15] Laravel LLC, “Laravel Documentation - Starter Kits,” https://laravel.com/docs/10.x/starter-kits, posjećeno 16.6.2023.

[16] “Laravel Jetstream - Introduction,” https://jetstream.laravel.com/3.x/introduction.html, posjećeno 16.6.2023.

[17] “Laravel Jetstream - Installation,” https://jetstream.laravel.com/3.x/installation.html, posjećeno 16.6.2023.

[18] “Laravel Jetstream - Livewire,” https://jetstream.laravel.com/3.x/stacks/livewire.html, posjećeno 16.6.2023.

[19] Laravel LLC, “Laravel Documentation - Configuration,” https://laravel.com/docs/10.x/configuration, posjećeno 16.6.2023.

[20] ——, “Laravel Documentation - Directory Structure,” https://laravel.com/docs/10.x/structure, posjećeno 16.6.2023.

[21] “Laravel Jetstream - Concept Overview: Actions,” https://jetstream.laravel.com/2.x/concept-overview.html#actions, posjećeno 16.6.2023.

[22] “Laravel Jetstream - Concept Overview: Layouts,” https://jetstream.laravel.com/2.x/concept-overview.html#layouts, posjećeno 16.6.2023.

[23] Laravel LLC, “Laravel Documentation - Database: Migrations,” https://laravel.com/docs/10.x/migrations, posjećeno 18.6.2023.

Dodatci