3. laboratorinis darbas. *Laravel* funkcionalumas

Laboratorinis darbas skirtas sukurti likusius modelius, šablonus bei valdiklius, svetainės turiniui atvaizduoti. Taip pat panaudoti *Pagination, Session*, sukurti *Migracijas*.

I. Darbo užduotis

1. Atvaizduoti bazėje saugojamus duomenis. Sukurti reikalingus modelius, šablonus ir valdiklius.
2. Panaudoti *Pagination, Session*.
3. Sukurti duomenų bazės lentelėms migracijas.

II. Darbo eiga

1. Duomenų vaizdavimas svetainėje

Trečio laboratorinio darbo metu susipažinote su *Laravel* architektūra bei kaip sukurti valdiklį, pagrindinį šabloną, vaizdus, modelius. Šiame darbe išmokesite pasinaudoti kitu *Laravel* karkaso funkcionalumu.

Visų pirma reikia sukurti likusius valdiklius, modelius bei vaizdus, kad būtų galima atvaizduoti visus DB lentelių duomenis. Tai galima padaryti pasinaudojant LD2.2 „*Laravel karkaso įvadas*“ skyrelyje pateiktais pavyzdžiais.

2. Laravel karkaso funkcionalumas

2.1. Laravel sesija

Šiuolaikinė *web* applikacija be sesijos sunkiai įsivaizduojama. *PHP* turi būrį funkcijų darbui su sesijom (detaliau: http://php.net/manual/en/book.session.php). Kam reikalingos sesijos? Baigus vykdyti *PHP* kodą vartotojui yra nusiunčiamas *HTML*, o visi kintamieji, objektai ir pan. yra sunaikinami, atmintis išvaloma. Todėl *PHP* yra vadinama *stateless language*. Sesijos - vienas iš būdų išsaugoti būseną.

Geriausiai informacija apie Laravel sesiją ir jų tipus pateikiama: https://laravel.com/docs/master/session

Naudojimas paprastas - *PHP* skripto viršuje sukūriame sesiją, o toliau ją naudojamės. Pavyzdžiui faile *set_session.php* į sesiją įdedame *UNIX timestamp'ą*:

set_session.php

```
<?php
// creating session
session_start();
```

```
// setting variable to session
$_SESSION['view_time'] = time();
?>
```

Tuo tarpu kituose failuose (ar kituose *PHP* skripto vietose) sesijoje patalpintą reikšmę galima pasiimti ir panaudoti:

get_session.php

```
<?php
// creating session
session_start();
// getting variable from session
$lastViewTime = $_SESSION['view_time'];
echo date('Y m d H:i:s', $lastViewTime);
// unsetting variable from session
unset($_SESSION['view_time']);
?>
```

Laravel karkase yra viskas labai panašu, skirtumas tik tas, kad naudojamos ne *PHP* funkcijos, o *Laravel* karkaso sesijos funkcijos. Žemiau pateiktas analogiškas pavyzdis tik su *Laravel* funkcijomis.

Funkcijoje index() nuskaitome laiką su pagalbine darbui su datomis skirta biblioteka Carbon datą ir laiką išsaugojame kintamajame $time jį pridedame prie sesijos, pasinaudodami raktu ‚entryTime‘ ištraukiame datą ir laiką iš sesijos, bei pašaliname su funkcija forget.

app/Htpp/controllers/HomeController.php

```
<?php
namespace App\Http\Controllers;
use Carbon\Carbon;
use Illuminate\Http\Request;
class HomeController extends Controller
{
    public function index()
    {
        $time = Carbon::now();
        session(['entryTime' => $time]);
        echo(session('entryTime'));
        session()->forget('entryTime');
    }
}
```

2.2. Laravel Pagination

Norint sukurti Laravel Pagination reikia keturių dalykų:

1. Modelio kurio duomenis mes norėsim atvaizduoti puslapiais.
2. Kreipinio į jau minėtą modelį su puslapiavimo funkcija.
3. Vaizdo papildymo kintamojo duomenų atvaizdavimu ir puslapio perėjimui skirtų mygtukų.

Daugiausiai informacijos apie Laravel Pagination funkciją ir jos naudojimą galima rasti čia: https://laravel.com/docs/master/pagination

Iš antrojo laboratorinio darbo turime modelį naujienų lentelės duomenims atvaizduoti **News.php**

app/News.php

```
<?php
namespace App;

use Illuminate\Database\Eloquent\Model;

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

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

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

    public function NewsOfCategory()
    {
        return $this->belongsTo('App\NewsCategory');
    }
}
```

Pakeičiame valdiklio funkciją News::all() kuri skirta visiems duomenims atvaizduoti į funkciją paginate(n). Kur n yra skaičius nurodantis kiek įrašų rodysime per puslapį.
app/http/controllers/NewsController.php

```
<?php

namespace App\Http\Controllers;

use App\News;
```

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

class BusinessController extends Controller
{
    public function Index()
    {
        $allNews= News::paginate(4);
        return view('news/business', compact('allNews'));
    }
}
```

Vaizde **news.blade.php** panaudojame @foreach blade šablono metodą atvaizduoti duomenims, bei pridedame eilutę **{{ $allNews->links() }}**, kuri sugeneruoja html kodą mygtukų perėjimui iš vieno puslapio į kitą.

resources/views/news.blade.phh

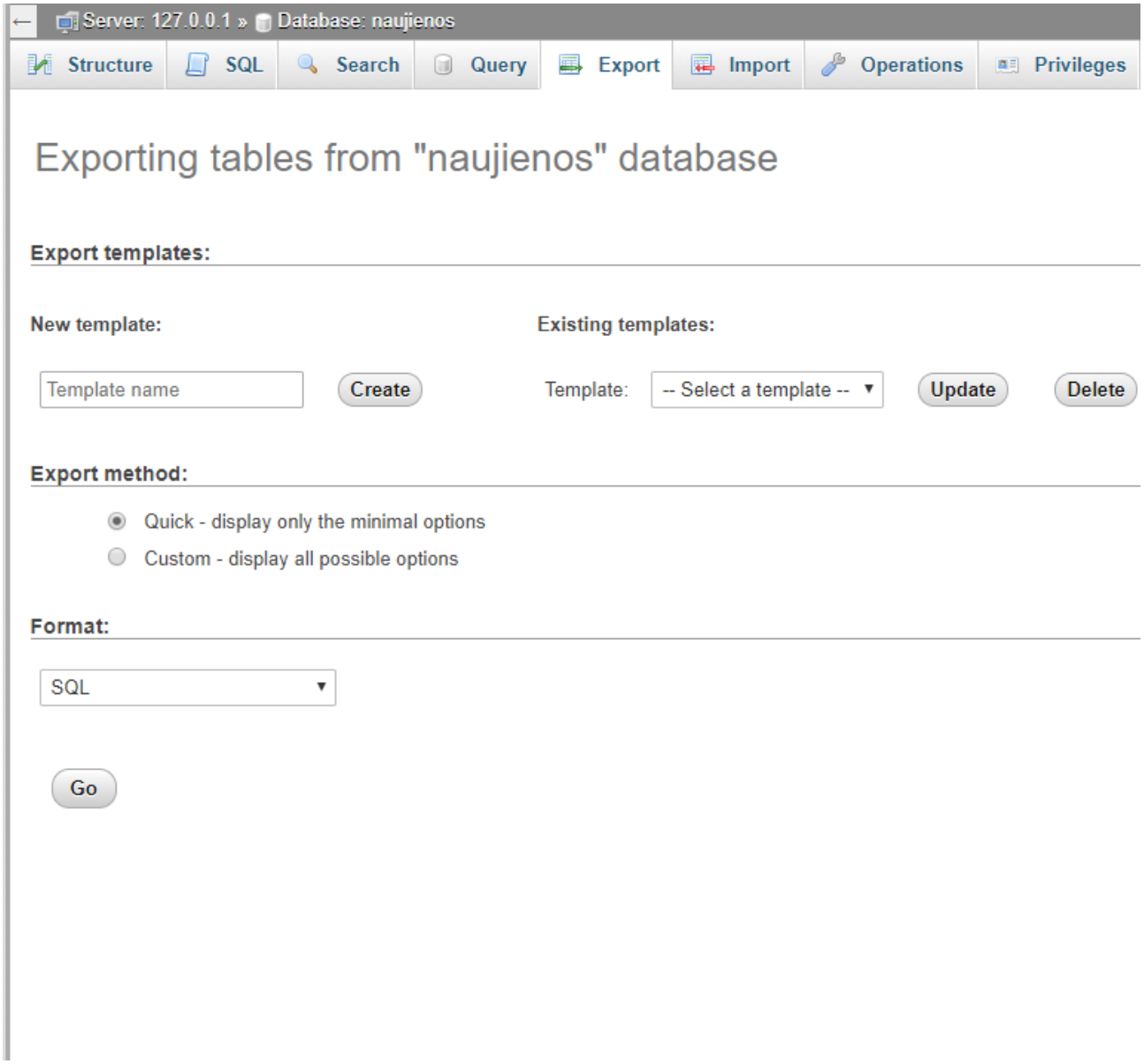
```
@extends('layouts.app')

@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 }} by <a href="#"> Jacob</a></p>
            <p>{{ $newsData->turinys }} </p>
            {{ $allNews->links() }}
        </div><!-- /.blog-post -->
        @endforeach
    </div><!--/row-->
@endsection
```

2.3. Laravel karkaso migracijos

Sklandžiausiai informaciją apie darbą su migracijomis pateikiama karkaso dokumentacijoje: https://laravel.com/docs/master/migrations

Migracijos tai duomenų bazių lentelių informacija saugojama karkaso struktūroje, leidžianti kurti, atnaujinti ir užpildyti duomenų bazių lenteles testavimo duomenimis. Prieš pradedant darbą su migracijomis rekomenduojamas pasidaryti esamos duomenų bazės atsarginę kopiją. Tai galima atlikti pasinaudojant PhpMyAdmin sistemos funkcionalumu pav. **1**. Tai atlikama pasirenkant norimą duomenų bazę atveriant jos langą, pasirankant skyrelį **export** ir nuspaudžiant mygtuką go.

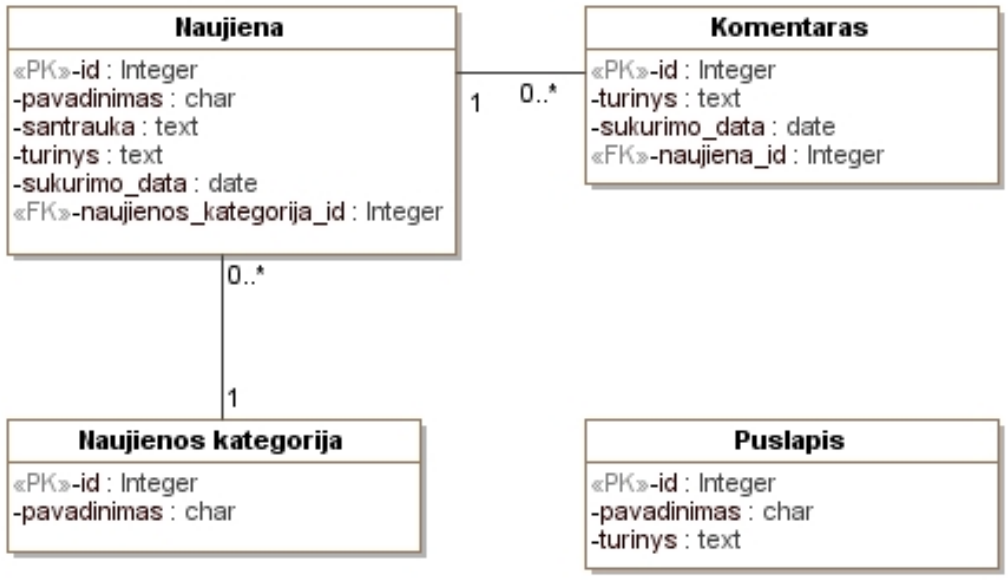


pav. 1 phpmyadmin duomenų bazės exportavimo langas

migracijos, kuriamos pasinaudojant komandine eilute ir komanda: **php artisan make:migration pavadinimas.** Migracijos faile aprašomos, bent dvi funkcijos up() bei down(). Up() komanda naudojama lentelei sukurti, down() komanda naudojama lentelės pašalinimo metu. Migracijos faile rekomenduojama užpildyti $tableName kintamąjį, kuriame nurodo kuriamso lentelės pavadinimą. Taip pat funkcijoje up() reikia nurodyti schemos sukūrimo parametrus, tokius kaip:

- Duomenų bazės stulpelių pavadinimai ir tipai
- Naudojamo duomenų bazės variklio tipas
- Lentelėje naudojami index'ai ir ryšiai.

Toliau pateikiamas duomenų bazės pavyzdys iš pirmojo laboratorinio darbo **pav. 2** bei šio pavyzdžio lentelei naujiena sukurtas migracijos failas.



pav. 2 Duomenų bazės lentelių pavyzdys.

```
<?php

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

class CreateNaujienaTable extends Migration
{
    /**
     * Schema table name to migrate
     * @var string
     */
```

```
    /**
     * Run the migrations.
     * @table Naujiena
     *
     * @return void
     */
    public function up()
    {
        Schema::create('naujiena', function (Blueprint $table) {
            $table->increments('id');
            $table->string('pavadinimas', 45)->nullable();
            $table->text('santrauka')->nullable();
            $table->longText('turinys')->nullable();
            $table->dateTime('sukurimo_data')->nullable();
            $table->unsignedInteger('naujienos_kategorija_id');

            $table->index(["naujienos_kategorija_id"],
'fk_Naujiena_Naujienos_kategorija1_idx');



            $table->foreign('naujienos_kategorija_id',
'fk_Naujiena_Naujienos_kategorija1_idx')
                ->references('id')->on('naujienos_kategorija')
                ->onDelete('no action')
                ->onUpdate('no action');
        });
    }

    /**
     * Reverse the migrations.
     *
     * @return void
     */
     public function down()
     {
       Schema::dropIfExists($this->tableName);
     }
}
```

Remiantis šiuo pavyzdžiu ir karkaso dokumentacija sukurkite migracijas visoms sistemoje naudojamoms duomenų bazės lentelėms. Migracijos iš karkaso į duomenų bazę perkeliamos pasinaudojant komanda:

php artisan migrate

Migracijos atšaukiamos komanda:

php artisan migrate:reset

2.4. Laravel karkaso modeliai.

Informacija apie modelius sklandžiausiai pateikiama karkaso dokumentacijoje:

https://laravel.com/docs/master/eloquent-relationships

Modeliai tai duomenų lentele atitinkantis objektai aprašantys duomenų bazės lentelės stulpelių informaciją ir ryšius su kitomis lentelėmis. Teisingai apsirašius visas duomenų bazės lenteles modeliais galima naudotis Elequent Objektų ryšių valdikliu (angl. Elequent ORM), kuris leidžia naudoti įvairias darbą pagreitinančias funkcijas tokias kaip objekto sukūrimas ir pašalinimas, be SQL kodo rašymo. Toliau pateikiamas pavyzdys kuriame aprašytas naujienos lentelės modelis.

```
<?php

namespace App;

use Illuminate\Database\Eloquent\Model;

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

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

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

    public function NewsOfCategory()
    {
        return $this->belongsTo('App\NewsCategory');
    }
}
```