
2.2. laboratorinis darbas. *Laravel* karkaso įvadas

I. Aprašymas

Kuriant interneto informacijos sistemas neapsieinama be universalių programinių funkcijų (vartotojų valdymas, įvedamų duomenų tikrinimas, duomenų bazės užklausų vykdymas ir kt.). Jų kūrimas užima pakankamai daug laiko, todėl sistemų kūrimui dažnai naudojami karkasai, kurie turi dažniausiai naudojamų universalių funkcijų bibliotekas.

Kitas karkasų privalumas – architektūra. Nuo to, kokie komponentai sudaro sistemą, kaip jie siejasi tarpusavyje, priklauso sistemos veikimo savybės. Bloga architektūra gali lemti didelius nuostolius, kuomet tenka brangiai palaikyti ar net perkurti prastai suprojektuotą sistemą. Karkasai turi gerai suprojektuotą ir išbandytą architektūrą – tai padeda sistemų kūrėjams išvengti klaidų.

Antroji šio laboratorinio darbo dalis skirta sukonfigūruoti *Laravel* karkaso projektą darbui žiniatinklio serveryje, suprojektuoti duomenų bazės schemą bei susipažinti su *Laravel*.

II. Darbo tikslas

Susipažinimas su karkaso diegimu ir programavimu panaudojant *Laravel* karkasą.

III. *MySQL* aprašymas

MySQL duomenų bazės kūrimui bei administravimui dažniausiai naudojamas per naršyklę veikiantis įrankis *phpMyAdmin*. Jis leidžia atlikti visus reikalingiausius administravimo veiksmus – kurti bei redaguoti duomenų bazines, lenteles, keisti lentelių struktūrą, importuoti/eksportuoti duomenis, nustatyti ryšius tarp lentelių ir kt.

Kuriant duomenų bazę reikia žinoti dažniausiai naudojamus *MySQL* duomenų tipus:

- INT – sveikiesiems skaičiams
- FLOAT – slankaus kablelio skaitinėms reikšmėms
- DECIMAL – fiksuoto kablelio skaitinėms reikšmėms. Naudojama tuomet, kai svarbus išsaugoti reikšmių tikslumą (pvz. saugant prekių kainas).
- VARCHAR – fiksuoto ilgio tekstiniams įrašams (pavadinimams, vardams ir pan.)
- TEXT – ilgiems tekstams (straipsniams, komentarams ir pan.)
- DATE – datai
- DATETIME – datai ir laikui
- BLOB – dvejetaini informacijai (failams)

MySQL vartotojo vadovas pateikiamas šiuo adresu:

<http://dev.mysql.com/doc/>

IV. *Laravel* karkaso diegimas

Detali informacija apie karkaso diegimą pasiekama faile **2.1_Laravel_karkaso_diegimas.docx**. Rekomenduojama detaliai išnagrinėti šio failo turinį prieš vykdant *Laravel* konfigūravimą.

Kuriant ir naudojantis Laravel karkasu patogiausias būdas gauti reikiamą informaciją yra naudotis oficialia karkaso dokumentacija, kuri yra pasiekama adresu:

<https://laravel.com/docs/master>

V. Laravel konfigūravimo paaiškinimai

Laravel karkasas paremtas *MVC (model, view, controller)* architektūra. Ši architektūra leidžia atskirti tris pagrindines sistemos dalis (modeliai, vaizdai ir valdikliai), atsakingas už duomenų valdymą, jų pateikimą vartotojui bei veiklos logiką.

1. Konfigūravimo failas

Visi pagrindiniai *Laravel* projekto nustatymai yra saugomi *.env* faile. Taip saugant nustatymus galima lengvai keisti nustatymus pritaikant juos skirtingoms aplinkoms.

```
/.env
```

Šiame faile saugomi visi pagrindiniai karkaso nustatymai:

- Aplikacijos nustatymai
- Duomenų bazės nustatymai
- El. pašto ir eilių variklių nustatymai
- Ir kiti.

Pavyzdinis failas gali atrodyti taip:

```
APP_NAME=Laravel
APP_ENV=local
APP_KEY=base64:fpV7IXR5F4xyBse4zxB6rG/g7p/naLqnjdSgrlkbGzE=
APP_DEBUG=true
APP_URL=http://localhost

LOG_CHANNEL=stack

DB_CONNECTION=mysql
DB_HOST=127.0.0.1
DB_PORT=3306
DB_DATABASE=homestead
DB_USERNAME=homestead
DB_PASSWORD=secret

BROADCAST_DRIVER=log
CACHE_DRIVER=file
QUEUE_CONNECTION=sync
```

```
SESSION_DRIVER=file
SESSION_LIFETIME=120

REDIS_HOST=127.0.0.1
REDIS_PASSWORD=null
REDIS_PORT=6379

MAIL_DRIVER=smtp
MAIL_HOST=smtp.mailtrap.io
MAIL_PORT=2525
MAIL_USERNAME=null
MAIL_PASSWORD=null
MAIL_ENCRYPTION=null
.....
```

Čia galima įrašyti ir kitus reikalingus nustatymus, bei paskui nesunkiai juos pasiekti iš valdiklio, modelio ar vaizdo. Pavyzdžiui vaizde aplikacijos pavadinimas pasiekiamas tokiu būdu:

```
<title>{{ config('app.name', 'Naujienų Svetainė') }}</title>
```

Konfigūruojant savo svetainę šiame nustatymų faile reikia nustatyti:

1. Aplikacijos pavadinimą (APP_NAME)
2. Aplikacijos URL adresą (APP_URL)
3. Duomenų bazės informaciją (DB_CONNECTION ir kt.)

3. Modelis

Modeliai – *PHP* klasės, skirtos dirbti su informacija, saugojama duomenų bazėje. Jose aprašomos funkcijos, skirtos išrinkti, įrašyti, atnaujinti ar pašalinti duomenis iš duomenų bazės. Modeliai saugojami kataloge:

```
/app/Models
```

Modeliai lengvai sukuriami naudojant `php artisan` komandą: **`php artisan make:model pavadinimas`**

Modelių failų vardus geriausia pavadinti duomenų bazės lentelės pavadinimu, pvz: *User.php*, *Post.php* ir t.t. Kiekvienas modelis tokiu būdu bus susietas su to paties pavadinimo duomenų bazės lentele, kurios objektus aprašys. Pavadinius modelius skirtingai nei lenteles, (kaip pavyzdžiui naudojant skirtingą kalbą duomenų basei ir karkaso programavimui) kiekvienam modeliui galima nurodyti su kokia duomenų bazės lentele jis turi susisieti pridedant kintamąjį:

```
protected $table = 'naujiena';
```

Ši kodo eilutė nurodo, kad modeliui News.php bus priskirta duomenų bazės lentelė „naujiena“.

Pirmojo laboratorinio darbo metu pateiktos lentelės naujiena atitikmuo modelyje atrodytų taip:

```
<?php

namespace App;

use Illuminate\Database\Eloquent\Model;

class News extends Model
{
    protected $table = 'naujiena';

    protected $fillable = ['pavadinimas', 'santrauka', 'turinys', 'sukurimo_data' ];

    /* nurodome kad nenaudosime created_at ir updated_at laukų šiame modulyje*/
    public $timestamps = false;
}
```

Modelio iškvietimas valdiklyje labai paprastas – įtraukiama modelį į valdiklį su *use* komanda, ir naudojama modelį su standartinėmis modelio funkcijomis.

```
<?php

namespace App\Http\Controllers\News;

use App\News;
use Illuminate\Http\Request;
use App\Http\Controllers\Controller;

class BusinessController extends Controller
{
    public function Index()
    {
        $news= News::all();
        return view('business', compact('news'));
    }
}
```

Viena modelio bei modelio DB klasė skirta vienai duomenų bazės lentelei.

4. Vaizdas

Vaizdas (angl. *view*) yra interneto puslapis ar puslapio fragmentas (puslapio viršutinė ar apatinė dalys, meniu ir pan.). Vaizdą galima įsivaizduoti kaip šabloną, aprašytą *HTML* formatu, kuriam reikalingus atvaizduoti duomenis perduoda valdiklis. Paprastai interneto svetainės turi *HTML* fragmentų, kurie gali būti pakartotinai panaudojami. Todėl *HTML* kodas išskaidomas – kuriamas pagrindinis šablonas, kuris aprašo svetainės struktūrą, bei papildomi šablonai, skirti tam

tikriems puslapio fragmentams atvaizduoti (menu, turinys ir pan.). Šablonai saugojami šiame *Laravel* kataloge:

```
resources/views/layouts/
```

Jei toks katalogas neegzistuoja jį galima sukurti. Daugiau informacijos apie vaizdų kūrimą ir jų įtraukimą į šablonus galima rasti: <https://laravel.com/docs/master/views>

Visi kiti sistemos naudojami vaizdai gali būti sukuriami **resources/views/** kataloge tačiau, plečiantis projektui rekomenduojama juos grupuoti į aplankus pagal posistemius. Pavyzdžiui jei turime daug naujienų skirtingomis kategorijomis būtų patogiau visi vaizdus susijusius su naujienomis saugoti:

```
resources/views/news/
```

Kiekvienas vaizdas *Laravel* karkase naudoja `.blade.php` plėtinį. `.blade.php` nurodo, kodo struktūrą php kodui įtraukti į HTML kodą ir kaip duomenys bus atvaizduojami vartotojams. Šio šablono aprašymą galima rasti <https://laravel.com/docs/master/blade>

Vaizdai niekuomet nėra naudojami tiesiogiai, jie visuomet kviečiami valdiklio (nebent reikia išvesti ne numatytąjį, bet kitą vaizdą). Duomenų perdavimas iš valdiklio į vaizdą yra paprastas ir nesudėtingas. Tarkime `$data` kintamąjį *UserController.php* faile `index()` funkcijoje perduodame į vaizdą:

```
// Failas NewsController.php
...
public function index () {
    // Į vaizdą perduodamas $data kintamasis (jis gali būti bet kokio tipo)
    return ('news', compact('data'));
}
```

Iš valdiklio perduotų duomenų naudojimas vaizde taip pat nesudėtingas. Aukščiau perduotą `$data` kintamąjį *news.blade.php* vaizde galime pasiekti taip:

```
// application/views/scripts/users/index.phtml
<div>
    <h2>naujienų sarašas</h2>
    <p> {{ $data }} </p>
</div>
```

Išvedant duomenis vaizde reikėtų vengti tiesioginio jų išvedimo naudojant *echo*, nes vartotojai gali būti įvedę specialių simbolių, tokių kaip:

- `'&'` (ampersand)
- `''''` (double quote)
- `''''` (single quote)
- `'<'` (less than)
- `'>'` (greater than)

Šiai problemai spręsti rekomenduojama naudoti `.blade.php` šablone pateikiamą informacijos išvedimo trumpinį „`{{}}`“

5. Valdiklis

Valdikliai aprašo bet kokią sistemos veiklos logiką, t.y. veikia kaip tarpininkas tarp modelio ir vaizdo dalių. Kaip pavyzdį galima įsivaizduoti klasę, kurios paskirtis – nuskaityti iš duomenų bazės duomenis, suformuoti ataskaitą bei pateikti ją vartotojui. Valdikliai *Laravel* karkase saugojami šiame kataloge:

```
app/Http/controllers/
```

Sistemoje gali būti daugybė valdiklių. Kurį valdiklį vykdyti *Laravel* karkase nusprendžia į svetainę įvestas adresas

```
http://localhost/LabExample/public/news
```

Šį adresą apdorojęs karkasas ieškos `/news` adreso **web.php** faile. Šį failą galima rasti **/routes**. **web.php** faile surašyti visi pasiekiami URL adresai ir nurodomas, kuris valdiklis yra atsakingas už užklauso apdorojimą. Paprasčiausios GET tipo užklauso pavyzdys galėtų būti toks:

```
Route::get('/', 'HomeController@index')
```

Šis kodas nurodo kad nuėjus adresu <http://localhost/LabExample/public/> bus kreipiamasi į valdiklį pavadinimu **HomeController** ir iškviečiama funkcija **index()**. Karkasas leidžia daug įvairių būtų aprašyti šiuos kelius daugiau informacijos galima rasti: <https://laravel.com/docs/master/routing>

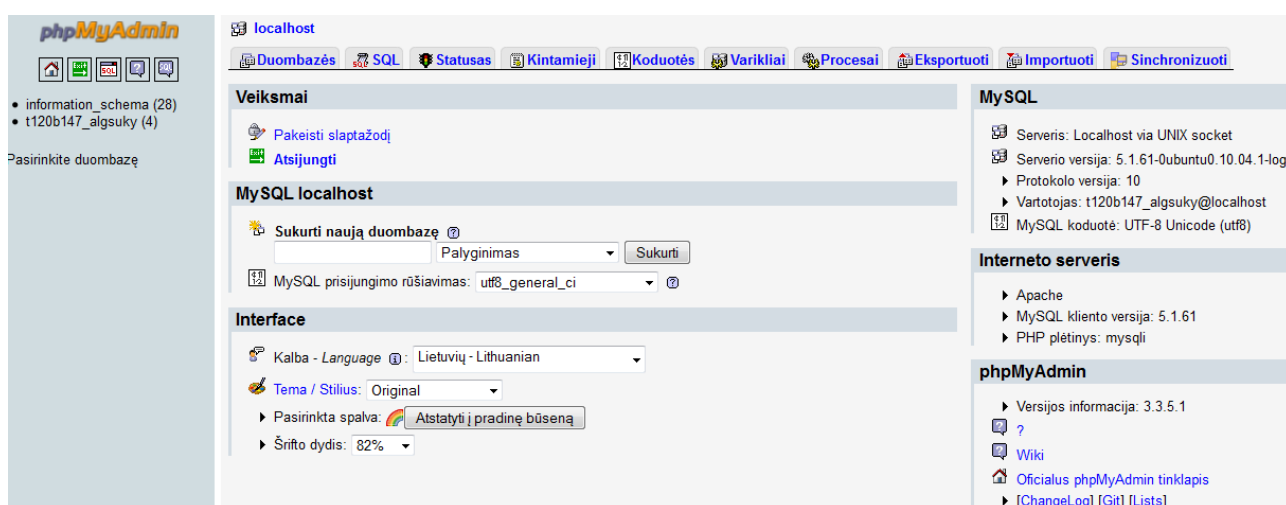
Tame pat **/routes** kataloge galima rasti ir kitų failų, kurie naudojami užklauso ne iš naršyklės kreipinių apdorojimui.

VI. Darbo eiga

1. Duomenų bazės kūrimas

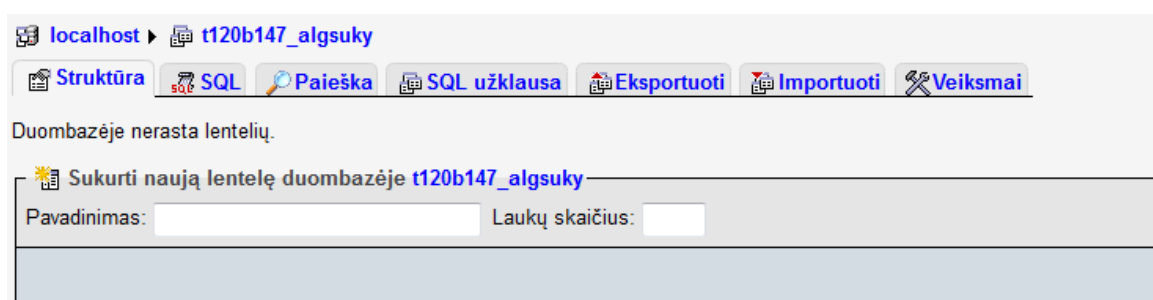
Duomenų bazė turėtų būti sukurta tiesiogiai iš *MySQL workbench* įrankio arba sugeneruotas *SQL* užklausa su *MagicDraw* įrankiu pasinaudojant *forward engineering* funkcionalumu. Nepavykus to padaryti duomenų bazė kūriama „rankiniu“ būdu, kuris aprašomas toliau.

Naudodami *phpMyAdmin* įrankį sukurkite duomenų bazę. Duomenų bazė kuriama pradiname *phpMyAdmin* lange. Palyginimą bei prisijungimo rūšiavimą nustatykite *UTF8_general_ci*. Jei naudojotės katedros serveriu, duomenų bazė jau bus sukurta.



1 pav. *phpMyAdmin* įrankio pradinis langas

Kairiajame *phpMyAdmin* meniu pasirinkite duomenų bazę ir sukurkite bazės lenteles:



2 pav. DB lentelių kūrimas

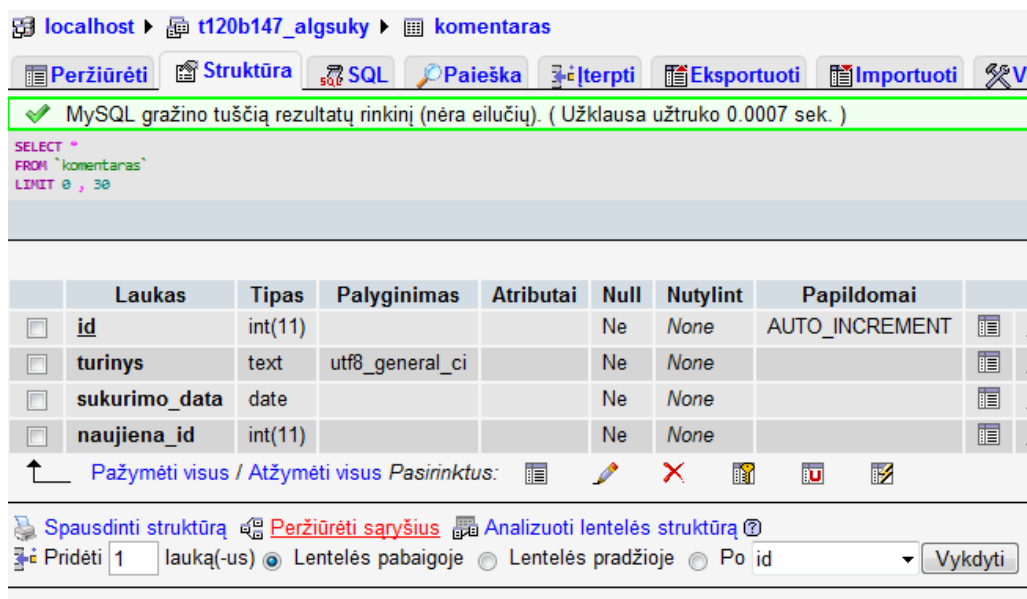
Kurdami lenteles atkreipkite dėmesį į šiuos punktus:

- pagal nutylėjimą naudojamas *MyISAM* saugojimo variklis. *MyISAM* nepalaiko ryšių tarp lentelių, todėl naudokite *InnoDB* variklį.
- Kiekviena lentelė privalo turėti lauką, saugantį įrašo pirminį raktą. Šiems laukams sukuriamas indeksas *PRIMARY*. Pirminių raktų laukams paprastai naudojamas INT duomenų tipas. Jam naudinga priskirti *AUTO_INCREMENT* požymį – tuomet įterpus

įrašų reikšmė nustatoma automatiškai. Indeksai reikalingi ir laukams, saugojantiems įrašų išorinius raktus – jiems kuriamas indeksas *INDEX*.

- Tekstiniams laukams nustatykite simbolių koduotę *UTF-8* bei palyginimą *general_ci* (*UTF8_general_ci*).

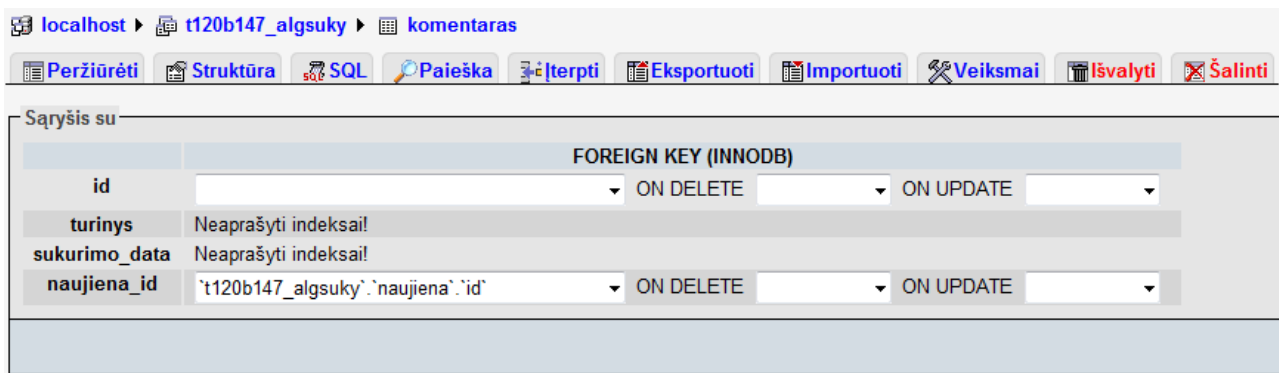
Sukūrę duomenų bazės lenteles, nustatykite ryšius tarp lentelių. Ryšiai nustatomi lentelės struktūros skiltyje, paspaudus nuorodą „Peržiūrėti sąryšius“:



The screenshot shows the MySQL Workbench interface for the 'komentaras' table. The 'Struktūra' (Structure) tab is active, displaying the table's schema. The table has four columns: 'id' (int(11), AUTO_INCREMENT), 'turinys' (text, utf8_general_ci), 'sukurimo_data' (date), and 'naujiena_id' (int(11)). The 'id' column is the primary key. The 'turinys' column has a collation of 'utf8_general_ci'. The 'sukurimo_data' column is of type 'date'. The 'naujiena_id' column is of type 'int(11)'. The 'id' column is marked as 'AUTO_INCREMENT'.

	Laukas	Tipas	Palyginimas	Atributai	Null	Nutylint	Papildomai
<input type="checkbox"/>	id	int(11)			Ne	None	AUTO_INCREMENT
<input type="checkbox"/>	turinys	text	utf8_general_ci		Ne	None	
<input type="checkbox"/>	sukurimo_data	date			Ne	None	
<input type="checkbox"/>	naujiena_id	int(11)			Ne	None	

3 pav. Lentelės struktūra



The screenshot shows the MySQL Workbench interface for the 'komentaras' table, specifically the 'Sąryšis su' (Relationships) tab. It displays the foreign key relationships for the 'komentaras' table. The 'id' column is the primary key. The 'naujiena_id' column is a foreign key that references the 'id' column of the 'komentaras' table. The relationship is defined with 'ON DELETE' and 'ON UPDATE' actions.

	FOREIGN KEY (INNODB)
id	ON DELETE ON UPDATE
turinys	Neaprašyti indeksai!
sukurimo_data	Neaprašyti indeksai!
naujiena_id	'komentaras'. 'id' ON DELETE ON UPDATE

4 pav. Lentelės ryšių kūrimas

Sukūrę lenteles bei nustatę ryšius lentelėms, sukurkite bent po 5 įrašus kiekvienoje lentelėje:

localhost ▶ t120b147_algsuky ▶ komentaras

Peržiūrėti Struktūra SQL Paieška Įterpti Eksportuoti Importuoti Veiksmai Išvalyti Šalinti

Laukas	Tipas	Funkcija	Null	Reikšmė
id	int(11)			
turinys	text			
sukurimo_data	date			
naujiena_id	int(11)			

Vykdyti

5 pav. Įrašų įterpimas į lentelę

Kadangi nustatėte ryšius tarp lentelių, pirmiausia sukurkite pagrindinių lentelių (neturinčių išorinių raktų) įrašus. Sistema tikrina įvedamų duomenų sąryšių integralumą, todėl neleis sukurti tokios išorinio rakto reikšmės, kurios neturi pagrindinės lentelės pirminis raktas.

2. Laravel MVC komponentų kūrimo instrukcija

2.1. Pradiniai nustatymai

Pirmas žingsnis susikūrus projektą reikia nustatyti nustatymus `.env` faile. Kaip nurodyta 1. šios instrukcijos skyriuje.

2.2. Pagrindinio šablono (layout'o) kūrimas

Šablonų kataloge sukuriamas pagrindinis svetainės šablonas – **app.blade.php** Šablonų kiekis neribotas, galima sukurti norimą kiekį šablonų ir juos naudoti tam tikrose svetainės vietose. Pagrindinis šablonas projekto kūrimo pradžioje turėtų atrodyti taip:

resources/views/layouts/layout.blade.php

```
<!DOCTYPE html>
<html lang="{{ str_replace('_', '-', app()->getLocale()) }}">
<head>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <!-- CSRF Token -->
    <meta name="csrf-token" content="{{ csrf_token() }}">

    <title>{{ config('app.name', 'Laravel') }}</title>

    <!-- Scripts -->
    <script src="{{ asset('js/app.js') }}" defer></script>

    <!-- Fonts -->
    <link rel="dns-prefetch" href="//fonts.gstatic.com">
    <link href="https://fonts.googleapis.com/css?family=Nunito" rel="stylesheet">

    <!-- Styles -->
    <link href="{{ asset('css/app.css') }}" rel="stylesheet">
</head>
<body>
    <div id="app">
        <nav class="navbar navbar-expand-md navbar-light bg-white shadow-sm">
            <div class="container">
                <a class="navbar-brand" href="{{ url('/') }}">
                    {{ config('app.name', 'Laravel') }}
                </a>
                <button class="navbar-toggler" type="button" data-toggle="collapse" data-
target="#navbarSupportedContent" aria-controls="navbarSupportedContent" aria-expanded="false" aria-
label="{{ __('Toggle navigation') }}">
                    <span class="navbar-toggler-icon"></span>
                </button>

                <div class="collapse navbar-collapse" id="navbarSupportedContent">
                    <!-- Left Side Of Navbar -->
                    <ul class="navbar-nav mr-auto">

                    </ul>

                    <!-- Right Side Of Navbar -->
                    <ul class="navbar-nav ml-auto">

                    </ul>
                </div>
            </div>
        </nav>

        <main class="py-4">
            @yield('content')
        </main>
    </div>
</body>
</html>
```

Į pagrindinį šabloną įkopijuokite *HTML* kodą iš pirmojo laboratorinio darbo.

Šablonas kitu turiniu gali būti papildytas kaip pavyzdžiui:

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

@section('content')
<div class="container">
  <div class="row justify-content-center">
    <div class="col-md-8">
      <div class="card">
        <div class="card-header">Dashboard</div>

        <div class="card-body">
          @if (session('status'))
            <div class="alert alert-success" role="alert">
              {{ session('status') }}
            </div>
          @endif

          You are logged in!
        </div>
      </div>
    </div>
  </div>
</div>
</div>
@endsection
```

2.3. Stiliaus ir veikimo skriptų papildymas

Karkaso CSS ir JavaScript skriptus galima rasti:

```
/public/css/
```

```
/public/js/
```

Rekomenduojama sukurti custom.css ir custom.js failus atitinkamuose aplankuose.

Sukūrus šiuos failus, juos reikia įtraukti į:

```
resources/views/layouts/app.blade.php
```

Šie skriptai papildomi į failą įtraukiant šias eilutes:

```
<script src="{{ asset('js/custom.js') }}" defer></script>
```

```
<link href="{{ asset('css/custom.css') }}" rel="stylesheet">
```

Stiliaus failas vėliau bus kviečiamas pagrindinio šablono.

Jei neplanuojate naudoti karkase CSS ir JavaScript turinio galite ištrinti failus arba įtraukimo įrašus pagrindiniame šablone.

2.4. Valdiklių kūrimas

Laravel karkase patogiausias būdas sukurti naujus valdikius yra pasinaudojant komandinės eilutės komanda: **php artisan make:controller *pavadinimas***. Nors ir neprivaloma tačiau pagal gerąsias failų pavadinimų kūrimo taisykles rekomenduojama prie valdiklio failo pavadinimo pridėti žodį **Controller** (pvz. `pavadinimasController`). Visus valdikius Laravel karkase galima rasti ***app/controllers/*** kataloge ir šio katalogo pakatalogiuose. Valdiklio pavyzdys pateikiamas apačioje.

```
<?php
namespace App\Http\Controllers;
use Illuminate\Http\Request;
class HomeController extends Controller
{
    public function index()
    {
        return view('home');
    }
}
```

Norint, kad pasiektume šią funkciją per naršyklę ***/routes/web.php*** failą reikia papildyti eilutėmis:

```
Route::get('/home', 'HomeController@index')
```

Valdiklio Home funkcijai *index()* ką tik sukurtas vaizdas galėtų atrodyti taip:

resources/views/home.blade.php

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

@section('content')
<div class="container">
    <div class="row justify-content-center">
        <div class="col-md-8">
            <div class="card">
                <div class="card-header">Dashboard</div>

                <div class="card-body">
                    <p> Hello world </p>
                </div>
            </div>
        </div>
    </div>
</div>
@endsection
```

Valdiklis pasiekiamas adresu:

```
http://localhost/pavadinimas/public/home
```

Sistema turėtų išvesti pagrindinį šabloną su stiliumi ir atvaizduoti tekstą hello world. Jei puslapio struktūra atrodo prastai, gali būti kad pašalinote standartinį CSS ir dar ne pridėjote savo stiliaus.

2.5. Modelių kūrimas

Sukurkime modelį duomenims išrinkti iš pasirinktos lentelės, tarkime „naujienos“. Pasinaudojus komandinės eilutės komanda **php artisan make:model Naujiena**

app /Naujiena.php

```
<?php
namespace App;
use Illuminate\Database\Eloquent\Model;

class Naujiena extends Model
{
    //
}
```

Jei sukurto modelio pavadinimas nesutampa su lentelės pavadinimu, galime tai nurodyti modelyje pridėdami \$table kintamąjį.

Taip pat reikia nurodyti ir kokius laukus lentelėje gali būti užpildyti. Šiais aspektais papildytas modelio failas gali atrodyti taip:

app /Naujiena.php

```
<?php
namespace App;

use Illuminate\Database\Eloquent\Model;
class Naujiena extends Model
{
    protected $table = 'PvzKitaNaujiena';
    protected $fillable= ['pavadinimas','santrauka', 'turinys','sukurimo_data'];
}
```

Taip pat jei nesiruošame naudoti Laravel karkaso valdomų sukūrimo ir atnaujinimo kintamųjų – created_at bei updated_at

```
public $timestamps = false;
```

Modelis taip pat turėtų būti papildomas funkcijomis, skirtomis išrinkti susijusius įrašus, tačiau tai priklauso nuo modelio ir duomenų bazės konteksto. Daugiau apie modelius ir sudėtingesnius panaudojimo variantus galite rasti: <https://laravel.com/docs/master/eloquent>

2.6. Duomenų išrinkimas ir atvaizdavimas

Dažnu atveju modeliai iškviečiami valdikliuose ar kituose specializuotose klasėse. Žemiau pateikiamas valdiklis ir vaizdas visiems duomenims iš „naujiena“ duomenų bazės lentelės atvaizduoti.

app/Http/controllers/NewsController.php

```
<?php
namespace App\Http\Controllers;
use App\News;
use Illuminate\Http\Request;
use App\Http\Controllers\Controller;

class NewsController extends Controller
{
    public function Index()
    {
        $allNews= News::all();
        return view('news', compact('allNews'));
    }
}
```

Valdiklio NewsController.php *index()* funkcijoje gauname visus vartotojus ir perduodame duomenis vaizdui, kuris galėtų atrodyti taip:

resources/views/news.blade.php

```
@extends('layouts.app')
@section('content')
    <div class="row">
        <h3 class="pb-3 mb-4 font-italic border-bottom">
            Verslas
        </h3>
        @foreach($allNews as $newsData )
            <div class="blog-post">
                <h2 class="blog-post-title"> {{ $newsData->pavadinimas }} </h2>
                <p class="blog-post-meta">{{ $newsData->sukurimo_data }} </p>
                <p>{{ $newsData->turinys }} </p>
            </div><!-- /.blog-post -->
        @endforeach
    @endsection
```

Siekiant, kad šis vaizdas būtų pasiekiamas iš naršyklės **web.php** failą papildome

```
Route::get('/news', NewsController@index)->name('news');
```

Valdiklyje `News::all()`; ištraukia visus įrašus iš lentelės `news`, pasinaudojant `compact()` funkcija perduoda kintamąjį `allNews` į vaizdą `news.balde.php` kur panaudojant `balde` šablono sintaksę vykdomas ciklas ir atvaizduojami duomenys iš lentelės. Tokie kaip pavadinimas, sukūrimo data, bei turinys.

Susiduriant su klaidomis rekomenduojama rekomenduojama pabandyti išvesti duomenis į ekraną su Laravel karkaso pagalbine funkcija `dd()`; daugiau informacija apie Laravel pagalbines funkcijas: <https://laravel.com/docs/master/helpers#method-dd>