|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | | | | |
| **TECHNOLOGIJŲ FAKULTETAS**  **INŽINERIJOS IR INFORMATIKOS** **katedra** | | | | |
|  | | | |  |
| **PROJEKTŲ SKELBIMO PORTALAS** | | | | |
|  | | | |  |
| Kursinis darbas | | | | |
|  | | | |  |
| Informatikos studijų programos | | | | |
| valstybinis kodas 6531BX004 | | | | |
| Informatikos studijų krypties | | | | |
|  | | | |  |
| Autorius Vitalijus Lovkis | |  |  |  |
|  |  | *(parašas)* |  | *(data)* |
| Vadovas doc. dr. Aleksas Narščius | |  |  |  |
|  |  | *(parašas)* |  | *(data)* |
| Klaipėda, 2023 | | | | |

**TURINYS**

[PAVEIKSLŲ SĄRAŠAS 2](#_Toc131427298)

[ĮVADAS 3](#_Toc131427299)

[1. PROGRAMŲ SISTEMOS PROJEKTAVIMAS 6](#_Toc131427300)

[1.1. Duomenų bazės projektavimas 6](#_Toc131427301)

[1.2. Naudojami duomenys 7](#_Toc131427302)

[1.2.1. Duomenų objektai 8](#_Toc131427303)

[1.2.2. Duomenų struktūros 9](#_Toc131427304)

[1.3. Programinis projektas 10](#_Toc131427305)

[1.4. Projektavimo šablonai 11](#_Toc131427306)

[2. PROGRAMŲ SISTEMOS REALIZACIJA 14](#_Toc131427307)

[2.1. JPA realizavimas 14](#_Toc131427308)

[2.2. DB užklausos 15](#_Toc131427309)

[2.3. Algoritmai 15](#_Toc131427310)

[2.4. Grafinė naudotojo sąsaja 16](#_Toc131427311)

[3. PROGRAMŲ SISTEMOS KOKYBĖS UŽTIKRINIMAS 18](#_Toc131427312)

[3.1. Testavimas 18](#_Toc131427313)

[3.2. Kodo versijų kontrolė 21](#_Toc131427314)

[IŠVADOS 22](#_Toc131427315)

[Bibliografija 23](#_Toc131427316)

# 

# PAVEIKSLŲ SĄRAŠAS

[1 pav. Duomenų bazė 7](#_Toc131083394)

[2 pav. Duomenų bazės lentelių objektai 9](#_Toc131083395)

[3 pav. Projekto architektūrinis modelis 11](#_Toc131083396)

[4 pav. Skelbimo atnaujinimo funkcija 16](#_Toc131083397)

[5 pav. Paieškos algoritmas 17](#_Toc131083398)

[6 pav. Sort metodas 17](#_Toc131083399)

[7 pav. Grafinė vartotojo sąsają (kompiuteris) 18](#_Toc131083400)

[8 pav. Grafinė vartotojo sąsaja (mobilus įrenginys) 18](#_Toc131083401)

[9 pav. Kodo testavimo padengimo puslapis. 19](#_Toc131083402)

[10 pav. Namų puslapio greitis 20](#_Toc131083403)

[11 pav. Prisijungimo puslapio greitis 21](#_Toc131083404)

[12 pav. Kontaktų formos puslapio greitis 21](#_Toc131083405)

[13 pav. Kodo versijavimo istorija 22](#_Toc131083406)

# ĮVADAS

**Tikslas*.*** Įsisavinti duomenų struktūrų, duomenų bazių, grafinės naudotojo sąsajos programavimą, projektavimo šablonų taikymą, kodo versijų kontrolės, dokumentavimo bei testavimo įrankių naudojimą, programuojant dalykinės srities vientisą taikomąją programą.

Siekiant įgyvendinti numatytą tikslą, keliami šie praktikos **uždaviniai**:

|  |
| --- |
| 1. Suprojektuoti programų sistemą:    1. Suprojektuoti sistemos pradinius duomenis ir rezultatą;    2. Apibrėžti programoje naudojamas duomenų struktūras;    3. Aprašyti programinio projekto struktūrą;    4. Parinkti ir pritaikyti programavimo šablonus projektuojant architektūrą. |
| 1. Realizuoti programų sistemą:    1. Suprogramuoti duomenų įvedimo/išvedimo srautus;    2. Suprogramuoti programos skaičiavimų algoritmus;    3. Suprogramuoti grafinę naudotojo sąsają; |
| 1. Užtikrinti programų sistemos kūrimo proceso valdymą ir kokybės užtikrinimą:    1. Sukurti kodo tikrinimo automatinių testus;    2. Naudoti kodo versijų kontrolės įrankius.   Programavimo kursinio darbo kūrimui buvo remiamasi duota minimalių reikalavimų lentele (žr. 1 lentelę). |

1 lentelė. Minimalių reikalavimų lentelė

|  |  |
| --- | --- |
| **Minimalūs reikalavimai:** | **Pildo dėstytojas** |
| Laikomasi kodo **vardinimo taisyklių** |  |
| **Ataskaita** be gramatinių ar formatavimo klaidų |  |
| **Gitlab** pateiktas kodas ir ataskaita |  |
| Yra ataskaita su užpildytais visais duoto šablono skyriais |  |
| Kiekviename ataskaitos skyriuje yra aiškiai nurodyta kurioje kodo dalyje yra realizuotas rezultatas |  |

Taip pat buvo remiamasi ir vertinimo kriterijų lentele (žr. 2 lentelę).

# PROGRAMŲ SISTEMOS PROJEKTAVIMAS

Kuriamas projektas yra internetinis portalas, kuris leidžia klientui peržiūrėti projektų skelbimus arba juos talpinti į svetainę. Visas projektas yra daromas programavimo kalba „PHP“ ir naudojamas su karkasu „Laravel“. Projekte numatyta, kad klientas galėtų rasti projektą prie kurio galėtų prisidėti ir tokiu atveju susisiektų tiesiogiai su užsakovu be tarpininkų, jeigu atitinka užsakovo reikalavimus. Taip pat, klientas pats gali įdėti projekto skelbimą į svetainę, jeigu norėtų surasti darbuotojus, kurie galėtų tą projektą įgyvendinti. Svetainė nėra tiesioginė susisiekimo priemonė ir neturi jokių bendrų ryšių tarp užsakovo ir vykdytojo. Svetainė yra tik priemonė, kuri leidžia ieškoti ir skelbti projektų skelbimus. Tolimesnis bendravimas vyktų per užsakovo duotus duomenis pavyzdžiui kaip telefono numeris, el-paštas, svetainė.

## Duomenų bazės projektavimas

Projekto duomenų bazė susideda iš penkių lentelių, kuriuose kaupiasi duomenys apie projektų užsakymus arba užregistruotus klientus, kurie naudojasi svetainės paslaugomis. Rankiniu būdu duomenų bazės nereikėjo kurti, kadangi programavimo karkasas turi komandinę eilutę su kurios pagalba galima lengvai sukurti lenteles, kuriuos iškart talpinamos į duomenų bazę. Visas procesas vyksta naudojant „Php artisan migrate“ komandinę eilutę. „Migration“ (liet. *perkėlimas/migracija*) yra duomenų bazės versijos valdymo tipas. Jis leidžia komandai keisti duomenų bazės schemą ir nuolat atnaujinti esamą schemos būseną. Perkėlimai paprastai susiejami su schemų kūrimo priemone, kad būtų galima lengvai valdyti programos duomenų bazės schemas (n.d, Laravel 2022).

Paveikslėlis, kuriame yra tekstas, vidaus

Automatiškai sugeneruotas aprašymas

1. Duomenų bazės projektas

Pirmame paveikslėlyje galima matyti sukurtas duomenų bazės lenteles, kurios buvo sukurtos nenaudojant „phpMyAdmin“ programos, o naudojant „Migrate“ funkciją. Privalumas naudojant „Migrate“ funkciją yra tas, kad kuriant naują lentelę ir bandant rašyt jos pavadinimą, Laravel karkasas bando pats atpažinti lentelės pavadinimą ir automatiškai sukurti, ir suprojektuoti lentelės stulpelių schemą. Sekančiame kodo pavyzdyje galima matyti kaip parašius komandinę eilutę php artisan make:migration create\_users\_table yra sukuriamas naujas failas pavadinimu „create\_users\_table“ ir automatiškai sugeneruojami nauji laukai, kurie skirti užpildyti duomenis.

class CreateUsersTable extends Migration

{

public function up()

{

Schema::create('users', function (Blueprint $table) {

$table->id();

$table->string('name');

$table->string('email')->unique();

$table->timestamp('email\_verified\_at')->nullable();

$table->string('password');

$table->rememberToken();

$table->timestamps();

});

}

public function down()

{

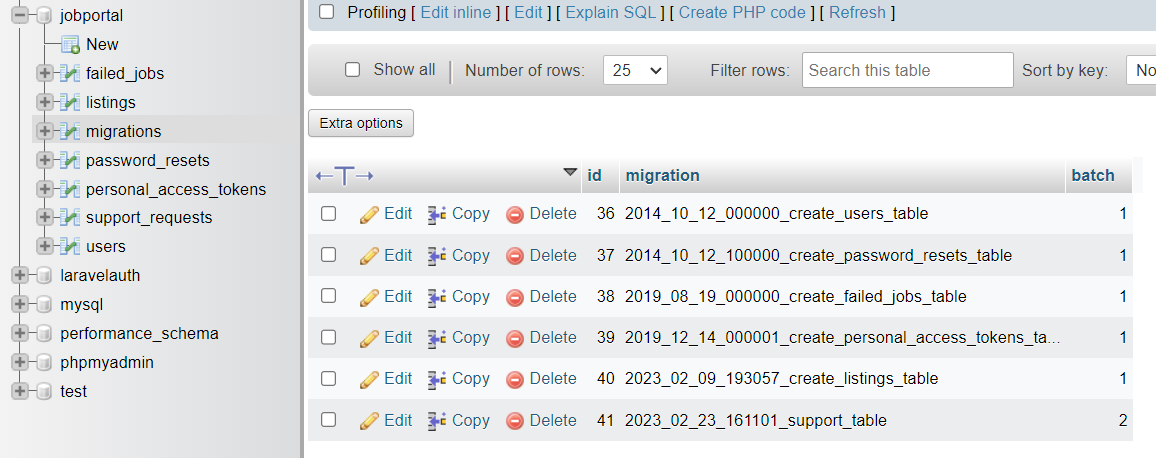
Schema::dropIfExists('users');

}

}

## Naudojami duomenys

Duomenų kaupimas yra organizuotas pagal gerąsias praktikas. Yra daug protokolų, skirtų užšifruoti duomenis, kad būtų užkirstas kelias visiems, kurie juos gauna (Abdulazeez Mousa, 2020). Jautrūs duomenys kaip slaptažodžiai yra užšifruojami specialiu kodu, kad nebūtų įmanoma jų sužinoti net svetainės administratoriui. Taip pat projekte yra kaupiami duomenys kelių variacijų. Pirmoji variacija yra vartotojų duomenys. Vartotojų duomenys sudaro vardas, pavardė, el. paštas, slaptažodis. Kita variacija duomenų yra projektų skelbimai. Skelbimų duomenys susidaro iš įmonės vardo, skelbimo pavadinimo, paveiksliuko, vietovės, susisiekimo kontaktų ir žymių.



1. Duomenų bazės lentelių objektai

Pav 2 galime pamatyti kokios lentelės buvo kuriamos duomenų bazėje naudojant „php artisan migrate“ kodo eilutę. Visos lentelės buvo sukeliamos automatiniu būdu. Lentelių nereikėjo kurti įvedinėjant ranka visus stulpelius ir pavadinimus, kadangi tai atlieka „Laravel“ karkasas už programuotoją.

## Duomenų objektai

Duomenų objektas susideda iš klasės rinkinio, kuriame yra laukai, kurie nustato tam tikrą reikšmę apie objektą. Šitame projekte buvo naudojami keli duomenų objektai, kurie leisdavo sukurti objektus leidžiančius naudoti pačiame projekte. Pagrindinis duomenų objektas buvo kliento objektas (angl. *User*), kuris susideda skirtas sukurti vartotoją:

class User extends Authenticatable

{

use HasApiTokens, HasFactory, Notifiable;

protected $fillable = [

'name',

'email',

'password',

];

}

Šitas duomenų objektas susideda iš trijų pagrindinių laukų: vardo, el pašto ir slaptažodžio. Objektas leidžia sukurti vartotoją saugiu pavidalu, kadangi naudoja „protected“ lauko apsaugą, kuri reiškia, kad pati klasė negali būti modifikuojama kažkur išorėje ir išliks tokia pati. Tokia lauko apsauga dažniausia vadinama inkapsuliacija. Pagal apibrėžimą inkapsuliavimas apibūdina duomenų ir metodų, naudojančių tuos duomenis, sujungimą į vieną vienetą. Ši sąvoka dažniausiai naudojama norint paslėpti vidinį objekto vaizdą ar būseną nuo išorės.

Antras duomenų objektas lygiai taip pat svarbus kaip ir pirmas. Duomenų objektas „Listing“ (liet. *sąrašas*) skirtas sukurti projekto skelbimą. Pačiame objekte yra irgi laukų, kurie privalo būti užpildyti, norint patalpinti skelbimą į svetainę. Iš viso yra devyni laukai: pavadinimas, kompanijos pavadinimas, lokacija/vieta, svetainė, el-paštas, aprašymas, žymės, logotipas ir kliento unikalus identifikatorius.

class Listing extends Model

{

use HasFactory;

protected $fillable = ['title', 'company', 'location', 'website', 'email', 'description', 'tags', 'logo', 'user\_id'];

}

Antrasis duomenų objektas taip pat turi funkcijas, kurios skirtos filtruoti sąrašą ir rodyti tik tam tikrus objektus, kurie turi ryšį su kliento identifikatoriumi.

## Duomenų struktūros

Viso projekto tikslas sukurti duomenis, juos peržiūrėt arba skaityti, atnaujinti ir ištrinti. Keturios pagrindinės operacijos, kurias galima atlikti su duomenimis duomenų bazėje arba programoje, vadinamos CRUD (angl. *Create, Read, Update ir Delete*). Terminas "kūrimo operacija" apibūdina naujų duomenų įrašų įtraukimo į duomenų bazę procesą. Pavyzdys - naujos naudotojo paskyros sukūrimas šitame projekte. Sąvoka "skaitymo operacija" apibūdina dabartinių duomenų įrašų gavimo iš sistemos procesą. Pavyzdys - žvilgsnis į naudotojo sukurtus darbo arba projekto skelbimus. Sistemoje jau esamų duomenų įrašų atnaujinimas arba keitimas vadinamas atnaujinimo operacija. Pavyzdžiui, naudotojo sukurtų darbo arba projektų skelbimų atnaujinimas. Terminas "ištrynimo operacija" apibūdina duomenų įrašų pašalinimą iš duomenų bazės. Pavyzdžiui, naudotojo sukurtų skelbimų platformoje ištrynimas. CRUD naudojimo pavyzdį galima matyti žemiau esančioje kode ištraukoje, kuri randasi projekte.

Listing::create($formFields);

return redirect('/')->with('message', 'Listing Created');

}

public function edit(Listing $listing){

return view('listings.edit', ['listing' => $listing]);

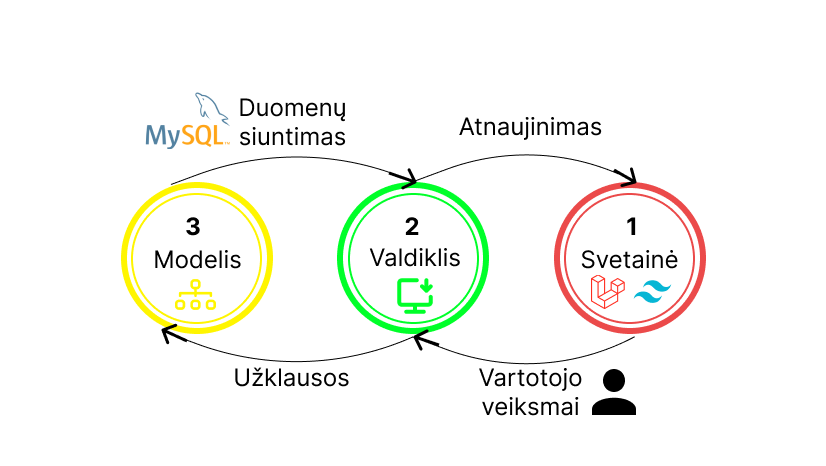
}

...

## Programinis projektas

Projekto žiniatinklio programa buvo sukurta naudojant "Laravel" PHP karkasą, "Tailwind" CSS karkasą ir "MySQL" duomenų bazę. Maršrutizavimas, autentiškumo patvirtinimas ir ORM (liet. *Objektinis reliacinis atvaizdavimas*). Objektinis reliacinis atvaizdavimas - tai metodas, naudojamas kuriant "tiltą" tarp objektinių programų ir dažniausiai reliacinių duomenų bazių. Kitaip tariant, ORM galima laikyti sluoksniu, jungiančiu objektinį programavimą (OOP) su reliacinėmis duomenų bazėmis. (Abba, 2022).

Tai tik kelios iš funkcijų, kurias siūlo PHP karkaso sistema "Laravel". Tailwind, kaip naudingoji CSS sistema, siūlo iš anksto paruoštas CSS klases, kurias galima naudoti norint greitai sukurti unikalią naudotojų patirtį. Žiniatinklio programose dažnai naudojama "MySQL" - populiari reliacinė duomenų bazių valdymo sistema, kuri yra atvirojo kodo. Siekiant valdyti HTTP užklausas ir atsakymus, palaikyti ryšį su "MySQL" duomenų baze ir atvaizduoti vaizdus naudojant "Tailwind" CSS klases, projekte naudojamos "Laravel" integruotos galimybės. Šios technologijos gali būti naudojamos ir yra patikimos, veiksmingos, gali valdyti didžiulius duomenų kiekius ir užtikrinti sklandžią naudotojo patirtį.



1. Projekto architektūrinis modelis

Pirmame paveikslėlyje galima pamatyti viso projekto architektūrinį modelį. Jame visas procesas vyksta per tris pagrindinius komponentus („Svetainė“, „Valdiklis“, „Modelis“). Pirmame komponente, atėjus į svetainę vaizdas bus atvaizduojamas laravel karkaso ir tailwind bibliotekos pagalba. Kai vartotojas atliks bet kokius veiksmus, visos instrukcijos bus siunčiamos į projekto valdiklių sistemą, kurios nustato kokie bus tolimesni serverinės pusės veiksmai. Paskutinis komponentas, kuris svarbus šitame projekte yra modelis. Jis nustato kokie duomenys iš užklausų bus siunčiami į valdiklius ir tada tiesiai į svetainę, kad būtų atvaizduojami vizualių rezultatų atnaujinimai.

## Projektavimo šablonai

Kad projekto kodas nebūtų surašytas bet kaip, tam buvo naudojami projektavimo šablonai. Projektavimo šablono modelis - tai konkrečiam kontekstui būdingas abstraktus, pasikartojantis bendros programinės įrangos projektavimo problemos sprendimas. (Washizaki, 2009). Dažniausiai projektavimo šablonai yra rašomi naudojant objektinę programavimo kalbą (angl. *OOP(Object Oriented Programing)*). Šitame projekte irgi naudojami projektavimo šablonai, kurie leidžia lengviau spręsti projektavimo problemas, kadangi jos pasikartojančios.

Projekte yra naudojama kelių tipų projektavimo šablonai, pirmas tipas, kurį naudoju ir aprašysiu yra **„Kūrimo“ projektavimo šablonai**.

Vienas iš kūrimo projektavimo šablonų, kuris buvo panaudotas yra „Factory“ projektavimo šablonas, kuris suteikia poklasiams galimybę keisti kuriamų objektų tipą, tačiau vis tiek suteikia sąsają, skirtą super klasės objektams kurti. Šitą projektavimo šabloną naudojau kuriant duomenų bazių objektus, kurie atvaizduotų netikrus vartotojus.

class ListingFactory extends Factory

{

public function definition()

{

return [

'title' => $this->faker->sentence(),

'tags' => 'laravel, api, backend',

'company' => $this->faker->company(),

...

Kitas kūrimo projektavimo šablonas, kuris buvo naudojamas projekte yra „Builder“ projektavimo šablonas. Jis dažnai naudojamas kai norima sukurti sudėtingus objektus žingsnis po žingsnio. Šitame projekte builder projektavimo šablonas buvo skirtas atrasti ryšius tarp vartotojo ir jo sukurtų skelbimų ir juos visus atkurti kitame puslapyje. Apačios pavyzdyje galime matyti kaip funkcija liepia sukurti skelbimus, kurie turėtų ryšį tarp su vartotojo id. Noriu pabrėžti, kad pati funkcija nėra builder šablonas, kadangi „Laravel“ karkasas savyje jau turi sukurtą builder šabloną pavadinimu „Query Builder“, kuri yra dalis „Elequent ORM“ sistemos ir kuri yra karkaso gamykliniuose nustatymuose ir jos kurti nereikia, tik panaudoti.

public function listings(){

return $this->hasMany(Listing::class, 'user\_id');

}

...

Kitas tipas šablonų, kuriuos naudojau projekte yra pavadinimu **„Elgesio“ projektavimo šablonai**. Pirmas elgesio projektavimo šablonas, kuris buvo panaudotas, tai buvo „Strategy“ (angl. *strategijos*) projektavimo šablonas. Jis turi šeimą algoritmų, kurios leidžia naudoti vieną iš pasirinktų metodų. Apatiniame pavyzdyje galima matyti, kad yra keli algoritmai sukurti pagal kuriuos būtų daroma skelbimų filtraciją

public function scopeFilter($querry, array $filters){

if($filters['tag'] ?? false){

$querry->where('tags', 'like', '%' . request('tag') . '%');

}

if($filters['search'] ?? false){

$querry->where('title', 'like', '%' . request('search') . '%')

->orWhere('description', 'like', '%' . request('search') . '%')

->orWhere('tags', 'like', '%' . request('search') . '%');

}

}

Svetainėse, kur yra registracijos forma dažniausiai būna ir siuntimai ir įspėjimai apie sėkmingą registraciją arba pasikeitimus. Todėl šiame projekte buvo implementuotas sekantis projektavimo šablonas pavadinimu „Observer“. Šitas šablonas leidžia įspėti vartotojus apie svarbius pasikeitimus. Observer projektavimo šablonas šitame projekte išsiunčia elektroninio pašto žinutę, kurį įspėja vartotoją, kad jis sėkmingai prisiregistravo prie svetainės. Apačioje galima pamatyti kodo ištrauką kaip veikia observer šablonas.

class UserObserver

{

public function created(User $user)

{

Mail::raw('Congratulations, you have just created account on Laravel Breeze.', function ($message) use ($user){

$message->from(env('MAIL\_FROM\_ADDRESS'), env('MAIL\_FROM\_NAME'));

$message->to($user->email);

$message->subject('Welcome to Laravel Breeze.');

});

}

Paskutinis šablonų tipas, kurį panaudojau projekte buvo **„Struktūrinių“ projektavimo šablonų tipai**. Pirmasis struktūrinis šablonas, kuris buvo panaudotas yra vadinamas „Composite“ projektavimo šablonu. Naudodami struktūrinį projektavimo modelį „Composite“, galima grupuoti daiktus į medžio struktūras ir elgtis su šiomis struktūromis kaip su atskirais objektais. Vienas iš pavyzdžių būtų puslapio išdėstymas. Svetainės puslapiai nėra sudaryti nuo nulio. Kai kurie komponentai yra kaip objektai tėvai (angl. parent object) ir jie yra naudojami po kelis kartus. Tokius atveju juos galima naudoti iš naujo nekurti puslapių kopijuojant kodą iš naujo. Apačioje pavyzdyje galima matyti kaip dvi pirmos kodavimo žymės (angl. tag) naudoja pavadinimus su „x“ raide pradžioje, ką reiškia jog jie naudoja tėvų objektus, kurie sukurti kituose failuose ir atvaizduoja tokius pat objektus, kurie buvo sukurti tėvų klasėje.

<x-layout>

<x-card class="p-10 max-w-lg mx-auto mt-16">

...

Paskutinis šablonas, kuris buvo panaudotas šitame projekte, tai „Decorator“ šablonas. Naudodami struktūrinį projektavimo šabloną, vadinamą dekoratoriumi, galima suteikti esamiems objektams naują elgseną, įtraukdami juos į specialius įvyniojamuosius objektus, kuriuose taip pat yra naujos elgsenos. Pats šablonas yra jau iš anksto įtrauktas į gamyklinius „Laravel“ parametrus, kuris yra dalis „Middleware“ (liet. *tarpinė programos įranga*). Su šiuo šablonu yra galima nustatyti naują elgseną, kuri nurodo ką gali matyti neprisijungęs vartotojas ir prisijungęs vartotojas naudojant anksčiau minėta „Middleware“. Pavyzdys apačioje parodo kad prisijungęs vartotojas mato kitą meniu išdėstymą nei, kad neprisijungęs vartotojas.

@auth <------ nurodo, kad tai prisijungusio vartotojo elgesnos vaizdas

<nav class="px-2 bg-lavanderBlush border-gray-200 dark:bg-gray-900 dark:border-gray-700">

<div class="container flex flex-wrap items-center justify-between mx-auto">

<a href="/"><img class="w-20" src="{{asset('images/logo.png')}}" alt="" class="logo"

...

@else <------ nurodo, kad tai neprisijungusio vartotojo elgsenos vaizdas

....

# PROGRAMŲ SISTEMOS REALIZACIJA

Projekto sistemos realizacija buvo sukurta naudojant „Laravel“ karkasą, kuris gali sujungti serverinę projekto pusę su kliento puse. Visi svetainės duomenys yra saugomi lokalioje mašinoje, naudojant „MySQL“ duomenų bazę. Ryšiai tarp duomenų bazės ir projekto taip pat yra sukurti su „Laravel“ karkaso pagalba naudojant „Laravel elequent ORM“ proceso pagalbą. Duomenys iš duomenų bazės yra gaunami siunčiant užklausas skirtas kurti naujus vartotoju, gauti duomenis apie skelbimus, prisijungimus.

"MySQL" yra bene populiariausia duomenų bazė, kurioje įdiegta daugiau nei dešimt milijonų duomenų bazių valdymo sistemų, skirtų žiniatinklio serveriams. „MySQL“ buvo Sukurta dešimtojo dešimtmečio viduryje, dabar ji yra brandi technologija, kuria naudojasi daugelis šiandien lankomiausių interneto vietų. Viena iš jos sėkmės priežasčių, matyt, yra tai, kad, kaip ir PHP, ją galima naudoti nemokamai. Tačiau ji taip pat labai galinga ir itin greita - ji gali veikti net pačiose paprasčiausiose kietosiose ir beveik nenaudoja sistemos išteklių. Be to, "MySQL" labai lengvai keičiama, o tai reiškia, kad ji gali augti kartu su jūsų svetaine (Nixon, 2012).

## JPA realizavimas

Projekte yra sukurta alternatyva JPA karkasui, naudojant „Laravel Eloquent ORM“ karkasą. Žiniatinklio veikimas pagrįstas duomenimis, todėl kūrėjams labai svarbu galvoti apie greitą ir veiksmingą būdą, kaip su jais dirbti. "Eloquent" - tai puiki ORM, kuri pateikiama kartu su Laravel PHP karkasu. Ji yra unikali ir labai naudinga kūrėjams, nes leidžia apibrėžti modelius, ryšius ir daugybę sudėtingų operacijų su tikrai lengva ir intuityvia sintakse, neprarandant našumo. Ši ORM leidžia atlikti operacijas su keliomis lentelėmis nerašant ilgų užklausų su objektais (Malatesta, 2015). Taip pat, projektas naudoja „CRUD“ funkcijas. Žodis „CRUD“ verčiasi iš programavimo kalbos kaip kurti (angl. *create*), skaityti (angl. *read*), atnaujinti (angl. *update*), ištrinti (angl. *delete*). Visos funkcijos leidžia vienam vartotojui atlikti veiksmus su jo sukurtu skelbimu. Taip pat, prieš darant kokius nors veiksmus yra atliekamas patikrinimas ir ryšys tarp vartotojo ir skelbimo, kad įsitikinti jog vartotojas yra savininkas to skelbimo. **Pav 2** Galima pamatyti kaip yra atliekama viena iš CRUD funkcijų, atnaujinimo funkcija, kai vartotojas nori atnaujinti savo skelbimo duomenis.

Paveikslėlis, kuriame yra tekstas

Automatiškai sugeneruotas aprašymas

1. Skelbimo atnaujinimo funkcija

Atnaujinus funkcija vartotojas bus įspėjamas žinute, kad jo skelbimas buvo atnaujintas sėkmingai ir duomenų bazėje informacija atsinaujins to skelbimo, kurio informacija buvo keičiama.

## DB užklausos

Duomenų bazė turi kelias lenteles. Kiekviena lentelė turi skirtingą užklausą, kuri yra siunčiama iš projekto. Pirmoji užklausa siunčiama sukuriant vartotoją. Pavyzdžiui jeigu vartotojas jau registravosi su identišku el-paštu, jam neleis antrą kartą registruotis su tokiu el-paštu, kadangi siunčiama užklausa tikrina. Tuo pačiu, jeigu pirmoji užklausa yra neigiama, tada automatiškai yra siunčiama kita užklausa, kuri talpina naujo vartotojo duomenis į duomenų bazę. Taip pat, talpinimo užklausa yra siunčiama, kai yra kuriamas skelbimas.

Užklausa, kuri apima kelias lentelės yra naudojama kai yra norima identifikuoti ar vartotojas yra skelbimo savininkas. Pirma užklausa yra siunčiama į duomenų bazę ir tikrinama koks vartotojo id numeris. Kiekvieną kartą sukūrus skelbimą, skelbimas turi id numerį, kuris yra nurodomas toks pats, kaip ir vartotojo id. Todėl sekanti užklausa vyksta iš skelbimų lentelės, kadangi norima gauti informaciją apie id. Jeigu vartotojo id ir skelbimo id sutampa, yra atvaizduojamas skelbimas jau iš vartotojo pusės.

## Algoritmai

Projekte yra sukurtas algoritmų rinkinys, kuris gali skelbimus ieškoti, rūšiuoti, filtruoti. Ieškojimo algoritmas yra ganėtinai standartinis, t.y įvedus raktinius žodžius į paieškos lauką, vartotojui bus išvedami visi skelbimai, turintis identiškų įvesties atitikmenų. 2 pav. galima pamatyti pavyzdį kaip atliekamas paieškos algoritmas

Paveikslėlis, kuriame yra tekstas

Automatiškai sugeneruotas aprašymas

1. Paieškos algoritmas

Paveikslėlyje galima matyti, kad paieška yra atliekama ne tik pagal paieškos lauko įvedimą, bet ir pagal žymes (angl. „tag“), kurios filtruoja skelbimus pagal paspaustas žymes. Pavyzdžiui vartotojui paspaudus žymę „react“ jam svetainė išfiltruos visus skelbimus, kurie susiję su „React“ karkasu.

Paskutinis rūšiavimo būdas projekte yra naudojant „sort“ funkciją ir metodą. 3 pav. galima pamatyti, kad skelbimus galima rūšiuoti nuo seniausių iki naujausių ir atvirkščiai.

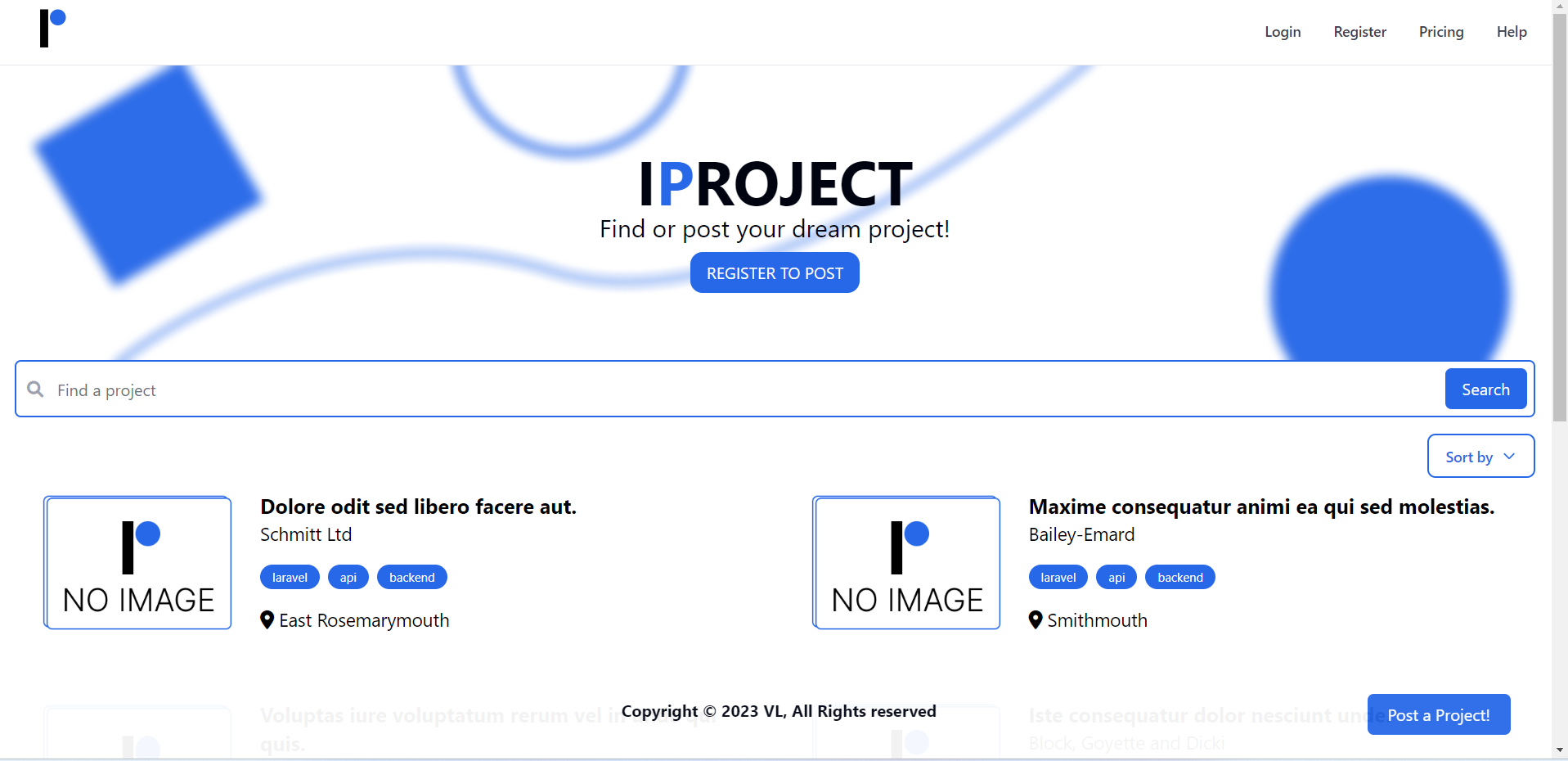


1. Sort metodas

Visi trys rūšiavimai ir filtravimai yra sujungti su duomenų bazės duomenimis. Taip pat, funkciją „sort()“ galima papildyti ateityje ir kitų tipų rūšavimais. Pavyzdžiui pagal kompanijos populiarumą, pagal atsiliepimus kompanijos.

## Grafinė naudotojo sąsaja

Naudojant karkasą „Laravel“, reikėjo apgalvoti kaip sukurti patrauklią grafinę sąsają ir tuo pačiu metu sugalvoti kaip viską pritaikyti mobiliems įrenginiams. Viena iš alternatyvų, kurias panaudojau buvo „Tailwind“ grafinės sąsajos biblioteka. „ "Tailwind CSS" - tai naudingoji CSS struktūra, sukurta tam, kad naudotojai galėtų greičiau ir paprasčiau kurti programas. Naudodami naudingumo klases galite valdyti išdėstymą, spalvas, tarpus, tipografiją, šešėlius ir kt., kad sukurtumėte visiškai individualų komponentų dizainą - nepalikdami HTML ir nerašydami nė vienos pasirinktinės CSS eilutės“ (Fitzgerald, 2022). „Tailwind“ pagalba buvo galima sukurti ne tik patrauklią grafinę vartotojo sąsają, bet ir priderinta mobiliems įrenginiams. **4 Pav.** galime pamatyti kaip grafinė sąsaja atrodo iš kliento pusės naudojant kompiuterį.



1. Grafinė vartotojo sąsają (kompiuteris)

Prieš darant svetainės grafinę sąsaja, buvo atlikta analizė kokios spalvos dominuoja šių dienų kontekste, kai yra kuriame svetainės grafinė sąsaja. Atlikus analizę, buvo išsiaiškinta, kad populiariausios spalvos, kurias naudoja kitos svetainės yra:

* Mėlyna. Spalvos kodas: #2768e7,
* Balta. Spalvos kodas: #ffffff,

Pritaikius šitas dvi spalvas ir sukūrus grafinę sąsają kompiuterių vartotojams, buvo pereinama prie grafinės sąsajos, kuri skirta mobiliems įrenginiams. **5 pav**. galima pamatyti kaip atrodo grafinė sąsaja iš vartotojų pusės, kurie naudoja mobiliuosius įrenginius.

Paveikslėlis, kuriame yra tekstas

Automatiškai sugeneruotas aprašymas

1. Grafinė vartotojo sąsaja (mobilus įrenginys)

Iš viso yra net trys nustatyti ekrano tipai, bet labiausiai dėmesio buvo akcentuojama mobiliems ekranams ir kompiuterių ekranams, kadangi žinoma yra, kad šiomis dienomis didelė žmonių dalis naudoja mobiliuosius įrenginius ir kompiuterius, naršant internete.

# PROGRAMŲ SISTEMOS KOKYBĖS UŽTIKRINIMAS

Kad projekto kokybė buvo užtikrinama maksimaliai, buvo naudojami testavimo įrankiai bei kodo versijavimo įrankiai. Pirmiausiai pradėti, manau, yra tikslingiausia nuo to, kad lengviau „PHP“ kalbos testavimo būdai yra tiesioginis išvedimas „echo“ į svetainę. Tokiu būdu galima suprasti ar kodas atlieka savo funkciją. Bet su projekto augimu yra suprantama, kad kiekvienos eilutės nebus galima testuoti vien tik „echo“ komandos pagalba, kadangi tai ilgai ir varginantis procesas. Todėl šitame projekte naudojau „PHPUnit“ ir „Xdebug“, testavimo įrankius, kurie leido greitai ir efektyviai parašyti kodo testavimus.

Naudodami PHP plėtinį "Xdebug" programuotojai gali derinti ir pagreitinti savo projektų kūrimą, ieškodami ir taisydami klaidas. Atnaujinta PHP funkcija var dump(), pridėtos pranešimų, įspėjimų, klaidų ir išimčių kamino sekos. Norėdami ištirti kintamųjų išvestį vos po vienos kodo iteracijos, galima pridėti nutraukimo taškų ir sustabdyti kodo vykdymą ties kiekvienu nutraukimo tašku (Nawaz, 2023).

Dar vienas įrankis, kuris leido sužinoti projekto greičio efektyvumą, buvo „ClockWork“. "PHP programavimo įrankis, kurį galima naudoti tiesiogiai naršyklėje. HTTP užklausų, komandų, eilės užduočių ir testų atveju "Clockwork" suteikia galimybę susipažinti su jūsų taikomosios programos veikimo laiku, įskaitant užklausų duomenis, našumo rodiklius, žurnalo įrašus, duomenų bazės užklausas, talpyklos užklausas, "Redis" komandas, išsiųstus įvykius, eilėje esančias užduotis, atvaizduotus vaizdus ir dar daugiau (Urlich & Feloidea, 2023).

## 3.1. Testavimas

Šiame projekte tikslas buvo padengti penkiasdešimt procentų viso kodo automatiniais testavimai. Kad matyti kiek kodas buvo padengtas procentas, „Xdebug“ įrankis kiekvieną kartą po testo sukurdavo atskirą puslapį, kuriame buvo išsami informacija apie tai, kiek yra procentaliai yra padengta kodo. Pav 9 galima pamatyti puslapį, kuriame rašoma informacija apie kodo testavimą.

Paveikslėlis, kuriame yra Interneto svetainė

Automatiškai sugeneruotas aprašymas

1. Kodo testavimo padengimo puslapis.

Pav 9 galima pastebėti, kad skiltyje „Total“ kodo eilučių testo padengimas siekia daugiau nei penkiasdešimt procentų, todėl užduotis su kodo versijavimu buvo atlikta sėkmingai. Testuojant kodą, buvo panaudotos daugiau nei penkios „assert“ rūšys. Sekančiame kodo bloke bus galima pamatyti keletą „assert“ rūšių, kurias naudojau testuojant kodą.

$response->assertStatus(200);

$response->assertRedirect('/');

$response->assertOk();

$this->assertEquals(['Laravel Job 1', 'Laravel Job 3'], $filteredTitles);

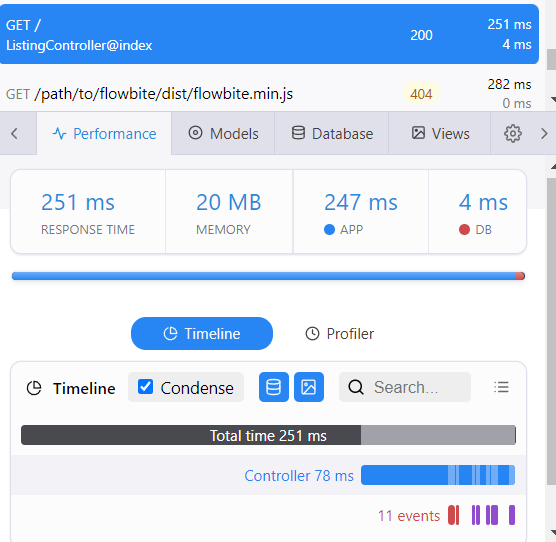
$this->assertDatabaseHas([...]);

$response->assertSessionHas('message', 'Listing Created');

$this->assertTrue($userListings->contains($listing));

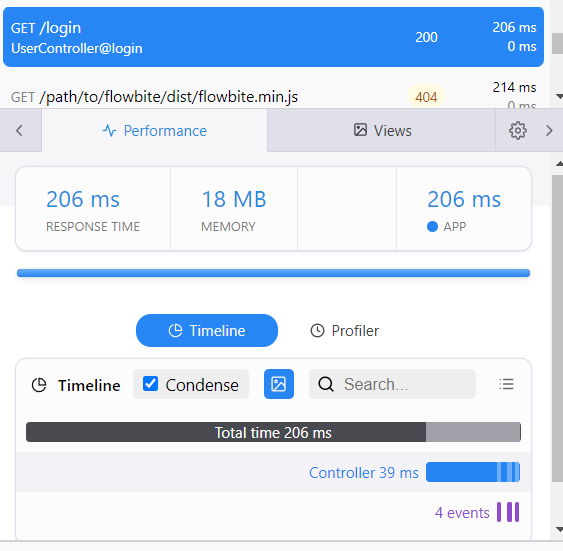
...

Pratestavus serverinės pusės funkcionalumą, buvo pereita prie klientinės pusės testavimo. Tam buvo pasitelkta „ClockWork“ įrankio pagalba. Testuojant puslapių greičių efektyvumą buvo atkreipiamas dėmes kiek užtrunka puslapio užkrovimas. Sekančiose paveiksėliuose bus galima pamatyti kokie greičiau vyrauja puslapiuose.



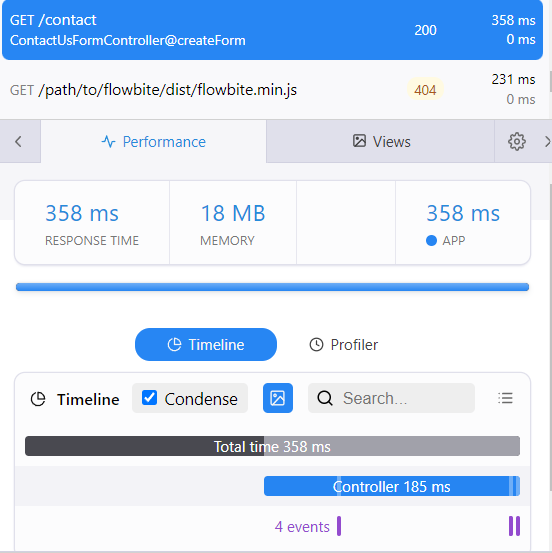
1. Namų puslapio greitis

10 pav galima pamatyti, kad namų puslapis yra užkraunamas per 251ms (milisekundes) ir pats puslapis turi dar papildomus vienuoliką užduočių, kurios susijusios su duomenų baze.



1. Prisijungimo puslapio greitis

11 pav. galima pastebėti, kad užkrovimo laikas prisijungimo puslapyje trunka šiek tiek mažiau, 206 milisekundes. Tai vyksta todėl, kad pats puslapis turi tik keletą klientinės pusės komponentų, ir turi mažiau užduočių, susijusių su duomenų baze.



1. Kontaktų formos puslapio greitis

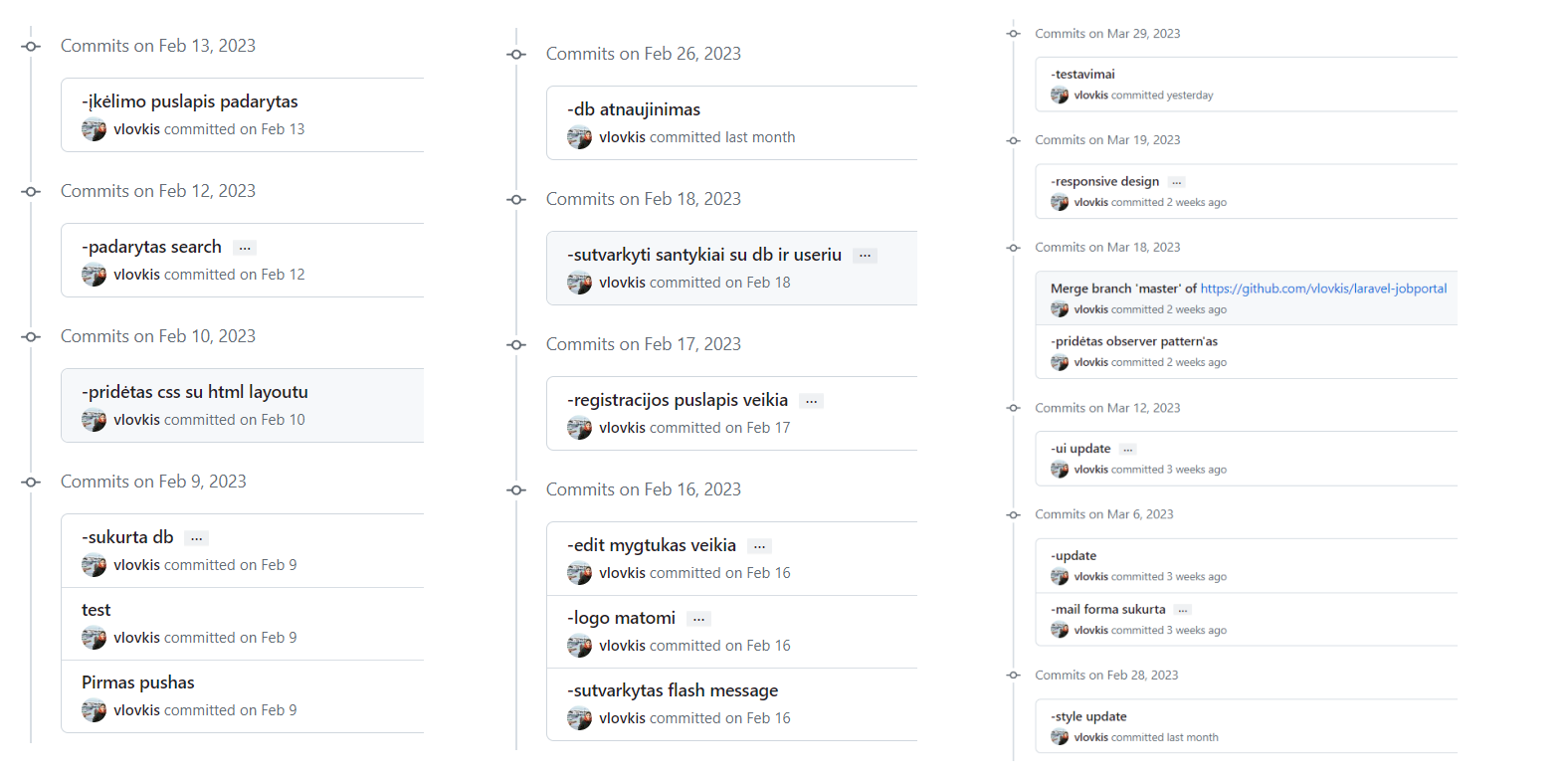
Kontaktų formos puslapis užtrunka ilgiausiai užkrauti, 358 milisekundes. Tai vyksta todėl, kad čia puslapis turi keletą užduočių, susijusių ne tik su lokalia serverine puse, bet ir su išorinę, kadangi kontaktų formos užpildymo metu, klientas siunčia žinutę ne tik į lokalią duomenų bazę, bet ir į išorinę el-pašto žinučių sistemą pavadinimu „Mailtrap“. „Mailtrap“ yra „SMTP“ serveris, kuris leidžia siųsti laiškus į serverį, kuris randasi ne lokalioje aplinkoje.

## 3.2. Kodo versijų kontrolė

Kodo versijavimo kontrolė buvo daroma kas savaitė arba net dažniau, kadangi prie projekto buvo sėdima itin intensyviai tiek iš kodo pusės, tiek iš testavimo pusės. Kodo versijavimui pasirinkau „Github“ praltformą. „Github“ yra internetinė svetainė, kurioje galima laikyti, sekti ir kolaboruoti projektuose. Pati platforma suteikia lengva galimybę programuotojam dalintis savo kodu arba jų failais su kitais programuotojais. (Juviler, 2022)

„Github“ pasirinkau dėl itin greito jo veikimo. Skirtingai nuo daugelio versijų valdymo sistemų, „Github“ veikia kaip momentinės nuotraukos, ir neieško skirtumų. Tai reiškia, kad github neseka skirtumo tarp dviejų failo versijų, bet nufotografuoja dabartinę projekto būseną. Štai kodėl „Github“ yra labai greitas, palyginti su kitais paskirstytaisiais VCS. Dėl to taip pat greitai ir lengvai perjungiama tarp versijų ir šakų (Tsitoara, 2020).

Kad sėkmingai būtų įmanoma sekti visus kodo pasikeitimus, buvo visąlaik rašomi komentarai prieš siunčiant kodą į „Github“ platformą. Kodo versijavimo talpinimus galima pamatyti sekančiose paveikslėliuose.



1. Kodo versijavimo istorija

Kiekvieni kodo siuntimai į versijavimo platformą buvo talpinami į pagrindinį „medį“, kadangi esu tik vienas projekto dalyvis ir nereikėjo patvirtinti išankstinio kodo talpinimo. Visas kodas yra saugomas „Github“ saugykloje, kurios nuoroda yra: <https://github.com/vlovkis/laravel-jobportal>

# IŠVADOS

1. Rezultate buvo suprojektuota programų sistema, kuri yra darbo ir projektų skelbimų svetainė naudojant „Laravel“ karkasą. Kad pasiekti šio rezultato, buvo suprojektuoti programos pradiniai duomenys ir rezultatai, apibrėžtos naudojamos duomenų struktūros naudojant „Laravel“ karkasą ir „Mysql“ duomenų bazę. Taip pat buvo išsiaiškinta, kad geriausi šablonai tokiam projektui naudoti buvo: „Factory“, „Builder“, „Strategy“, „Composite“, „Decorator“, „Observer“ projektavimo šablonai.
2. Buvo realizuota programų sistema, kurioje buvo suprogramuoti duomenys įvedimo ir išvedimo srautai naudojant alternatyva pavadinimu „Laravel Elequent ORM“. Buvo suprogramuoti įvedimo ir išvedimo srautai. Buvo sukurti skaičiavimo algoritmai, kurie leidžia rūšiuoti skelbimus pagal jų sukūrimo datą, paiešką ir žymes. Padaryta grafinė naudotojo sąsaja, naudojant „Tailwind“ bibliotekos įrankius.
3. Buvo užtikrintas programų sistemos kūrimo proceso valdymas ir kokybės užtikrinimas naudojant „Xdebug“ ir „ClockWork“ testavimo įrankius, kurie leido matyti kodo testavimo procentalų padengimą ir puslapio greičio efektyvumą. Tam rezultatui pasiekti buvo išsiaiškinta kaip kurti kodo tikrinimo automatinius testus. Projekto versijavimo kontrolei buvo naudojama „Github“ versijavimo kontrolės platforma ir joje talpinama visa kodavimo istorija ir versijos.

|  |
| --- |
|  |

# PRIEDAI

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Vertinama dalis (skyrius ataskaitoje)** | **Vertė** | **Slenkstinis (5-6)** | **Tipinis (7-8)** | **Puikus (9-10)** |
| **Duomenų bazės projektavimas (1.1)** | 5 % | DB su bent 3 lentelėmis (po min 20 įrašų po min 3 laukus). | DB su bent 4 lentelėmis (po min 20 įrašų po min 3 laukus). Panaudoti kelių tipų ryšiai. | Yra skaitymas iš DB su bent 4 lentelėmis (po min 20 įrašų po min 3 laukus). Panaudoti kelių tipų ryšiai.  Sutvarkytos lietuvių kalbos rašmenų koduotės. |
| **Duomenų objektai (1.2.1)** | 5 % | Dirbama bent su vienu duomenų objektu sudarytu iš min 3 savybių | Dirbama su keliais duomenų objektais arba vienu sudėtiniu duomenų objektu | Dirbama su keliais duomenų objektais ir bent vienas yra sudėtinis duomenų objektas |
| **Duomenų struktūros (1.2.2)** | 10 % | Parinkta viena duomenų struktūra ir pagrįstas jos tinkamumas | Panaudotos kelios duomenų struktūros, apibrėžta sava jų kombinacija. Arba yra galimybė papildyti naujais objektais. | Panaudotos kelios duomenų struktūros, apibrėžta sava jų kombinacija. Ir yra galimybė papildyti naujais objektais. |
| **Programinis projektas (1.3)** | 5 % | Aprašytas programinis projektas ir naudojamos technologijos | Aprašytas programinis projektas ir naudojamos technologijos, nurodytas pilnas projekto architektūrinis modelis | Aprašytas programinis projektas ir naudojamos technologijos, nurodytas pilnas projekto architektūrinis modelis. Aprašyti veikimo algoritmų modeliai. |
| **Projektavimo šablonai (1.4)** | 10 % | Programos kode pritaikyta po 1 kūrimo, struktūrinį bei elgesio projektavimo šabloną | Programos kode pritaikyti 5 kūrimo, struktūrinį bei elgesio projektavimo šablonai | Programos kode pritaikyti 7 (po minimum 2 kūrimo, struktūrinį bei elgesio) projektavimo šablonai |
| **JPA realizavimas (2.1)** | 5 % | Sukonfiguruota JPA, naudojant Hibernate arba alternatyvų karkasą. Sukurta bent viena esybė ir atliekamos CRUD operacijos. | Sukonfiguruota JPA, naudojant Hibernate arba alternatyvų karkasą. Sukurtos bent viena sudėtingesnė esybių struktūra (viena esybė sudaryta iš kelių kitų, viena esybė privalomai turi turėti kitą esybę ir pan.). | Sukonfiguruota JPA, naudojant Hibernate arba alternatyvų karkasą. Sukurtos kelios sudėtingesnės esybių struktūros (viena esybė sudaryta iš kelių kitų, viena esybė privalomai turi turėti kitą esybę ir pan.) |
| **DB užklausos (2.1)** | 7 % | Atliekamos visos CRUD operacijos. | Atliekamos sudėtinės užklausos apimančios kelis parametrus arba kelias DB lenteles | Atliekamos sudėtinės užklausos apimančios kelis parametrus ir kelias DB lenteles |
| **Algoritmai (2.2)** | 8 % | Atliekama viena operacija iš:  Elementų paieška kolekcijoje,  Elementų atranka (filtravimas) kolekcijoje,  Elementų rūšiavimas kolekcijoje | Atliekamos dvi operacijos iš:  Elementų paieška kolekcijoje,  Elementų atranka (filtravimas) kolekcijoje,  Elementų rūšiavimas kolekcijoje | Atliekamos trys operacijos:  Elementų paieška kolekcijoje,  Elementų atranka (filtravimas) kolekcijoje,  Elementų rūšiavimas kolekcijoje |
| **Grafinė naudotojo sąsaja  (2.3)** | 10 % | Yra grafinė naudotojo sąsaja, atvaizduojanti duomenis ir leidžianti manipuliuoti jais. | Yra sudėtinė grafinė naudotojo sąsaja, atvaizduojanti duomenis ir leidžianti manipuliuoti jais. | Yra kelios skirtingos sudėtinės grafinės naudotojo sąsajos (mobile app, web, kt.), atvaizduojančios duomenis ir leidžiančios manipuliuoti jais. |
| **Testavimas (3.1)** | 4 % | Sukurtas kodas testuojamas automatiniais testais (padengimas 20 proc.) | Sukurtas kodas testuojamas automatiniais testais (padengimas 50 proc.). | Sukurtas kodas testuojamas automatiniais testais (padengimas 70 proc.) |
| 3 % | Yra panaudoti bent 3 rūšių assert metodai. | Yra panaudoti bent 4 rūšių assert metodai. | Yra panaudoti bent 5 rūšių assert metodai. |
| 3 % | Yra panaudotos bent 3 rūšių anotacijos. | Yra panaudotos bent 4 rūšių anotacijos. | Yra panaudotos bent 5 rūšių anotacijos. |
| 4 % | Realizuota viena iš testavimo kategorijų:  Išimčių testavimas, Performance testavimas,  Parametrizuoti testai. | Realizuotos dvi iš testavimo kategorijų: Išimčių testavimas, Performance testavimas,  Parametrizuoti testai. | Realizuotos trys iš iš testavimo kategorijų:  Išimčių testavimas, Performance testavimas,  Parametrizuoti testai. |
| **Kodo versijų kontrolė (3.2)** | 8 % | Minimum 25% kassavaitiniai tarpiniai kodo pateikimai | Minimum 50% kassavaitinių tarpinių kodo pateikimų | Minimum 75% kassavaitinių tarpinių kodo pateikimų |
| **Sprendimų pagrindimas (Informacinių šaltinių sąrašas)** | 13 % | Projektuojant ir realizuojant pacituoti bent 5 moksliniai šaltiniai | Projektuojant ir realizuojant pacituoti bent 8 moksliniai šaltiniai | Projektuojant ir realizuojant pacituoti bent 10 mokslinių šaltinių |

# Bibliografija

Abba, I. V. (2022 m. Spalis 21 d.). *What is an ORM – The Meaning of Object Relational Mapping Database Tools*. Nuskaityta iš Free Code Camp: https://www.freecodecamp.org/news/what-is-an-orm-the-meaning-of-object-relational-mapping-database-tools/

Abdulazeez Mousa, M. K. (2020 m. birželis 21 d.). *IEEEXplore*. Nuskaityta iš Database Security Threats and Challenges: https://ieeexplore.ieee.org/document/9116436

Fitzgerald, A. (2022 m. Gegužė 30 d.). *Tailwind CSS: What It Is, Why Use It & Examples*. Nuskaityta iš HubSpot: https://blog.hubspot.com/website/what-is-tailwind-css

Juviler, J. (2022 m. Spalis 31 d.). *What Is GitHub? (And What Is It Used For?)*. Nuskaityta iš hubSpot: https://blog.hubspot.com/website/what-is-github-used-for

Malatesta, F. (2015). *Learning Laravel`s Eloquent: Develop amazing data-based applications with Eloquent, the Laravel framework ORM.* Packt Publishing.

n.d. (2022). *Laravel*. Nuskaityta iš Migration & Seeding: https://laravel.com/docs/4.2/migrations#:~:text=Migrations%20are%20a%20type%20of,easily%20manage%20your%20application's%20schema.

Nawaz, S. (2023 m. Vasaris 22 d.). *Learn how to Debug PHP with Xdebug and Vscode*. Nuskaityta iš CloudWays: https://www.cloudways.com/blog/php-debug/#:~:text=Xdebug%20uses%20the%20DBGp%20debugging,otherwise%20lost%20in%20code%20debugging.

Nixon, R. (2012). *Learning PHP, MySQL, JavaScript, and CSS: A Step-by-Step Guide to Creating Dynamic Websites.* O'Reilly Media.

Tsitoara, M. (2020). *Beginning Git and GitHub: A Comprehensive Guide to Version Control, Project Management, and Teamwork for the New Developer.* Apress.

Urlich, A., & Feloidea, M. (2023 m. Sausis 30 d.). *ClockWork*. Nuskaityta iš madeWithLaravel: https://madewithlaravel.com/clockwork#:~:text=Clockwork%20gives%20you%20an%20insight,first%2Dclass%20Laravel%20support.%22

Washizaki, H. F. (2009). Detecting Design Patterns Using Source Code of Before Applying Design Patterns. *International Conference on Computer and Information Science* (p. 933-938). Tokyo: National Institute of Infomatics.