



Geekbrains

Создание интернет-магазина с системой фильтрации товаров и отзывами пользователей

Программа: Разработчик
Специализация: Fullstack разработчик
Воронин Виктор Владимирович

Ревда
2025

Содержание

Введение	3
Теоретическая и практическая главы	16
Заключение	49
Список литературы:	51

Введение

Актуальность

В современном мире бизнес немыслим без активного присутствия в интернете. В условиях жёсткой конкуренции создание собственного веб-ресурса для кафе перестало быть просто возможностью выделиться среди конкурентов — это стало насущной необходимостью для успешного развития предприятия общественного питания.

Онлайн-присутствие заведения позволяет не только привлечь новых клиентов, но и значительно повысить лояльность уже существующей аудитории. В эпоху цифровых технологий потенциальный посетитель в первую очередь обращается к интернету для поиска подходящего места для обеда или ужина.

Сайт для кафе — это не просто визитная карточка в интернете, а мощный инструмент развития бизнеса, который позволяет автоматизировать многие процессы и существенно расширить аудиторию заведения.

Цели и задачи

Основная цель данной дипломной работы заключается в разработке функционального веб-сайта для небольшого кафе с использованием современного PHP-фреймворка Laravel, который позволит оптимизировать работу заведения и повысить его конкурентоспособность в современных условиях.

Задачи исследования:

Для достижения поставленной цели необходимо решить следующие задачи:

- Изучить особенности и преимущества фреймворка Laravel для разработки веб-приложений
- Создать архитектуру веб-сайта на базе Laravel
- Реализовать основные функциональные модули
- Обеспечить безопасность и защиту данных

Объект и предмет исследования

Объект исследования в данной дипломной работе представляет собой процесс разработки веб-сайта для малого бизнеса с применением современного PHP-фреймворка Laravel.

Предмет исследования охватывает следующие аспекты:

- Технические возможности и особенности фреймворка Laravel
- Процесс создания и внедрения системы управления контентом
- Механизмы обеспечения безопасности веб-приложения

Важно отметить, что **теоретической базой** исследования служат:

- Документация и руководства по Laravel
- Практические материалы из открытых интернет-источников
- Примеры реализованных проектов на Laravel

Таким образом, объектом исследования является целостный процесс разработки веб-сайта, а предметом — конкретные технологические и методологические аспекты этого процесса, связанные с использованием фреймворка Laravel в контексте создания сайта для кафе.

Практическая значимость работы

Практическая ценность данной дипломной работы заключается в следующем:

Создание **готового прототипа** веб-приложения, который может быть использован как основа для разработки полноценных сайтов кафе и ресторанов

Формирование **базовой модели** для дальнейшего масштабирования и расширения функционала

Получение **практических навыков** работы с современным PHP-фреймворком Laravel, включая:

Освоение принципов MVC-архитектуры

Работу с базами данных

Реализацию системы аутентификации

Инструменты и технологии

В процессе выполнения дипломной работы будут использованы следующие **программные средства и инструменты разработки**:

Visual Studio Code — современная среда разработки.

Инструменты разработчика браузеров:

Microsoft Edge Developer Tools.

Firefox Developer Tools.

Postman — инструмент для работы с API.

Дополнительно в процессе разработки будут использованы:

Система контроля версий Git для управления изменениями кода

PHP-сервер для локального тестирования

База данных SQLite.

Что такое SQLite

SQLite — это легкая встраиваемая система управления базами данных, которая хранит всю информацию в одном файле и не требует отдельного сервера для работы. Это делает её идеальным решением для небольших проектов и приложений, работающих локально.

SQLite — отличный выбор для небольших проектов, где не требуется сложная система управления базами данных. Она проста в использовании, надежна и не требует значительных ресурсов. Однако для крупных проектов с высокой нагрузкой рекомендуется использовать более мощные СУБД.

Особенности работы

- **Встраиваемый движок:** работает как библиотека внутри приложения
- **Локальное хранение:** все данные находятся в одном файле на устройстве
- **Простота использования:** не требует сложной настройки и администрирования
- **Высокая производительность:** быстрый доступ к данным благодаря отсутствию сетевого взаимодействия

Composer для управления зависимостями проекта.

Vite — современная система разработки фронтенд-приложений

Все перечисленные инструменты обеспечат:

Качественную разработку веб-приложения

Эффективное тестирование функционала

Удобство поддержки и развития проекта

Соответствие современным стандартам разработки

Создание сайтов

Создание сайта для кафе

Сайт для кафе должен быть не просто информационным ресурсом, а удобной и привлекательной площадкой для взаимодействия с потенциальными посетителями. Основные особенности, которые стоит учитывать при разработке сайта для кафе, включают:

Привлекательный дизайн. Внешний вид сайта играет важнейшую роль в формировании первого впечатления о заведении. Дизайн должен быть стильным, соответствовать атмосфере кафе и вызывать у посетителя желание его посетить. Отображение фирменных цветов, логотипа и уникального стиля кафе – это первый шаг к созданию доверия и узнаваемости бренда.

Удобная навигация. Сайт должен быть интуитивно понятен и легок в использовании. Клиенты не должны тратить много времени на поиск нужной информации, будь то меню, контакты или информация о местоположении.

Меню с фотографиями блюд. Размещение полного меню с привлекательными фотографиями блюд – один из ключевых факторов, способствующих увеличению заказов. Люди воспринимают визуальную информацию гораздо лучше, и качественные фотографии блюд могут увеличить желание сделать заказ.

Функция онлайн-заказа и бронирования. Для современного кафе наличие опции онлайн-заказа еды с доставкой или функции бронирования

столика через сайт – необходимое условие. Это значительно улучшает клиентский опыт и повышает шансы на привлечение посетителей.

Отзывчивая (адаптивная) верстка. Учитывая, что большинство пользователей посещают сайты с мобильных устройств, крайне важно, чтобы сайт был оптимизирован для различных устройств – смартфонов, планшетов и ПК.

Кто такой Fullstack-разработчик?

Fullstack-разработчик — это универсальный специалист в области веб-разработки, обладающий компетенциями как в создании клиентской (фронтенд), так и серверной (бэкенд) части веб-приложений. Термин происходит от английского *full stack* — «полный набор» инструментов разработки.

Вариаций Full-stack разработчиков, на самом деле, множество: PHP Full-stack Developer, Node.js Full-stack Developer, Java Full-stack Developer и так далее. Название, которое стоит в самом начале специальности, говорит о том, какой язык/платформа берется за основу во время реализации Backend части. Стек технологий FrontEnd-a практически всегда одинаков и отличается лишь используемыми JavaScript-фреймворками — Angular, React или Vue.js. А вот бекенд предоставляет гораздо больше возможностей для реализации своих амбиций.

Еще раз проговорим, что Full-stack Developer — это разработчик, который принимает непосредственное участие во всех этапах разработки веб-приложений — от создания клиентской части (визуальная часть + пользовательская логика) до реализации серверной (базы данных, серверная архитектура, программная логика). Какой стек технологий и языков

находится в распоряжении данного специалиста? Если говорить о **FrontEnd составляющей** (клиентская сторона), то она у всех примерно одинакова:

- язык вёрстки HTML и язык стилей CSS;
- языки программирования JavaScript и TypeScript;
- препроцессоры SASS и LESS;
- фреймворк Angular//Vue.js или библиотека React;
- технологии DOM, AJAX, REST API, знания об интернете и веб-технологиях в целом;
- навыки адаптивной и кроссбраузерной вёрстки.

Что такое система контроля версий

Система контроля версий (СКВ) — это программное обеспечение, позволяющее отслеживать изменения в файлах и управлять различными версиями одного и того же документа или набора файлов.

Преимущества использования СКВ

Безопасность данных

Контроль изменений

Командная работа

Откат к предыдущим версиям

Документация изменений

В проекте использовалась система Git.

Что такое Figma

Figma — это современный графический онлайн-редактор для совместной работы над проектами. Сервис позволяет создавать прототипы сайтов, интерфейсы приложений и другие дизайнерские продукты в режиме реального времени.

Основные возможности

Создание прототипов сайтов и приложений с интерактивными элементами

Разработка интерфейсов с кнопками, иконками, формами обратной связи

Работа с векторной графикой и создание иллюстраций

Совместная работа команды над проектом

Облачное хранение всех файлов и автоматическое сохранение изменений

Что такое Postman

Postman — это мощный инструмент для тестирования, разработки и документирования API. Программа позволяет отправлять HTTP-запросы к различным API и проверять их ответы, что помогает убедиться в корректной работе приложений.

Основные возможности

- **Тестирование API:** отправка различных HTTP-запросов (GET, POST, PUT, DELETE)
- **Автоматизация тестов:** создание коллекций запросов для автоматического тестирования
- **Совместная работа:** облачное хранение и обмен запросами между членами команды
- **Документация:** автоматическое создание документации по API
- **Mock-серверы:** создание имитаций API для тестирования

Composer: менеджер зависимостей для PHP

Что такое Composer

Composer — это инструмент для управления сторонними библиотеками в PHP-проектах. Он позволяет автоматически устанавливать, обновлять и управлять зависимостями проекта, делая процесс разработки более эффективным и организованным.

Основные возможности

Управление зависимостями: автоматическое скачивание и обновление библиотек

Контроль версий: фиксация стабильных версий пакетов

Автозагрузка классов: автоматическая настройка автозагрузки

Скрипты: запуск пользовательских команд при определённых событиях

Плагины: расширение функционала через дополнительные модули

Vite — это современный инструмент для настройки среды разработки и сборки веб-проектов. Он использует передовые технологии JavaScript для обеспечения быстрой и эффективной разработки.

Основные возможности

Быстрая разработка: мгновенный запуск сервера и горячая замена модулей

Модульная архитектура: поддержка разделения проекта на компоненты

Оптимизация: автоматическая сборка для продакшена

Транспиляция: поддержка современных стандартов JavaScript

Интеграция: работа с различными фреймворками и библиотеками

Выбор инструмента.

При обучении мы изучали три основных технологии для веб-разработки:

- Nodejs;
- React;
- Laravel;

React - библиотека для языка программирования JavaScript. Цель React — предоставить высокую скорость, простоту и работоспособность приложений на разных платформах. У него высокая скорость разработки — если обычное веб-приложение на запрос от браузера возвращает ему HTML-разметку с CSS, чтобы тот отрисовал страницу, то в случае приложений на React, браузер сначала скачивает набор скриптов, которые выполняются на устройстве пользователя. Короче говоря, это снимает нагрузку с сервера, улучшая производительность проекта. React может использоваться без каких-либо фреймворков, но чаще нужен кто-то дополнительный, чтобы улучшить его эффективность и гибкость.

Node.js - платформа на C++, которая превращает JavaScript в язык общего назначения, поэтому с Node.js можно писать код и для бэкенда, и для фронтенда. Чаще всего используем его для разработки проектов с постоянным обменом информацией с пользователем: социальных сетей, чатов.

Laravel. Внутри него есть много готовых вещей: маршрутизация, HTML-шаблонизаторы, ORM, CLI и другое. Чем хороши продукты на нем:

- Они производительны — будут быстро и четко выполнять задачи. Почему? Потому что фреймворк содержит оптимизированные компоненты и инструменты. Например, механизмы кэширования, которые уменьшают нагрузку на сервер и ускоряют загрузку страниц;
- Можно не переживать за безопасность своего продукта, потому что у Laravel есть встроенные базовые механизмы для защиты от угроз и атак.
- Продукт легко масштабировать. У фреймворка есть нативные инструменты, через которые можно расширить функционал. А так как у него большое комьюнити, то найти разработчиков для поддержки проекта будет несложно.

Итого — проект на Laravel безопасен и надежен, его можно без проблем масштабировать.

Моим выбором стал Laravel, как инструмент для создания безопасного и масштабируемого сайта.

Теоретическая и практическая главы

Что такое Laravel?

Философия Laravel состоит в том, чтобы дать вам из коробки максимальное количество удобных инструментов, причем желательно однозначно удобных, надежных и простых в эксплуатации. Именно поэтому в документации одним из первых упоминается Homestead.

Что такое Homestead

Laravel Homestead — это официальная предустановленная виртуальная машина на базе Vagrant, созданная специально для разработки приложений на Laravel. Она предоставляет полностью настроенную среду разработки со всеми необходимыми компонентами.

Основные компоненты

Базовая конфигурация включает:

- PHP с необходимыми расширениями
- Веб-сервер Nginx
- СУБД MySQL
- Redis
- Другие инструменты для разработки Laravel-приложений

Преимущества использования

Ключевые достоинства Homestead:

- Единая среда разработки для всей команды
- Отсутствие необходимости установки ПО на локальную машину

- Полная изоляция проекта
- Совместимость с различными ОС (Linux, Mac, Windows)
- Простота настройки и обновления

Фреймворк гарантирует хорошую производительность, отдельно стоит упомянуть кэширование. Благодаря соответствующему драйверу файловая система сохраняет в себе большое количество различных элементов. Подобный подход способствует более быстрой разработке самых разных по сложности приложений. Система аутентификации в Laravel очень удобна, с ее помощью можно даже контролировать доступ к имеющимся ресурсам. То есть неавторизованные пользователи, как говорится, не останутся незамеченными.

Платформа дружит с архитектурным дизайном MVC. Он оказывает помощь в разделении бизнес-логики и интерфейса для обычного пользователя. Движок таких шаблонов тоже поддерживается фреймворком Ларавел. Поэтому тут зеленый свет гарантирован нативному PHP-коду.

Миграция баз данных

Итак, миграция — это нечто вроде системы контроля для переноса ваших таблиц в БЗ с помощью конструктора таблиц. Миграция позволит вам избежать ошибок и конфликтов во время конструирования таблиц в базе данных для большого проекта вместе с участниками другой команды. Кроме того, это позволит взаимодействовать с базой данных не с помощью таких инструментов, как MySQL WorkBench или PhpMyAdmin, а напрямую из кода, в зависимости от потребностей вашего проекта в таблицах данных. Добавление и удаление таблиц записывается в истории миграций

ORM

ORM — система объектно-реляционного отображения, которая связывает базы данных с концепциями объектно-ориентированного программирования. Звучит на первый взгляд страшно, однако это напрямую связано с миграцией баз данных: на каждую таблицу создается свой класс — модель, который используется только для работы с этой таблицей. Это позволяет не разводить лишней работы в самой базе данных, а взаимодействовать с ней опять же напрямую из проекта. В итоге это получается и удобней, и надежней. Конечно, на освоение команд и особенностей генерации моделей уйдет некоторое время, но на создание большого проекта с огромным количеством таблиц его уйдет куда больше. Речь идёт о Eloquent. Данная система дает возможность работать с различными БД. Это достигается путем реализации шаблона под названием ActiveRecord. То есть можно работать, но при этом не создавать SQL-запросы повышенного уровня сложности.

```
10  */
11  use Illuminate\Database\Eloquent\Builder;
12
13  User::where('city', '=', 'Berlin')
14      →orWhere('city', '=', 'Chicago')
15      →orWhere('city', '=', 'New York')→get();
16
17
18
19  User::where(['Country', '=', 'Germany'])
20      →where(function($q){
21          $q→where('city', '=', 'Berlin')
22             →orWhere('city', '=', 'München');
23      })→get();
24
25
```

Blade — шаблоны: удобное представление вида

Blade — это мощный шаблонизатор, встроенный в фреймворк Laravel, который предоставляет удобный способ создания и управления представлениями веб-приложений.

На самом деле, все шаблоны Blade компилируются в обычный PHP-код и кешируются до тех пор, пока не будут изменены, что означает, что Blade добавляет фактически нулевую нагрузку вашему приложению. Файлы шаблонов Blade используют расширение файла `.blade.php` и обычно хранятся в каталоге `resources/views`. Шаблоны Blade могут быть возвращены из маршрутов или контроллера с помощью глобального помощника `view`.

Вы можете отображать данные, которые передаются в шаблоны Blade, заключив переменную в фигурные скобки.

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

    @slot('title')

        Home Page

    @endslot

    <div class="col-6">

        @component('inc.alert')

            This is the alert message here.

        @endcomponent

        <h1>Welcome</h1>

    </div>
```

Вы можете создавать операторы `if`, используя директивы `@if`, `@elseif`, `@else`, и `@endif`. Эти директивы работают так же, как и их аналоги в PHP.

Директивы аутентификации

Директивы `@auth` и `@guest` могут использоваться для быстрого определения, является ли текущий пользователь аутентифицированным или считается гостем.

Artisan — это интерфейс командной строки, который поставляется вместе с Laravel. Он позволяет генерировать модели, контроллеры, новые тесты, уведомления — прямо из командной строки. Это куда удобнее, чем каждый раз копировать откуда-то шаблон класса или даже писать его ручками.

```
Command Prompt
C:\Users\Anonymous>cd C:\Users\Anonymous\Desktop\GfG\Laravel-Artisan
C:\Users\Anonymous\Desktop\GfG\Laravel-Artisan>php artisan
Laravel Framework 6.6.0

Usage:
  command [options] [arguments]

Options:
  -h, --help                Display this help message
  -q, --quiet               Do not output any message
  -V, --version             Display this application version
  --ansi                   Force ANSI output
  --no-ansi                Disable ANSI output
  -n, --no-interaction     Do not ask any interactive question
  --env[=ENV]              The environment the command should run under
  -v|vv|vvv, --verbose     Increase the verbosity of messages: 1 for normal output, 2 for more verbose output and 3 for debug

Available commands:
  clear-compiled            Remove the compiled class file
  down                     Put the application into maintenance mode
  env                      Display the current framework environment
  help                     Displays help for a command
```

Эффективная работа с трафиком. Чем известнее сайт, тем большее число запросов в секунду обязано принимать и пускать в обработку приложение. Соответственно, сервер получает приличную нагрузку, а хостинг увеличивается в цене. В таком темпе сервер иногда решает не отвечать, и данные могут потеряться. Но подобные риски с Laravel сведены к минимуму. Причина – реализация в фреймворке интересной системы информационной очереди. С ее помощью нагрузка на сервер упорядочивается. И работа не прерывается, и с данными все в порядке.

Валидация. Предусматривает проверку всех входящих данных. Позволяет обнаружить и устранить случайную ошибку. Ввод неверных данных не означает «падения» сайта, так как приводит к возврату на предыдущую страницу.

Маршрутизация — это процесс определения, какие действия должны выполняться при обращении к определённым URL-адресам веб-приложения. Laravel предоставляет мощный и гибкий механизм маршрутизации для управления URL и их обработчиками.

В Laravel маршруты определяются в следующих файлах:

- **web.php** — для веб-приложений с сессиями и cookies
- **api.php** — для API-запросов
- **console.php** — для консольных команд
- **channels.php** — для событий вещания

Laravel поддерживает все основные HTTP-методы:

```
Route::get('/user', function () {  
  
    return 'GET запрос';  
  
});
```

```
Route::post('/user', function () {  
  
    return 'POST запрос';  
  
});
```

```
Route::put('/user', function () {  
  
    return 'PUT запрос';  
  
});
```

```
Route::delete('/user', function () {  
  
    return 'DELETE запрос';  
  
});
```

Преимущества и недостатки

По многочисленным опросам среди программистов Laravel стабильно занимает высокие места (часто – попросту первое) в качестве самого популярного фреймворка. Причем в самых разных направлениях программирования – как для личных целей, так и для бизнеса. Помимо упомянутого выше обширного функционала, другими важными преимуществами программной платформы выступают:


1. Очень развитое и многочисленное комьюнити, делающее сообщество пользователей Laravel и созданных ими интернет-ресурсов настоящей и полноценной экосистемой. Вершиной подобной деятельности стало проведение с 2013 года ежегодных мировых конференций под названием Laracon.
2. Отменная производительность. Фреймворк ориентирован на поддержку формата баз данных NoSQL, обеспечивающих высокую скорость обработки и обмена информацией.
3. Безопасность. Достигается за счет нескольких встроенных опций защиты, простой и эффективной.
4. Открытый код. Предоставляет возможность вносить изменения в собственную версию ПО.



5. Понятный и лаконичный синтаксис. Обеспечивает удобство чтения программного кода, в котором не используются длинные или сложные конструкции.
6. Мультиязычность. Laravel поддерживает разные языки и предусматривает удобную настройку многоязычности.
7. Широкий спектр разнообразных библиотек и пакетов. В распоряжении программиста находится множество вспомогательных инструментов, делающих его работу проще, быстрее и удобнее.

Недостатки, которые можно встретить на тематических сайтах, не существенны.

Установка Laravel.

Первым делом, необходимо установить PHP с официального сайта php.net, следуя инструкциям для операционной системы.

 www.php.net/downloads.php

 Downloads Documentation Get Involved Help 

Downloads & Installation Instructions

[Installing PHP](#) is covered thoroughly in the PHP documentation.

Binaries

Binaries are available for [Microsoft Windows](#). The PHP project does not currently release binary packages for macOS, but they are packaged by distributions and other providers. For more information, see:

- [Installation instructions for Unix systems](#)
- [Installation instructions for macOS](#)

Source Code

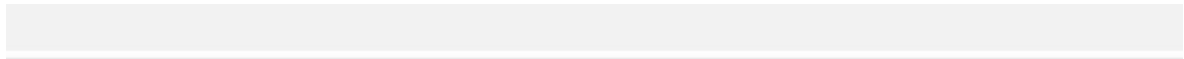
Current Stable PHP 8.4.10 (Changelog)

- [php-8.4.10.tar.gz \(sig\)](#) [21,222Kb]
sha256: bd25c40ece60d1b3c879c11f517d335b8d6a872174c32ebb088b9494d8bb2cf2
- [php-8.4.10.tar.bz2 \(sig\)](#) [17,172Kb]
sha256: 8815d10659cde5f03be4d169205d62b7b29ed0edc7cdd84b6384cda0310c3108
- [php-8.4.10.tar.xz \(sig\)](#) [13,306Kb]
sha256: 14983a9ef8800e6bc2d920739fd386054402f7976ca9cd7f711509496f0d2632
- [Windows downloads](#)

[GPG Keys for PHP 8.4](#)

Установить Composer с официального сайта getcomposer.org. После установки проверить, что Composer установлен правильно, выполнив команду в терминале:

```
composer --version
```



A Dependency Manager for PHP

Latest: **2.8.9** ([changelog](#))

[Getting Started](#)

[Download](#)

[Documentation](#)

[Browse Packages](#)

[Issues](#)

[GitHub](#)

Authors: [Nils Adermann](#), [Jordi Boggiano](#) and many [community contributions](#)

Sponsored by:

Установить Laravel с помощью Composer: выполнить команду

composer global require laravel/installer

— это установит Laravel Installer как глобальный инструмент, что позволяет создавать новые проекты Laravel из любого места на машине.

Необходимо отметить, что Laravel активно использует пакетный менеджер NPM. NPM, он же Node Package Manager — это менеджер пакетов, который идет в комплекте с Node.js. Он нужен, чтобы не писать весь код проекта с нуля, а использовать уже готовые решения. Таким образом, для нормальной работы Laravel, желательно установить Node.js.

ckage-manager/all

5 [Contribute](#) [Certification](#)

Installing Node.js via Package Managers

The packages on this page are maintained and supported by their respective packagers, not the Node.js core team. Please report any issues you encounter to the package maintainer. If it turns out your issue is a bug in Node.js itself, the maintainer will report the issue upstream.

Alpine Linux

Node.js LTS and npm packages are available in the Main Repository.

```
1 apk add nodejs npm
```

Shell [Copy to clipboard](#)

Node.js Current can be installed from the Community Repository.

```
1 apk add nodejs-current
```

Shell [Copy to clipboard](#)

Android

Vite – это современный инструмент сборки фронтенда, который обеспечивает чрезвычайно быстрое окружение разработки и собирает ваш код для продакшена. При создании приложений с использованием Laravel вы обычно используете Vite для сборки файлов CSS и JavaScript вашего приложения в готовые к продакшену ресурсы.

Laravel интегрируется с Vite без проблем, предоставляя официальный плагин и директиву Blade для загрузки ваших ресурсов как для разработки, так и для продакшена. В новой установке Laravel в корне структуры каталогов вашего приложения вы найдете файл `package.json`. В файле `package.json` уже содержится все необходимое для начала работы с Vite и плагином Laravel. Вы можете установить зависимости фронтенда вашего приложения через NPM.

```
npm install
```

Vite настраивается с помощью файла `vite.config.js` в корне вашего проекта. Вы можете настраивать этот файл по своему усмотрению, а также устанавливать любые другие плагины, необходимые для вашего приложения, такие как `@vitejs/plugin-vue` или `@vitejs/plugin-react`.

Выполните команду `composer create-project laravel/laravel <имя проекта>`, где имя проекта — это имя вашего проекта.

Установка Breeze.

[Laravel Breeze](#) – это минимальная и простая реализация всех функций [аутентификации](#) Laravel, включая вход в систему, регистрацию, сброс пароля, подтверждение по электронной почте и подтверждение пароля.

вы можете вручную установить Laravel Breeze с помощью Composer:

```
composer require laravel/breeze --dev
```

После установки пакета Laravel Breeze с помощью Composer, вы должны выполнить команду *Artisan breeze:install*. Эта команда публикует представления аутентификации, маршруты, контроллеры и другие ресурсы в вашем приложении.

К сожалению, установка Breeze обновляет таблицу стилей, и если вы уже настроили файлы CSS, придётся проделать работу заново. К тому же, новые CSS файлы придётся регистрировать в файле `vite.config.js` и запускать установку

```
npm run build
```

```
import { defineConfig } from 'vite';
import laravel from 'laravel-vite-plugin';

export default defineConfig({
  plugins: [
    laravel({
      input: ['resources/css/app.css', 'resources/js/app.js', 'resources/css/style.css', 'resources/css/team.css', 'resources/css/reg.css'],
      refresh: true,
    }),
  ],
});
```

Настройка базы данных.

Настройка базы производится в файле .env.

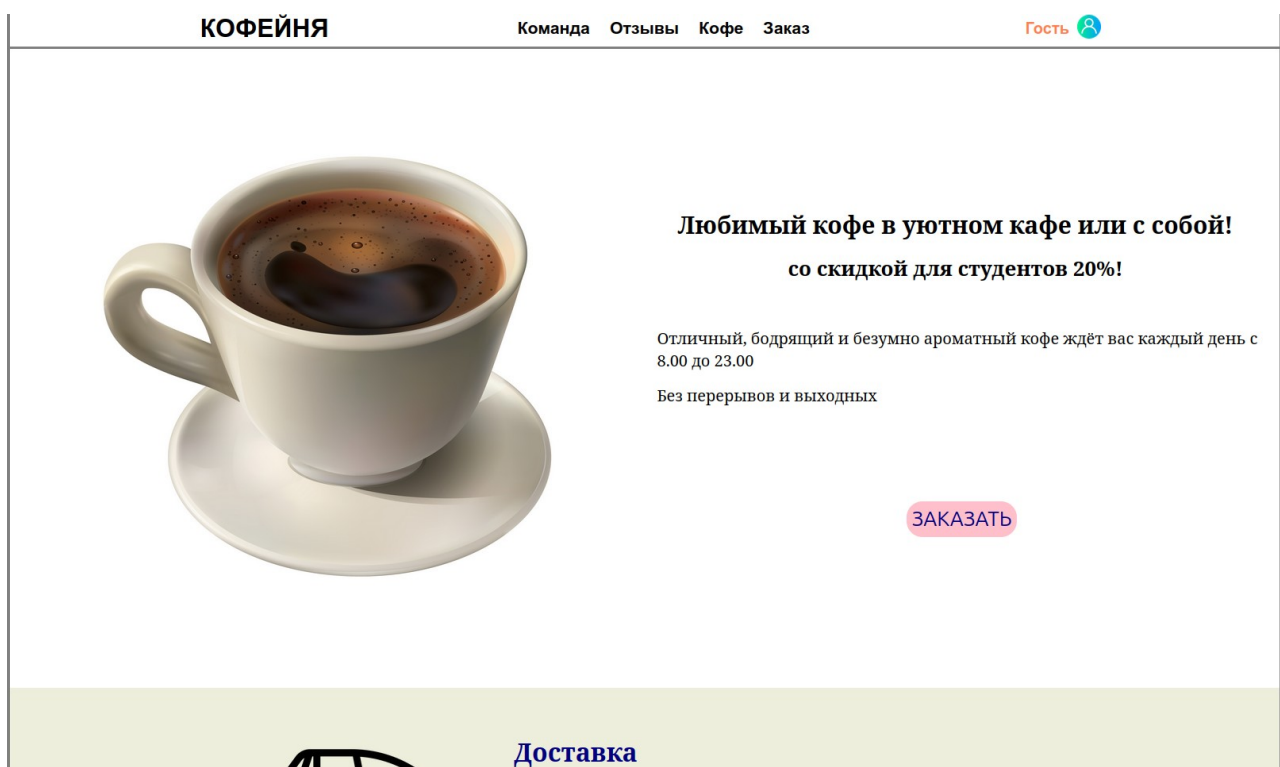
```
DB_CONNECTION=sqlite
DB_HOST=127.0.0.1
DB_PORT=3306
DB_DATABASE=database/data.db
# DB_USERNAME=root
# DB_PASSWORD=
```

Для проекта выбрана база данных SQLite3, как легко настраиваемая и перемещаемая.

Главные страницы.

Для отображения страниц был создан шаблон `layout.blade.php` в папке `resources/views/layouts`, который в свою очередь, состоит из компонентов: `head.blade.php`, `header.blade.php` и `footer.blade.php`. Это позволяет не создавать `header` и `footer` на каждой странице.

Главная страница сайта отображает режим работы



Она связана со страницей информации о работниках (Команда)

КОФЕЙНЯ

Команда

Отзывы

Кофе

Заказ

Гость

Наша команда

<p>Lorem, ipsum. Lorem ipsum dolor sit amet consectetur adipiscing elit. Explicabo, voluptate.</p>	<p>Lorem, ipsum. Lorem ipsum dolor sit amet consectetur adipiscing elit. Explicabo, voluptate.</p>	<p>Lorem, ipsum. Lorem ipsum dolor sit amet consectetur adipiscing elit. Explicabo, voluptate.</p>	<p>Lorem, ipsum. Lorem ipsum dolor sit amet consectetur adipiscing elit. Explicabo, voluptate.</p>

Отзывы

КОФЕЙНЯ

Команда

Отзывы

Кофе

Заказ

Гость

Отзывы

Добавить отзыв

<p>Кофе вкусный, а вот блины пересушены</p> <p>Алефтина</p>	<p>Добрые напитки, добрые бариста</p> <p>Михаил</p>
<p>Неплохо</p> <p>Гость</p>	

Каталог (Кофе)

Кофе



Американо

Акварельный набросок в пастельных тонах

190py6



Капучино

Lorem ipsum dolor sit amet consectetur adipisicing elit.
 Mollitia, ipsum.

210руб



Мокачино

Lorem ipsum dolor sit amet consectetur adipisicing elit.
 Mollitia, ipsum.

190py6



Ирландский кофе

Ирландский кофе

190py6



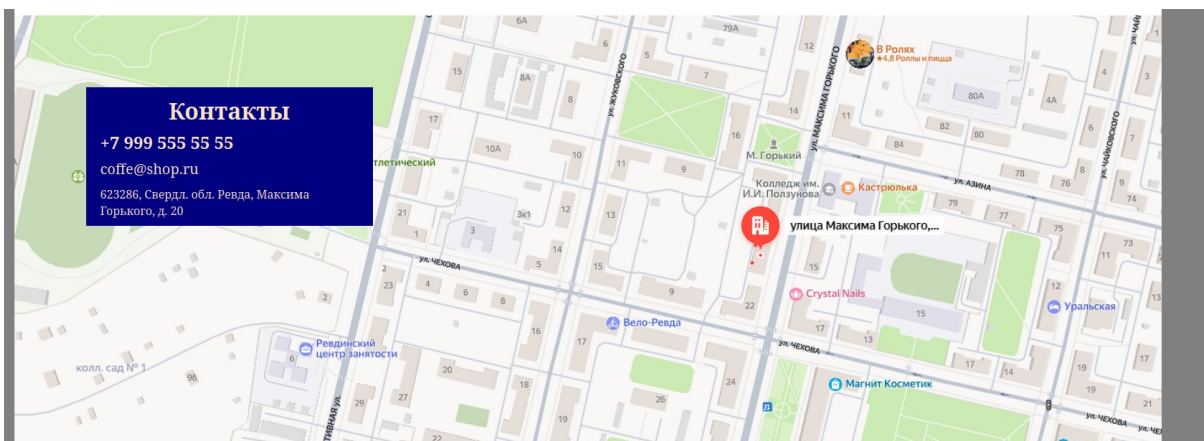
► Показать ещё

Гость

Ваша корзина пуста

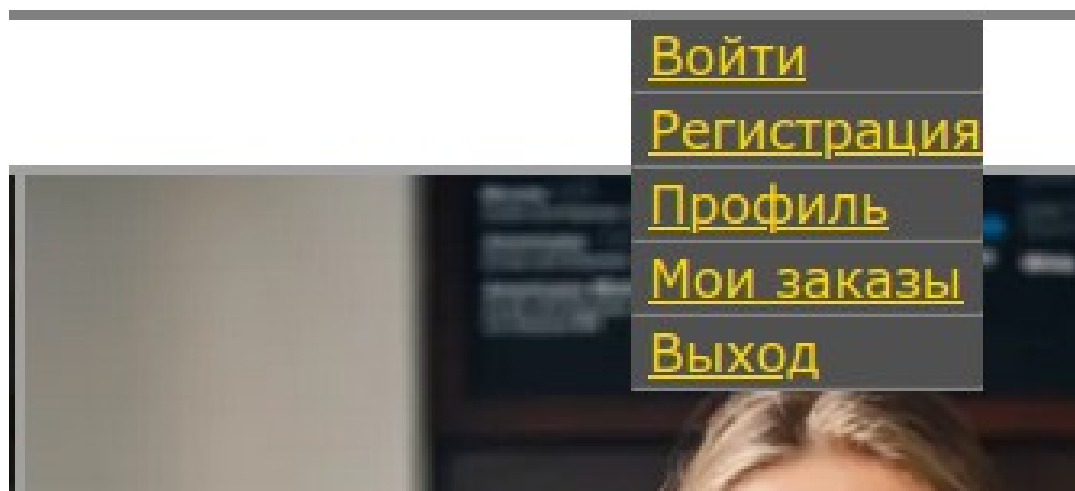


33

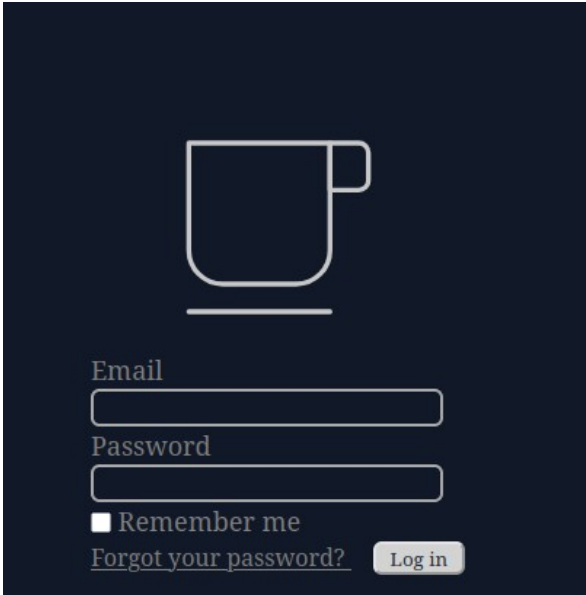


Header — меню для навигации, а также выпадающее меню пользователя.

Гость 



Страницы пользователя, в основном, сделано на основе blade шаблонов Breeze. Вход



A login form on a dark blue background. At the top is a white line-art icon of a cup. Below it are two input fields labeled "Email" and "Password". Under the password field is a checkbox labeled "Remember me" and a link "Forgot your password?". To the right of these is a "Log in" button.


Регистрация:




A registration form on a dark blue background. At the top is a white line-art icon of a cup. Below it are input fields for "Name", "Email", "Phone", "Password", and "Confirm Password". At the bottom is a text area labeled "Ваш адрес" (Your address). Below the address field are a link "Already registered?" and a "Register" button.

Личная страница:

Профиль




Загрузить файл

ИНФО

Name

Михаил

Email

user1@mail.ru

Phone

1-11-11

user user user


Сохранить

Заказы:

Ваши заказы					
2	Капучино - 1; ИТОГО: 210 руб.	Оплата СБП Оплатить	Самовывоз	Контакты: user1@mail.ru 1-11-11	Заказ создан
3	Тирамису - 1; ИТОГО: 100 руб.	Оплата При получении Оплатить	Заказать столик	Контакты: user1@mail.ru 1-11-11	Заказ создан

А также Выход, для смены пользователя.

Страница Заказы связана с оформлением



Блинчики с
повидлом
60 руб.

Ваша корзина

1. Блинчики с повидлом - Количество: 1 - Сумма - 60 руб.

ИТОГО: 60 руб.

Оформить

При нажатии оформить:

Оформление заказа

Состав:

1. Блинчики с повидлом - Количество: 1 - Сумма - 60 руб.

ИТОГО: 60 руб.

1. Способ оплаты

- ☐ При получении
☐ СБП
☐ Банковской картой

2. Персональные данные

Ваш e-mail:

Ваш телефон:

3. Способ доставки

- ☐ Заказать столик
☐ Самовывоз
☐ Курьером

[Подтвердить](#)

Ваш заказ №4 создан

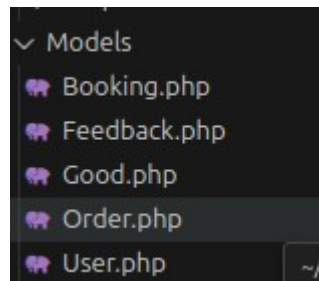
4	Блинчики с повидлом - 1; ИТОГО: 60 руб.	Оплата СБП Оплатить	Самовывоз	Контакты: user1@mail.ru 1-11-11	Заказ создан
---	---	--	-----------	---------------------------------------	-----------------

Ваши заказы

2	Капучино - 1; ИТОГО: 210 руб.	Оплата СБП Оплатить	Самовывоз	Контакты: user1@mail.ru 1-11-11	Заказ создан
3	Тирамису - 1; ИТОГО: 100 руб.	Оплата При получении Оплатить	Заказать столик	Контакты: user1@mail.ru 1-11-11	Заказ создан

Модели

Для работы сайта были созданы модели:



User — модифицированная модель, созданная Breeze.

Поля:

```
protected $fillable = [  
    'id',  
    'name',  
    'email',  
    'password',  
    'role',  
    'onDesc',  
    'address',  
    'phone',  
    'avatar'  
];  
protected $hidden = [
```

id — идентификатор,

name — отображаемое имя,

email, address и phone — контактные данные,

avatar — картинка для самоидентификации,

role — для определения клиента (0), сотрудника (1) и админа(2).

```
class Order extends Model  
{  
    protected $fillable = [  
        'id',  
        'count' ,  
        'cat'  
    ];  
}
```

id — идентификатор товара,

count — количество товара,

cat — категория товара (кофе, десерты, блины).

Используется для фильтрации при автосборке каталога.

Класс order — это корзина, неоформленный заказ. Обнуляется после оформления.

```
class Good extends Model
{
    protected $fillable = [
        'id',
        'name',
        'desc',
        'price',
        'new',
        'src'
    ];
}
```

id — идентификатор товара,

name — наименование товара,

desc -описание товара,

price — стоимость товара,

new — новинка, отображается в ленте,

src — путь к фотографии товара.

```
class Feedback extends Model
{
    protected $fillable = [
        'id',
        'user_id',
        'text',
    ];
}
```

id — идентификатор,

user_id — связь с пользователем, если никто не вошел в систему, то равен -1 (Гость),

text — содержание отзыва.

```
class Booking extends Model
{
    protected $fillable = [
        'id',
        'user_id',
        'composition',
        'time_creation',
        'time_modification',
        'status',
        'address',
        'paid',
        'pay_method',
        'phone',
        'email',
        'amount'
    ];
}
```

Booking — это модель оформленного товара

id — идентификатор,

user_id — связь с пользователем, если никто не вошел в систему, то связь с заказом осуществляется через e-mail,

`composition` — состав заказа в виде json строки,

`ststus` — состояние обработки заказа,

`address` — указывает куда доставить заказ, предусматривает заказ столика или выдачу на руки (самовывоз),

`paid` — оплачен заказ или нет,

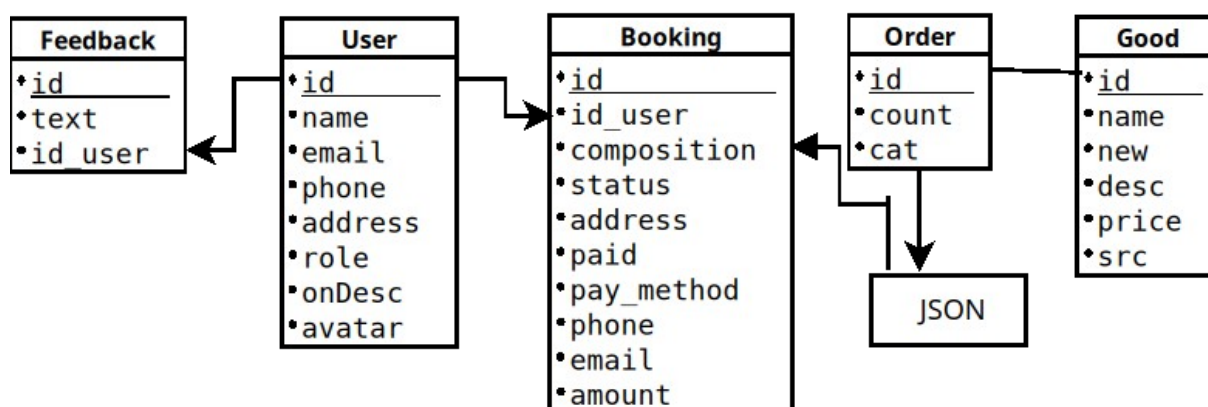
`pay_method` — метод оплаты,

`phone, email` — контакты заказчика.

База данных.

Каждой модели в базе соответствует таблица. За связь отвечает модуль **Eloquent**, который позволяет получать данные из таблиц без sql запросов. Такой подход защищает данные от sql инъекций и несанкционированного доступа.

Схема базы данных



Связь между моделями и таблицами базы создаётся при помощи миграции. Пример миграции расположен ниже.

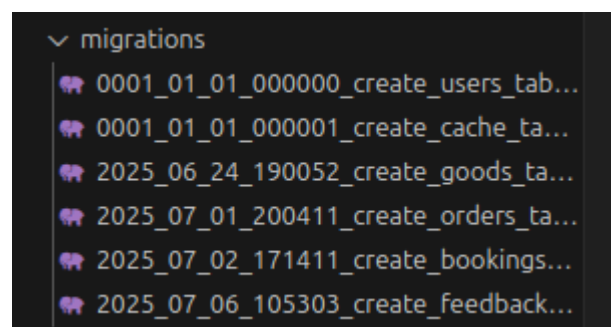
```

return new class extends Migration
{
    /**
     * Run the migrations.
     */
    public function up(): void
    {
        Schema::create('cache', function (Blueprint $table) {
            $table->string('key')->primary();
            $table->mediumText('value');
            $table->integer('expiration');
        });

        Schema::create('cache_locks', function (Blueprint $table) {
            $table->string('key')->primary();
            $table->string('owner');
            $table->integer('expiration');
        });
    }

    /**
     * Reverse the migrations.
     */
    public function down(): void
    {
        Schema::dropIfExists('cache');
        Schema::dropIfExists('cache_locks');
    }
};

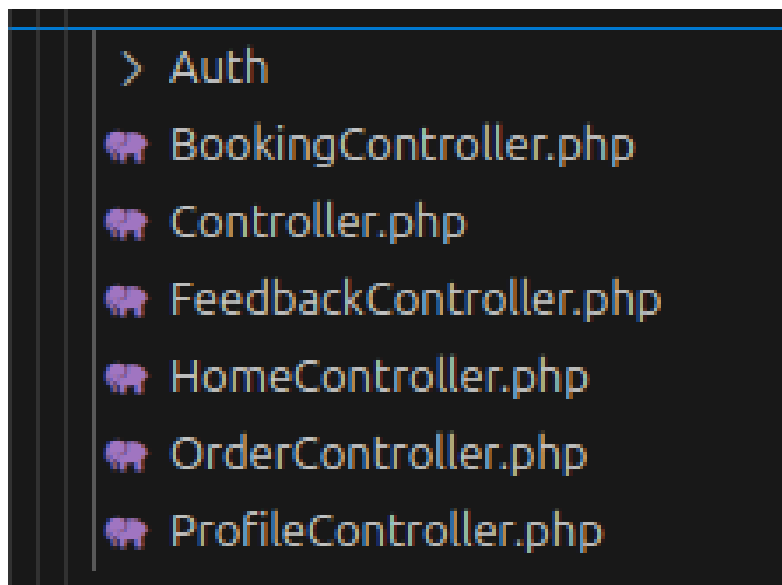
```



Контроллеры.

Контроллеры — основные обработчики запросов и преобразователи данных. Именно при помощи контроллеров вызываются шаблоны для отображения страниц.

В проекте созданы следующие контроллеры:



Создаются контроллеры командой

```
php artisan make:model <имя> -mc
```

Эта команда создаёт модель, контроллер и миграцию.

В папке Auth содержатся контроллеры Breeze, отвечающие за авторизацию.

Пример контроллера:

```
<?php

namespace App\Http\Controllers;
use Illuminate\Support\Facades\Auth;
use Illuminate\Http\Request;
use App\Models\Order;
use Illuminate\Support\Facades\Storage;
    use App\Models\User;
    use App\Models\Booking;
    use App\Models\Feedback;

class HomeController extends Controller
{
    public function exit(Request $request, $parent) {
        $parent = '/' . $parent;
        Auth::guard('web')->logout();

        $request->session()->invalidate();

        $request->session()->regenerateToken();

        //return $this->parent;
        return redirect(url($parent)) ;
    }
}
```

Безопасность

За безопасность данных в базе, как отмечалось отвечает система Eloquent. По умолчанию Eloquent ORM Laravel защищает от SQL-инъекций путём параметризации запросов и использования привязок SQL. [laravel.su](https://laravel.com/docs/5.4/queries)

Некоторые другие механизмы безопасности в Laravel:

Защита от межсайтового скриптинга (XSS). При выводе представлений все выходные данные автоматически экранируются, что предотвращает внедрение вредоносных скриптов на страницы.

Защита от атак Cross-Site Request Forgery (CSRF). Для отправки форм Laravel автоматически генерирует токены CSRF и проверяет их, чтобы убедиться в легитимности запроса.

Хэширование паролей. По умолчанию пароли хэшируются с использованием алгоритма bcrypt, поэтому даже если злоумышленники получат доступ к данным в базе, они не смогут прочитать пароли.

Шифрование. Laravel предоставляет простые и эффективные меры шифрования для защиты чувствительных данных, которые присутствуют в приложениях.

За безопасность пользователей отвечает фасад Auth. Он позволяет

Laravel Auth — это комплексная система безопасности, обеспечивающая защиту веб-приложений от несанкционированного доступа.

Ключевые элементы системы:

Auth Guard — механизм проверки подлинности пользователей

Система сессий — управление состоянием пользователя

Хеширование паролей — защита конфиденциальных данных

Middleware — фильтрация запросов

Авторизация — управление правами доступа

Принцип работы системы

Механизм аутентификации включает:

Проверку учетных данных пользователя

Управление состоянием сессии

Контроль доступа к ресурсам

Защиту от несанкционированного доступа

В Laravel есть встроенная файловая система (Laravel File Storage), которая абстрагирует работу с различными системами хранения файлов. Система построена на основе пакета Flysystem PHP и предоставляет единый интерфейс для взаимодействия с разными вариантами хранения

Особенности:

позволяет хранить файлы локально (на сервере) или в облаке (например, Amazon S3);

поддерживает другие удалённые варианты (FTP, Rackspace и др.);

позволяет управлять видимостью файлов (public/private) и шифрованием. По-умолчанию файлы сохраняются по пути *project/storage/app/private*, если необходимо обеспечить видимость файлов из интернета (например, аватарки пользователей), необходимо настроить общедоступное (public) хранилище.

Это можно сделать настроив файл `filesystems.php` из папки `config`.

Я внёс в массив `disks` значение `public_upload`.

```
'disks' => [  
    'local' => [  
        'driver' => 'local',  
        'root' => storage_path('app/private'),  
        'serve' => true,  
        'throw' => false,  
        'report' => false,  
    ],  
    'public' => [  
        'driver' => 'local',  
        'root' => storage_path('app/public'),  
        'url' => env('APP_URL').'/storage',  
        'visibility' => 'public',  
        'throw' => false,  
        'report' => false,  
    ],  
    'public_uploads' => [  
        'driver' => 'local',  
        'root' => public_path() . '/uploads',  
    ],  
]
```


Заключение

Что сделано.

В процессе создания сайта получен дополнительный опыт по работе с фреймворком Laravel.

Созданы основные страницы сайтов.

Настроены локальное и публичное хранилища.

Созданы основные модели.

К моделям подключена база данных.

Осуществлена система создания заказов.

Создана страница для публикации отзывов пользователей.

Что не сделано.

Не доконца реализована админская панель.

Не все страницы удовлетворительно стилизованы.

Не доконца реализована адаптивность.

Причина: недостаточно опыта, и, как следствие, не хватило времени для реализации всех задумок.

Выводы:

Результатом выпускной квалификационной работы является веб-приложение для кафе, предназначенная для повышения товарооборота и представительства своей деятельности в сети Интернет.

Было проведено проектирование веб-приложения. Выделены основные сущности для хранения информации при осуществлении деятельности магазина.

Разработана структура базы данных, построены модели данных.

Цель выпускной квалификационной работы достигнута.

Список литературы:

1. **GitHub: Laravel** [Электронный ресурс] / Официальный репозиторий Laravel.
- 2 Кузнецов, Максим Самоучитель PHP 5 - М.: БХВ-Петербург, 2004. - 560 с.
- 3 Котеров, Симдянов. PHP 7 - М.: БХВ-Петербург, 2010. - 1160 с.