**PAMOKA PAGAL IT PROGER** (Youtube): [https://www.youtube.com/watch?v=0Be0fX9wbXc](https://www.youtube.com/watch?v=0Be0fX9wbXc&t=690s)

*Giliau į Laravel failų struktūrą: package.json - dependencies frontendui o composer.json - backendui (serverinei daliai, plug-inai ir t.t.). package.json reikalingi dependencies instaliuojami npm -i komanda (windows.Ir dar reikia node.js susivesti). Na o Linux aplinkoje komandos tokios (instaliavau į Ubuntu aplinką - Win wsl2.):*

sudo apt-get install -y nodejs #suvedama node.js

sudo apt-get install curl #jei kartais nesuvestas curl

curl -o- https://raw.githubusercontent.com/nvm-sh/nvm/v0.39.1/install.sh | bash #šita komanda nesuveikė

sudo apt install npm #o čia jau susiveda priklausomybės frontend daliai

*phpunit.xml - projekto unit testavimui, o webpack.mix.js - nurodo kokiu būtent būdu bus perduoti/verčiami js ir css failai. (Tarkim eilutė .saas('resources/sass/app.scss','public/css'); reiškia, jog css failams naudosime preprocesorių saas o vėliau viskas automatitku būdu bus paverčiama įprastu css failu)*

1. web.php pridedame Routes: Route: :get('/about', function () { return view('about');}); ir pakeičiame pagrindinio šablono "/" į Route::get('/', function () { return view('home');});

2. *tarkim darysime tokį url adresą, kurio struktūra sekanti: /user/2/John. Tai yra vartotojai/vartotojoID/VartotojoVardas - kurie yra* ***kintami/dinaminiai****. Tada web.php įrašome* **Route: :get**('/user/{id}/{name}', function ($id, $name) {return 'ID: '.$id.'. Name: '.$name;}); *dinaminis kintamasis žymimas {}, pagal tai išvedama ID: 2. Name: John . Dinaminių kint. pagalba galima gauti duomenis iš url adresų.*

3. Ištriname welcome.blade.php - sukuriame home.blade.php . Standartinis šablonas su ! ir Tab. Pridedame <h1>Labas</h1>. Pajungiame bootstrap per link:

*<link href="https://cdn.jsdelivr.net/npm/bootstrap@5.2.0/dist/css/bootstrap.min.css" rel="stylesheet" integrity="sha384-gH2yIJqKdNHPEq0n4Mqa/HGKIhSkIHeL5AyhkYV8i59U5AR6csBvApHHNl/vI1Bx" crossorigin="anonymous">*

*Šito neįrašau, nes js nenaudojama <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.2.0/dist/js/bootstrap.bundle.min.js" integrity="sha384-A3rJD856KowSb7dwlZdYEkO39Gagi7vIsF0jrRAoQmDKKtQBHUuLZ9AsSv4jD4Xa" crossorigin="anonymous"></script>*

4. <https://getbootstrap.com/docs/5.2/examples/pricing/> . F12. Pasirenkam selector. Paspaudžiame header. Copy Element. Įklijuojame į body dalį ir paliekame kelis html elementus, pridedame mygtuką, pakeičiame href'uose "#" į "/" "/about" "/review". Pakeičiame foną ir teksto spalvą. Turime:

<!doctype html>  
<html lang="lt">  
<head>  
 <meta charset="UTF-8">  
 <meta name="viewport"  
 content="width=device-width, user-scalable=no, initial-scale=1.0, maximum-scale=1.0, minimum-scale=1.0">  
 <meta http-equiv="X-UA-Compatible" content="ie=edge">  
 <title>Main page</title>  
 <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.2.0/dist/css/bootstrap.min.css" rel="stylesheet" integrity="sha384-gH2yIJqKdNHPEq0n4Mqa/HGKIhSkIHeL5AyhkYV8i59U5AR6csBvApHHNl/vI1Bx" crossorigin="anonymous">  
</head>  
<body class="bg-dark">  
<header>  
 <div class="container-fluid py-3">  
 <div class="d-flex flex-column flex-md-row align-items-center pb-3 mb-4 border-bottom bg-dark">

<span class="fs-4">Mokausi laravel iš ITProger https://youtu.be/0Be0fX9wbXc</span>  
 <nav class="d-inline-flex mt-2 mt-md-0 ms-md-auto">  
 <a class="me-3 py-2 text-white text-decoration-none" href="/">Pagrindinis</a>  
 <a class="me-3 py-2 text-white text-decoration-none" href="/about">Apie mus</a>  
 </nav>  
 <a class="btn btn-outline-primary" href="/review">Atsiliepimai</a>  
 </div>  
 </div>  
</header>  
</body>  
</html>

5. Sukuriame layout.blade.php į kurį iškerpame visą html kodą iš home.blade.php . Šitas kodas kartosis kiekviename puslapyje ir papildomai reikia nurodyti vietą, kuri nesikartos kiekviename puslapyje. Todėl virš </body> įrašome eilutę @yield('main\_content') - ir įkišame į <div class="container"> **į tą vietą galėsime įrašyti kitų failų turinį, kurie paveldės šį html šabloną**. *Pavadinimą galime sugalvoti kokį tik norime, svarbiausia, kad jis nesikartotų tame pačiame faile.* O į tuščią home.blade.php įrašome eilutę @extends('layout') - **t.y. paveldime viską iš failo layout.blade.php**. ***Jei failai yra toje pačioje direktorijoje, nereikia nurodyti išsamesnio kelio kaip 'subdir/layout' ar panašiai****.* Toliau ***reikia nurodyti ką reikės įterpti į layout.blade.php šablono main\_content dalį, kurią iškviečia yield('main\_content')***. Tam žemiau kreipiamės į @section('main\_content'), nurodydami, kad būtent dirbsime su 'main\_content' ir reikia uždaryti sekciją su @endsection . O į sekcijos vidų įdedame <p>lorem30+TAB</p>.

{{--home.blade.php--}}  
@extends('layout')   
@section('main\_content')  
<p>Lorem ipsum dolor sit amet, consectetur adipisicing elit. Numquam quis quo ullam vitae voluptas. Consectetur iusto molestiae nam optio provident! Atque deserunt ducimus harum ipsum porro quasi reiciendis sint temporibus.</p>  
@endsection

6. html kodą iš home.blade.php nukopijuojame į about.blade.php ir dadedame *<h1>Puslapis apie mus</h1>*. Tada jei paspausti pagrindiniame puslapyje ant nuorodos headeryje Apie mus - parodo about.blade.php turinį.

{{--about.blade.php--}}  
@extends('layout')  
@section('main\_content')  
 <h1>Puslapis apie mus</h1>  
 <p>Lorem ipsum dolor sit amet, consectetur adipisicing elit. Numquam quis quo ullam vitae voluptas. Consectetur iusto molestiae nam optio provident! Atque deserunt ducimus harum ipsum porro quasi reiciendis sint temporibus.</p>  
@endsection

7. *Bet nuėjus į kitas nuorodas* ***title*** *išlieka tas pats - koks nurodytas šablone*. Tam, kad **to nebūtų** ir **title keistusi**, reikia layout.blade.php <title> tage nurodyti dar vieną dinaminę sekciją, kuri keistusi pagal paduodamą puslapį. T.y. įrašome: <title>@yield('title')</title> , kur 'title' - bet koks pavadinimas tik ne 'main\_content', kuris šitame html jau yra. Esmė - **kartotis** tame pačiame puslapyje **negali**.

{{--layout.blade.php--}}  
<!doctype html>  
<html lang="lt">  
<head>  
 <meta charset="UTF-8">  
 <meta name="viewport"  
 content="width=device-width, user-scalable=no, initial-scale=1.0, maximum-scale=1.0, minimum-scale=1.0">  
 <meta http-equiv="X-UA-Compatible" content="ie=edge">  
 <title>@yield('title')</title>  
 <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.2.0/dist/css/bootstrap.min.css" rel="stylesheet" integrity="sha384-gH2yIJqKdNHPEq0n4Mqa/HGKIhSkIHeL5AyhkYV8i59U5AR6csBvApHHNl/vI1Bx" crossorigin="anonymous">  
</head>  
<body class="bg-dark text-white">  
<header>  
 <div class="container-fluid py-3">  
 <div class="d-flex flex-column flex-md-row align-items-center pb-3 mb-4 border-bottom bg-dark">  
 <span class="fs-4">Mokausi laravel iš ITProger https://youtu.be/0Be0fX9wbXc</span>  
 <nav class="d-inline-flex mt-2 mt-md-0 ms-md-auto">  
 <a class="me-3 py-2 text-white text-decoration-none" href="/">Pagrindinis</a>  
 <a class="me-3 py-2 text-white text-decoration-none" href="/about">Apie mus</a>  
 </nav>  
 <a class="btn btn-outline-primary btn-warning text-black" href="/review">Atsiliepimai</a>  
 </div>  
 </div>  
</header>  
<div class="container">  
 @yield('main\_content')  
</div>  
</body>  
</html>

8. Įrašome home.blade.php po @extends @section('title')Pagrindinis puslapis@endsection o about.blade.php - @section('title')Apie mus@endsection. *Atskirose eilutėse įrašiau, nes kitaip nesuveikdavo kodas - komanda būdavo atvaizduota html'e o ne title pavadinime.*

{{--home.blade.php--}}  
@extends('layout')  
  
@section('title')  
Pagrindinis puslapis  
@endsection  
  
@section('main\_content')  
<p>Lorem ipsum dolor sit amet, consectetur adipisicing elit. Numquam quis quo ullam vitae voluptas. Consectetur iusto molestiae nam optio provident! Atque deserunt ducimus harum ipsum porro quasi reiciendis sint temporibus.</p>  
@endsection

{{--about.blade.php--}}  
@extends('layout')  
  
@section('title')  
Apie mus  
@endsection  
  
@section('main\_content')  
 <h1>Puslapis apie mus</h1>  
 <p>Lorem ipsum dolor sit amet, consectetur adipisicing elit. Numquam quis quo ullam vitae voluptas. Consectetur iusto molestiae nam optio provident! Atque deserunt ducimus harum ipsum porro quasi reiciendis sint temporibus.</p>  
@endsection

{--layout.blade.php--}}  
<!doctype html>  
<html lang="lt">  
<head>  
 <meta charset="UTF-8">  
 <meta name="viewport"  
 content="width=device-width, user-scalable=no, initial-scale=1.0, maximum-scale=1.0, minimum-scale=1.0">  
 <meta http-equiv="X-UA-Compatible" content="ie=edge">  
 <title>@yield('title')</title>  
 <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.2.0/dist/css/bootstrap.min.css" rel="stylesheet" integrity="sha384-gH2yIJqKdNHPEq0n4Mqa/HGKIhSkIHeL5AyhkYV8i59U5AR6csBvApHHNl/vI1Bx" crossorigin="anonymous">  
</head>  
<body class="bg-dark text-white">  
<header>  
 <div class="container-fluid py-3">  
 <div class="d-flex flex-column flex-md-row align-items-center pb-3 mb-4 border-bottom bg-dark">  
 <span class="fs-4">Mokausi laravel iš ITProger https://youtu.be/0Be0fX9wbXc</span>  
 <nav class="d-inline-flex mt-2 mt-md-0 ms-md-auto">  
 <a class="me-3 py-2 text-white text-decoration-none" href="/">Pagrindinis</a>  
 <a class="me-3 py-2 text-white text-decoration-none" href="/about">Apie mus</a>  
 </nav>  
 <a class="btn btn-outline-primary btn-warning text-black" href="/review">Atsiliepimai</a>  
 </div>  
 </div>  
</header>  
<div class="container">  
 @yield('main\_content')  
</div>  
</body>  
</html>

9. web.php - web routes failo, nereikėtų apsunkinti bereikalingu kodu - tam yra skirtas kontroleris, kuriame implementuojamas funkcionalumas ir apdorojami duomenys, o **tik paskui išvedamas pats šablonas**. *Kontrolerių galima susikurti kiek reikalinga. Tarkim jeigu yra vartotojų registracija, autorizacija ir pasijungimas į saitą tai jam galima (ir reiktų) sukurti atskirą Controller. Taigi,* ***kursime naują Controller*** *pagal komandą* **php artisan make:controller** *Main*Controller - ***BŪTINAI*** *Controller gale, o* ***pavadinimas - iš didžiosios ir vienaskaita****. Galima rašyti ir rankomis, bet daug rašliavos.* Jame parašome metodus(functions):

<?php  
namespace App\Http\Controllers**;**use Illuminate\Http\Request**;**class MainController extends Controller  
{

public function home(){return view('home')**;**}  
public function about(){return view('about')**;**}  
}

10. išvedinėti views ne per web.php, bet per Controller galima sekančiai (ŠITAS VARIANTAS NESUVEIKĖ,

*web.php pakeičiame:* Route::get('/', function () { return view('home');}); į **Route: :get**('/','MainController@home'); *kai kreipiamasi į /, ~~tai MainController suveikia (action, kurį nurodo @) funkcija home.~~ Tam tikslui Main Controller įrašome f-ją:* **public function** home() {**return** **view**('home')}; Atitinkamai: web.php: ~~Route:: get ('/about','MainController@about')~~; o MainController - public function **about**() {return view( view: 'about'); DĖL TO PASINAUDOJAU DĖSTYTOJO VARIANTU: Route**::**get('/',[MainController::**class**, 'home']); IR Route::get('/about', [MainController::**class**, 'about']);

ir **BŪTINAI**: use App\Http\Controllers\MainController; viršuje.

11. **Kuriame atsiliepimų puslapį**. Pridedame web: Route::get('/review', [MainController::class, 'review']); MainController.php: public function review(){return view('review');}

12. **Sukuriame** review.blade.php. **Nukopijuojame** kodą iš home.blade.php, kurio išvaizda prieš tai buvo truputį pakeista. Paliekame tik @extends('layout') @section('title')**Atsiliepimai**@endsection @section('main\_content') čia įrašome formą, atkreipti dėmesį : **method="post"** - perdavimo metodas o **action="/review/check"** nurodo kokiame url adrese bus apdorojami formos duomenys. Formoje įrašome kelis įvedimo laukus, kuriuose **būtini** atributai - name ir id, tokiomis pačiomis reikšmėmis, placeholder - tiesiog informacija vartotojui ką įvesti o class="form-control" tiesiog bootstrap stilius, gražesniai išvaizdai. **Būtinas button -** TIK JAM NEVARTOJAMAS atributas "name" o rašosi **type="submit"** - o tai reiškia, jog paspaudus mygtuką - puslapis persikrauna o duomenys - išsiunčiami į serverį. **Neįrašius** **@csrf** formoje - duomenys nebus išsiūsti, klaida 404 nes ši funkcija generuoja token o be jo laravelis nesiunčia duomenų dėl saugumo.

{{--review.blade.php--}}  
@extends('layout')  
  
@section('title')  
Atsiliepimai  
@endsection  
  
@section('main\_content')  
 <h2>Atsiliepimo pridėjimo forma</h2>  
 <form method="post" action="/review/check">

@csrf  
 <input type="email" name="email" id="email" placeholder="Įveskite el.pašto adresą" class="form-control mb-2">  
 <input type="text" name="subject" id="subject" placeholder="Parašykite atsiliepimą" class="form-control mb-2">  
 <textarea name="message" id="message" class="form-control mb-2" placeholder="Įveskite žinutę"></textarea><br>  
 <button type="submit" class="btn btn-success mb-2">Išsiūsti</button>  
 </form>  
@endsection

13. web pridedame maršrutą - Route::**post**('/review/check', [MainController::**class**, 'review\_check']); atkreipti dėmesį - get - tiesiog nukreipiame į kažkokį url o post - gauname duomenis.

14. MainController - įrašome naują metodą: public function **review\_check**(Request **$request**){dd(**$request**);} - metodas pavadinimu **review-check** laukia duomenų (Tai nurodo **Request**) į kintamąjį pavadinimu *request* ir **dd** (dump and die) funkcija juos išveda bei sustabdo tolesnį programos vykdymą. Duomenis, kuriuos suvedame į formą galima rasti dd išvedimo išskleidžiamoje request skiltyje. Tai labai padeda analizei ir klaidų nustatymui. Šiuo atveju ateina kaip associative array.

15. Prieš pradėdami dirbti su duom.bazėm, nustatome duomenų tikrinimą-**VALIDACIJĄ**. Tiesiog jei vartotojas įves neatitinkančius nustatytų validacijos sąlygas duomenis - bus išvedami pranešimai, tarkim neįvestas email ir kt. **Validacijos sąlygos nustatomos Controller**. Tam pakeičiame metodą MainController **review\_check** ir išdėstome jį taip:

public function review\_check(Request $request){  
 $valid = $request->validate([  
 'email' => 'required|min:4|max:100'**,** 'subject' => 'required|min:4|max:100'**,** 'message' => 'required|min:15|max:500'**,**])**;**}

kur sukuriame kintamąjį $valid ir jam priskiriame reišmes gautas sekančiai: kreipiamės į objektą $request, **kuriame yra** funkcija vadinama **validate**. Šios funkcijos pagalba galime nurodyti kokius laukus ir kokius sąlygų tikrinimus mes jai nustatome. Tai padarome sukurdami validate funkcijoje asociatyvų masyvą, kuriame nurodome **raktą/key** - tai **lauko iš formos name** ir(arba?) **id** pavadinimas o reikšmes priskiriame pagal validacijos sąlygas: required - būtinas(negali būti tuščias), min:4 - mažiausiai 4 simboliai, max:100 - daugiausiai 100 simbolių. Kelios reikšmės atskiriamos "|" simboliu.

16. Tada review.blade.php **realizuojame informacijos apie formos validacijos neatitikimus išvedimą** įrašydami sekantį kodą:

@if($errors->any())  
 <div class="alert alert-danger">  
 <ul>  
 @foreach($errors->all() as $error)  
 <li>{{$error}}</li>  
 @endforeach  
 </ul>  
 </div>  
@endif

Per sąlygos operatorių @if kreipiamės į objektą $errors, tikrindami ar jis nėra tuščias any() funkcijos pagalba. Jei taip - vykdomas kodo blokas kuris įdėtas į **<div>** su pritaikyta *alert alert-danger bootstrap klase* **išvaizdai**, kuriame **ul** - nenumeruotame sąraše visos klaidos iš $errors po vieną @foreach pagalba yra išvedamos vartotojui virš formos. *{{kintamasis}} veikia kaip echo.* Išbandau neužpildęs jokių laukų. Klaidas meta - validacija veikia.

17. SQL Serverio pagal esamą pamoką mėginu nesuvesti o naudotis integruotais įrankiais, pagal Dėstytojo pamoką: env faile nurodome kuri db bus naudojama. Tam jame pakeičiame reikšmes DB\_DATABASE=laravel į mysql , DB\_HOST=127.0.0.1 į db DB\_USERNAME=root į db , DB\_PASSWORD= įrašome db. JEIGU DAROMI KOKIE NORS SVARBŪS PAKEITIMAI, KAIP ŠIS AR APSKRITAI - REIKIA IŠVALYTI CACHE. Tai daroma komanda: **php artisan config:cache** . *man išveda ne Configuration cache cleared! Configuration cached successfully! o tik Configuration cached successfully! Kažin kodėl? Yra kitų komandų, bet kol kas nenaudoju. Jau panaudojau:* php artisan cache:clear php artisan config:clear php artisan config:cache .

18. Kad manipuliuoti db duomenimis CRUD reikalingas modelis. Dar reikalingas migracijos failas. Skirtumas tas, kad migracijos faile įrašomi laukai, kurie bus table. Imant dabartinę pamoką - migracijos faile reikės aprašyti table ir laukus iš vartotojo suvestos formos - email, subject, message. O modelio faile bus galima dirbti jau su pačiomis laukų,table reikšmėmis. Kuriame abu: **php artisan make:model ContactModel -m** , kur ContactModel - pasirinktas modelio pavadinimas (Model gale nebūtina rašyti, bet paskui paprasčiau tiesiog skaityti savo kodą - aiškumo principas), o -m raktas gale komandos reiškia, kad tuo pačiu bus sukurtas migracijos failas. Atsiranda du failai: App\Models\ContactModel.php ir database\migrations\2020\_09\_01\_053734\_create\_contact\_models\_table.php , kur data ir laikas yra migration file timestamp*. Pagal nutylėjimą laravel karkase jau yra modelis User.php ir trys migracijos failai: user - darbui su vartotojais, password - slaptažodžių atsatymui ir failed jobs - visų buvusių migracijų aprašymui (?).*

19. **DARBAS SU MIGRACIJOS FAILU**: jame yra tik du metodai **up()** ir **down()** . **up()** - kviečiamas kai migracijos failo pagrindu norime sukurti konkrečią lentelę. Tame metode aprašomas lentelės pavadinimas ir nustatomi lentelės laukai. Taigi, redaguojame eilutę sekančiai: Schema::create('contact\_models', function (Blueprint $table) lentelės laukas - id() unikalus autoincrement įrašo numeris - reikalingas, o lentelės laukas timestamps(); - saugo lauko sukūrimo datą/laiką (jei nereikalingas - galime ištrinti). Įrašome reikalingus laukus iš formos - email,subject - string tipo (iki 255 simbolių) o message nurodome text tipą (simbolių daugiau). Na o down() metodas reikalingas - kai norime pašalinti kokią nors lentelę migracijos failo pagrindu, taigi toje vietoje pakeičiame ir įrašome eilutę taip: Schema::dropIfExists('contacts'); o tai reiškia, kad jeigu tokia lentelė egzistuoja *IfExists*, tai iškviesdami šitą metodą mes ją galime pašalinti *drop*. Metodai up ir down:

public function up()  
{  
 Schema::create('contact\_models'**,** function (Blueprint $table) {  
 $table->id()**;** $table->string('email')**;** $table->string('subject')**;** $table->text('message')**;** $table->timestamps()**;** })**;**}

public function down()  
 {  
 Schema::dropIfExists('contact\_models')**;** }  
}**;**

Po šitų manipuliacijų duomenų bazėje fiziškai niekas nepasikeičia. Tam, kad atsirastų reikalinga db, reikia įvykdyti komandą **php artisan migrate** . **LABAI SVARBU, KAD LAUKŲ PAVADINIMAI MIGRACIJOS FAILE NESIDUBLIUOTŲ, NES VAL. GALIMA UŽTRUKTI IEŠKANT!**

Po migracijos db atsiranda daugiau lentelių. Jos yra kuriamos laravel pagal nutylėjimą ir trinti jų nereikia.

pasitikriname ar po migracijos duomenų bazėje db atsirado lentelė contacts ir laukai email, subject, message komandomis:

ddev mysql; show databases; (sąraše turi būti db) use db; show tables (sąraše turi būti contacts); show columns from contacts from db; (sąraše turi būti visi laukai) exit; *Jei negalima atlikti veiksmų - permission denied, tada - įvairūs sprendimo būdai. Naudojau* (dar viena komanda: **ddev artisan migrate:refresh** - kai reikia atnaujinti migracijos failą. Po šitos komandos dingsta duomenys ir juos reikia suvesti iš naujo) O su vartotojais ir jų pasijungimais įdomios linux komandos: **ddev mysql -u root -p** (pasijungiama prie db root teisėmis -p nurodo suvesti password: pas mane root)/ **select user from mysql.user;** (parodo user lentelę)/ **select user();** (parodo lentelėje pasijungusį user)/ **drop user 'db'@'localhost';** (išmesti user iš priėjimo prie db)/ **flush privileges;** (šita komanda reikalinga po kokių nors svarbių pakeitimų su user account)/ **CREATE USER 'db'@'localhost';** (per naują sukuriame vartotoją)/ **grant all on \*.\* to 'db'@'localhost'** with grant option; (nustatoma, kad vartotojas db turės visišką priėjimą prie duombazių, priėjimo lygį galima pakeisti)/ **flush privileges;** (dar kartą, pakeitimų įtvirtinimui)/ **SHOW GRANTS FOR db@localhost;** (parodo priėjimo prie db lygį atskiram vartotojui db@localhost;). Tada pasijungiu kaip esamas user o ne root per ddev mysql -p (password tiesiog enter) ir tikrinuosi select current\_user(); / show databases; / use db; / show tables; / show columns from contact\_models from db;

20. MainController **atkreipti dėmesį, kad turėjo atsirasti use App\Models\ContactModel; ir jei nėra tai reikia įrašyti šią eilutę.** Jau praėjome validaciją, taigi gausime duomenis ir įrašysime juos į db. Tam po validacija įrašome sekantį bloką, kuriame susikuriame objektą pasirenkamu pavadinimu - renkamės $review ir jam išskiriame atmintį, tai yra kreipiamės į ContactModel ir priskiriame formos lauko reikšmę konkrečiam db laukui. id perduoti nereikia, nes jis yra **autoincrement** - automatiškai didėjanti dešimtainė reikšmė. O kitiems laukams tiesiog per objektą $review perduodame db lauko name, tada šitam laukui kreipiamės į objektą $request, tai yra duomenis, gautus iš vartotojo užpildytos formos ir panaudojame metodą input('email'), kuris pagal nustatytą raktą ('email' lauką vartotojo užpildytoje html formoje) leidžia gauti reikšmę. $review->email = $request->input('email'); Taip pat parašome ir subject ir message. Po viso to kreipiamės į objektą $review ir panaudojame metodą save(), kuris išsaugo įrašą į db.contacts eilute $review->save(); . Po viso to veiksmo vartotoją peradresuojame kur norime, bet šiuo atveju tiesiog gražiname į puslapį /review darašydami eilutę: return redirect()->route('/review');

21. Po viso to **gauname klaidas**, kurias pasitaisome sekančiai: 1) jei nebūtumėme parašę migracijos faile timestamp lauko (bet jis paliktas), reikėtų modelyje darašyti kode eilutę class ContactModel extends Model{ public $timestamps = false;use HasFactory;} 2) negalėjome panaudoti return redirect()->route('/review'); MainControleryje, nes pateiktas '/review' šiuo atveju turi būti ne url adresas bet Route vardas, kurio mes web.php maršrutui nepriskyrėme, tai ištaisome taip: web.php: **Route::get('/review', [MainController::class, 'review'])**->name('review'); - priskiriame maršrutui vardą. Jis gali būti bet koks, bet šis kaip ir logiškas ir tada MainControler jau galima nurodyti priskirtą vardą: **return redirect()->route(**'review'**);**

22. Norint pasitikrinti visą db susikurtą lentelę galime įrašyti MainController metodo review pradžioje $reviews = new ContactModel(); dd($reviews->all()); t.y. ContactModel() pagrindu sukuriame objektą $reviews bei dd funkcijos pagrindu kreipiamės į ši objektą panaudodami metodą all(). Ir atnaujinus /review puslapį naršyklėje gauname dd išvedamą lentelę, kur išskleidę items, susirandame original skiltį - o ten yra mūsų įvestas įrašas ir taip kiekvienai db table eilutei. Jei įrašome šitą kodą iš karto po metodo tai nereikia užkomentuoti sekančio kodo - dd, skirtingai nuo analogiškos dump f-jos sustabdo kodo vykdymą.

23. Pasitikrinę per dd **MĖGINAME IŠVESTI DUOMENIS Į ŠABLONĄ**. **MainController** užkomentuojame dd eilutę ir **review** metode įrašome: return view('review', **['reviews' => $reviews->all()]**); t.y. į šabloną **'review'** paduodame asociatyvų masyvą kaip antrą parametrą - kuriame nurodome, kokias reikšmes, objektus, elementus, t.t. siūsime į šabloną: šiuo atveju siunčiame savo susikurtu pavadinimu raktą **'reviews'** į kurį bus galima kreiptis pačiame šablone, o jo reikšmė **'$reviews'** (prieš tai eilutė - $reviews = new Contact() modelis - sukuriamas objektas) **->all()**, nurodo jam priskirti visas reikšmes iš duomenų bazės lentelės.

24. review.blade.php po </form> įrašome <h2>Visi atsiliepimai</h2> ir ciklą: **@foreach($reviews as $el)** , kur **$el** savo sugalvotas kintamasis o **$reviews** yra priskirtas raktas **'reviews'** iš ankstesnio punkto ir **<div class= "alert alert-warning"> <h3>{{$el->subject}} </h3> </div> @endforeach**

**Turėtų gautis taip:**

|  |  |
| --- | --- |
|  |  |
|  |