



МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РФ  
УЛЬЯНОВСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

Факультет Математики, Информационных и Авиационных технологий  
Кафедра Информационных технологий

**КУРСОВАЯ РАБОТА**  
**по дисциплине “ Интернет-программирование”**  
**по теме “Разработка WEB-приложения с помощью фреймворка Laravel”**

Математическое обеспечение и администрирование информационных систем  
(Бакалавриат) – 02.03.03

---

(наименование и номер специальности(направления))

Проект выполнил  
студент

МОАИС-0-19/1  
(группа)

---

(подпись, дата)

/Шарифзанов И. И.  
(Ф.И.О.)

Научный руководитель

доцент  
(должность)

---

(подпись, дата)

/Санкин Н. Ю.  
(Ф.И.О.)

---

оценка

УЛЬЯНОВСК  
2022 г.

# **Оглавление**

## **1. Введение**

- Теоретическая часть
- Выборы языка программирования
- Выборы и описания программных средств реализации задачи

## **2. Практическая часть.**

- Создание субд
- Реализация проекта
- Шаблоны и наработки

## **3. Заключение**

## **4. Список литературы**

## **5. Программный код приложения**

## **ВВЕДЕНИЕ**

Развитие персональных компьютеров привело к появлению компьютерных сетей, которые обеспечили обмен данными между вычислительными устройствами (компьютерами). Объединение локальных сетей в глобальную сеть повлекло за собой появление – Интернет.

Интернет проник во все сферы человеческой жизни, начиная от доступного источника информации и общения с друзьями в интерактивном режиме и кончая осуществлением покупок онлайн.

С развитием Интернета возникло такое понятие как Web-технология, то есть начали возникать веб-сайты различной тематической направленности.

Web-технология полностью перевернула представления о работе с информацией, да и с компьютером вообще. Традиционные параметры развития вычислительной техники - производительность, пропускная способность, емкость запоминающих устройств - не учитывали главного "узкого места" системы – простоты взаимодействия вычислительной техники с человеком.

С развитием персональных компьютеров и появлением новых технологий внедряемых в сфере информационных систем упростило взаимодействие человека и компьютера. [1]

С развитием глобальной сети Интернет возникла необходимость:

- удаленного общения между людьми посредством отправки моментальных сообщений,

- получение нужной информации,
- научные расчеты на удаленных кластерах (кластер – группа компьютеров, объединённых высокоскоростными каналами связи и представляющая с точки зрения пользователя единый аппаратный ресурс, который может использоваться для расчетов) без непосредственного посещения места расположения самого кластера.

Создание Web-сайтов является одной из важнейших технологий разработки ресурсов в Интернете. Сайты могут быть тематическими, содержит полезную информацию для конкретной целевой аудитории, сайты широкого спектра (например, новостные сайты, сайты с прогнозом погоды), сайты коммерческой фирмы, частного лица или образовательного учреждения. При этом сайт должен обеспечивать доступ к размещенной на нем информации круглосуточно.[1]

**Актуальность** курсового проекта состоит в том, что в настоящее время нет сайтов по данной тематике, либо они предоставляют пользователю минимум информации, поэтому пользователям необходимо пользоваться несколькими сайтами, что не всегда удобно.

**Цель работы:** Создание Web-приложение для создания статей с использованием php-фреймворка Laravel. То есть можно выделить несколько основных пунктов, которые должны быть выполнены. Во-первых, приложение должно стабильно работать. Во-вторых, корректное отображение он должен нормально открываться во всех основных браузерах, а также быть доступным к просмотру с помощью разных устройств. В-третьих, создать функционал администратора: возможность добавления, удаления, просмотра.

Для достижения поставленной цели необходимо решить следующие задачи:

- провести анализ предметной области
- определение сущностей в базе данных
- разработать структуру и дизайн веб-приложения
- выполнить реализацию разработанных макетов
- провести тестирование пользовательского интерфейса

Вся работа выполнена с помощью Visual Studio Code 2022. Для написания сценариев использовался PHP 8.0.2 и использовался фреймворк Laravel 9.19. Работа с базами данных осуществлялась с помощью поставщика БД MySQL через локальный сервер XAMPP и phpMyAdmin. Для разработки

фронтенда в качестве базовых инструментов используются: HTML (для создания базовой структуры страниц и контента), CSS (для стилизации внешнего вида) и JavaScript (для добавления интерактивности)

## ТЕРМИНЫ И ОПРЕДЕЛЕНИЯ

Web-приложение - это приложение, которое использует клиент-серверное соединение, где в качестве клиента выступает браузер, а в качестве сервера - веб-сервер. Основная суть веб-приложения заключается в том, что его логика заключена между сервером и клиентом так, что вся необходимая информация содержится на сервере, соединение и обмен информацией происходит по сети. [11]

PHP — это серверный язык сценариев. это используется для разработки статических веб-сайтов или динамических веб-сайтов или веб-приложений.

Гипертекст – это текст, заключенный в теги HTML

Фронтенд (англ. frontend) — это разработка пользовательских функций и интерфейса.

Пользовательский интерфейс — это набор программных и аппаратных средств, обеспечивающих взаимодействие пользователя с компьютером

Бэкенд (англ. backend) — это внутренняя часть сайта и сервера, то есть это программно-аппаратный комплекс, который позволяет сайту и серверу корректно работать.

Веб-сервер — это компьютерное программное обеспечение и базовое оборудование, которое принимает запросы через HTTP

HTTP (англ. HyperText Transfer Protocol — «протокол передачи гипертекста») — это протокол связи, используемый для просмотра веб-страниц.

Фреймворк (англ. framework – каркас) — программная платформа, определяющая структуру программной системы; программное обеспечение, облегчающее разработку и объединение разных компонентов большого программного проекта.

Миграции — это что-то вроде системы контроля версий для базы данных. Они позволяют изменять структуру БД, в то же время оставаясь в курсе изменений прошлых данных.

## ИСПОЛЬЗУЕМЫЕ ЯЗЫКИ ПРОГРАММИРОВАНИЯ

В этой главе введем общие понятия используемых языков программирования в разработке сайтов и также область их применения. Более подробно будет описано все ниже.

Для создания Web-приложений существует много специализированных языков программирования. Перечислим некоторые из них и далее дадим характеристику:

### 1.1 HTML

Это язык гипертекстовой разметки текста, предназначенных для отображения в веб-браузере элементы: текст, картинки, таблицы и видео. При сохранении HTML-файла в месте размещения документа создается папка, в которую помещаются сопутствующие ему графические элементы оформления.

С помощью языка разметки HTML браузер делает запрос по адресу, который ввел пользователь, и получает файл в формате «.html». Браузер распознаёт код, выбирает знакомые для себя сигналы: понимает, что написать словами, где поставить заголовок и какой именно. Таким образом код из файла преобразуется в необходимые визуальные объекты.

Также можно посмотреть за компьютером любой сайт:

1. Заходим в любой браузер
2. Нажимаем F12

The screenshot shows the Yandex browser window. At the top, there's a dark header bar with various icons and a 'Log in' button. Below it is a navigation bar with back, forward, and search buttons. The main content area has a Yandex logo and a search bar labeled 'Finds everything'. At the bottom, there are language selection buttons for Russia, Belarus, Kazakhstan, Uzbekistan, and Turkey. The developer tools are open, specifically the CSS panel, which displays the current styles for the selected element. The code editor at the bottom shows the HTML and CSS for the page being viewed.

Рис 1. Вывод конкретной HTML-код страницы через браузер  
И мы можем увидеть, как выглядит HTML этой конкретной страницы блога.

## Возможности HTML

Язык HTML помогает структурировать информацию. Это могут быть команды добавить картинку, разбить текст на абзацы, добавить заголовки и подзаголовки, создать список, вставить таблицу, нарисовать какой-либо элемент. Вёрстка страницы будет выглядеть так, как её закодирует разработчик.

Что представляет собой HTML: набор команд, который после обработки превращается в визуальное представление. Команда — это тег, состоящий из имени, расположенного между символами «меньше» и «больше» (<h1>). Есть парные теги. Правила вложенности у каждого свои. Например, вот так выглядит строка в списке: [13]

```
<ul>  
<li>Название элемента в списке</li>  
</ul>
```

Из-за незакрытого или неправильно закрытого тега вёрстка может ломаться.

Простая HTML-страница состоит из трёх тегов: <html>, <head> и <body>. В документе <head> и <body> используют только раз.

<html> идёт в документе сразу после «доктайпа» — обозначения типа документа. По нему браузер определяет версию HTML и понимает, как правильно отобразить страницу. (<!DOCTYPE html>)

<head> хранит важную служебную информацию — заголовок страницы и кодировку.

<body> хранит содержимое страницы. Именно так код отображается в браузере. Тексты и картинки добавляются внутрь этого тега.

Рассмотрим некоторые из других наиболее часто используемых тегов HTML

<header> определяет вводную часть веб-страницы. Содержит логотип, элементы навигации, панель поиска.

**<nav>** управляет элементами навигации: контакты, информация, часто задаваемые вопросы и другое.

**<main>** содержит основные разделы HTML-документа, кроме **<header>** и **<footer>**. В идеале во всём HTML-документе используется только один раз.

**<img>** помогает добавить картинку, но сам по себе он не имеет смысла. Нам потребуется прописать внутри адрес, ведущий к картинке. Это делается с помощью атрибута `src`:

```

```

**<article>** структурирует информацию, работая с комбинацией текста, изображений, видео.

**<section>** определяет конкретный раздел веб-страницы. Это может быть раздел «Витрина», «О нас», «Контакты» или другие. [13]

Теги передают фактическое значение — элементы не размещаются автоматически там, где они должны быть. Это делается уже с помощью CSS.

## Преимущества HTML

- Подходит почти для всех браузеров, а код можно написать в любом текстовом редакторе.
- HTML можно использовать бесплатно.
- Чтобы освоить азы языка разметки гипертекста, достаточно пары часов. У него простой и понятный синтаксис.
- Подходит для профессиональных разработчиков сайтов и для любителей.
- Статические сайты на HTML, которым не нужно обращаться к базам данных, быстро загружаются, потому что меньше весят и в них отсутствует «мусорный код». Они дешевле в разработке, и для таких сайтов подойдёт самый дешёвый хостинг.
- HTML интегрируется с другими языками программирования.

## Недостатки HTML

- Все повторяющиеся изменения на сайте придётся вносить вручную. Например, заменить на каждой странице пункты меню и навигации. То же самое с созданием новых страниц — даже если структура повторяется, нужно создавать каждую страницу отдельно.

- Если нужно, чтобы сайт выглядел красиво и реагировал на действия пользователя, придётся учить ещё CSS и JS.

## 1.2 JAVASCRIPT

Язык программирования JavaScript разработан фирмой Netscape для создания интерактивных HTML-документов. Это объектно-ориентированный язык разработки, выполняющейся как на стороне клиента, так и на стороне сервера. Синтаксис языка очень похож на синтаксис Java – поэтому его называют – Java-подобным.

Основные области применения JavaScript делятся на следующие категории:

- динамическое создание документа с помощью сценария;
- оперативная проверка достоверности заполняемых пользователем полей форм HTML до передачи их на сервер;
- создание динамических HTML-страниц совместно с каскадными таблицами стилей и объектной моделью документа;
- взаимодействие с пользователем при решении "локальных" задач, решаемых приложением JavaScript, встроенным в HTML-страницу.

## 1.3 CSS

CSS — формальный язык описания внешнего вида документа (веб-страницы), написанного с использованием языка разметки (чаще всего HTML или XHTML).

CSS используется создателями веб-страниц для задания цветов, шрифтов, стилей, расположения отдельных блоков и других аспектов представления внешнего вида этих веб-страниц. Основной целью разработки CSS является ограждение и отделение описания логической структуры веб-страницы (которое производится с помощью HTML или других языков разметки) от описания внешнего вида этой веб-страницы (которое теперь производится с помощью формального языка CSS). Такое разделение может увеличить доступность документа, предоставить большую гибкость и возможность управления его представлением, а также уменьшить сложность и повторяемость в структурном содержимом. [14]

Кроме того, CSS позволяет представлять один и тот же документ в различных стилях или методах вывода, таких как экранное представление, печатное представление, чтение голосом (специальным голосовым браузером или программой чтения с экрана) или при выводе устройствами.

### **1.3 MySQL**

MySQL – является одной из самых популярных и распространенных СУБД (система управления базами данных) в Интернете. СУБД MySQL не предназначена для работы с большими объемами информации, но ее применение идеально подходит для веб-сайтов, как небольших, так и достаточно крупных.

MySQL отличается хорошей скоростью работы, надежностью, гибкостью. Работа с ней не вызывает больших трудностей. Поддержка сервера MySQL автоматически включена в поставку PHP.

К достоинствам СУБД MySQL следует также отнести тот факт, что она распространяется на условиях общей лицензии GNU (GPL, GNU Public License), то есть бесплатно.

Раньше для долговременного хранения информации использовали файлы, в которые записывали необходимую информацию, которую затем считывали для последующей работы либо обработки.

Очень часто в ряде задач программирования встречаются требования длительного хранения информации, с последующей обработкой сохраненной информации. Для этой цели лучше всего подходят базы данных. Базы данных сами заботятся о безопасности информации, и её сортировке и позволяют извлекать и размещать информацию при помощи одного запроса. Код с использованием базы данных получается более компактным, и отлаживать его гораздо легче. Скорость выборки информации из базы данных происходит значительно быстрее, чем из файлов.

Базы данных хорошо подходят и при создании Web-приложений, где возникают задачи: подсчёт посетителей в счётчике, хранение сообщений на форуме, удалённое управление содержанием информации на сайте и т.д.

Приложения, написанные на PHP, использующие для хранения информации базу данных (в частности MySql), всегда работают быстрее приложений, использующие файлы.

Основным достоинством MySQL является то, что она берёт на себя всю работу с жёстким диском и делает это очень эффективно.

### **1.4 PHP**

Главная область применения PHP - написание скриптов, работающих на стороне сервера; таким образом, PHP способен выполнять все то, что выполняет любая другая программа CGI, например, обрабатывать данные html-форм, генерировать динамические html-страницы и чтение/запись в базу данных. Но PHP способен выполнять намного больше. [3]

Существуют три основных области применения PHP.

1) Создание скриптов для выполнения на стороне сервера. PHP традиционно и наиболее широко используется именно таким образом. Для этого вам будут необходимы три вещи. Интерпретатор PHP (в виде программы CGI или серверного модуля), веб-сервер и браузер. Для того чтобы можно было просматривать результаты выполнения PHP-скриптов в браузере, нужен работающий веб-сервер и установленный PHP. Просмотреть вывод PHP-программы можно в браузере, получив PHP-страницу, сгенерированную сервером. В случае, если вы просто экспериментируете, вы вполне можете использовать свой домашний компьютер вместо сервера.

2) Создание скриптов для выполнения в командной строке. Вы можете создать PHP-скрипт, способный запускаться без сервера или браузера. Все, что вам потребуется - парсер PHP. Эти скрипты также могут быть использованы в задачах простой обработки текстов.

3) Создание оконных приложений (GUI-приложения), выполняющихся на стороне клиента. Возможно, PHP является не самым лучшим языком для создания подобных приложений. Также, вы можете создавать и кросс-платформенные приложения. PHP-GTK является модулем PHP и не поставляется вместе с основным дистрибутивом PHP.

PHP доступен для большинства операционных систем, включая Linux, многие модификации Unix Microsoft Windows, macOS, RISC OS и многие другие. Также в PHP включена поддержка большинства современных веб-серверов, таких как Apache, IIS и многих других. В принципе, подойдёт любой веб-сервер, способный использовать бинарный файл FastCGI PHP, например, lighttpd или nginx. PHP может работать в качестве модуля или функционировать в качестве процессора CGI.

Таким образом, выбирая PHP, вы получаете свободу выбора операционной системы и веб-сервера. Более того, у вас появляется выбор между использованием процедурного или объектно-ориентированного программирования (ООП) или же их сочетания.

PHP способен генерировать не только HTML. Доступно формирование изображений, файлов PDF и даже роликов Flash, создаваемых «на лету». PHP также способен генерировать любые текстовые данные, такие, как XHTML и другие XML-файлы. PHP может осуществлять автоматическую генерацию таких файлов и сохранять их в файловой системе вашего сервера вместо того, чтобы отдавать клиенту, организуя, таким образом, серверный кеш для вашего динамического контента.

Одним из значительных преимуществ PHP является поддержка широкого круга баз данных. Создать скрипт, использующий базы данных - невероятно просто. Можно воспользоваться модулем, специфичным

для отдельной базы данных (таким как mysql) или использовать уровень абстракции от базы данных, такой как PDO, или подсоединиться к любой базе данных, поддерживающей Открытый Стандарт Соединения Баз Данных (ODBC), с помощью одноимённого модуля ODBC.

PHP также поддерживает взаимодействие с другими сервисами через такие протоколы, как LDAP, IMAP, SNMP, NNTP, POP3, HTTP, COM (на платформах Windows) и многих других. Кроме того, вы получаете возможность работать с сетевыми сокетами напрямую. PHP поддерживает стандарт обмена сложными структурами данных WDDX практически между всеми языками веб-программирования. Обращая внимание на взаимодействие между различными языками, следует упомянуть о поддержке объектов Java и возможности их использования в качестве объектов PHP.

PHP имеет много возможностей по обработке текста, включая регулярные выражения Perl (PCRE) и много других модулей и инструментов для обработки и доступа к XML-документам. В PHP обработка XML-документов стандартизирована и происходит на базе мощной библиотеки libxml2, расширив возможности обработки XML добавлением новых модулей SimpleXML, XMLReader и XMLWriter. [15]

## ОСОБЕННОСТИ АРХИТЕКТУРЫ LARAVEL

Идеальным вариантом для разработки нашей системы был бы фреймворк, который бы поставлялся с частью уже готового «стандартного» функционала, чтобы свести минусы выбранного способа разработки к минимуму. Для облегчения и ускорения процесса написания кода и запуска Web-приложения будем использовать PHP-фреймворк Laravel.

Используя фреймворк, не нужно самостоятельно писать большое количество кода и тратить время на поиск потенциальных просчетов и ошибок [2].

### 2.1 ОБЩИЙ ОБЗОР

Laravel — бесплатный веб-фреймворк с открытым кодом, предназначенный для разработки с использованием архитектурной модели MVC.

Архитектурная модель(каркас) MVC (Model-View-Controller, Модель-Представление-Контроллер) – схема совместного использование нескольких шаблонов проектирования, позволяющая разделить данные, представление и обработку действий пользователя на три отдельных компонента:

1) Модель – предоставляет знания: данные и методы работы с этими данными, реагирует на запросы, изменяя своё состояние. Не содержит информации, как эти знания можно визуализировать (папка app/models). [4]

2) Представление – отвечает за визуализацию информации, то есть html-файлы (папка resources/views)

3) Контроллер – обеспечивает связь между пользователем и системой: контролирует ввод данных пользователем и использует модель и представление для реализации необходимой реакции (папка app/http/controllers)

Основная цель применения этой схемы состоит в отделении бизнес-логики (модели) от её визуализации (представления, вида). За счет такого разделения повышается возможность повторного использования.

Диаграмма использования MVC в Laravel хорошо представлена на рисунке 2.

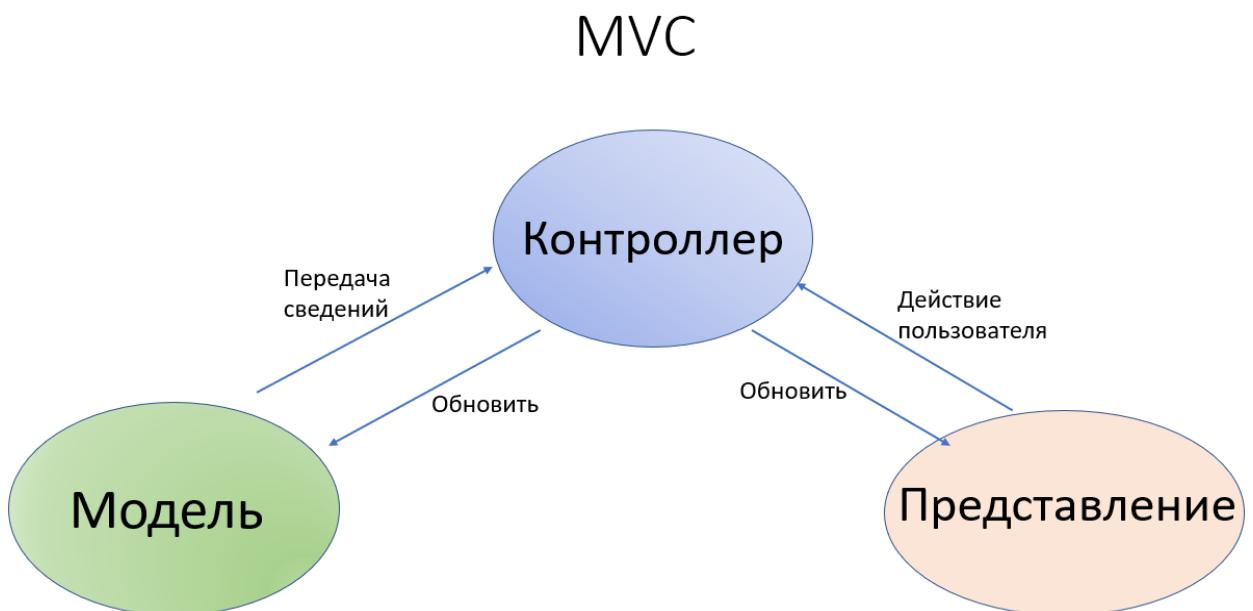


Рис. 2 MVC в Laravel

## 2.2 КЛЮЧЕВЫЕ ОСОБЕННОСТИ LARAVEL

В основе архитектуры Laravel, лежат следующие возможности:

1) Пакеты — позволяют создавать и подключать модули в формате Composer [10] к приложению на Laravel. Многие дополнительные возможности уже доступны в виде таких модулей.

2) Eloquent ORM — реализация шаблона проектирования ActiveRecord на PHP. Позволяет строго определить отношения между объектами базы данных.

3) Логика приложения — часть разрабатываемого приложения, объявленная либо при помощи контроллеров, либо маршрутов (функций-замыканий).

4) Обратная маршрутизация связывает между собой генерируемые приложением ссылки и маршруты, позволяя изменять последние с автоматическим обновлением связанных ссылок. При создании ссылок с помощью именованных маршрутов Laravel автоматически генерирует конечные URL.

5) REST-контроллеры — дополнительный слой для разделения логики обработки GET- и POST-запросов HTTP.

6) Автозагрузка классов — механизм автоматической загрузки классов PHP без необходимости подключать файлы их определений в include. Загрузка по требованию предотвращает загрузку ненужных компонентов; загружаются только те из них, которые действительно используются.

7) Составители представлений (англ. view composers) — блоки кода, которые выполняются при генерации представления (шаблона).

8) Инверсия управления (англ. Inversion of Control) — позволяет получать экземпляры объектов по принципу обратного управления. Также может использоваться для создания и получения объектов-одиночек (англ. singleton).

9) Миграции — система управления версиями для баз данных. Позволяет связывать изменения в коде приложения с изменениями, которые требуется внести в структуру БД, что упрощает развёртывание и обновление приложения.

10) Модульное тестирование (юнит-тесты) — играет очень большую роль в Laravel, который сам по себе содержит большое число тестов для предотвращения регрессий (ошибок вследствие обновления кода или исправления других ошибок).

11) Страницочный вывод (англ. pagination) — упрощает генерацию страниц, заменяя различные способы решения этой задачи единым механизмом, встроенным в Laravel.

Также в стандартный установщик Laravel встроен модуль регистрации и аутентификации плюс уже сформированные миграции для таблиц пользователей с ролями, что сводит минусы выбора данного фреймворка вместо CMS, практически к нулю.

### **2.3 РАЗВОРАЧИВАНИЕ ПРОЕКТА**

Если на компьютере уже установлены PHP и Composer, то вы можете создать новый проект Laravel напрямую с помощью Composer. Composer — пакетный менеджер, который управляет зависимостями проекта и генерирует

autoload файл, который определяет механизм загрузки классов зависимостей и классов проекта.[10] После того как приложение было создано, вы можете запустить локальный сервер разработки Laravel с помощью команды serve Artisan CLI:

```
composer create-project laravel/laravel [example-app]
cd example-app
php artisan serve
```

Вы также можете увидеть созданные каталоги в проекте:

```
> app
> bootstrap
> config
> database
> lang
> public
> resources
> routes
> storage
> tests
> vendor
```

Рис. 3 Структура каталогов в Laravel

Опишем какую функцию выполняет каждый из каталогов:

Каталог *app* содержит основной код вашего приложения. Почти все классы в приложении будут в этом каталоге.

Каталог *bootstrap* содержит файл app.php, который загружает фреймворк. В этом каталоге также находится каталог cache, содержащий файлы, сгенерированные фреймворком для оптимизации производительности, например, файлы кеша маршрутов и служб. Обычно вам не нужно изменять какие-либо файлы в этом каталоге.

Каталог *config*, как следует из названия, содержит все файлы конфигурации вашего приложения. Для понимания работы с данными можно прочитать все эти файлы и ознакомиться со всеми доступными параметрами.

Каталог *database* содержит миграции баз данных, фабрики моделей и наполнители. При желании можно использовать этот каталог для хранения SQLite БД.

Каталог *public* содержит файл index.php и запускает его, также поступают запросы и содержит картинки, JavaScript, CSS, которые будут доступны пользователям. [10]

Каталог *resources* содержит шаблоны, а также необработанные, нескомпилированные ресурсы, например, JavaScript или CSS. В этом каталоге также находятся все языковые файлы.

Каталог *routes* содержит все определения маршрутов для приложения. По умолчанию в Laravel входит несколько файлов маршрутов: **web.php**, **api.php**, **console.php** и **channels.php**.

Теперь рассмотрим файлы внутри каталога *routes*:

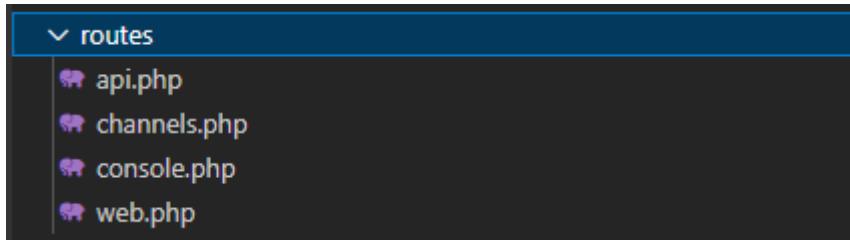


Рис. 4 Структура внутри каталога *routes*

Файл **web.php** содержит маршруты, которые RouteServiceProvider размещает в группе посредников *web*, обеспечивающие состояние сессии, защиту CSRF и шифрование файлов куки. Если в приложении не предполагается сохранение состояния и RESTful API, то, все ваши маршруты будут определены в файле *web.php*.

Файл **api.php** содержит маршруты, которые RouteServiceProvider размещает в группе посредников *api*. Эти маршруты не предполагают сохранения состояния, поэтому запросы, поступающие в приложение через эти маршруты, предназначены для аутентификации с помощью токенов и не имеют доступа к состоянию сессии.

В файле **console.php** можно определить все анонимные консольные команды. Каждое замыкание привязано к экземпляру команды, что обеспечивает простой подход к взаимодействию с методами ввода-вывода каждой команды. Несмотря на то, что этот файл не определяет маршруты HTTP, он определяет точки входа (маршруты) в консольное приложение.

В файле **channels.php** можно зарегистрировать все каналы трансляции событий, которые поддерживает приложение.

Каталог *storage* содержит журналы (логи), скомпилированные шаблоны Blade, файлы сессий, кеша и другие файлы, созданные фреймворком. Этот каталог разделен на каталоги *app*, *framework*, и *logs*.

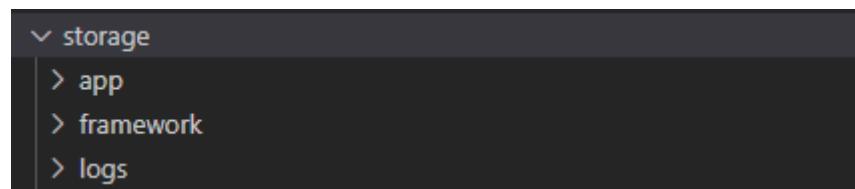


Рис. 5 Структура внутри каталога *storage*

Каталог *app* может использоваться для хранения любых файлов, созданных приложением. [10]

Каталог *framework* используется для хранения файлов и кешей, сгенерированных фреймворком.

Каталог *logs* содержит файлы журнала приложения.

Каталог *storage/app/public* может использоваться для хранения файлов, созданных пользователями, таких как аватары профиля, которые должны быть общедоступными.



Рис. 6 Структура внутри каталога *storage/app/public*

Каталог *tests* содержит автоматизированные тесты. Примеры модульного PHPUnit и функционального тестов предоставляются из коробки. Каждый тестовый класс должен иметь суффикс *Test*. Можно запускать свои тесты с помощью команд *rphunit* или *php vendor/bin/rphunit*.

Каталог *vendor* содержит ваши Composer-зависимости.

### Каталог пространства App

Большая часть приложения находится в каталоге *app*. По умолчанию этот каталог находится в пространстве имен App и автоматически загружается Composer.



Рис. 7 Структура внутри каталога *app*

Каталог *app* содержит множество дополнительных каталогов, например, имеющих пространство *Console*, *Http*, и *Providers*. Каталоги *Console* и *Http* являются API-проводниками ядра приложения. Протокол HTTP и интерфейс командной строки являются механизмами взаимодействия с приложением, но фактически не содержат логики приложения. Другими словами – это два способа передачи команд вашему приложению. Каталог *Console* содержит все Artisan-команды, тогда как каталог *Http* содержит контроллеры, посредники и запросы. [4]

Множество других каталогов будет создано внутри каталога `app`, если использовать команды `make Artisan` для создания классов. Так, например, каталог `app/Jobs` не будет существовать, пока не выполнишь команду `make:job Artisan` для создания класса задания.

Многие классы в каталоге `app` могут быть созданы с помощью команд `Artisan`. Чтобы просмотреть доступные команды, выполните команду `php artisan list` в консоли.

Рассмотрим отдельно каждый подкаталог внутри каталога `app`:

Каталог `Console` содержит все `Artisan`-команды приложения. Эти команды могут быть созданы с помощью команды `make:command`. В этом каталоге также находится ядро консоли, в котором регистрируются ваши пользовательские команды `Artisan` и определены ваши запланированные задачи.

Каталог `Exceptions` содержит обработчик исключений вашего приложения, а также является хорошим местом для размещения любых исключений, генерируемых вашим приложением. Если нужно скорректировать как исключения будут отображаться или записываться в журнал, следует изменить класс `Handler` в этом каталоге.

Каталог `Http` содержит контроллеры, посредники и запросы форм. Практически вся логика обработки запросов, поступающих в приложение, будет размещена в этом каталоге.

Каталог `Models` содержит все классы моделей `Eloquent`. Laravel содержит библиотеку `Eloquent ORM`, предоставляющую красивую и простую реализацию `ActiveRecord` для работы с базой данных. Каждая таблица базы данных имеет соответствующую «Модель», которая используется для взаимодействия с этой таблицей. Модели позволяют запрашивать данные в таблицах, а также вставлять новые записи в таблицу.

Каталог `Providers` содержит всех поставщиков служб приложения. Поставщики служб загружают приложение, связывая службы в контейнере служб, регистрируя события или выполняя любые другие задачи для подготовки приложения к входящим запросам.

## 2. РАЗРАБОТКА И РЕАЛИЗАЦИЯ ПРОГРАММНОГО ВЕБ-ПРОЕКТА

### 2.1 ПРОЕКТИРОВАНИЕ БАЗЫ ДАННЫХ

База данных является важнейшим элементом любой информационной системы. Laravel на данный момент поддерживает работу со следующими СУБД:

- MySQL
- Postgres
- SQLite
- SQL Server

Для реализации всех необходимых возможностей нашего веб-сервиса вполне будет достаточно свободной реляционной СУБД MySQL.[2]

В ходе изучения предметной области были выделены следующие основные сущности:

· Категории — используется для разделения статей, генерируемых

администрацией сервиса, на категории. Здесь под категорией имеется в виду разделение на разные тематики (наука, культура, рисование, рецепта и т.д.). Пример атрибутов: id, parent\_id, name, slug, created\_at, updated\_at

· Посты — администрацией веб-сервиса подразумевается написание и поиск статей на различную тематику. Такие статьи пользователь сможет использовать в своих целях, чтобы поделиться используя блог. Пример атрибутов: id, author\_id, category\_id, title, excerpt, body, image, slug, meta\_description, meta\_keywords, status, created\_at, updated\_at.

· Пользователи — содержит основную информацию о пользователях веб-сервиса. Пользователем выступает администратор сайта или редактор статей. Пример атрибутов: id, role\_id, name, email, avatar, password, remember\_token, settings, created\_at, updated\_at.

· Роли — содержит информацию об типах доступа к данным на сайте, можно самому создавать роли и

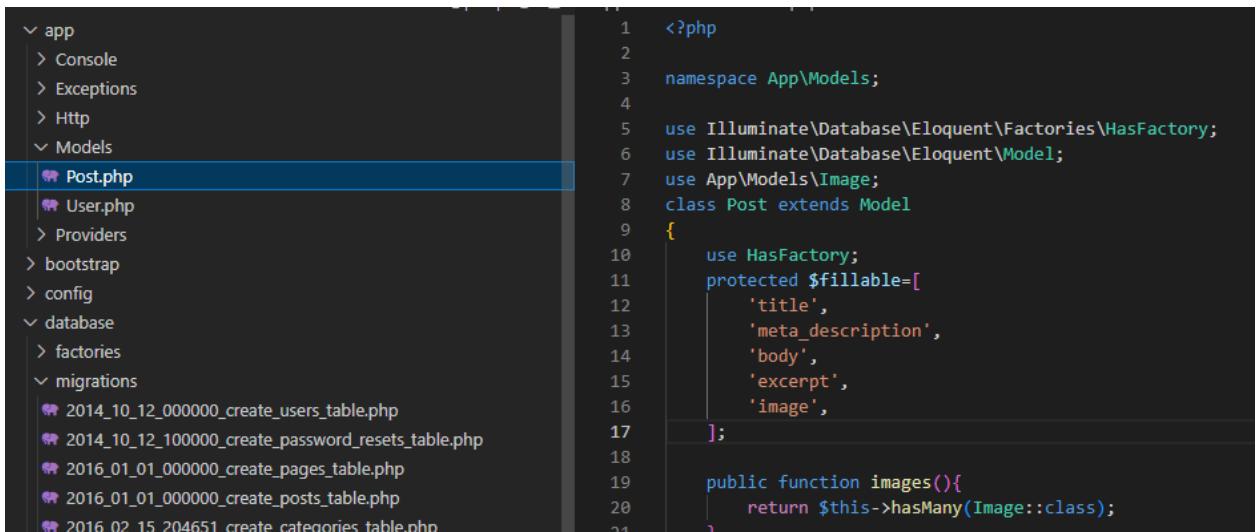
присваивать к пользователю. Пример атрибутов: id, name, display\_name, created\_at, updated\_at.

## Миграции Базы Данных

Используем миграцию для определения таблицы базы данных для хранения всех наших задач. Миграции БД в Laravel позволяют простым способом определить структуру таблицы базы данных и выполнять модификации с использованием простого и выразительного PHP кода. Вместо того чтобы вручную добавлять столбцы в свои локальные копии БД, можно просто запустить миграции, которые мы поместили в систему управления версиями.

Laravel содержит ORM-библиотеку Eloquent, предоставляющую способ работы с базой данных, который часто удобнее обычного построителя запросов. При использовании Eloquent каждая таблица БД имеет соответствующую «Модель», которая используется для взаимодействия с этой таблицей. Помимо получения записей из таблицы БД, модели Eloquent также позволяют вставлять, обновлять и удалять записи из таблицы.

Модели расширяют класс Illuminate\Database\Eloquent\Model. Чтобы сгенерировать новую модель Eloquent, используем Artisan-команду make:model. Эта команда поместит новый класс модели в каталог app/Models приложения: *php artisan make:model Post*



```
1 <?php
2
3 namespace App\Models;
4
5 use Illuminate\Database\Eloquent\Factories\HasFactory;
6 use Illuminate\Database\Eloquent\Model;
7 use App\Models\Image;
8 class Post extends Model
9 {
10     use HasFactory;
11     protected $fillable=[
12         'title',
13         'meta_description',
14         'body',
15         'excerpt',
16         'image',
17     ];
18
19     public function images(){
20         return $this->hasMany(Image::class);
21     }
}
```

Рис. 8 Создание модели Posts

При создании модели генерируем миграцию базы данных, используя параметр --migration или -m, который указывает на создание миграций:

*php artisan make:model Post --migration*

В результате создается модель в папке *app/Models* и файл миграции для таблицы *Posts* в папке *database/migrations* (Рис. 9). Миграции используются, чтобы без потерь обновить структуру базы данных при изменении модели. [8] Файлы миграций содержат информацию, какие поля нужно создать.

С помощью миграций можно создать таблицу и изменить структуру таблиц - функция *up()*. Или откатить изменения – функция *down()*

```

1 <?php
2
3 use Illuminate\Database\Migrations\Migration;
4 use Illuminate\Database\Schema\Blueprint;
5 use TCG\Voyager\Models\Page;
6
7 class CreatePagesTable extends Migration
8 {
9
10    /**
11     * Run the migrations.
12     *
13     * @return void
14     */
15    public function up()
16    {
17        // Create table for storing roles
18        Schema::create('pages', function (Blueprint $table) {
19            $table->increments('id');
20            $table->integer('author_id');
21            $table->string('title');
22            $table->text('excerpt')->nullable();
23            $table->text('body')->nullable();
24            $table->string('image')->nullable();
25            $table->string('slug')->unique();
26            $table->text('meta_description')->nullable();
27            $table->text('meta_keywords')->nullable();
28            $table->enum('status', Page::$statuses)->default(Page::STATUS_INACTIVE);
29            $table->timestamps();
30        });
31    }
32
33    /**
34     * Reverse the migrations.
35     *
36     * @return void
37     */
38    public function down()
39    {
40        Schema::dropIfExists('pages');
41    }
42}

```

Рис. 9 Миграция таблицы *Posts*

Для запуска всех необходимых миграций используем Artisan-команду *migrate*.

*php artisan migrate*

Так же сущность создаётся на локальном сервере phpMyAdmin, в которой мы можем добавлять, изменять и удалять данные, менять тип данных удалять или добавлять столбцы в более удобном виде.

Рис. 10 Сущность *Posts* на локальном сервере phpMyAdmin

## Маршрутизация

В данном приложении будем использовать контроллеры для организации наших маршрутов. Контроллеры позволяют нам лучше организовать логику HTTP-запросов в нескольких файлах. [12]

Будет несколько маршрутов: `/articles` представляющий из себя лендинг для гостей приложения. Заполним наши маршруты `/`. По этому маршруту мы запускаем функцию `index` класса `PostController`(*Http/Controllers*), которая в свою очередь отрисовывает HTML-шаблон, который содержит страницу «`index`» с переданной переменной `Posts` в нем находится выборка из базы необходимые столбцы: `'title', 'meta_description', 'body', 'excerpt', 'image'`. Все маршруты находятся в каталоге `routes`.

```
Route::get('/',[PostController::class,'index']);
```

```
Route::get('/ articles ,[PostController::class,'indexA']);
```

В Laravel все HTML-шаблоны хранятся в каталоге `resources/views`.

Последний маршрут `/admin` ссылается на пакет Voyager, который перекидывает на свои маршруты в каталоге `vendor/tcg/voyager/routes`

```
Route::group(['prefix' => 'admin'], function () {
```

```
    Voyager::routes();
```

```
});
```

Voyager — это административная панель для фреймворка Laravel. При создании Web-приложения с помощью Laravel полезно иметь возможность управлять данными, которые обрабатываем. [9]

Самый простой способ сделать это — использовать консоль MySQL для выбора, обновления, вставки и удаления данных. Тем не менее, использование терминала для этого неудобно. Нужно запомнить и набрать правильные команды, или столкнуться с ошибками форматирования из-за размера ваших таблиц и т. д.

Следующий метод, который можно использовать, который немного более удобен для пользователя, — это phpMyAdmin. Очень полезно управлять структурой базы данных Web-приложения, так как теперь есть графический интерфейс с флажками, кнопками и списками вместо командных строк, но по-прежнему довольно неэффективно управлять самими данными, а также пользователями, которые имеют доступ к этой панели администратора.

Voyager для Laravel продвинулся по уровню вперед и предоставляет современную и адаптивную панель администратора, схожую с phpMyAdmin, с поддержкой BREAD/CRUD и точным контролем доступа пользователей к Web-приложению Laravel.

Это дает возможность настроить панель в соответствии со структурой базы данных и позволяет продвинуться дальше в создании Web-приложения, не написав ни единой строки кода.

Преимущества Voyager:

- Современный, красивый и адаптивный пользовательский интерфейс.
- Управление данными (поддержка BREAD/CRUD)
- Связи между таблицами без кода
- Управление доступом пользователей, которое может быть распространено на весь Web-приложение с поддержкой Passport
- Возможность настройки панели под другие функции администратора

Voyager очень прост в установке. После создания нового приложения Laravel можно включить пакет Voyager следующей командой:

```
composer require tcg/voyager
```

Далее убедимся, что создалась новая база данных и добавились учетные данные базы данных в .env файл, а также добавим URL приложения в переменную APP\_URL:

```
APP_URL=http://localhost:8000  
DB_HOST=127.0.0.1  
DB_PORT=3306  
DB_DATABASE=admin-panel  
DB_USERNAME=root  
DB_PASSWORD=
```

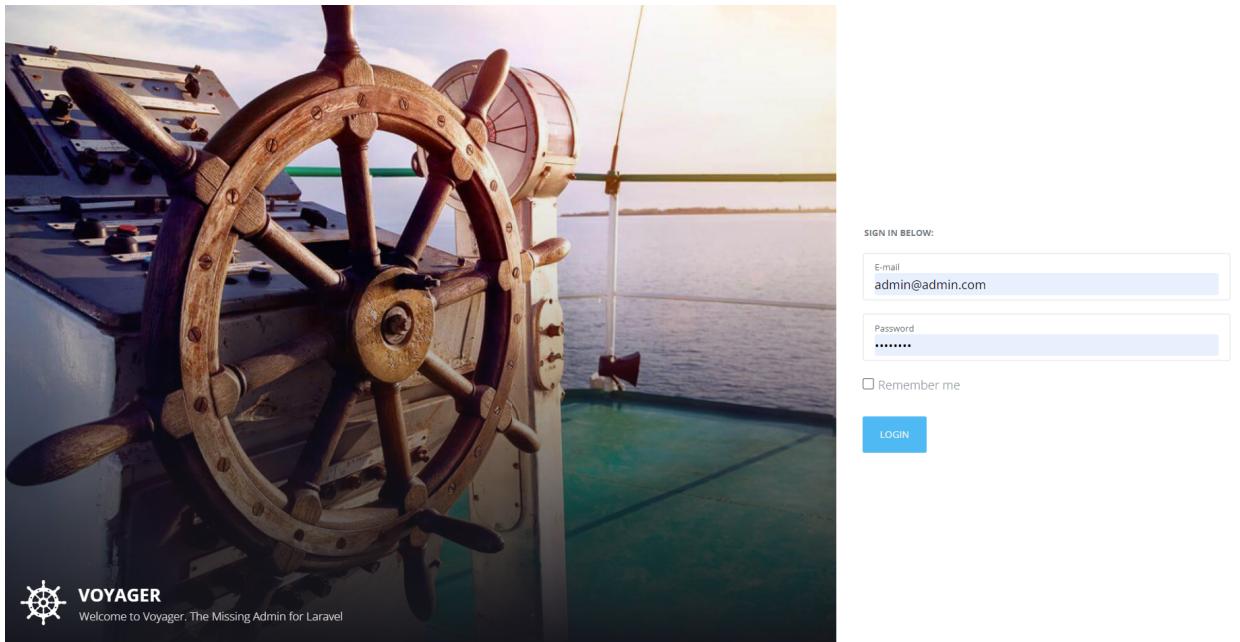
Теперь устанавливаем Voyager. Можно выбрать установку Voyager с фиктивными данными или без них. В состав данных будет входить 1 аккаунт администратора (если нет уже существующих пользователей), 1 демо-страница, 4 демо-записи, 2 категории и 7 настроек.

Устанавливаем его с фиктивными данными, выполняя следующую команду:

```
php artisan voyager:install --with-dummy
```

Создаем нового пользователя с правами администратора, передавая флаг --create, как показано ниже.

```
php artisan voyager:admin your@email.com --create
```



The image shows the Voyager admin dashboard. The left sidebar includes links for Admin, Dashboard, Roles, Users, Media, Posts, Pages, Categories, Tools, and Settings. The main content area has a breadcrumb navigation bar showing "Dashboard". It displays three cards: one for "Users" (3 Users, View all users), one for "Posts" (6 Posts, View all posts), and one for "Pages" (1 Page, View all pages). Below these cards, a note states: "To view analytics you'll need to get a google analytics client id and add it to your settings for the key |google\_analytics\_client\_id|. Get your key in your Google developer console: https://console.developers.google.com".

**VOYAGER**

Admin

- Dashboard
- Roles
- Users
- Media
- Posts**
- Pages
- Categories
- Tools
- Settings

**Edit Post**

**Post Title**  
The title for your post  
Сколько времени живет фотон? Может ли фотон «умереть»?

**Post Content**

В человеческом сознании все в мире подчиняется течению времени: от нашей собственной жизни до величественного танца галактик на просторах Вселенной. Но когда мы говорим о таких странных частицах как фотонах, наши представления о времени имеют мало общего с реальностью.

Известно, что фотоны – частицы света – имеют кумулюю массу покоя. Именно это позволяет им перемещаться со скоростью света. Чем быстрее передвигается объект в пространстве, тем медленнее для него идет время относительно наблюдателя. Для фотона, летящего со скоростью света, сию вовсе останавливается.

Что это означает? Попробуй, фотон всегда не воспринимает течение времени. Миллиарды лет существования Вселенной, не говоря уже о времени жизни человека, для этой частицы проходит как одно мгновение. Более того, для фотона, летящего со скоростью света, все расстояния по направлению его движения сокращаются до точки – вступает в действие эффект релативистского сокращения длины. Получается, фотон с его точки зрения живет и существует лишь мгновение, никак при этом не перемещаясь – пока не будет поглощен каким-либо объектом.

А с точки зрения нас – наблюдателей – фотон может существовать бесконечно, пересекая космическое пространство со скоростью света. Испущенный в самом начале рождения Вселенной, он может существовать до самого ее конца. С нашей точки зрения – вечность.

**Excerpt** Small description of this post  
2 мин. чтения

**Post Details**

URL slug: сколько-времени-живет-фотон-может-ли-фотон-умереть

Post Status: published

Post Category: Наука

Featured:

**Post Image**

Выбор файла [Нажмите, чтобы выбрать один файл]

**VOYAGER**

Admin

- Dashboard
- Roles
- Users
- Media
- Posts**
- Pages
- Categories
- Tools
- Settings

**Posts**

Add New Bulk Delete

Show: 10 entries	Search:					
Title	Post Image	Status	Created At	SEO Title	Featured	Actions
Почему вдоль железной дороги лежат камни?		published	2022-12-02 18:34:33	4	0	<a href="#">View</a> <a href="#">Edit</a> <a href="#">Delete</a>
Первая фотография Земли из космоса		published	2022-12-02 18:31:03	0	0	<a href="#">View</a> <a href="#">Edit</a> <a href="#">Delete</a>
Сколько времени живет фотон? Может ли фотон «умереть»?		published	2022-11-17 14:52:55	0	0	<a href="#">View</a> <a href="#">Edit</a> <a href="#">Delete</a>
Сколько фотонов во Вселенной?		published	2022-11-17 14:52:55	0	0	<a href="#">View</a> <a href="#">Edit</a> <a href="#">Delete</a>
Первая народная галерея в Лыжном. История		published	2022-11-17 14:52:55	0	0	<a href="#">View</a> <a href="#">Edit</a> <a href="#">Delete</a>
Хенчукова деревянная архитектура русского севера		published	2022-11-17 14:52:55	0	0	<a href="#">View</a> <a href="#">Edit</a> <a href="#">Delete</a>

Showing 1 to 6 of 6 entries

< Previous 1 Next >

Файл Правка Вид Журнал Закладки Инструменты Справка

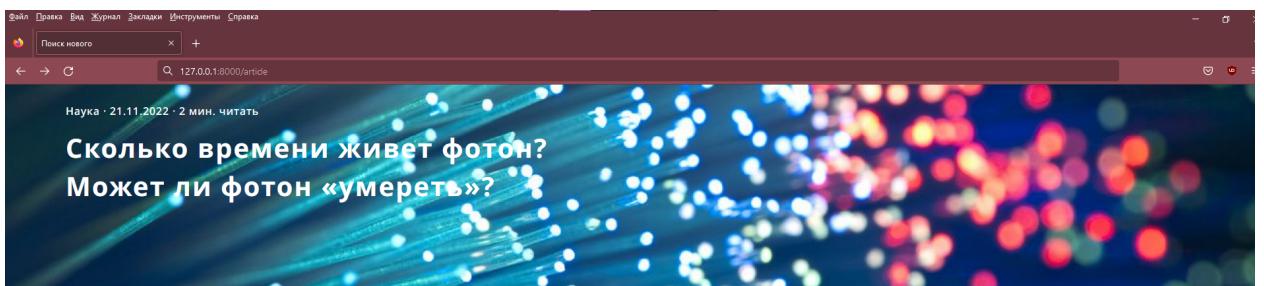
Поиск нового +

127.0.0.1:8000/article

# Поиск нового

Наука · 21.11.2022 · 2 мин. читать

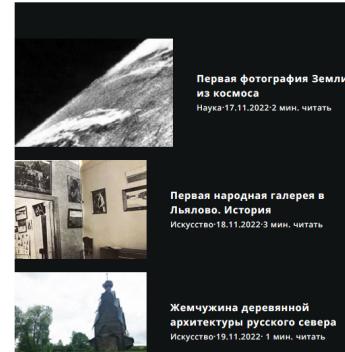
## Сколько времени живет фотон? Может ли фотон «умереть»?



В человеческом сознании все в мире подчиняется течению времени: от нашей собственной жизни до величественного танца галактик на просторах Вселенной. Но когда мы говорим о таких странных частицах как фотонах, наши представления о времени имеют мало общего с реальностью.

Известно, что фотоны – частицы света – имеют нулевую массу покоя. Именно это позволяет им перемещаться со скоростью света. Чем быстрее передвигается объект в пространстве, тем медленнее для него идет время относительно наблюдателя. Для фотона, летящего со скоростью света, оно вовсе останавливается.

Что это означает? Похоже, фотон вовсе не воспринимает течение времени. Миллиарды лет существования Вселенной, не говоря уже о времени жизни человека, для этой частицы проходят как одно мгновение. Более того, для фотона, летящего со скоростью света, все расстояния по направлению его движения сократятся до точки – вступает в действие эффект релятивистского сокращения длины. Получается, фотон с его точки зрения живет и существует лишь мгновение, нигде при этом не перемещаясь – пока не будет поглощен каким-либо объектом..



A screenshot of the Voyager CMS admin dashboard. The left sidebar has a dark theme with icons for Admin, Dashboard, Roles, Users, Media, Posts, Pages, Categories, Tools, and Settings. The main content area is titled 'Roles' and shows a table of users. The table has columns for 'Name' and 'Display Name'. It lists two users: 'admin' (Administrator) and 'user' (Normal User). Each user row has 'View', 'Edit', and 'Delete' buttons. At the bottom, it says 'Showing 1 to 2 of 2 entries'.

**VOYAGER**

**Admin**

- Dashboard
- Roles
- Users**
- Media
- Posts
- Pages
- Categories
- Tools
- Settings

**Users**

Add New Bulk Delete

Show 10 entries Search:

	Name	Email	Created At	Avatar	Role	Roles	Actions
<input type="checkbox"/>	admin	admin@admin.com	2022-12-01 07:26:45		Administrator	No results	<a href="#">View</a> <a href="#">Edit</a> <a href="#">Delete</a>
<input type="checkbox"/>	nastya	n@v.com	2022-11-17 15:49:09		Normal User	Normal User	<a href="#">View</a> <a href="#">Edit</a> <a href="#">Delete</a>
<input type="checkbox"/>	Анастасия Бирюкова	nastya22bir@gmail.com	2022-11-17 14:52:55		Administrator	No results	<a href="#">View</a> <a href="#">Edit</a> <a href="#">Delete</a>

**VOYAGER**

**Admin**

- Dashboard
- Roles
- Users
- Media
- Posts
- Pages
- Categories**
- Tools
- Settings

**Categories**

Add New Bulk Delete

Show 10 entries Search:

	Order	Name	Slug	Actions
<input type="checkbox"/>	1	Искусственный интеллект	iskusstvennyj-intellekt	<a href="#">View</a> <a href="#">Edit</a> <a href="#">Delete</a>
<input type="checkbox"/>	1	История	istoriya	<a href="#">View</a> <a href="#">Edit</a> <a href="#">Delete</a>
<input type="checkbox"/>	1	Наука	category-1	<a href="#">View</a> <a href="#">Edit</a> <a href="#">Delete</a>
<input type="checkbox"/>	1	Искусство	category-2	<a href="#">View</a> <a href="#">Edit</a> <a href="#">Delete</a>

Showing 1 to 4 of 4 entries

**VOYAGER**

**Admin**

- Dashboard
- Roles
- Users
- Media
- Posts
- Pages
- Categories
- Tools**
- Menu Builder
- Database
- Compass
- BREAD
- Settings

**Database**

Create New Table

Table Name Table Actions

<a href="#">categories</a>	<a href="#">Browse BREAD</a>	<a href="#">Edit BREAD</a>	<a href="#">Delete BREAD</a>	<a href="#">View</a> <a href="#">Edit</a> <a href="#">Delete</a>
<a href="#">failed_jobs</a>	<a href="#">Add BREAD to this table</a>			<a href="#">View</a> <a href="#">Edit</a> <a href="#">Delete</a>
<a href="#">menus</a>	<a href="#">Browse BREAD</a>	<a href="#">Edit BREAD</a>	<a href="#">Delete BREAD</a>	<a href="#">View</a> <a href="#">Edit</a> <a href="#">Delete</a>
<a href="#">pages</a>	<a href="#">Browse BREAD</a>	<a href="#">Edit BREAD</a>	<a href="#">Delete BREAD</a>	<a href="#">View</a> <a href="#">Edit</a> <a href="#">Delete</a>
<a href="#">permissions</a>	<a href="#">Add BREAD to this table</a>			<a href="#">View</a> <a href="#">Edit</a> <a href="#">Delete</a>
<a href="#">posts</a>	<a href="#">Browse BREAD</a>	<a href="#">Edit BREAD</a>	<a href="#">Delete BREAD</a>	<a href="#">View</a> <a href="#">Edit</a> <a href="#">Delete</a>
<a href="#">roles</a>	<a href="#">Browse BREAD</a>	<a href="#">Edit BREAD</a>	<a href="#">Delete BREAD</a>	<a href="#">View</a> <a href="#">Edit</a> <a href="#">Delete</a>
<a href="#">translations</a>	<a href="#">Add BREAD to this table</a>			<a href="#">View</a> <a href="#">Edit</a> <a href="#">Delete</a>
<a href="#">user_roles</a>	<a href="#">Add BREAD to this table</a>			<a href="#">View</a> <a href="#">Edit</a> <a href="#">Delete</a>
<a href="#">users</a>	<a href="#">Browse BREAD</a>	<a href="#">Edit BREAD</a>	<a href="#">Delete BREAD</a>	<a href="#">View</a> <a href="#">Edit</a> <a href="#">Delete</a>

## и html-шаблоны

The screenshot shows the Voyager web application interface. On the left is a dark sidebar with a user icon labeled 'Admin', navigation links for 'Dashboard', 'Roles', 'Users', 'Media', 'Posts', 'Pages', 'Categories', 'Tools' (with 'Menu Builder', 'Database', 'Compass', and 'BREAD' listed), and 'Settings'. The main area is titled 'BREAD' and shows a list of database tables: 'categories', 'failed\_jobs', 'menus', 'pages', 'permissions', 'posts', 'roles', 'translations', 'user\_roles', and 'users'. Each table entry includes a small icon, a 'Table Name' label, and three action buttons: 'Browse' (orange), 'Edit' (blue), and 'Delete' (red). Below the table list are two buttons: 'Add BREAD to this table' and 'Add BREAD to this table' (repeated).

## ЗАКЛЮЧЕНИЕ

Целью данной работы являлась разработка Web-приложения, который бы предоставлял частным работникам медицинской сферы необходимый функционал для управления своими инструментами для интернет-маркетинга из одного места. В ходе выполнения работы был выполнен анализ существующих систем, выявлены требования к разрабатываемому приложению, спроектирована архитектура и реализован веб-сервис для интернет-маркетинга частных работников медицины. Таким образом, все поставленные задачи выполнены и цель работы достигнута. Разработанный веб-сервис введен в эксплуатацию на поддомене сайта [www.doctorhero.com](http://www.doctorhero.com) по адресу [app.doctorhero.com](http://app.doctorhero.com).

Был создан удобный и простой в использовании web-интерфейс для решения системы дифференциальных уравнений, который при дальнейшем развитии может расширять диапазон своих функций и оказывать помощь в решении уравнений и исследовании физических явлений. Полученные данные (значения и графики) были проверены в программе Mathematica и являются надежными.

У данной программы также есть множество перспектив, например:

- Возможность выбора разных видов дифференциальных уравнений
- Увеличение количества начальных параметров, задаваемых пользователем
- Возможность выбора зависимости для построения графиков
- Возможность сохранять полученные таблицы и графики пользователем в файл.

В данной курсовой работе рассмотрены актуальные вопросы разработки и создания современного Web-сайта.

При этом нами были решены следующие частные задачи:

- ознакомление с современными Интернет-технологиями и их использование в настоящей разработке;
- изучение программного инструментария, применяемого для разработки и создания Web-сайтов;
- выявление и учет методов и способов представления на Web-страницах различных видов информации, не препятствующих их доступности;
- ознакомление с основными правилами и рекомендациями по разработке и созданию Web-сайтов и неукоснительное следование им на практике;
- определение структуры Web-страниц;
- выбор стратегии разработки и создания Web-сайта.

1. При создании сайта была применена древовидная структура, т.е. с главной страницы можно перейти по ссылкам на другие страницы. Сайт разработан на языке программирования - PHP, с помощью среды разработки - Macromedia Dreamweaver.
2. Разработана структура сайта и структура, используемой базы данных. Рассмотрены основные сущности и их атрибуты.
3. Разработан сайт, который предоставляет возможность регистрации, просмотра списка статей, форумов и пользователей. Также предоставляется возможность добавлять, редактировать и удалять статьи и прочие данные, а также личную информацию, которые ранее были добавлены данным пользователем. Зарегистрированные пользователи могут добавлять статьи и личную информацию о себе.
4. Произведено тестирование, в результате которого были найдены некоторые не существенные неполадки и приняты все меры по их устранению.
5. При выполнении данной курсовой работы были изучены:
  - язык сценариев PHP;
  - СУБД MySQL;

Результатом курсовой работы стал готовый к работе web - сайт в котором размещена информация о замечательной восточной стране.. В ходе проведения работы были решены поставленные в курсовой работе задачи. Оценивая проделанную работу, можно сделать следующие выводы:

- сайт предоставляет пользователям наиболее нужную ему информацию, а именно возможность ознакомиться с информацией о культуре другой страны.
- дизайн сайта соответствует предполагаемым предпочтениям целевой группы, времени и целям нахождения потенциальных клиентов на сайте;
- реализована навигация с помощью меню по страницам сайта.
- возможность пользователей оставить комментарий к статье или отправить личное сообщение
- Для разработки и реализации сайта был выбран бесплатный конструктор сайтов Wix.
- В дальнейшем разработанный мной web-сайт можно будет использовать в реальной жизни. Так же возможно расширение тематической информации.
- Данный ресурс стабилен в работе и не требует высоких знаний для работы с ним.

Laravel – действительно удивительная структура. Это быстро, просто, элегантно и так просто в использовании. Это абсолютно заслуживает рассмотрения в качестве основы, которая будет использоваться для следующего проекта.

## **СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ**

1. Эспозито Д. Разработка современных веб-приложений. Анализ предметных областей и технологий – Вильямс. – 2017. – 464 стр.
2. Страффер М. Laravel. Полное руководство. 2-е издание – СПб. – 2020. – 514 стр.
3. Никсон Р. Создаем динамические веб-сайты с помощью PHP, MySQL, JavaScript, CSS и HTML5 – СПб. – 2019. – 816 стр.
4. Дронов В. Laravel 9. Быстрая разработка веб-сайтов на PHP – BHV. – 2021. – 688 с.
- 5.
- 6.
- 7.
8. Миграция по Laravel [Электронный ресурс] // URL: <https://laravel.com/docs/9.x/migrations> (дата обращения: 20.10.2022).
9. Справочник по Voyager [Электронный ресурс] // URL: <https://voyager-docs.devdojo.com/> (дата обращения: 20.11.2022).
10. Документация по работе с php-фреймворком Laravel [Электронный ресурс] // URL: <https://laravel.su/docs/9.x> (дата обращения: 22.10.2022).

11. Понятие Web-приложения [Электронный ресурс] // URL:  
<https://studwood.net/667002/informatika/vvedenie> (дата обращения: 01.10.2022).

12. Маршрутизация в Laravel [Электронный ресурс] // URL:  
<https://laravel.su/docs/9.x/routing> (дата обращения: 10.11.2022).

13. Справочник по HTML [Электронный ресурс] // URL:  
<https://practicum.yandex.ru/blog/zachem-nuzhen-html/> (дата обращения: 12.10.2022).

14. Справочник по CSS [Электронный ресурс] // URL:  
<http://htmlbook.ru/css> (дата обращения: 12.10.2022).

15. Справочник по PHP [Электронный ресурс] // URL:  
<https://www.php.net/manual/ru/intro-whatcando.php> (дата обращения: 10.10.2022).

## ПРИЛОЖЕНИЯ

*app\Http\Controllers\PostControllers.php*

```
<?php  
  
namespace App\Http\Controllers;  
  
use App\Models\Post;  
  
use Illuminate\Http\Request;  
  
use Illuminate\Support\Facades\File;  
  
class PostController extends Controller  
  
{  
  
    public function index()  
  
    {  
  
        $posts=Post::all();
```

```
        return view('index')->with('posts',$posts);  
    }  
  
    public function article()  
    {  
        $posts=Post::all();  
  
        return view('article')->with('posts',$posts);  
    }  
  
    public function myadmin()  
    {  
        $posts=Post::all();  
  
        return view('myadmin')->with('posts',$posts);  
    }  
  
    public function create()  
    {  
        //  
    }  
  
    public function store(Request $request)  
    {  
        if($request->hasFile("cover")){  
            $file=$request->file("cover");  
  
            $imageName=time().'.'.$file->getClientOriginalName();  
  
            $file->move(\public_path("cover/"),$imageName);  
  
            $post =new Post([  
                "title" =>$request->title,  
            ]);  
            $post->save();  
        }  
    }  
}
```

```
"author" =>$request->author,  
"body" =>$request->body,  
"cover" =>$imageName,  
]);  
  
$post->save();  
  
}  
  
if($request->hasFile("images")){  
  
$files=$request->file("images");  
  
foreach($files as $file){  
  
$imageName=time().'_'.$file->getClientOriginalName();  
$request['post_id']=$post->id;  
$request['image']=$imageName;  
$file->move(public_path("/images"),$imageName);  
Image::create($request->all());  
}  
}  
  
return redirect("/");  
}  
  
public function show(Post $post)  
{  
//  
}  
  
public function edit($id)  
{
```

```
$posts=Post::findOrFail($id);

return view('edit')->with('posts',$posts);

}

public function update(Request $request,$id)

{

$post=Post::findOrFail($id);

if($request->hasFile("cover")){

    if (File::exists("cover/".$post->cover)) {

        File::delete("cover/".$post->cover);

    }

$file=$request->file("cover");

$post->cover=time()."_" . $file->getClientOriginalName();

$file->move(\public_path("/cover"),$post->cover);

$request['cover']=$post->cover;

}

$post->update([
    "title" =>$request->title,
    "author"=>$request->author,
    "body"=>$request->body,
    "cover"=>$post->cover,
]);

if($request->hasFile("images")){
    $files=$request->file("images");
    foreach($files as $file){


```

```
$imageName=time().'.'.$file->getClientOriginalName();

$request["post_id"]=$id;

$request["image"]=$imageName;

$file->move(\public_path("images"),$imageName);

Image::create($request->all());

}

}

return redirect("/");

}

public function destroy($id)

{

$posts=Post::findOrFail($id);

if (File::exists("cover/".$posts->cover)) {

    File::delete("cover/".$posts->cover);

}

$images=Image::where("post_id",$posts->id)->get();

foreach($images as $image){

if (File::exists("images/".$image->image)) {

    File::delete("images/".$image->image);

}

}

$posts->delete();

return back();
```

```
}

public function deleteimage($id){

    $images=Image::findOrFail($id);

    if (File::exists("images/".$images->image)) {

        File::delete("images/".$images->image);

    }

    Image::find($id)->delete();

    return back();

}

public function deletecover($id){

    $cover=Post::findOrFail($id)->cover;

    if (File::exists("cover/".$cover)) {

        File::delete("cover/".$cover);

    }

    return back();

}

}
```

*app/Models/Post.php*

```
<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Factories\HasFactory;

use Illuminate\Database\Eloquent\Model;

class Post extends Model

{
```

```
use HasFactory;

protected $fillable=[

    'title',
    'meta_description',
    'body',
    'excerpt',
    'image',
];

public function images(){

    return $this->hasMany(Image::class);

}

}
```

### ***database|migrations|create\_posts\_table.php***

```
<?php

use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
class CreatePostsTable extends Migration
{
    public function up()
    {
        // Создание таблиц
        Schema::create('posts', function (Blueprint $table) {
            $table->increments('id');
            $table->integer('author_id');
        });
    }
}
```

```

$table->integer('category_id')->nullable();

$table->string('title');

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

$table->text('excerpt');

$table->text('body');

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

$table->string('slug')->unique();

$table->text('meta_description');

$table->text('meta_keywords');

$table->enum('status', ['PUBLISHED', 'DRAFT',
'PENDING'])->default('DRAFT');

$table->boolean('featured')->default(0);

$table->timestamps();

});

}

public function down()
{
    Schema::drop('posts');
}

}

resources\views\index.blade.php

resources\views\article.blade.php
<!DOCTYPE html>

```

```
<html lang="en">

<head>

    <meta charset="UTF-8">

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

    <meta name="viewport" content="width=device-width,
initial-scale=1.0">

    <link rel="preconnect" href="https://fonts.googleapis.com">

    <link rel="preconnect" href="https://fonts.gstatic.com" crossorigin>

    <link
        href="https://fonts.googleapis.com/css2?family=Noto+Sans:wght@500;700&display=swap" rel="stylesheet">

    <link rel="stylesheet" href="../css/style.css">

    <title>Поиск нового</title>

</head>

<body>

    <header class="header">

        <div class="header__inner">

            <h1 class="logo__title">Поиск нового</h1>

        </div>

    </header>

    <main>

        <div class="content">

            <div class="content__photo">

                <div class="content__info">
```

```
<p class="content__data">{{ $post->category }}·  
,{{ $table->timestamps() }}· {{ $post->excerpt }}</p>  
  
<h2 class="content__title"> {{ $post->title }}</h2>  
  
</div>  
  
</div>  
  
<div class="content__item">  
  
<div class="container">  
  
<div class="content__inner">  
  
<div class="content__text"> {{ $post->body }}</div>  
  
</div>  
  
<div class="content__posts">  
  
<div class="posts__item">  
  
<a class="post" href="#">  
  
  
  
<div class="post__text">  
  
<h4 class="post__title"> {{ $post->title }}</h4>  
  
<p class="post__data">{{ $post->category }}{{ $table->timestamps() }}· {{ $post->excerpt }}</p>  
  
</div>  
  
</a>  
  
<a class="post" href="#">  
  
  
  
<div class="post__text">
```

```
<h4 class="post__title"> {{ $post->title }}</h4>

<p class="post__data">{{ $post->category }} , {{ $table->timestamps() }} , {{ $post->excerpt }}</p>

</div>

</a>

<a class="post" href="#">



<div class="post__text">

<h4 class="post__title"> {{ $post->title }}</h4>

<p class="post__data">{{ $post->category }} , {{ $table->timestamps() }} , {{ $post->excerpt }}</p>

</div>

</a>

<a class="post" href="#">



<div class="post__text">

<h4 class="post__title"> {{ $post->title }}</h4>

<p class="post__data">{{ $post->category }} , {{ $table->timestamps() }} , {{ $post->excerpt }}</p>

</div>

</a>

<a class="post" href="#">


```

```
<div class="post__text">

    <h4 class="post__title"> {{ $post->title }}</h4>

    <p class="post__data"> {{ $post->category }} · {{ $table->timestamps() }} · {{ $post->excerpt }}</p>

</div>

</a>

</div>

</div>

</div>

</main>

<footer class="footer">

    <h1 class="logo__title">Поиск нового</h1>

</footer>

</body>

</html>
```

### ***routes/web.php***

```
<?php

use App\Http\Controllers\PostController;

use Illuminate\Support\Facades\Route;

Route::get('/',[PostController::class,'index']);

Route::get('/article',[PostController::class,'article']);

Route::group(['prefix' => 'admin'], function () {

    Voyager::routes();
```

});