Для создания страницы со всеми продуктами, для редактирования продуктов, для просмотра каждого продукта в отдельном окне нам понадобится все тоже самое, что мы делали при создании главной страницы и инструментов для работы с ней и таблицей БД от которой она связана. Поэтому в этой главе будет много кода страниц, контроллера, модели и сервис-класса запроса. Постараюсь как можно больше объяснить, но в тоже время не хочется повторяться.

Перечислю все, что нам понадобится сразу. Главная страница всех товаров – resources\views\dogs.blade.php; страница для просмотра информации по каждой позиции – resources\views\dog\view.blade.php; страница для ввода и редактирования каждой позиции – resources\views\dog\input.blade.php; контроллер, который будет обрабатывать все действия, связанные с продуктами нашего проекта – app\Http\Controllers\DogController.php. Так же нам понадобится таблица в БД, миграция, которая ее создает, заполнитель для таблицы, модель, связанная с нашей таблицей. Вобщем, работы много, давайте уже приступать.

И начнем с миграции. Создадим миграцию командой:

php artisan make:migration create\_ourDogs\_table

Исправим код в файле database\migrations\<метка\_даты>\_create\_ourDogs\_table.php

<?php

use Illuminate\Support\Facades\Schema;

use Illuminate\Database\Schema\Blueprint;

use Illuminate\Database\Migrations\Migration;

class CreateOurDogsTable extends Migration

{

public function up()

{

Schema::create('our\_dogs', function (Blueprint $table) {

$table->increments('id\_dogs');

$table->unsignedTinyInteger('sex');

$table->string('name');

$table->date('date\_age');

$table->string('family', 3)->nullable();

$table->string('dbres') ->nullable();

$table->unsignedTinyInteger('sale');

$table->timestamps();

$table->unique('id\_dogs');

});

}

public function down()

{

Schema::dropIfExists('our\_dogs');

}

}

$table->unique('id\_dogs');

Эта строка была добавлена, чтобы избежать ошибки предотвращения создания нескольких полей первичного ключа и автоприращения. Дело в том, что когда мы указываем поля цифрового формата (integet, bigInteger и т.д.), при запуске миграции, с ними будет произведена попытка преобразования в поля с автоприращением и первичного ключа. На что сразу выскочит ошибка, блокирующая это действие, и таблица не будет создана. Указав же для нашего поля первичного ключа, что значение должно быть уникальным (unique), другие цифровые поля не будут преобразованы и ошибки не возникнет.

Запускаем миграцию:

php artisan migrate

Таблица ‘our\_dogs’ в нашей базе данных создана. Просмотрим ее с помощью инструмента phpMyAdmin.

Теперь перейдем к созданию модели, которая будет связана с этой таблицей. Модель назовем Our\_dog и создадим с вызвав в консоли команду :

php artisan make:model Our\_dog

Для начала в моделе app\Our\_dog.php пропишем поля, которые будут защищены от массового присвоения и укажем первичный ключ:

<?php

namespace App;

use Illuminate\Database\Eloquent\Model;

use Illuminate\Support\Facades\DB;

class Our\_dog extends Model

{

protected $fillable = ["name", "sex", "date\_age", "family", "dbres"];

protected $primaryKey = 'id\_dogs';

}

файле database\factories\ModelFactory.php. Откроем его и добавим к уже имеющемуся коду следующий фрагмент:

<?php

use Faker\Provider\ru\_RU;

$factory->define(App\Content::class, function (Faker\Generator $faker) {

return [

'title' => $faker-> sentence,

'pageContent' => $faker-> realText . ' || ' . $faker-> realText,

'photo' => $faker->imageUrl($width = 960, $height = 427),

];

});

**$factory->define(App\Our\_dog::class, function (Faker\Generator $faker) {**

**$faker = Faker\Factory::create('ru\_RU');**

**return [**

**'sex' => $faker->randomElement($array = array (0, 1)),**

**'name' => $faker->name,**

**'date\_age' => $faker->date($format = 'Y-m-d', $max = 'now'),**

**'family' => $faker->randomElement($array = array ("A","B","C","D","E","F","G")),**

**'dbres' => $faker->url,**

**'sale' => $faker->randomElement($array = array (0, 1)),**

**];**

**});**

Создадим seeder класс для наполнения данными

php artisan make:seeder OurDogTableSeeder

Отредактируем файл database\seeds\ OurDogTableSeeder.php

<?php

use Illuminate\Database\Seeder;

use Illuminate\Support\Facades\DB;

use App\Our\_dog;

class OurDogTableSeeder extends Seeder

{

public function run()

{

DB::statement("SET foreign\_key\_checks=0");

Our\_dog::truncate();

DB::statement("SET foreign\_key\_checks=1");

factory(Our\_dog::class, 14)->create();

}

}

Изменим файл database\seeds\DatabaseSeeder.php:

<?php

use Illuminate\Database\Seeder;

class DatabaseSeeder extends Seeder

{

public function run()

{

$this->call('LitterTableSeeder');

$this->command->info('operation completed successfully!');

$this->call('ContentTableSeeder');

$this->command->info('operation completed successfully!');

**$this->call('OurDogTableSeeder');**

**$this->command->info('operation completed successfully!');**

}

}

Запускаем из Консоли наполнение таблицы данными:

php artisan db:seed

Таблица для фото и связывание таблиц в моделях и в SQLite.

Добавим код в модель Our\_dog.php:

<?php

namespace App;

use Illuminate\Database\Eloquent\Model;

**use Illuminate\Support\Facades\DB;**

class Our\_dog extends Model

{

protected $fillable = ["name", "sex", "date\_age", "family", "dbres"];

protected $primaryKey = 'id\_dogs';

**public function dogs\_photos() {**

**return $this->hasMany("App\Dogs\_photo", "id\_dogs", "id\_dogs");**

**}**

**public static function ourDogs() {**

**return DB::table('our\_dogs')->select('dogs\_photos.photo', 'our\_dogs.\*')**

**->leftJoin('dogs\_photos', 'dogs\_photos.id\_dogs', 'our\_dogs.id\_dogs')**

**->orderBy('date\_age', 'desc')->groupBy('name')->distinct('name')**

**->paginate(8);**

**}**

}

Миграция для таблицы с фотографиями

php artisan make:migration create\_dogsPhoto\_table

<?php

use Illuminate\Support\Facades\Schema;

use Illuminate\Database\Schema\Blueprint;

use Illuminate\Database\Migrations\Migration;

class CreateDogsPhotoTable extends Migration

{

public function up()

{

Schema::create('dogs\_photos', function (Blueprint $table) {

$table->increments('id');

$table->integer('id\_dogs');

$table->string('photo');

$table->timestamps();

$table->unique('id');

});

}

public function down()

{

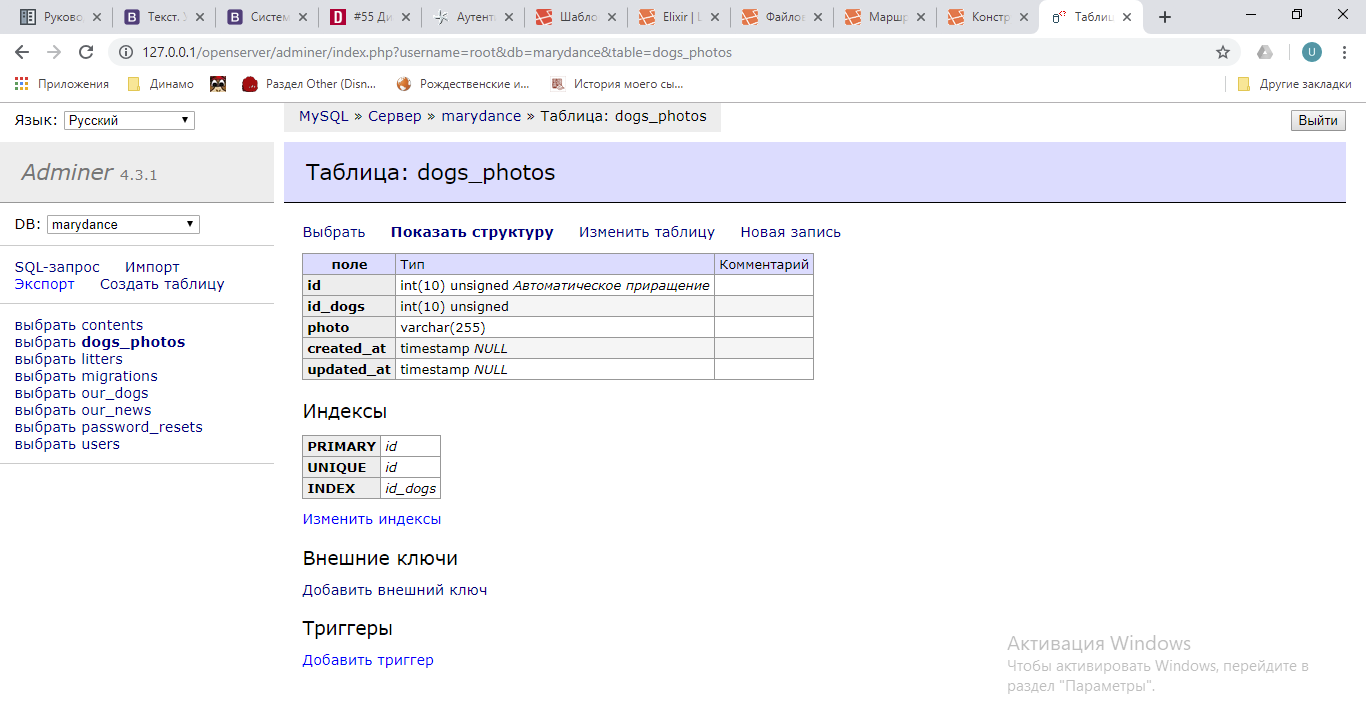
Schema::dropIfExists('dogs\_photos');

}

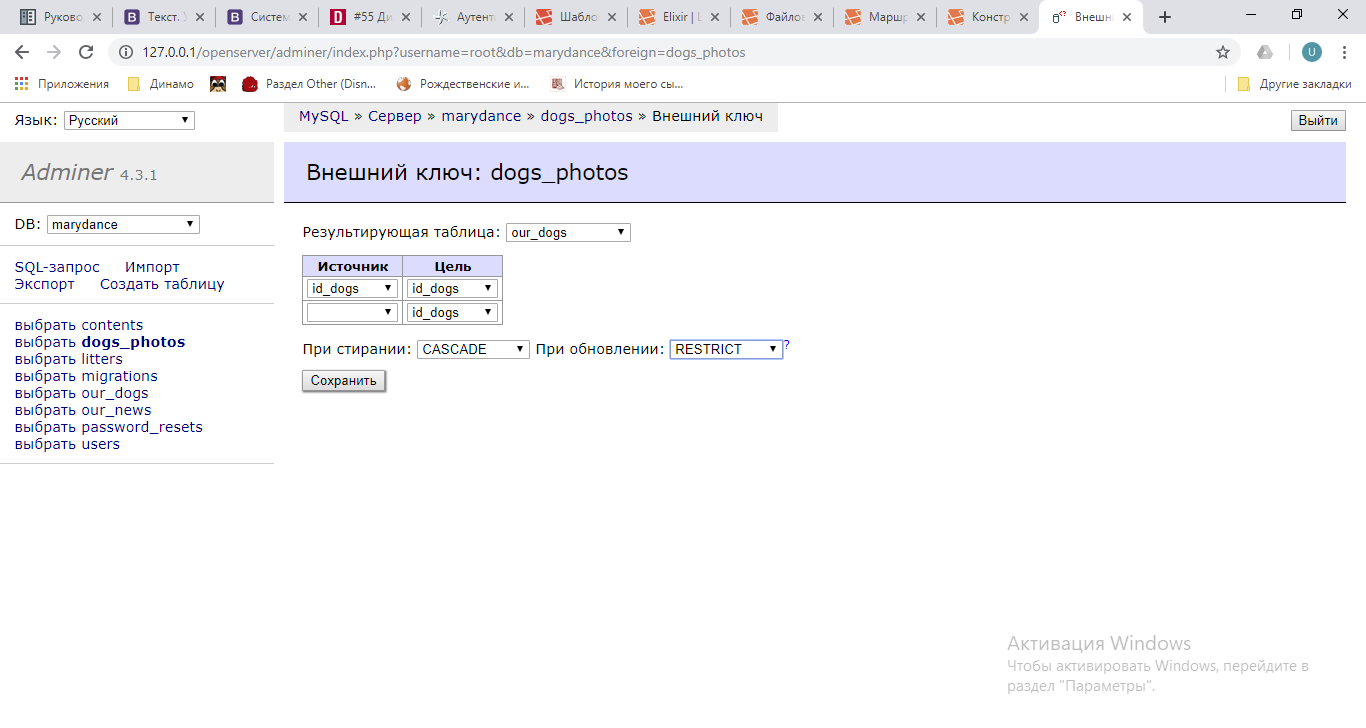
}

Запустив миграцию создадим таблицу и добавим внешний ключ, связав две таблицы:

Вначале приведем поле id\_dogs таблицы dogs\_photos к такому же виду как и поле id\_dogs таблицы our\_dogs:



Добавим внешний ключ:



Теперь нужна модель для таблицы с фотографиями

php artisan make:model Dogs\_photo

Откроем и заполним файл app\Dogs\_photo.php

<?php

namespace App;

use Illuminate\Database\Eloquent\Model;

class Dogs\_photo extends Model

{

public function our\_dogs() {

return $this->belongsTo("App\Our\_dog", "id\_dogs", "id\_dogs");

}

protected $primaryKey = 'id\_dogs';

}

Создаем заполнитель:

php artisan make:seed DogsPhotoTableSeeder

<?php

use Illuminate\Database\Seeder;

use Illuminate\Support\Facades\DB;

use App\Dogs\_photo;

class DogsPhotoTableSeeder extends Seeder

{

public function run()

{

DB::statement("SET foreign\_key\_checks=0");

Dogs\_photo::truncate();

DB::statement("SET foreign\_key\_checks=1");

factory(Dogs\_photo::class, 30)->create();

}

}

Вносим изменения в database\factories\ModelFactory.php:

<?php

$factory->define(App\Content::class, function (Faker\Generator $faker) {

**$faker = Faker\Factory::create('ru\_RU');**

return [

'title' => $faker-> sentence,

'pageContent' => $faker-> realText . ' || ' . $faker-> realText,

'photo' => $faker->imageUrl($width = 960, $height = 427),

];

});

$factory->define(App\Our\_dog::class, function (Faker\Generator $faker) {

$faker = Faker\Factory::create('ru\_RU');

return [

'sex' => $faker->randomElement($array = array (0, 1)),

'name' => $faker->name,

'date\_age' => $faker->date($format = 'Y-m-d', $max = 'now'),

'family' => $faker->randomElement($array =

array ("A","B","C","D","E","F","G")),

'dbres' => $faker->url,

'sale' => $faker->randomElement($array = array (0, 1)),

];

});

**$factory->define(App\Dogs\_photo::class, function (Faker\Generator $faker) {**

**return [**

**'id\_dogs' => $faker->numberBetween($min = 1, $max = 14),**

**'photo' => $faker->imageUrl($width = 250, $height = 150),**

**];**

**});**

Напоследок редактируем database\seeds\DatabaseSeeder.php:

<?php

use Illuminate\Database\Seeder;

class DatabaseSeeder extends Seeder

{

public function run()

{

$this->call('LitterTableSeeder');

$this->command->info('operation Litter completed successfully!');

$this->call('ContentTableSeeder');

$this->command->info('operation Content completed successfully!');

$this->call('OurDogTableSeeder');

$this->command->info('operation OurDog completed successfully!');

**$this->call('DogsPhotoTableSeeder');**

**$this->command->info('operation DogsPhoto completed successfully!');**

}

}

Запускаем заполнение:

php artisan db:seed

Создадим маршрут на просмотр начальной страницы routes\web.php:

Route::get('/dogs', 'DogController@dogs');

Создадим контроллер командой в Консоли а потом изменим его:

php artisan make:controller DogController

Добавим метод dogs() в контроллер app\Http\Controllers\DogController.php:

<?php

namespace App\Http\Controllers;

use App;

**use App\Our\_dog;**

class DogController extends Controller

{

**public function dogs() {**

**return view("dogs", ["dogs" => Our\_dog::ourDogs()]);**

**}**

}

И представление resources\views\dogs.blade.php:

@extends('layouts.app')

@section("title", "Собаки")

@section("main")

<?php

function format\_interval(DateInterval $interval) {

$result = "0 мес.";

if ($interval->y) {

$result = $interval->format("%y г. ");

} else {

$result = "";

}

if ($interval->m) {

$result .= $interval->format("%m мес.");

} else {

$result .= "0 мес.";

}

return $result;

}

?>

<h1>Наши собаки</h1>

<p>

@if (auth()->check())

<a href="{{ route('dog.create', ['page' => '1']) }}" class="btn btn-outline-info">Добавить собаку</a>

@endif

</p>

<div class="container">

<div class="row">

@foreach ($dogs as $dog)

<div class="col-3">

<div class="card">

<a href="/dog/{{$dog->id\_dogs}}">

@if($dog->photo)

<div class="card-img-top mh-8" style="background: url('{{$dog->photo}}') no-repeat 50% 50%; background-size: contain;">

</div>

@else

<div class="card-img-top mh-8" style="background: url('./img/nophoto.jpg') no-repeat 50% 50%; background-size: cover;">

</div>

@endif

<div class="card-body">

<h4 class="card-title"><b>{{ $dog->name }}</b></h4>

<p class="card-text"><b>Пол: </b>

@if ($dog->sex == 0)

Сука

@else

Кобель

@endif

</p>

<p class="card-text">Помёт: {{ $dog->family }}</p>

@if ($dog->dbres)

<p class="card-text"><b>Ссылка на родословную:</b> <small class="text-muted">{{ $dog->dbres }}</small></p>

@endif

</div>

<div class="card-footer">Возраст:

<small class="text-muted">

<?php

$date1 = new DateTime("now");

$date3 = new DateTime($dog->date\_age);

$interval2 = $date1->diff($date3);

$nowT2 = format\_interval($interval2);?>

<?=$nowT2;?></small>

</div>

</a>

@if (auth()->check())

<a href="{{ action('DogController@input', ['id' => $dog->id\_dogs]) }}" class="btn btn-primary">

Редактировать

</a>

<a href="{{ action('DogController@destroy', ['id' => $dog->id\_dogs]) }}" class="btn btn-success" onclick="return confirm('Подтверждаете удаление?')">

Удалить

</a>

@endif

</div>

</div>

@endforeach

</div>

{{ $dogs->links() }}

</div>

@endsection

Можем проверить, перейдя на страницу http://mary-dance.lc/dogs

Мы сделали ссылку <a href="/dog/{{$dog->id\_dogs}}"> что означает, что будет создана ссылка по всей карточке элемента http://mary-dance.lc/dog/1/, перейдя по которой мы должны увидеть развернутую карточку элемента с id равным 1. Для этого нам надо создать маршрут, метод в контроллере и представление.

Маршрут:

Route::get("/dog/{id\_dogs}", "DogController@view");

Метод в контроллере:

<?php

namespace App\Http\Controllers;

use App;

use App\Our\_dog;

use App\Dogs\_photo;

use Illuminate\Support\Facades\DB;

class DogController extends Controller

{

public function dogs() {

return view("dogs", ["dogs" => Our\_dog::ourDogs()]);

}

**public function view(Our\_dog $id\_dogs, $id=null) {**

**if (isset($id)) {**

**$photos = DB::table("dogs\_photos")->where("id\_dogs", "=", $id\_dogs->id\_dogs)->get();**

**$mainPhoto = DB::table("dogs\_photos")->where("id", "=", $id)->get();**

**$photika = DB::table("dogs\_photos")->select("photo")->where("id", "=", $id)->get();**

**return view ("dog.view", ["dog" => $id\_dogs, "photo" => $photos, "firstPhoto" => $photika]);**

**} else {**

**$photos = DB::table("dogs\_photos")->where("id\_dogs", "=", $id\_dogs->id\_dogs)->get();**

**$mainPhoto = null;**

**$photika = null;**

**if (isset($photos)) {**

**return view ("dog.view", ["dog" => $id\_dogs, "photo" => $photos, "firstPhoto" => null]);**

**} else {**

**return view ("dog.view", ["dog" => $id\_dogs, "photo" => null, "firstPhoto" => null]);**

**}**

**}**

**}**

}

Теперь надо создать представление для просмотра каждой позиции, но так как мы предусмотрели переключение между фотографиями объекта, при наличии у него нескольких изображений, средствами PHP (можно сделать средствами JavaScript без перезагрузки страницы – это личный выбор каждого), то нам необходимо добавить новый маршрут на обработку запроса http://адрес\_сайта/ид\_объекта/номер\_изображения/.

Route::get("/dog/{id\_dogs}/{firstPhoto}", "DogController@view");

Код представления resources\views\dog\view.blade.php:

@extends('layouts.app')

@push("head")

<?php

function format\_interval(DateInterval $interval) {

$result = "0 мес.";

if ($interval->y) {

$result = $interval->format("%y г. ");

} else {

$result = "";

}

if ($interval->m) {

$result .= $interval->format("%m мес.");

} else {

$result .= "0 мес.";

}

return $result;

}

?>

@endpush

@section('main')

<?php $h = "Информация :: " . ($dog->name) ?>

@section("title", $h . "")

<div class="p-3 mb-2 bg-secondary text-white text-center">

<h1>{{ $dog->name }}</h1>

</div>

<div class="container">

<div class="row">

<div class="col-md-7 col-md-offset-2">

<div class="panel panel-default">

<div class="panel-heading">

<div class="panel-body">

<div class="card-body">

<h5 class="card-title"><strong>{{ $dog->name }}</strong></h5>

<p class="card-text"><strong>Пол: </strong>

@if ($dog->sex == 0)

Сука

@else

Кобель

@endif

</p>

<p class="card-text"><strong>Помет: </strong>{{ $dog->family }}</p>

@if ($dog->dbres)

<a href="{{ $dog->dbres }}" target="\_blank">

<div class="text-center">{{ $dog->dbres }}</div>

</a>

@endif

</div>

<div class="card-footer">

<small class="text-muted"><?php

$date1 = new DateTime("now");

$date3 = new DateTime($dog->date\_age);

$interval2 = $date1->diff($date3);

$nowT2 = format\_interval($interval2);?>

<?=$nowT2;?></small>

</div>

</div>

</div>

</div>

<div class="container-fluid">

<div class="row justify-content-md-center">

<?php if (isset($photo)) { ?>

@foreach ($photo as $photoDog)

<div class="col-md-3 col-md-offset-2">

<a href="/dog/{{$dog->id\_dogs}}/{{$photoDog->id}}">

<img class="img-thumbnail" src="{{ $photoDog->photo }}" alt="{{ $photoDog->photo }}">

</a>

</div>

@endforeach

<?php } ?>

</div>

</div>

</div>

<div class="col-md-5 col-md-offset-2 row">

<div class="col-12 justify-content-md-center justify-content-md-center">

@if (@isset($firstPhoto))

@foreach($firstPhoto as $firstPhoto1)

<img class="img-fluid" src="{{ $firstPhoto1->photo }}" alt="{{ $firstPhoto1->photo }}">

@endforeach

@elseif (($photo->count()) >= 1)

<?php $valPhoto = $photo[0]->photo; ?>

<img class="img-fluid" src="{{ $valPhoto }}">

@endif

</div>

</div>

</div>

</div>

@endsection

Остается только реализовать механизм ввода нового объекта и редактирования существующего.