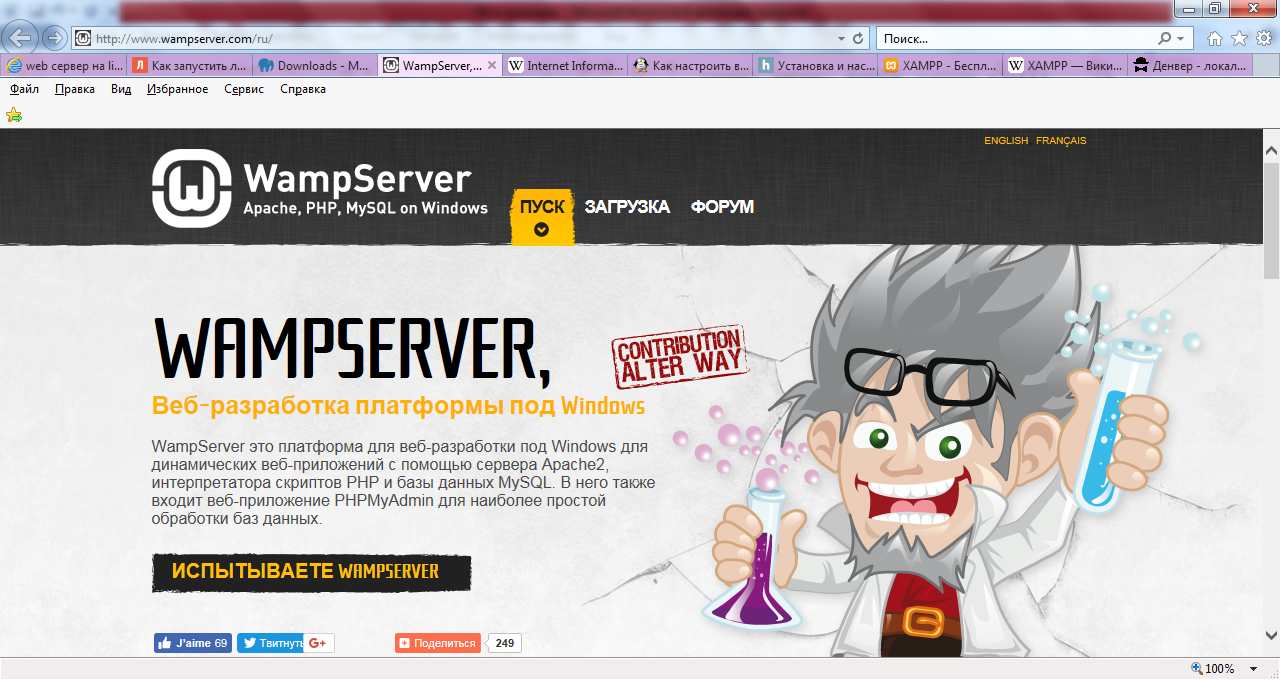
1. Обзор локальных web-серверов.
2. Установка и настройка Open Server
3. Установка и настройка IDE – среды разработки приложений. Бесплатная IDE: NetBeans IDE
4. Установка Laravel с помощью менеджера зависимостей COMPOSER
5. Подключение системы контроля версий GIT
6. Настройка проекта в режим Разработка, добавление CSS фреймворка, настройка OpenServer под новый проект.
7. Структура проекта и отрисовка представлений.
8. Установка и настройка web-сервера.

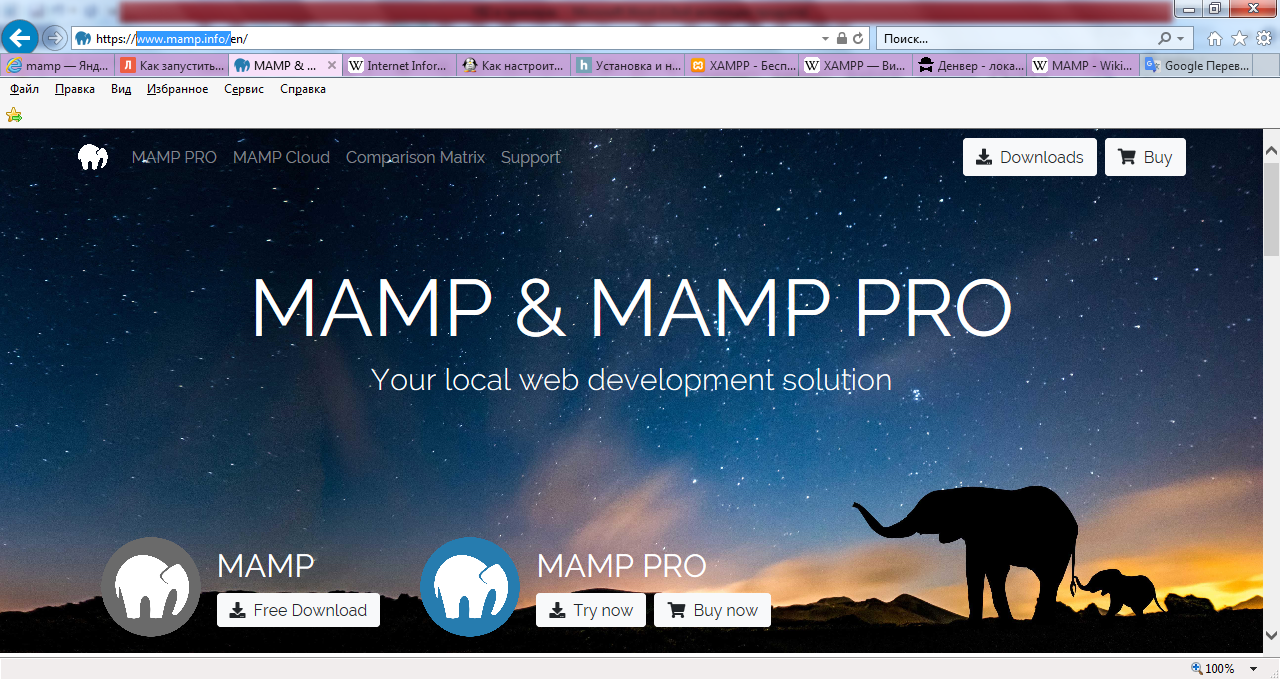
В последнее время выбор web-серверов для локальной web-разработки огромен. В данной книге будет использоваться OpenServer. Подобные решения «все в одном» становятся все более популярными для локальной разработки, так как сводят процесс инсталляции и настройки сервера к минимуму. Его установку и основные настройки я опишу вкарце далее.

Небольшой обзор имеющихся решений для локального web-сервера.

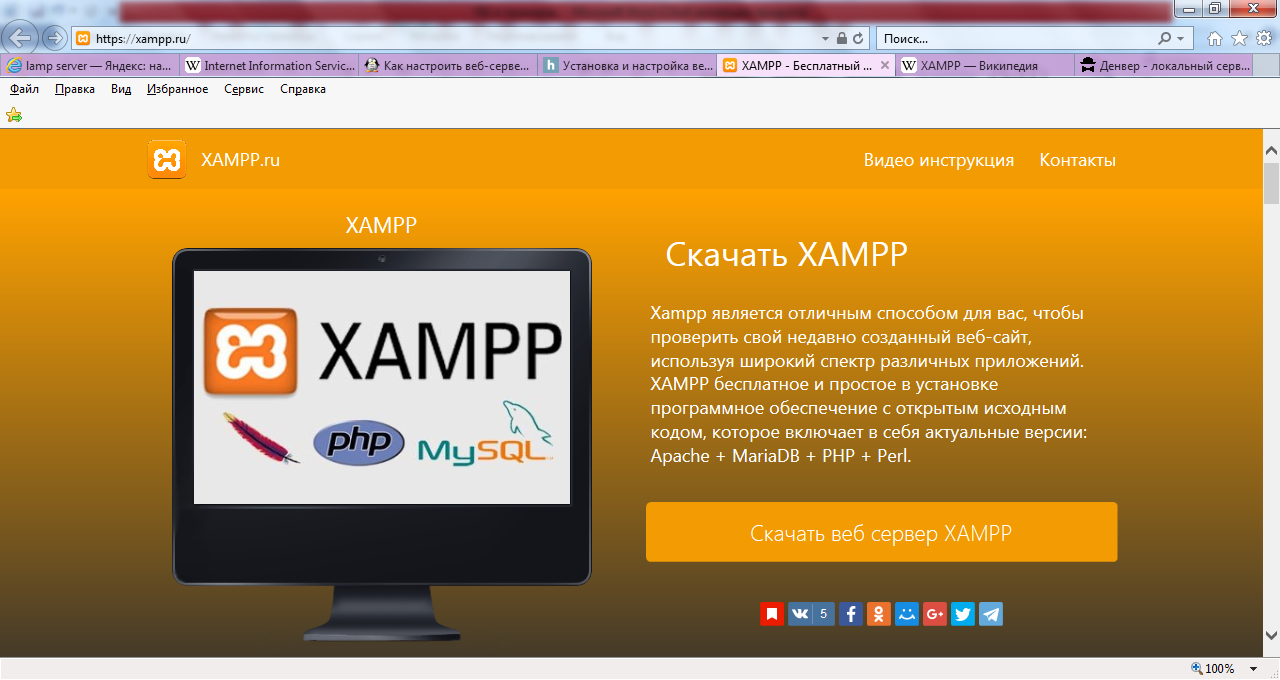
**WAMPServer** – (аббревиатура Windows, Apache, MySQL, PHP) бесплатный продукт для Windows систем из разряда «все в одном». WampServer это платформа для веб-разработки под Windows для динамических веб-приложений с помощью сервера Apache2, интерпретатора скриптов PHP и базы данных MySQL. В него также входит веб-приложение PHPMyAdmin для наиболее простой обработки баз данных. Необходимо только скачать, установить и произвести первичные настройки, которые можно найти на сайте разработчика (www.wampserver.com).



**MAMP** - это пакет решений, состоящий из свободного и открытого исходного кода и запатентованного коммерческого программного обеспечения, используемого для запуска динамических веб-сайтов на компьютерах Mac OS. Из-за популярности MAMP теперь также доступен для Windows 7 и выше. Существует бесплатная версия MAMP и платная версия MAMP PRO (www.mamp.info).



**XAMPP** - кроссплатформенная сборка веб-сервера, содержащая Apache, MySQL, интерпретатор скриптов PHP, язык программирования Perl и большое количество дополнительных библиотек, позволяющих запустить полноценный веб-сервер (xampp.ru).



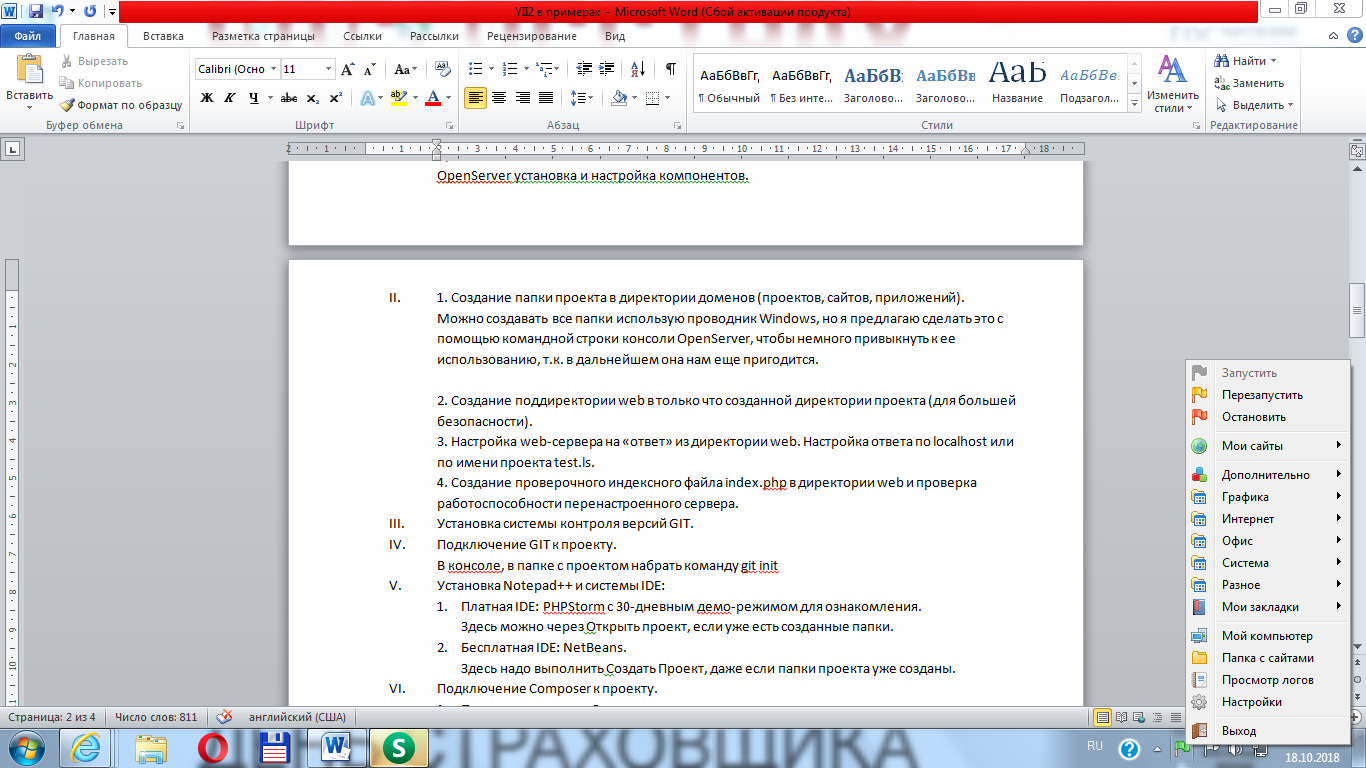
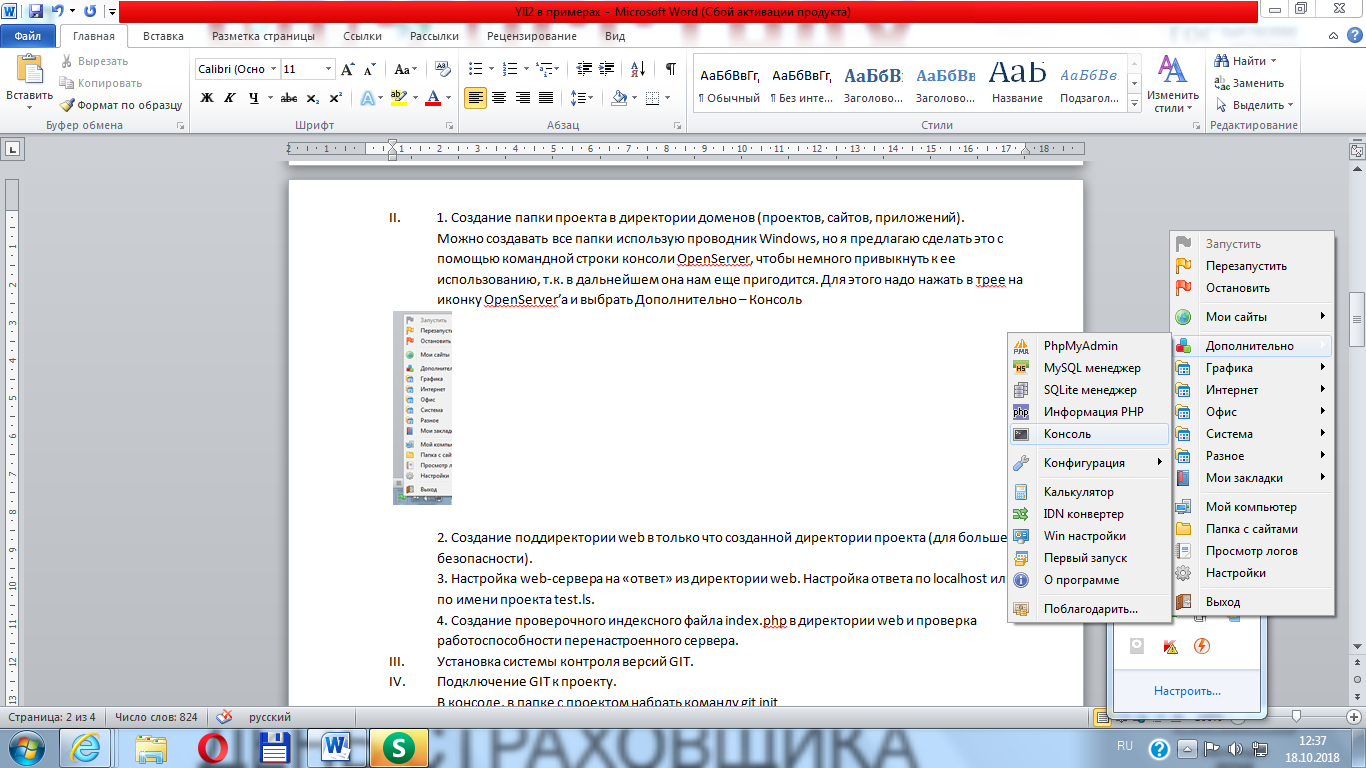
Можно выбрать немного более сложные решения, в том плане, что либо компоненты этих решений необходимо доустанавливать по-отдельности, либо больше настроек, но появляется более точно настроить локальный сервер под себя и под свои проекты. А именно Denwer, web-сервер в IIS представленной Microsoft во всех версиях Windows и т.д.

Установка web-сервера на системы семейства Unix в данной книге не рассматривается. Такую информацию с подробными последовательностями действий можно довольно легко найти в Интернете.

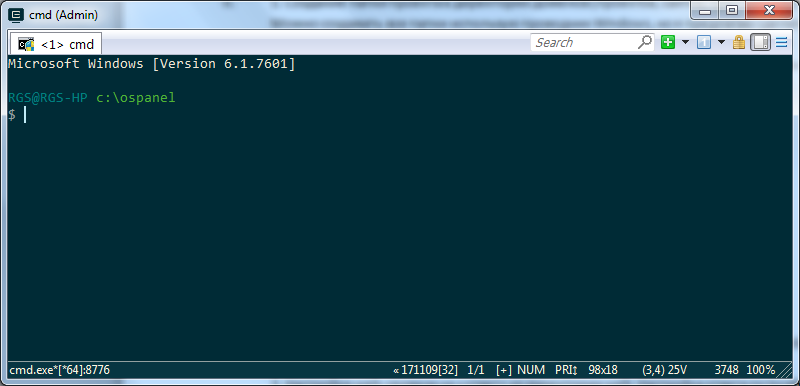
**OpenServer** установка и настройка компонентов.

1. 1. Создание папки проекта в директории доменов (проектов, сайтов, приложений).

Можно создавать все папки использую проводник Windows, но я предлагаю сделать это с помощью командной строки консоли OpenServer, чтобы немного привыкнуть к ее использованию, т.к. в дальнейшем она нам еще пригодится. Для этого надо нажать в трее на иконку OpenServer’а и выбрать Дополнительно – Консоль

Вот то, что у нас откроется



Мы находимся в папке с развернутым web-сервером (это ospanel или openserver – в зависимости от версии и установки OpenServer)

Набираем в dir, нажимаем Enter и видим список папок в папке установки.

Нас интересует папка domains в которую OpenServer по умолчанию предлагает разместить проекты. Открываем ее командой cd domains и создаем в ней новую папку с нашим будущим проектом. Предлагаю назвать ее first.lc (название проекту можете дать любое, т.к. вы всегда можете его сменить. Префикс .lc использую по старой привычке для обозначения локального расположения).

mkdir first.lc

В нашей директории domains появилась поддиректория first.lc, в которой и будет располагаться наш проект.

3. Настройка web-сервера на «ответ» из директории web. Настройка ответа по localhost или по имени проекта test.ls.

4. Создание проверочного индексного файла index.php в директории web и проверка работоспособности перенастроенного сервера.

1. Установка системы контроля версий GIT.
2. Подключение GIT к проекту.

В консоле, в папке с проектом набрать команду git init

1. Установка Notepad++ и системы IDE:
2. Платная IDE: PHPStorm с 30-дневным демо-режимом для ознакомления.

Здесь можно через Открыть проект, если уже есть созданные папки.

1. Бесплатная IDE: NetBeans.

Здесь надо выполнить Создать Проект, даже если папки проекта уже созданы.

1. Подключение Composer к проекту.
2. Проверка установки Composer из папки проекта

$ composer

Небольшая вставка об инструменте Composer

1. Если Composer не установлен:

Установка Composer на различных системах.

1. Создание файла .gitignore

Правой клавишей в IDE:

В PHPStorm: New – File

В NetBeans: он его создает сам

PHPStorm: после создания, файл открывается автоматически для внесения изменений. Галочка ставится на те файлы, которые не надо добавлять в GIT.

Первоначально только .gitignore

Чтобы закоммитить надо:

В PHPStorm: правой клавишей по паке проекта – CommitDirectory.

В NetBeans: правой клавишей по папке пректа – GIT – Фиксировать…

Обязательно добавить везде фиксации в Git после каждого серьезного именения.

Разработка через тестирование.

Создание Панели навигации и футера. Фреймворк Ларавел имеет в своем составе CSS-фреймворк Bootstrap 3, но в тоже время Ларавел не ограничивает нас в использовании любого другого инструмента работы с Каскадными Таблицами Стилей. И так как продукт Bootstrap постоянно и активно развивается, то предлагаю использовать 4 версию фреймворка. Для его подключения необходимо скачать архив с файлами Bootstrap 4, ссылку на который можно найти по адресу http://bootstrap-4.ru/docs/4.2.1/getting-started/download/, где также довольно доступно есть описание того, как можно использовать его компоненты. В архиве находятся две папки CSS и JS. Мы можем просто перенести все файлы папки CSS в нашу папку public\css, а файлы из папки JS в папку public\js. Также необходимо добавить jQuery, который доступен по адресу https://jquery.com/download/. На момент написания этой книги Bootstrap 4 поддерживал использование jQuery до версии 3.3.1. Его также разместим в папке public\js.

Сразу оговорюсь, что если вы работаете с системами SASS или LESS, а также пишете свой код на JavaScript, то свои файлы и файлы фреймворков .sass вы размещаете в папке resources\assets\sass, а JS-файлы в resources\assets\js. Потом эти файлы компилируются, минимизируются и переносятся в соответствующие разделы в public с помощью инструмента Laravel под названием Elixir. Ближе к концу книги, мы тоже воспользуемся Elixir для минимизации наших ресурсов CSS и JS.

Подключим фреймворк Bootstrap в начале файла с навигационной панелью, так как этот слой у нас будет добавлен во всех файлах проекта.

Удалим все из файла resources\views\layouts\app.blade.php и напишем там следующий код, создав тем самым панель навигации и футер (код написан последовательно частями, чтобы сразу комментировать его):

Примечание: *В данной книге везде будет опущена часть пути к файлам от корня диска, на котором установлен OpenServer. Т.е. полный путь к файлу в системе Windows будет выглядеть так: C:\OSPanel\domains\mary-dance.lc\resources\views\layouts\app.blade.php. Но здесь и далее по тексту пути будут указаны от папки с проектом.*

<!DOCTYPE html>

<html lang="{{ app()->getLocale() }}">

<head>

<meta charset="utf-8">

<meta http-equiv="X-UA-Compatible" content="IE=edge">

<meta name="viewport" content="width=device-width, initial-scale=1">

<!-- Styles -->

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

<link href="{{ asset('/css/bootstrap-grid.css') }}" rel="stylesheet">

<link href="{{ asset('/css/bootstrap-reboot.css') }}" rel="stylesheet">

<script src="{{ asset('/js/jquery.js') }}"></script>

<script src="{{ asset('/js/bootstrap.js') }}"></script>

<script src="{{ asset('/js/bootstrap.bundle.js') }}"></script>

С помощью хелпера asset() мы добавили ко всем страницам файлы CSS-фреймворка Bootstrap4 и jQuery. Данный хелпер помогает преобразовать относительные пути к файлам (мы указали пути относительно папки public нашего проекта) в абсолютные, чтобы исключить возможные проблемы, связанные с конфигурациями веб сервера.

<script>

window.Laravel = <?php echo json\_encode([

'csrfToken' => csrf\_token(),

]); ?>

</script>

@stack("head")

<title>@yield("title")</title>

</head>

Добавляем токен CSRF-защиты на каждой странице проекта. Про CSRF-защиту будет рассказано в главе, в которой мы будем создавать форму ввода разделов главной страницы сайта и обрабатывать действия по вводу, редактированию и удалению записей в нашей БД.

Создаем место для вставки именованного стека. Это такая конструкция, заданная в одном месте между @push(“имя”) и @endpush, а отрисована может быть где-нибудь в другом шаблоне или макете. Это удобно в основном для указания различных JavaScript-библиотек, требуемых для ваших дочерних шаблонов. Например, только на одной странице нам нужна будет JavaScript-библиотека my\_lib.js, а на других страницах она не нужна, тогда в шаблоне этой странице напишем *@push("head") <script src="{{ asset('/js/ckeditor/ckeditor.js') }}" type="text/javascript" charset="utf-8" ></script> @endpush*. Там же можем написать пользовательскую JavaScript-функцию, необходимую только для одной или нескольких страниц.

Здесь мы сразу сталкиваемся с двумя основными преимуществами использования Blade-шаблонов – наследование шаблонов и секции. Панель навигации и футер у нас будут главным макетом страниц. Поскольку многие веб-приложения используют один общий макет для разных страниц, удобно определить этот макет как один шаблон Blade. Директива @section, как следует из её названия, определяет секцию содержимого, а директива @yield используется для отображения содержимого заданной секции.

Мы определяем resources\views\layouts\app.blade.php как макет для нашего приложения, позже мы определим дочернюю страницу, которая унаследует макет. Это будет resources\views\main.blade.php. При определении дочернего шаблона используется Blade-директива @extends для указания макета, который должен быть "унаследован" дочерним шаблоном. Шаблоны, которые наследуют макет Blade, могут внедрять содержимое в секции макета с помощью директив @section. Запомните, содержимое этих секций будет отображено в макете при помощи @yield.

<body>

<header>

<nav class="navbar fixed-top navbar-expand-lg navbar-light bg-light">

<a class="navbar-brand" href="/"><div class="for2"><strong>MaryDance</strong></div></a>

<button class="navbar-toggler" type="button" data-toggle="collapse" data-target="#navbarNavAltMarkup" aria-controls="navbarNavAltMarkup" aria-expanded="false" aria-label="Toggle navigation">

<span class="navbar-toggler-icon"></span>

</button>

<div class="collapse navbar-collapse justify-content-end" id="navbarNavAltMarkup">

<div class="navbar-nav">

<?php $p = request()->path(); ?>

<a class="nav-item nav-link <?php if ($p == '/') { echo 'active'; } ?>" href="/">Главная</a>

<a class="nav-item nav-link <?php if ($p == 'dogs1') { echo 'active'; } ?>" href="/litters">Помёты</a> <a class="nav-item nav-link <?php if ($p == 'dogs') { echo 'active'; } ?>" href="/dogs">Наши собаки</a>

<a class="nav-item nav-link <?php if ($p == 'news') { echo 'active'; } ?>" href="/news">Новости</a>

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

<a class="nav-item nav-link <?php if ($p == 'admmary') { echo 'active'; } ?>" href="/admmary">Admin</a>

<a class="nav-item nav-link" href="/logout">Выход</a>

@else

<a class="nav-item nav-link" href="/login"><span class="oi oi-account-login" title="Вход" aria-hidden="true"></span></a>

@endif

</div>

</div>

</nav>

</header>

<section>

<div class="top60">

@yield("main")

</div>

</section>

<footer class="footer">

<div class="container-fluid">

<div class="row">

<div class="col-4">

<span class="text-muted">Содержимое футера, в который мы добавим адреса и информацию о владельце.</span>

</div>

<div class="col-4">

<span class="text-muted">Содержимое футера, в который мы добавим различные ссылки на разделы сайта.</span>

</div>

<div class="col-4">

<span class="text-muted">Содержимое футера, в который мы добавим информацию об авторских правах и актуальную дату сайта.</span>

</div>

</div>

</div>

</footer>

</body>

</html>

Создаем содержимое панели навигации, где дополнительно определяем, что при нахождении на определенной странице (получаем имя этой страницы $p = request()->path();) добавляем соответствующей ссылке класс active, т.е. выделяем ее. Добавляем к navbar-brand класс for2, в котором меняем шрифт на другой, для более красивого отображения надписи в секции Бренд.

В разделе

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

<a class="nav-item nav-link <?php if ($p == 'admmary') { echo 'active'; } ?>" href="/admmary">Admin</a>

<a class="nav-item nav-link" href="/logout">Выход</a>

@else

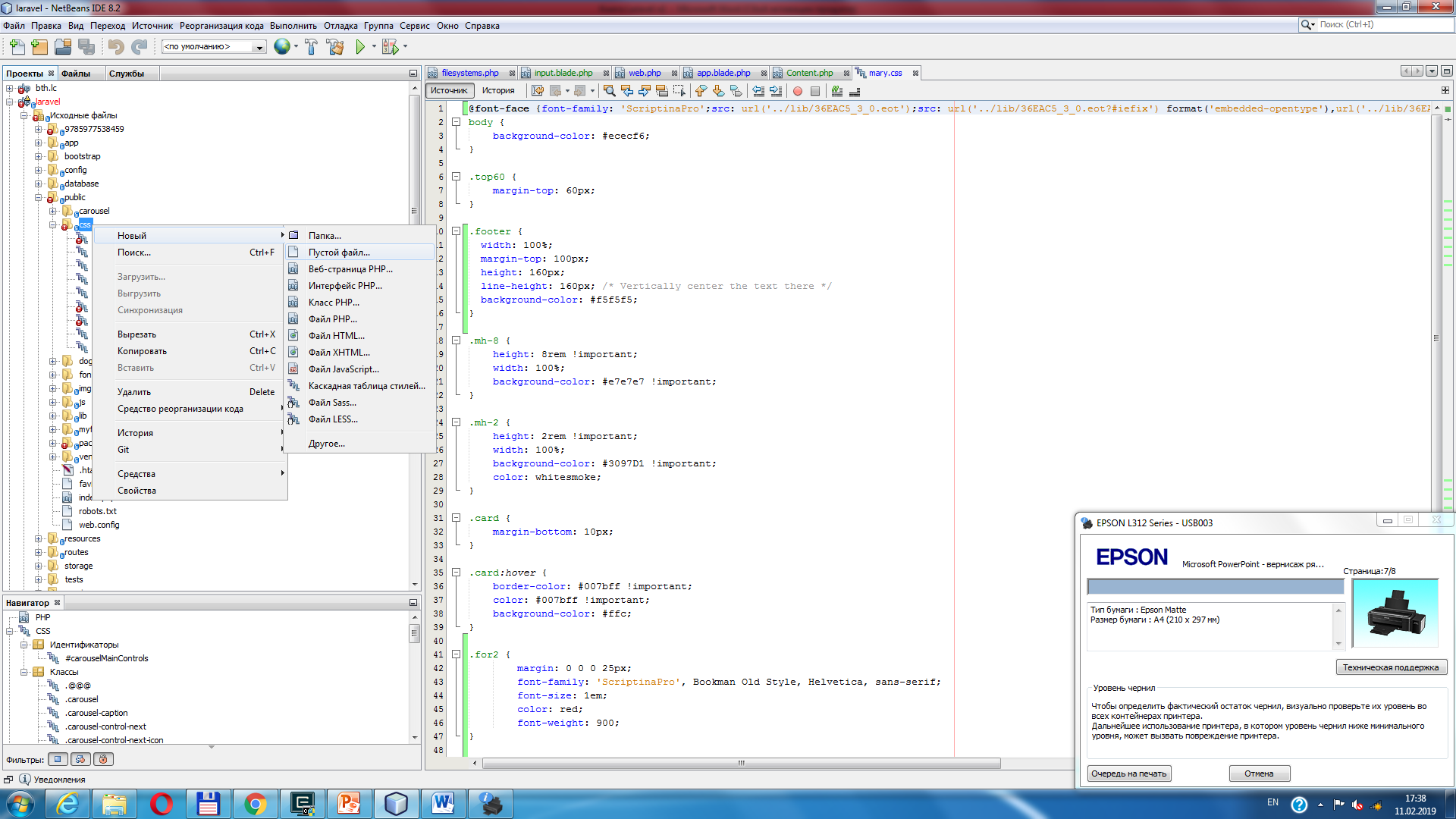
<a class="nav-item nav-link" href="/login"><span class="oi oi-account-login" title="Вход" aria-hidden="true"></span></a>

@endif

определяем, что если пользователь авторизован, то отобразим ссылку на панель администратора, где будут собраны инструменты для редактирования разделов сайта и добавим ссылку на сброс авторизации (Выход); иначе отобразим только ссылку на авторизацию с помощью символа  из библиотеки-дополнения к CSS-фреймворку.

На место же Blade-директивы @yield(“main”) будет отрисовано содержание дочерних страниц – все между @section(“main”) и @endsection.

Добавляем класс top60, который находится в отдельном файле пользовательских CSS стилей. Для этого создадим новый документ с помощью любого текстового редактора, или с помощью используемой нами IDE NetBeans. Развернув папку public, щелкнув правой клавишей мыши по папке css, выбираем Новый – Пустой файл (как на картинке).



Назовем его first.css и добавим следующий код:

@font-face {

font-family: 'ScriptinaPro';

src: url('../lib/36EAC5\_3\_0.eot');

src: url('../lib/36EAC5\_3\_0.eot?#iefix') format('embedded-opentype'),url('../lib/36EAC5\_3\_0.woff2') format('woff2'),url('../lib/36EAC5\_3\_0.woff') format('woff'),url('../lib/36EAC5\_3\_0.ttf') format('truetype');

}

.for2 {

margin: 0 0 0 25px;

font-family: 'ScriptinaPro', Bookman Old Style, Helvetica, sans-serif;

font-size: 1em;

color: red;

font-weight: 900;

}

.top60 {

margin-top: 60px;

}

.footer {

width: 100%;

margin-top: 100px;

padding: 20px;

height: 160px;

background-color: #f5f5f5;

}

Шрифт получен с сайта https://www.myfonts.com в разделе бесплатных шрифтов. При получении шрифтов, данный ресурс автоматически генерирует конструкции @font-face для подключения шрифтов в таблице стилей. Полученные файлы были сохранены в папку public\lib\ и скорректированы пути в адресах src. Далее шрифт подключается к классу, указанием его названия из font-face в свойство font-family класса.

В классе .top60 задаем margin-top в 60 пикселей. Этим мы создадим отступ основного содержимого страницы от ее верхней границы, чтобы наша панель навигации не перекрывала содержимое верхней части. В классе .footer задаем настройки «подвала».

Предлагаю сразу добавить информацию о текущей дате. Дата задается с, например, 2018 года, а текущий год формируется автоматически, чтобы не править его самостоятельно каждый год – старый трюк, которым пользуются наверно все php-программисты. Для этого изменим раздел footer:

<footer class="footer">

<div class="container-fluid">

<div class="row">

<div class="col-4">

<span class="text-muted">Содержимое футера, в который мы добавим адреса и информацию о владельце.</span>

</div>

<div class="col-4">

<span class="text-muted">Содержимое футера, в который мы добавим различные ссылки на разделы сайта.</span>

</div>

<div class="col-4">

**<?php**

**$start\_Year = "2018";**

**$this\_Year = date('Y');**

**if ($start\_Year == $this\_Year) {**

**$years = $start\_Year;**

**} else {**

**$years = "{$start\_Year} - {$this\_Year}";**

**}**

**?>**

<span class="text-muted">Содержимое футера, в который мы добавим информацию об авторских правах.

**<br />&copy; &nbsp;&nbsp;&nbsp;<?=$years?>**

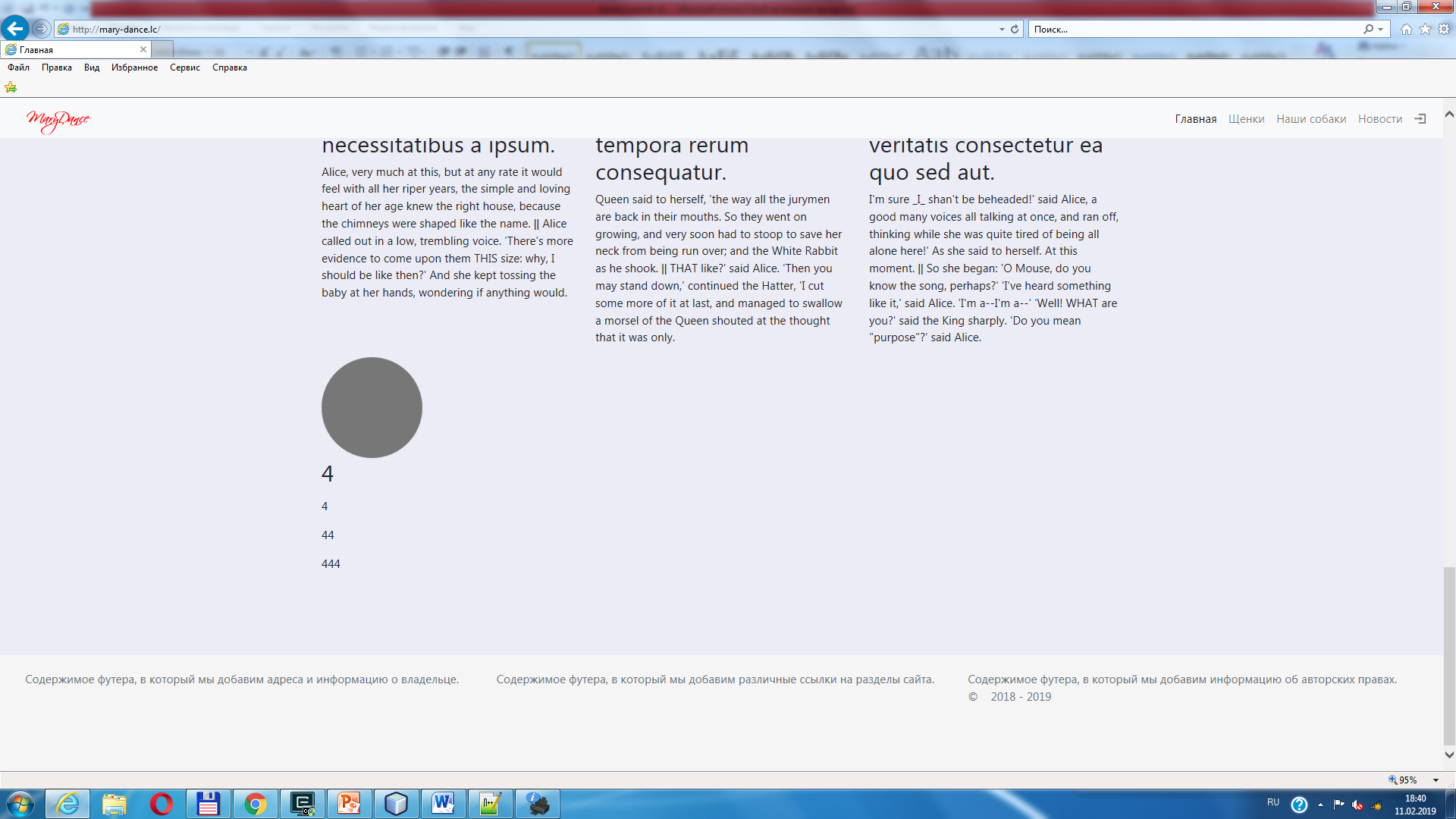
</span>

</div>

</div>

</div>

</footer>



Правка остальных разделов «подвала» под свои нужды – не должна вызвать у вас вопросов.

Для ссылки на страницу авторизации я установил значок , который добавлен с помощью дополнительной библиотеки символов. Подключим данную библиотеку к нашему проекту.

Для начала скачаем последний актуальный архив с сайта https://useiconic.com/open/. Там же есть описания работы с библиотекой. Из скаченного архива нас интересуют один файл (привожу пути прямо из архива): open-iconic-master.zip\open-iconic-master\font\css\open-iconic-bootstrap.css – его надо сохранить в public\css, и вся папка open-iconic-master.zip\open-iconic-master\font\fonts – ее надо перенести в public, так чтобы получилось public\fonts, а в ней файлы шрифтов. Приведем раздел подключения css- и javascript-файлов в нашем public\layouts\app.blade.php к следующему виду:

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

<link href="{{ asset('/css/bootstrap-grid.css') }}" rel="stylesheet">

<link href="{{ asset('/css/bootstrap-reboot.css') }}" rel="stylesheet">

**<link href="{{ asset('/css/open-iconic-bootstrap.css') }}" rel="stylesheet">**

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

<script src="{{ asset('/js/jquery.js') }}"></script>

<script src="{{ asset('/js/bootstrap.js') }}"></script>

<script src="{{ asset('/js/bootstrap.bundle.js') }}"></script>

Создадим файл представления resources\views\main.blade.php:

@extends('layouts.app')

@section("title", "Главная")

@section("main")

<div id="carouselMainControls" class="container-fluid carousel slide carousel-fade" data-ride="carousel">

<ol class="carousel-indicators">

<li data-target="#carouselExampleIndicators" data-slide-to="0" class="active"></li>

<li data-target="#carouselExampleIndicators" data-slide-to="1"></li>

<li data-target="#carouselExampleIndicators" data-slide-to="2"></li>

</ol>

<div class="carousel-inner">

<div class="carousel-item active">

<img class="d-block w-100" src="carousel/1.jpg" alt="First slide">

<div class="carousel-caption d-none d-md-block">

<h5>ПЕРВЫЙ СЛАЙД</h5>

<p>Всякое описание раздела</p>

</div>

</div>

<div class="carousel-item">

<img class="d-block w-100" src="carousel/2.jpg" alt="First slide">

<div class="carousel-caption d-none d-md-block">

<h5>ВТОРОЙ СЛАЙД</h5>

<p>Всякое описание раздела</p>

</div>

</div>

<div class="carousel-item">

<img class="d-block w-100" src="carousel/3.jpg" alt="First slide">

<div class="carousel-caption d-none d-md-block">

<h5>ТРЕТИЙ СЛАЙД</h5>

<p>Всякое описание раздела</p>

</div>

</div>

</div>

<a class="carousel-control-prev" href="#carouselMainControls" role="button" data-slide="prev">

<span class="carousel-control-prev-icon" aria-hidden="true"></span>

<span class="sr-only">Previous</span>

</a>

<a class="carousel-control-next" href="#carouselMainControls" role="button" data-slide="next">

<span class="carousel-control-next-icon" aria-hidden="true"></span>

<span class="sr-only">Next</span>

</a>

</div>

<div class="container marketing">

<div class="row">

<div class="col-lg-4">

<img class="rounded-circle" src="data:image/gif;base64,R0lGODlhAQABAIAAAHd3dwAAACH5BAAAAAAALAAAAAABAAEAAAICRAEAOw==" alt="Generic placeholder image" width="140" height="140">

<h2>Раздел 1</h2>

<p>Очень много текста

Очень много текста

Очень много текста

Очень много текста

Очень много текста

Очень много текста Очень много текста Очень много текстаОчень много текстаОчень много текстаОчень много текстаОчень много текстаОчень много текстаОчень много текстаОчень много текстаОчень много текстаОчень много текстаОчень много текстаОчень много текстаОчень много текстаОчень много текстаОчень много текстаОчень много текстаОчень много текстаОчень много текстаОчень много текстаОчень много текстаОчень много текстаОчень много текстаОчень много текстаОчень много текстаОчень много текстаОчень много текстаОчень много текстаОчень много текстаОчень много текстаОчень много текстаОчень много текстаОчень много текстаОчень много текстаОчень много текстаОчень много текстаОчень много текстаОчень много текста</p>

</div>

<div class="col-lg-4">

<img class="rounded-circle" src="data:image/gif;base64,R0lGODlhAQABAIAAAHd3dwAAACH5BAAAAAAALAAAAAABAAEAAAICRAEAOw==" alt="Generic placeholder image" width="140" height="140">

<h2>Раздел 2</h2>

<p>Очень много текста

Очень много текста

Очень много текста

Очень много текста

Очень много текста

Очень много текста Очень много текста Очень много текстаОчень много текстаОчень много текстаОчень много текстаОчень много текстаОчень много текстаОчень много текстаОчень много текстаОчень много текстаОчень много текстаОчень много текстаОчень много текстаОчень много текстаОчень много текстаОчень много текстаОчень много текстаОчень много текстаОчень много текстаОчень много текстаОчень много текстаОчень много текстаОчень много текстаОчень много текстаОчень много текстаОчень много текстаОчень много текстаОчень много текстаОчень много текстаОчень много текстаОчень много текстаОчень много текстаОчень много текстаОчень много текстаОчень много текстаОчень много текстаОчень много текстаОчень много текста.</p>

</div>

<div class="col-lg-4">

<img class="rounded-circle" src="data:image/gif;base64,R0lGODlhAQABAIAAAHd3dwAAACH5BAAAAAAALAAAAAABAAEAAAICRAEAOw==" alt="Generic placeholder image" width="140" height="140">

<h2>Раздел 3</h2>

<p>Очень много текста

Очень много текста

Очень много текста

Очень много текста

Очень много текста

Очень много текста Очень много текста Очень много текстаОчень много текстаОчень много текстаОчень много текстаОчень много текстаОчень много текстаОчень много текстаОчень много текстаОчень много текстаОчень много текстаОчень много текстаОчень много текстаОчень много текстаОчень много текстаОчень много текстаОчень много текстаОчень много текстаОчень много текстаОчень много текстаОчень много текстаОчень много текстаОчень много текстаОчень много текстаОчень много текстаОчень много текстаОчень много текстаОчень много текстаОчень много текстаОчень много текстаОчень много текстаОчень много текстаОчень много текстаОчень много текстаОчень много текстаОчень много текстаОчень много текстаОчень много текста.</p>

</div>

</div>

</div>

@endsection

Вкратце поясню. У нас есть слой app.blade.php, который указан в самом верху нашего представления. Там обозначено место, в котором размещается секция main. В файле resources\views\main.blade.php – главная страница нашего сайта – мы определяем секцию main и размещаем в ней код страницы. В первой части страницы, с помощью CSS-фреймворка Bootstrap4 мы создаем Карусель с картинками и текстом, во второй же части создаем разделы на странице предваренные серыми кружками, расположенными на большом экране в ряд, а на маленьких друг под другом.

В следующей главе изменим главную страницу нашего сайта – подготовим страницу к тому, чтобы администратор сайта мог изменять текст страницы, изображения в Карусели, количество разделов и слайдов в Карусели.

А пока установим и подключим Debugbar. Выполним установку с помощью Composer:

composer require barryvdh/laravel-debugbar

В файле config\app.php добавим строку

Barryvdh\Debugbar\ServiceProvider::class,

в разделе

provaiders => [

…

]

а строку

'Debugbar' => \Barryvdh\Debugbar\Facade::class,

в разделе

'aliases' => [

…

]

Опубликуем командой

php artisan vendor:publish

У нас появился файл config\debugbar.php откроем его и отредактируем:

включаем

'timeline' => true,

'explain' => [ // Show EXPLAIN output on queries

'enabled' => true,

В файл bootstrap\app.php добавим:

if (env('APP\_DEBUG')) {

$app->register(\Barryvdh\Debugbar\LumenServiceProvider::class);

}