Министерство образования и молодежной политики

Свердловской области

ГАПОУ СО «НИЖНЕТАГИЛЬСКИЙ ТОРГОВО-ЭКОНОМИЧЕСКИЙ

КОЛЛЕДЖ»

Комиссия информационных технологий

КУРСОВАЯ РАБОТА

ПРОЕКТИРОВАНИЕ И РАЗРАБОТКА ВЕБ-САЙТА ДЛЯ ЗАКАЗА ПРОДУКТОВ

Основы алгоритмизации и программирования

|  |  |
| --- | --- |
| Специальность: 09.02.07 «Информационные системы и программирование»  Дата защиты: \_\_\_\_\_\_\_\_\_\_\_\_\_\_  Оценка: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | Исполнитель: И. А. Кардапольцев  Группа 3ИС–6  Руководитель: Е.Д. Репьев  преподаватель |

Нижний Тагил 2024

СОДЕРЖАНИЕ

[ВВЕДЕНИЕ 3](#_Toc168576926)

[1. АНАЛИТИЧЕСКАЯ ЧАСТЬ 5](#_Toc168576927)

[1.1. Теоретические основы разработки веб-приложений 5](#_Toc168576928)

[1.2. Обзор современных технологий веб-разработки 8](#_Toc168576929)

[1.3. Основы фреймворка Laravel 10](#_Toc168576930)

[2. ПРАКТИЧЕСКАЯ ЧАСТЬ 13](#_Toc168576931)

[2.1. Проектирование веб-приложения 13](#_Toc168576932)

[2.2. Проектирование архитектуры приложения 14](#_Toc168576933)

[2.3. Проектирование пользовательского интерфейса 18](#_Toc168576934)

[2.4. Реализация веб-приложения и создание базовых миграций и моделей 18](#_Toc168576935)

[ЗАКЛЮЧЕНИЕ 30](#_Toc168576936)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 33](#_Toc168576937)

ВВЕДЕНИЕ

Актуальность темы исследования. Сервис по доставке еды в последние годы стал неотъемлемой частью городской жизни. С увеличением темпов жизни и высокой занятостью у людей остается все меньше времени на приготовление пищи. В такой ситуации сервисы по доставке еды предлагают удобное решение, позволяя пользователям заказывать еду из различных ресторанов и кафе прямо к себе домой или в офис. Это особенно актуально в условиях пандемии, когда многие люди предпочитают оставаться дома и минимизировать контакты с окружающими. Разработка эффективного и удобного веб-приложения для такого сервиса требует применения современных технологий и подходов, чтобы обеспечить надежность, масштабируемость и высокое качество пользовательского опыта.

Целью данной курсовой работы является разработка веб-приложения для сервиса по доставке еды с использованием современных технологий, таких как PHP, Laravel и Orchid. Для достижения поставленной цели необходимо выполнить следующие задачи:

Изучить существующие подходы и технологии для создания сервисов по доставке еды, включая анализ преимуществ и недостатков различных фреймворков и инструментов.

Спроектировать архитектуру веб-приложения, включающую базу данных, серверную часть и интерфейс пользователя.

Реализовать основные функциональные модули приложения, такие как регистрация и аутентификация пользователей, управление ресторанами и меню, создание и обработка заказов.

Обеспечить удобный и интуитивно понятный пользовательский интерфейс с помощью Orchid, интегрированного в Laravel.

Провести тестирование и оптимизацию веб-приложения для обеспечения его надежной работы и высокой производительности.

Оценить полученные результаты и определить перспективы дальнейшего развития приложения.

Предметом исследования является процесс разработки веб-приложения для сервиса по доставке еды, включающий проектирование архитектуры, реализацию функциональных модулей и тестирование.

Объектом исследования является веб-приложение, которое предоставляет пользователям возможность заказывать еду, а администраторам — управлять ресторанами, меню и заказами.

Методологическая база исследования включает следующие этапы. Анализ литературы: изучение нормативных документов, специализированных интернет-ресурсов, учебных пособий и научных статей по веб-разработке и использованию фреймворков Laravel и Orchid.

Проектирование и моделирование: разработка схем базы данных, проектирование архитектуры приложения и пользовательских интерфейсов.

Программирование: реализация функциональных модулей на языке PHP с использованием фреймворков Laravel и Orchid.

Тестирование: проведение функционального и нагрузочного тестирования приложения для выявления и устранения ошибок, а также оптимизации производительности.

Практическая значимость данного исследования заключается в разработке функционального и масштабируемого веб-приложения, которое может быть использовано как основа для создания реального сервиса по доставке еды. Разработанное приложение может быть адаптировано и доработано под конкретные потребности бизнеса, что делает его полезным инструментом для предпринимателей и разработчиков. Кроме того, результаты работы могут быть использованы в образовательных целях для изучения современных технологий веб-разработки.

1. АНАЛИТИЧЕСКАЯ ЧАСТЬ

## 1.1. Теоретические основы разработки веб-приложений

Информационная система (ИС) представляет собой совокупность взаимосвязанных компонентов, используемых для сбора, хранения, обработки и передачи информации. ИС предназначены для автоматизации и оптимизации различных процессов, происходящих в организациях, что позволяет повысить эффективность их деятельности.

База данных (БД) – это организованная коллекция данных, которая хранится и управляется с помощью систем управления базами данных (СУБД). БД обеспечивает структурированное хранение данных, позволяет выполнять запросы на получение, добавление, обновление и удаление данных.

Фреймворк – это набор библиотек и инструментов, предназначенных для упрощения и ускорения процесса разработки программного обеспечения. Фреймворки обеспечивают готовую архитектуру и стандартизированные решения для решения распространенных задач в разработке приложений.

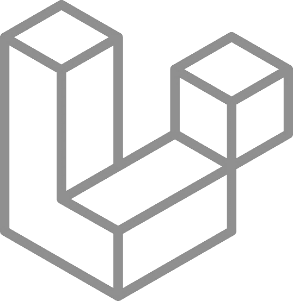


Рисунок 1 – Laravel

Laravel – это популярный фреймворк для разработки веб-приложений на языке программирования PHP. Он предоставляет разработчикам удобные инструменты для выполнения различных задач, таких как маршрутизация, аутентификация, управление базами данных и создание шаблонов [1].



Рисунок 2 – Laravel Orchid

Laravel Orchid — это популярный инструмент для разработки веб-приложений на Laravel, который предоставляет множество преимуществ [15]:

Мощный административный интерфейс: Laravel Orchid предоставляет готовый административный интерфейс, который позволяет управлять данными, пользователями и другими аспектами приложения.

Гибкость и расширяемость: Орхидея построена на базе Laravel, что обеспечивает высокую гибкость и возможность расширения. Вы можете легко добавлять свои модули и компоненты.

Интеграция с другими инструментами: Laravel Orchid интегрируется с другими популярными инструментами, такими как Vue.js, Tailwind CSS и Alpine.js, что облегчает разработку современных интерфейсов [14].

Быстрая разработка: благодаря готовым компонентам и функциональности, предоставляемой Orchid, вы можете быстро создавать приложения без необходимости писать много кода с нуля.

Поддержка множества типов полей: Orchid предоставляет множество типов полей, таких как текстовые поля, выпадающие списки, даты и многое другое, что упрощает работу с данными.

Безопасность: Laravel Orchid следует лучшим практикам безопасности, что обеспечивает защиту ваших данных и приложения.

Жизненный цикл программного обеспечения (ЖЦПО) – это последовательность этапов, через которые проходит программное обеспечение от момента возникновения идеи до его окончательного выведения из эксплуатации. Основные этапы ЖЦПО включают анализ требований, проектирование, разработку, тестирование, внедрение и сопровождение.

Каскадная модель – это линейная модель жизненного цикла программного обеспечения, в которой каждый этап разработки выполняется последовательно и начинается только после завершения предыдущего. Основные этапы каскадной модели включают анализ требований, проектирование, разработку, тестирование, внедрение и сопровождение.

V-образная модель – это расширение каскадной модели, в которой каждому этапу разработки соответствует этап тестирования. Это обеспечивает более строгое соответствие между этапами проектирования и тестирования, улучшая качество программного обеспечения.

Спиральная модель – это итеративная модель жизненного цикла программного обеспечения, которая включает многократное повторение циклов анализа, проектирования, разработки и тестирования. Спиральная модель позволяет более гибко реагировать на изменения требований и уменьшает риски за счет постепенного улучшения продукта.

Agile (гибкая методология) – это подход к разработке программного обеспечения, основанный на итеративном и инкрементальном развитии с тесным взаимодействием с пользователями. Основные принципы Agile включают адаптивное планирование, эволюционное развитие, раннюю доставку и постоянное улучшение.

Нефункциональные требования определяют качества и характеристики системы, такие как производительность, безопасность, надежность, удобство использования и масштабируемость. Эти требования важны для обеспечения высоких стандартов качества и удовлетворения потребностей пользователей.

Функциональные требования описывают конкретные функции и возможности, которые должна предоставлять система. Они включают описание действий, которые система должна выполнять, и условий, при которых эти действия должны происходить.

Стандарты и ГОСТы – это наборы правил и рекомендаций, которые регламентируют процессы разработки и внедрения информационных систем. Примеры стандартов включают ISO/IEC 12207 (процессы жизненного цикла программного обеспечения) и ГОСТ 34.601-90 (стадии создания автоматизированных систем).

Эти основные понятия являются фундаментальными элементами при написании курсовой работы, посвященной разработке веб-приложений для сервиса по доставке еды. Они помогают структурировать теоретическую и практическую часть исследования, обеспечивая логичное и последовательное изложение материала.

## 1.2. Обзор современных технологий веб-разработки

Веб-разработка является одной из наиболее динамично развивающихся областей информационных технологий. Существует множество фреймворков и инструментов, предназначенных для упрощения процесса создания веб-приложений, каждый из которых имеет свои особенности, преимущества и недостатки. В данной главе рассмотрим наиболее популярные фреймворки: Laravel, Django и Ruby on Rails.

Laravel – это фреймворк для веб-разработки на языке PHP, созданный для упрощения и ускорения разработки веб-приложений, благодаря использованию элегантного синтаксиса и мощных инструментов.

Преимущества:

* Простота и элегантность кода: Laravel предоставляет интуитивно понятный синтаксис, что позволяет разработчикам сосредоточиться на логике приложения, а не на технических деталях.
* Широкие возможности: включает в себя множество встроенных функций, таких как маршрутизация, аутентификация, управление сессиями, работа с базами данных через ORM Eloquent и многое другое.
* Расширяемость: благодаря использованию пакетов и модулей, разработчики могут легко добавлять новые функции и расширять возможности приложения.

Недостатки:

* Сложность: для новичков может показаться сложным освоение всех возможностей и инструментов Laravel.
* Производительность: в некоторых случаях Laravel может быть медленнее по сравнению с более легковесными фреймворками из-за большого количества встроенных функций.



Рисунок 3 – Django

Django – это высокоуровневый фреймворк для веб-разработки на языке Python, предназначенный для быстрого создания безопасных и поддерживаемых веб-приложений.

Преимущества:

* Полный стек: Django включает все необходимые компоненты для создания веб-приложений, что позволяет разработчикам сосредоточиться на разработке функциональности, а не на интеграции различных инструментов.
* Безопасность: Встроенные механизмы защиты от распространенных угроз, таких как SQL-инъекции, CSRF и XSS-атаки [10].
* Активное сообщество: Большое и активное сообщество разработчиков, предоставляющее множество дополнительных пакетов и библиотек.

Недостатки:

* Крутая кривая обучения: может быть сложным для новичков из-за большого количества встроенных инструментов и концепций.
* Производительность: подобно Laravel, Django может страдать от проблем с производительностью при использовании во высоконагруженных проектах.

Ruby on Rails – это веб-фреймворк на языке Ruby, предназначенный для упрощения и ускорения разработки веб-приложений благодаря философии «Convention over Configuration» и «Don't Repeat Yourself».

Преимущества:

Продуктивность: Rails позволяет разработчикам быстро создавать функциональные веб-приложения благодаря высокому уровню абстракции и умным стандартам.

Активное сообщество: Обширная экосистема геймплейтов, библиотек и инструментов, а также активное сообщество, предоставляющее поддержку и ресурсы.

Философия: Принципы «Convention over Configuration» и «Don't Repeat Yourself» помогают избежать дублирования кода и упрощают процесс разработки.

Недостатки:

Производительность: Rails может быть менее производительным по сравнению с более легковесными фреймворками.

Сложность масштабирования: в некоторых случаях масштабирование приложений на Rails может потребовать значительных усилий.

## 1.3. Основы фреймворка Laravel

Laravel был создан Тейлором Отвеллом и впервые выпущен в июне 2011 года. Целью разработки Laravel было создание фреймворка, который был бы более современным и функциональным по сравнению с существующими решениями для PHP, такими как CodeIgniter. Laravel быстро завоевал популярность благодаря своей элегантности, простоте использования и мощным инструментам.

Основные особенности Laravel:

Элегантный синтаксис: Laravel предоставляет лаконичный и читаемый синтаксис, что упрощает процесс разработки.

Модульная архитектура: Фреймворк легко расширяется благодаря использованию пакетов и модулей.

Встроенные инструменты: Laravel включает в себя множество встроенных инструментов, таких как маршрутизация, миграции баз данных, система шаблонов Blade, система управления зависимостями через Composer и другие [7].

Основные компоненты Laravel

Маршруты (Routing): определяют, как приложение должно реагировать на входящие запросы. Маршруты в Laravel могут быть определены в файлах маршрутизации, таких как web.php и api.php [3].

Контроллеры (Controllers): Контроллеры обрабатывают запросы, выполняют логику приложения и возвращают ответы. Они служат связующим звеном между моделями и представлениями [11].

Модели (Models): представляют данные и бизнес-логику приложения. В Laravel модели обычно взаимодействуют с базой данных через ORM Eloquent [4].

Представления (Views): отвечают за отображение данных пользователю. В Laravel используется система шаблонов Blade, которая упрощает создание и управление представлениями [5].

Преимущества использования Laravel

Простота и удобство: Laravel предоставляет интуитивно понятный и элегантный синтаксис, что упрощает процесс разработки и снижает порог входа для новых разработчиков.

Мощные встроенные инструменты: Laravel включает в себя множество готовых инструментов и функций, таких как аутентификация, авторизация, управление сессиями, маршрутизация, кеширование и другие, что позволяет быстро создавать функциональные веб-приложения.

Более 40 типов полей: В Laravel Orchid доступно более 40 различных типов полей, которые вы можете использовать в своих формах. Это включает в себя текстовые поля, выпадающие списки, даты и другие элементы управления. Выбор подходящего типа поля упрощает взаимодействие с данными в вашем приложении

Активное сообщество и документация: Laravel имеет большое и активное сообщество разработчиков, которое предоставляет поддержку, дополнительные пакеты и библиотеки. Официальная документация Laravel является одной из самых полных и доступных среди фреймворков.

Безопасность: Встроенные механизмы защиты от распространенных угроз, таких как SQL-инъекции, CSRF и XSS-атаки, обеспечивают высокий уровень безопасности приложений.

Производительность и масштабируемость: несмотря на наличие множества встроенных функций, Laravel оптимизирован для производительности и поддерживает масштабируемость приложений, что позволяет использовать его как для небольших проектов, так и для крупных корпоративных систем.

Таким образом, Laravel является мощным и удобным инструментом для разработки веб-приложений, предлагая разработчикам широкий спектр возможностей и упрощая процесс создания сложных систем.

Основы фреймворка Orchid

Orchid — это административная панель для Laravel, которая предоставляет мощные инструменты для создания пользовательских интерфейсов и управления данными. Orchid позволяет быстро создавать панели администрирования с минимальными усилиями.

История и особенности Orchid

Orchid был создан для того, чтобы упростить создание административных панелей в Laravel. Основные особенности включают:

Компонентный подход к созданию интерфейсов.

Поддержка CRUD-операций с минимальным количеством кода. Интеграция с Laravel и использование его возможностей. Встроенные компоненты для работы с формами, таблицами и диаграммами.

Основные компоненты Orchid

Экраны (Screens): Основные элементы интерфейса, которые отображают данные и позволяют взаимодействовать с ними.

Поля (Fields): Компоненты для создания форм и ввода данных.

Макеты (Layouts): Определяют структуру экранов и расположение элементов.

Сегменты (Sections): Части экрана, которые можно повторно использовать в разных местах.

Преимущества использования Orchid

Быстрое создание административных панелей. Компонентный подход упрощает разработку и поддержку. Гибкость и расширяемость. Легкость интеграции с Laravel и использование его возможностей.

2. ПРАКТИЧЕСКАЯ ЧАСТЬ

## 2.1. Проектирование веб-приложения

Анализ Описание функциональных и нефункциональных требований к системе

Для проектирования веб-приложения по доставке еды необходимо четко определить функциональные и нефункциональные требования к системе.

Функциональные требования:

Регистрация и аутентификация пользователей:

Пользователи должны иметь возможность регистрироваться в системе, предоставляя такие данные, как имя, адрес электронной почты и пароль.

Зарегистрированные пользователи должны иметь возможность входить в систему, используя адрес электронной почты и пароль.

Управление профилем пользователя:

Пользователи должны иметь возможность просматривать и редактировать информацию в своем профиле.

Просмотр и выбор еды:

В приложении должен быть доступен каталог еды.

Каждая категория должна иметь отдельную страницу с подробной информацией, которая включает в себя содержимые в ней продукты.

Оформление заказа:

Пользователи должны иметь возможность добавлять еду в корзину.

Приложение должно предоставлять возможность оформления заказа с выбором адреса доставки.

Управление заказами:

Пользователи должны иметь возможность просматривать историю своих заказов и оставлять на них откзывы.

Администраторы должны иметь возможность управлять пользователями, добавлять и редактировать информацию о категориях/продуктах.

Нефункциональные требования:

Приложение должно обеспечивать быстрое время отклика для основных операций, таких как регистрация, вход в систему, просмотр каталога еды и оформление заказов.

Система должна быть способна обрабатывать большое количество одновременных пользователей без значительных задержек.

Безопасность. Приложение должно защищать данные пользователей, включая личную информацию и данные о заказах.

Пароли пользователей должны храниться в зашифрованном виде.

Система должна быть защищена от распространенных веб-угроз, таких как SQL-инъекции и межсайтовый скриптинг (XSS).

Удобство использования:

Интерфейс приложения должен быть интуитивно понятным и удобным для пользователей.

Дизайн приложения должен быть адаптивным, обеспечивая корректное отображение на различных устройствах, включая мобильные телефоны и планшеты.

Масштабируемость. Приложение должно быть спроектировано таким образом, чтобы легко расширяться и модифицироваться, добавляя новые функции и компоненты.

Целевая аудитория включает две основные группы пользователей:

Конечные пользователи (покупатели): Люди, которые будут использовать приложение для заказа еды.

Администраторы: Сотрудники компании, которые будут управлять приложением, включая добавление и редактирование информации о еде и управление пользователями.

## 2.2. Проектирование архитектуры приложения

Схема базы данных для веб-приложения компании должна включать таблицы, обеспечивающие хранение информации о пользователях, их корзинах, истории покупок, товарах, категориях и отзывах.

Описание MVC-архитектуры приложения

Архитектура приложения основана на паттерне MVC (Model-View-Controller), который обеспечивает четкое разделение ответственности и упрощает поддержку и расширение системы.

«Model» (Модель): Модель представляет данные и логику приложения. В Laravel модели обычно связаны с таблицами базы данных и используются для взаимодействия с данными. Модели в Laravel используют ORM Eloquent для упрощения работы с базой данных.

«View» (Представление): Представление отвечает за отображение данных пользователю. В Laravel представления создаются с использованием системы шаблонов Blade, которая упрощает создание динамических веб-страниц и управление ими.

«Controller» (Контроллер): Контроллеры обрабатывают входящие запросы, выполняют бизнес-логику и определяют, какое представление должно быть возвращено пользователю. Контроллеры служат связующим звеном между моделями и представлениями, обеспечивая взаимодействие между ними.

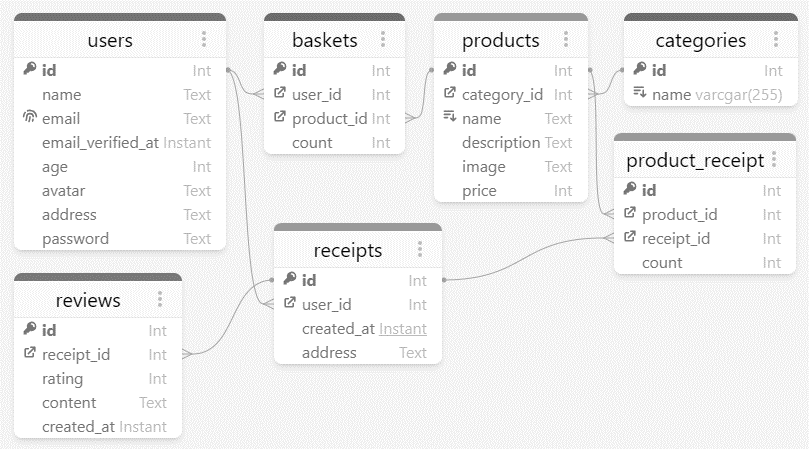


Рисунок 4 – Схема базы данных

Основные таблицы включают:

1. «reviews» – таблица для хранения отзывов:
   * «id (int, primary key)» – уникальный идентификатор отзыва
   * «foreignId('receipt\_id')->references('id')->on('receipts')» – Связывает таблицы «receipt» и «reviews»
   * «rating (int)» – Это поле будет использоваться для оценки (рейтинга) отзыва.
   * «string('comment')»: здесь пользователи смогут оставлять комментарий к отзыву.
   * «timestamp('created\_at')->useCurrent();»: Этот метод добавляет поле «created\_at» с типом данных «timestamp» (временная метка) в таблицу. Значение этого поля будет автоматически устанавливаться при создании записи.
2. «receipts» – таблица для хранения информации о чеках:
   * «id (int, primary key)» – уникальный идентификатор чека
   * «foreignId("user\_id")->references("id")->on("users");»: Этот метод добавляет внешний ключ «user\_id», который связан с полем «id» таблицы «users». Это позволяет связать записи в таблицах «receipts» и «users».
   * «timestamp("created\_at")->useCurrent();»: Этот метод добавляет поле «created\_at» с типом данных «timestamp» (временная метка) в таблицу. Значение этого поля будет автоматически устанавливаться при создании записи.
   * «string("address");»: Этот метод добавляет поле «address» с типом данных «string» (текстовое поле) в таблицу. Здесь будет храниться информация о адресе, связанная с чеком.
3. «products» – таблица для хранения информации о продуктах:
   * «id (int, primary key)» – уникальный идентификатор продукта
   * «string('name', 255)->index();»: Этот метод добавляет поле «name» с типом данных «string» (текстовое поле) в таблицу. Максимальная длина строки ограничена 255 символами. Также создается индекс для ускорения поиска по имени продукта.
   * «text('description');»: Этот метод добавляет поле «description» с типом данных «text» (текстовое поле) в таблицу. Здесь будет храниться описание продукта.
   * «unsignedBigInteger('price');»: Этот метод добавляет поле «price» с типом данных «unsignedBigInteger» (целое положительное число) в таблицу. Здесь будет храниться цена продукта.
   * «foreignId('category\_id')->references('id')->on('categories');»: Этот метод добавляет внешний ключ «category\_id», который связан с полем «id» таблицы «categories».
4. «categories» – таблица для хранения информации о категориях:
   * «id (int, primary key)» – уникальный идентификатор категории
   * «string('name', 255)->index();»: Этот метод добавляет поле «name» с типом данных string (текстовое поле) в таблицу.
   * «string('image')-> nullable();»: Этот метод добавляет поле «image» с типом данных string (текстовое поле) в таблицу. Здесь будет храниться путь к изображению категории.
5. «product\_receipt» – таблица для хранения информации о продуктах в чеке:
   * «id (int, primary key)» – уникальный идентификатор продукта в чеке
   * «foreignId('product\_id')->references('id')->on('products');»: Этот метод добавляет внешний ключ «product\_id», который связан с полем «id» таблицы «products».
   * «foreignId('receipt\_id')->references('id')->on('receipts');»: Аналогично, этот метод добавляет внешний ключ «receipt\_id», связанный с полем «id» таблицы «receipts».
   * «integer('count')->default(1);»: Этот метод добавляет поле «count» с типом данных integer (целое число) в таблицу. Здесь будет храниться количество продуктов в чеке. По умолчанию значение этого поля равно 1.
6. «baskets» – таблица для хранения информации о корзине пользователя:
   * «id (int, primary key)» – уникальный идентификатор продукта в корзине
   * «foreignId('user\_id')->references('id')->on('users');»: Этот метод добавляет внешний ключ «user\_id», который связан с полем «id» таблицы «users».
   * «foreignId('product\_id')->references('id')->on('products');»: Аналогично, этот метод добавляет внешний ключ «product\_id», связанный с полем «id» таблицы «products».
   * «integer('count')->default(1);»: Этот метод добавляет поле «count» с типом данных integer (целое число) в таблицу. Здесь будет храниться количество продукта в корзине. По умолчанию значение этого поля равно 1.

## 2.3. Проектирование пользовательского интерфейса

Для создания интуитивно понятного и удобного интерфейса будут разработаны макеты основных страниц приложения, таких как:

Главная страница: Выбор категорий продукта.

Категория: Добавление продукта в корзину с возможностью изменить количество нужного товара

Корзина: Список выбранных продуктов для заказа. В корзине пользователи смогут изменить количество товара, удалить ненужные позиции, оформление заказа после ввода адреса.

Профиль пользователя: Информация о пользователе и возможность редактирования. На странице профиля пользователи смогут изменить свои данные, такие как имя, адрес электронной почты, пароль, возрасти аватар. Также на странице присутствует история заказов с возможностью оценить определённый заказ.

## 2.4. Реализация веб-приложения и создание базовых миграций и моделей

Для начала разработки веб-приложения с использованием фреймворка Laravel необходимо выполнить несколько этапов по установке и настройке окружения.

Создание новый проект Laravel с помощью следующей команды:

Листинг 1 – Создание нового проекта

$ composer create-project --prefer-dist laravel/laravel

Эта команда создаст директорию и установит в неё все необходимые файлы и зависимости для Laravel.

Laravel использует файл «.env» для хранения конфигурационных параметров.

Миграции позволяют управлять схемой базы данных с помощью кода, что обеспечивает простоту и надежность при изменении структуры базы данных [9].

Для создания миграций, которые будут определять структуру таблиц корзины, отзывов, чеков, продуктов и категорий, выполните следующие команды:

Листинг 2 – Создание миграций

$ php artisan make:migration create\_baskets\_table

$ php artisan make:migration create\_reviews\_table

$ php artisan make:migration create\_receipts\_table

$ php artisan make:migration create\_products\_table

Продолжение листинга 2

$ php artisan make:migration create\_categories\_table

$ php artisan make:migration create\_product\_receipt\_table

Редактирование миграций

После создания миграций необходимо отредактировать файлы миграций, добавив в них столбцы, которые будут использоваться в таблицах.

Для создания таблиц в базе данных необходимо выполнить команду

Листинг 3 ­ Создание таблиц в базе данных

$ php artisan migrate

Данная команда инициирует процесс выполнения всех миграций, которые ещё не были применены к базе данных. В результате выполнения этой команды, в базе данных создаются структуры таблиц, соответствующие схемам, определенным в миграциях.

Для взаимодействия с таблицами базы данных в Laravel используются модели. Модели представляют собой классы, которые обеспечивают абстракцию над таблицами базы данных, позволяя осуществлять операции по созданию, чтению, обновлению и удалению данных (CRUD-операции). Для создания моделей для корзин, категорий, продуктов, чеков и отзывов необходимо выполнить следующие команды:

Листинг 4 – Создание моделей

$ php artisan make:model Basket

$ php artisan make:model Category

$ php artisan make:model Product

$ php artisan make:model Receipt

$ php artisan make:model Review

Каждая из этих команд создает файл модели в директории «app/Models». Эти файлы содержат базовый код, который можно расширить для реализации специфического поведения, связанного с соответствующей таблицей базы данных.

Разработка основных функциональных модулей. Реализация регистрации и аутентификации пользователей

Laravel предоставляет встроенные средства для реализации регистрации и аутентификации пользователей с помощью пакета «laravel/ui». Данный пакет включает в себя готовые компоненты для создания интерфейсов регистрации, входа в систему и сброса пароля. Чтобы использовать этот пакет, необходимо выполнить следующие шаги:

Листинг 5 – Установите пакет Orchid:

$ composer require orchid/platform

Создайте новую базу данных и обновите файл .env с учетными данными, а также добавьте URL-адрес вашего приложения в переменную «APP\_URL».

Создание пользователя:

Для создания пользователя с максимальными правами выполните команду, указав имя пользователя, электронный адрес и пароль:

Листинг 6 – подключение пользователя с админ-правами и запуск локального сервера:

$ php artisan orchid:admin admin admin@admin.com password

$ php artisan serve

Откройте браузер и перейдите по адресу http://localhost:8000/admin. Если все работает, вы увидите страницу входа в панель управления.

Перед началом работы убедитесь, что у вас установлены права на запись для каталогов «storage» и «bootstrap/cache».:

В Orchid маршруты и контроллеры аутентификации настраиваются в соответствии с фреймворком Laravel. Давайте рассмотрим, где и как это делается:

Маршруты в Orchid определяются в файлах маршрутов, таких как «routes/web.php» и «routes/platform.php».

В файле «routes/platform.php» вы можете зарегистрировать маршруты для административной панели.

Контроллеры в Orchid используются так же, как в обычном Laravel. Для аутентификации в Orchid используются стандартные методы Laravel.

Для реализации функционала создания и взаимодействия с заказами необходимо создать контроллеры, модели и представления.

Для управления маршрутами, связанными с заказом еды, необходимо добавить их в файл «routes/web.php». Используем ресурсные маршруты, которые автоматически создают маршруты для всех основных:

Эта запись создаёт набор маршрутов, соответствующих RESTful-методам для работы с ресурсом «pages».

В контроллере «ViewsController» необходимо реализовать методы для отображения страниц.

Рассмотрим каждый метод подробно.

Листинг 7 – Реализуем методы для отображения страниц

1 // app/Http/Controllers/ViewsController.php

2 namespace App\Http\Controllers;

3

4 use App\Models\Basket;

5 use App\Models\Category;

6 use Orchid\Attachment\Models\Attachment;

7

8 class ViewsController extends Controller

9 {

10 public function index()

11 {

12 $categories = Category::all()->map(function ($category) {

13 $image = Attachment::find($category->image);

14 $category->image = $image ? $image->url() : null;

15 return $category;

16 });

17

18 return view('index', compact('categories'));

19 }

20

21 public function register()

22 {

23 return view('pages.register');

24 }

25

26 public function login()

27 {

28 return view('pages.login');

29 }

30

31 public function profile()

32 {

33 return view('pages.profile');

34 }

35

36 public function basket()

37 {

38 $products = Basket::all();

39 return view('pages.basket', compact('products'));

40 }

41

42 public function category(Category $category)

43 {

44 return view('pages.category', compact('category'));

45 }

46 }

«index()»: метод для отображения списка всех категорий. Он извлекает все записи из таблицы «categories» и передаёт их в представление «index».

Основные шаги включают:

* Запрос к базе данных для извлечения всех категорий.
* Передача данных категорий в представление.
* Возвращение представления с переданными данными.

«register()»: метод для отображения формы регистрации для не авторизованных пользователей. Он возвращает представление «pages.register», где пользователи могут ввести свои данные для создания нового аккаунта.

Основные шаги включают:

* Проверка, что пользователь не авторизован.
* Возвращение представления формы регистрации.

«login()»: метод для отображения формы авторизации. Он возвращает представление «pages.login», где пользователи могут войти в систему.

Основные шаги включают:

* Проверка, что пользователь не авторизован.
* Возвращение представления формы авторизации.

«basket()»: метод для отображения списка всех товаров в корзине. Он извлекает все записи пользователя из таблицы «baskets» и возвращает представление «pages.basket». Основные шаги включают:

* Запрос к базе данных для получения всех товаров в корзине текущего пользователя.
* Передача данных корзины в представление.
* Возвращение представления с переданными данными.

«category()»: метод для отображения списка всех товаров в определённой категории. Он извлекает все данные, связанные с категорией, из таблицы «products» и возвращает представление «pages.category».

Основные шаги включают:

* Получение идентификатора категории из запроса.
* Запрос к базе данных для получения всех товаров, относящихся к данной категории.
* Передача данных о товарах в представление.
* Возвращение представления с переданными данными.

Создание представлений

Для полноценного функционирования веб-приложения, необходимо создать представления («views») с использованием Blade-шаблонов. Все шаблоны разместим в директории «resources/views/pages».

Листинг 8 – Создание контроллера для управления взаимодействиями с использованием Middleware [6]

$ php artisan make:controller ActionsController

$ php artisan make:controller CartController

Файл «routes/web.php» в Laravel используется для определения маршрутов («routes») веб-приложения. Эти маршруты управляют тем, какие контроллеры и методы будут обрабатывать запросы, поступающие на определённые «URL». Маршруты определяются с использованием различных методов, предоставляемых фасадом «Route», таких как «get», «post», «put», «delete» и «resource».

Листинг 9 – Добавим маршрутов файл «routes/web.php»:

1 Route::get('/', [ViewsController::class, 'index'])->name('index');

2 Route::get('/register', [ViewsController::class, 'register'])->name('register')->middleware('guest');

3 Route::get('/login', [ViewsController::class, 'login'])->name('login')->middleware('guest');

4 Route::post('/register', [ActionsController::class, 'register']);

5 Route::post('/login', [ActionsController::class, 'login']);

6 Route::get('/logout', [ActionsController::class, 'logout'])->name('logout')->middleware('auth');

7 Route::get('/profile', [ViewsController::class, 'profile'])->name('profile')->middleware('auth');

8 Route::get('/basket', [ViewsController::class, 'basket'])->name('basket')->middleware('auth');

9 Route::get('/category/{category}', [ViewsController::class, 'category'])->name('category.show');

10 Route::post('/cart/add/{product}', [CartController::class, 'add'])->name('cart.add')->middleware('auth');

11 Route::post('/cart/decrease/{product}', [CartController::class, 'decrease'])->name('cart.decrease')->middleware('auth');

12 Route::post('/cart/increase/{product}', [CartController::class, 'increase'])->name('cart.increase')->middleware('auth');

13 Route::post('/checkout', [CartController::class, 'checkout'])->name('cart.checkout')->middleware('auth');

14 Route::put('/profile', [ActionsController::class, 'profile\_update'])->name('profile.update')->middleware('auth');

15 Route::post('/receipts/{id}/review', [ActionsController::class, 'create\_review'])->name('receipts.review')->middleware('auth');

В контроллере «ActionsController» необходимо реализовать четыре основных метода:

Метод для регистрации пользователя: этот метод отвечает за создание нового пользователя. Когда пользователь заполняет форму регистрации и отправляет её, метод обрабатывает эти данные и создает запись нового пользователя в базе данных:

* Получение данных формы регистрации из запроса (например, имя, email, пароль).
* Валидация данных на соответствие необходимым требованиям (например, уникальность email, минимальная длина пароля).
* Хеширование пароля для безопасности.
* Создание новой записи пользователя в базе данных.
* Возврат ответа клиенту, подтверждающего успешную регистрацию или содержащего ошибки валидации.

Метод для авторизации пользователя: этот метод позволяет пользователю войти в систему, предоставив свои учетные данные (email и пароль):

* Получение данных формы авторизации из запроса.
* Поиск пользователя в базе данных по предоставленному email.
* Проверка соответствия хешированного пароля.
* Если авторизация успешна, создание сессии.
* Возврат ответа клиенту, подтверждающего успешный вход в систему или содержащего сообщение об ошибке.

Метод для обновления профиля пользователя: этот метод позволяет пользователю обновить свою личную информацию:

* Получение данных формы обновления профиля из запроса (например, имя, email, аватар).
* Валидация данных.
* Проверка авторизации пользователя для подтверждения, что он имеет право на выполнение обновления.
* Обновление записи пользователя в базе данных новыми данными.
* Возврат ответа клиенту, подтверждающего успешное обновление или содержащего ошибки валидации.

Метод для написания отзыва: этот метод позволяет пользователю оставить отзыв на товар или услугу:

* Получение данных формы отзыва из запроса (например, текст отзыва, рейтинг, идентификатор товара или услуги).
* Валидация данных [8].
* Проверка авторизации пользователя, чтобы убедиться, что он имеет право оставлять отзывы.
* Создание новой записи отзыва в базе данных.
* Возврат ответа клиенту, подтверждающего успешное создание отзыва или содержащего ошибки валидации.

Листинг 10 – Реализация методов для управления взаимодействиями

1 namespace App\Http\Controllers;

2

3 use App\Models\Basket;

4 use App\Models\Product;

5 use App\Models\User;

6 use App\Models\Blog\Post;

7 use Illuminate\Http\Request;

8 use Illuminate\Support\Facades\Auth;

9 use Illuminate\Support\Facades\Hash;

10 use Illuminate\Support\Facades\Storage;

11

12 class ActionsController extends Controller

13 {

14 public function register(Request $request)

15 {

16 // Валидация данных

17 ...

18 $user = User::create($request->input('user'));

19 Auth::login($user);

20 return redirect('/');

21 }

22

23 public function logout()

24 {

25 Auth::logout();

26 return redirect('/');

27 }

28

29 public function login(Request $request)

30 {

31 // Валидация данных

32 ...

33 if (Auth::attempt($request->input('user'))) {

34 return redirect('/');

35 } else {

36 return back()->withErrors([

37 'user.email' => 'Предоставленная почта или пароль не подходят'

38 ]);

39 }

40 }

41

Продолжение листинга 10

42 public function profile\_update(Request $request)

43 {

44 $user = Auth::user();

45 // Валидация данных

46 ...

47 $user->name = $request->name;

48 $user->email = $request->email;

49 $user->age = $request->age;

50 if ($request->password) {

51 $user->password = Hash::make($request->password);

52 }

53 if ($request->hasFile('avatar')) {

54 $path = $request->file('avatar')->store('avatars', 'public');

55 if ($user->avatar) {

56 Storage::disk('public')->delete($user->avatar);

57 }

58 $user->avatar = $path;

59 }

60 $user->save();

61 return redirect()->route('profile')->with('success', 'Профиль успешно обновлен');

62 }

63

64 public function create\_review(Request $request, $id)

65 {

66 // Валидация данных

67 ...

68 $receipt = Auth::user()->receipts()->findOrFail($id);

69 $receipt->review()->create($request->all());

70 return redirect()->route('profile')->with('success', 'Отзыв успешно добавлен');

71 }

72 }

«register (Request $request)»: Post метод для регистрации пользователя и переадресация пользователя на главную страницу.

«login (Request $request)»: Post метод для пользователя и переадресация пользователя на главную страницу.

«logout (Request $request)»: Метод для выхода из учетной записи.

«profile\_update (Request $request)»: Метод для обновления данных в профиле пользователя.

«create\_review (Request $request, $id)»: Метод для добавления отзыва к определённому заказу. Добавление записи в таблицу reviews и переадресация обратно в профиль.

В контроллере «CartController» необходимо реализовать три основных метода:

Метод для добавления товара в корзину: этот метод отвечает за добавление нового товара в корзину. Когда пользователь выбирает товар и указывает количество, данный метод обрабатывает запрос и добавляет товар в корзину. Если товар уже находится в корзине, метод увеличивает его количество:

* Получение идентификатора товара и количества из запроса.
* Поиск корзины пользователя (например, по идентификатору сессии или пользователя).
* Проверка, находится ли товар уже в корзине.
* Если товар уже есть, увеличение его количества.
* Если товара нет, добавление нового товара в корзину с указанным количеством.
* Сохранение изменений и возврат ответа клиенту.

Метод для увеличения или уменьшения количества товара в корзине: Этот метод позволяет пользователю изменять количество товара, который уже находится в корзине. Пользователь может либо увеличить, либо уменьшить количество товара:

* Получение идентификатора товара и нового количества из запроса.
* Поиск корзины пользователя.
* Проверка наличия товара в корзине.
* Если товар найден, обновление его количества на новое значение.
* Если новое количество равно нулю, удаление товара из корзины.
* Сохранение изменений и возврат ответа клиенту.

Метод для оформления заказа: Этот метод отвечает за создание заказа на основе текущего состояния корзины. Когда пользователь подтверждает заказ, метод обрабатывает все товары в корзине и создает соответствующий заказ:

* Получение корзины пользователя.
* Проверка наличия товаров в корзине.
* Создание нового заказа с информацией о пользователе и товарах.
* Очистка корзины после успешного создания заказа.
* Возврат подтверждения клиенту о создании заказа.



Рисунок 5 – Выбор иконки корзины [13]

Листинг 11 – Реализация методов для управления корзиной

1 namespace App\Http\Controllers;

2

3 use App\Models\Basket;

4 use App\Models\Product;

5 use Illuminate\Http\Request;

6 use App\Models\Receipt;

7

8 class CartController extends Controller

9 {

10 public function add(Request $request, Product $product)

11 {

12 $basket = Basket::where('user\_id', $request->user()->id)

13 ->where('product\_id', $product->id)

14 ->first();

15

16 if ($basket) {

17 $basket->count += 1;

18 $basket->save();

19 } else {

20 Basket::create([

21 'user\_id' => $request->user()->id,

22 'product\_id' => $product->id,

23 'count' => 1,

24 ]);

25 }

26

27 return redirect()->back()->with('success', 'Продукт добавлен в корзину');

28 }

29

30 public function decrease(Request $request, Product $product)

31 {

32 $basket = Basket::where('user\_id', $request->user()->id)

33 ->where('product\_id', $product->id)

34 ->first();

35

36 if ($basket) {

37 if ($basket->count > 1) {

38 $basket->count -= 1;

39 $basket->save();

40 } else {

41 $basket->delete();

42 }

43 }

44

45 return redirect()->back();

46 }

47

48 public function increase(Request $request, Product $product)

49 {

50 $basket = Basket::where('user\_id', $request->user()->id)

51 ->where('product\_id', $product->id)

52 ->first();

53

54 if ($basket) {

55 $basket->count += 1;

56 $basket->save();

57 }

58

59 return redirect()->back();

60 }

61

62 public function checkout(Request $request)

63 {

64 $basket = Basket::where('user\_id', $request->user()->id)->get();

Продолжение листинга 11

65

66 if ($basket->isEmpty()) {

67 return redirect()->back()->with('error', 'Ваша корзина пуста');

68 }

69

70 $receipt = new Receipt;

71 $receipt->user\_id = $request->user()->id;

72 $receipt->address = $request->address;

73 $receipt->save();

74

75 // Обновляем адрес пользователя

76 $user = $request->user();

77 $user->address = $request->address;

78 $user->save();

79

80 foreach ($basket as $item) {

81 $receipt->products()->attach($item->product\_id, ['count' => $item->count]);

82 $item->delete();

83 }

84

85 return redirect()->route('index')->with('success', 'Заказ оформлен');

86 }

87 }

«add(Request $request, Product $product)»: Добавляет продукт в корзину текущего пользователя. Если продукт уже в корзине, увеличивает его количество на единицу. Если нет, добавляет новый элемент в корзину.

«decrease(Request $request, Product $product)»: Уменьшает количество продукта в корзине текущего пользователя. Если количество больше одного, уменьшает на единицу. Если количество равно одному, удаляет продукт из корзины.

«increase(Request $request, Product $product)»: Увеличивает количество продукта в корзине текущего пользователя на единицу.

«checkout(Request $request)»: Оформляет заказ. Проверяет, что корзина не пуста, создает запись в таблице «receipt» с информацией о пользователе, обновляет адрес пользователя, переносит продукты из корзины в заказ и очищает корзину.

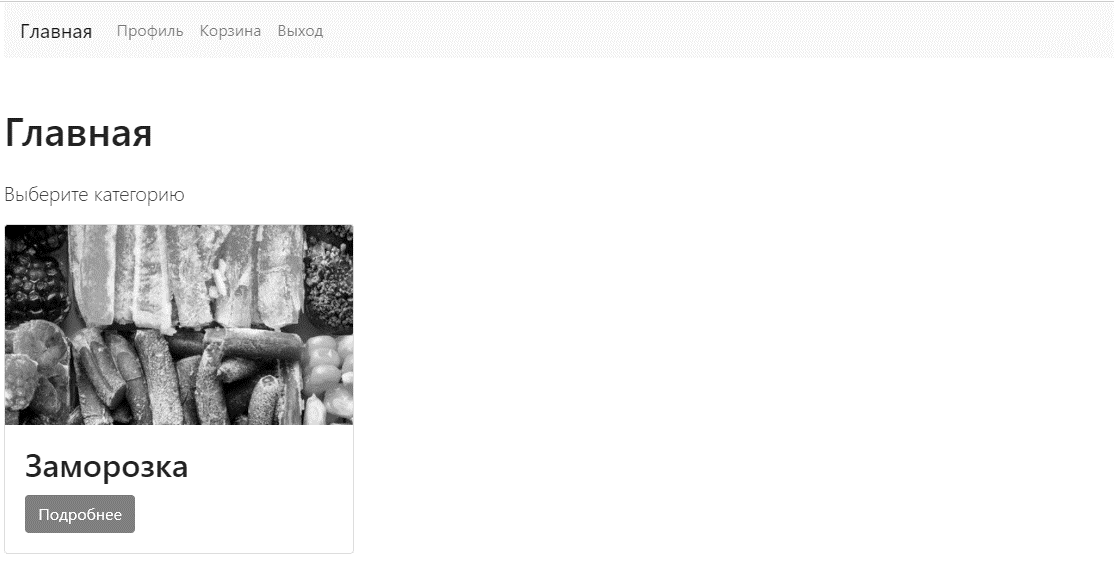


Рисунок 6 – Скриншот интерфейса

ЗАКЛЮЧЕНИЕ

В ходе выполнения данной курсовой работы была рассмотрена разработка веб-приложения для сервиса по доставке еды с использованием фреймворков Laravel и Orchid. В процессе работы были достигнуты следующие результаты:

Разработка веб-приложений для сервисов доставки еды является актуальной задачей в условиях современного мира, где значительное число людей предпочитает заказывать еду онлайн. Это обусловлено ростом числа пользователей интернета и мобильных устройств, а также стремлением людей экономить время и получать удобные и качественные услуги.

Целью работы было создание веб-приложения для сервиса по доставке еды с использованием современных технологий веб-разработки. В ходе работы были выполнены следующие задачи:

Анализ современных технологий веб-разработки.

Проектирование архитектуры приложения.

Реализация основных функциональных модулей приложения.

Оценка полученных результатов и определение перспектив развития проекта.

Проектирование и реализация веб-приложения:

На этапе проектирования были определены функциональные и нефункциональные требования к системе, а также разработана схема базы данных. Были созданы модели, контроллеры и представления для основных сущностей приложения, таких как пользователи, блюда и заказы. Особое внимание было уделено реализации регистрации и аутентификации пользователей, а также обеспечению безопасности данных.

Laravel был выбран в качестве основного фреймворка для разработки из-за его удобства, богатого функционала и широкого сообщества разработчиков. Фреймворк Orchid использовался для создания административной панели, что значительно упростило управление контентом и обработку заказов. Эти инструменты позволили создать устойчивое и масштабируемое приложение с интуитивно понятным интерфейсом.

Работа над курсовым проектом позволила мне углубить знания в области веб-разработки и познакомиться с современными инструментами и технологиями, такими как Laravel и Orchid. Я приобрел практический опыт проектирования и разработки полнофункционального веб-приложения, начиная с анализа требований и заканчивая реализацией функциональных модулей.

Особо стоит отметить удобство и мощность фреймворка Laravel, который предоставляет множество встроенных возможностей для быстрой и качественной разработки. Использование Orchid значительно упростило создание административной панели, что является важной частью любого коммерческого веб-приложения.

Полученные результаты соответствуют поставленным задачам и целям. Разработанное веб-приложение обладает всеми необходимыми функциями для комфортного использования как конечными пользователями, так и администраторами. Оно обеспечивает удобную регистрацию и аутентификацию, просмотр и заказ блюд, а также управление заказами через административную панель.

Возможные улучшения и новые функции для приложения. Интеграция с платежными системами: Добавление возможности оплаты заказов онлайн с использованием популярных платежных систем, таких как PayPal, Stripe или банковские карты.

Реализация мобильного приложения: Создание мобильного приложения для iOS и Android, что позволит пользователям заказывать еду еще удобнее.

Расширение функционала: Введение системы рейтингов и отзывов для блюд и ресторанов, а также реализация программы лояльности для постоянных клиентов.

Оптимизация производительности: Улучшение производительности приложения путем внедрения кэширования данных и оптимизации запросов к базе данных.

План дальнейшего развития проекта:

Постоянное обновление контента: Регулярное добавление новых блюд и акций, поддержание актуальности информации.

Мониторинг и аналитика: Внедрение системы аналитики для отслеживания пользовательской активности и анализа данных о продажах, что позволит принимать обоснованные бизнес-решения.

Обратная связь с пользователями: Создание механизма для сбора и обработки отзывов и предложений от пользователей, что поможет улучшить качество предоставляемых услуг.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Сидоров, П. А. Фреймворк Laravel: от новичка до профессионала. – М.: ДМК Пресс, 2021.
2. [Электронный ресурс] Installation – Официальный сайт фреймворка Laravel. URL: https://laravel.com/docs/11.x/installation (дата обращения: 06.06.2024)
3. [Электронный ресурс] Routing – Официальный сайт фреймворка Laravel. URL: https://laravel.com/docs/11.x/routing (дата обращения: 06.06.2024)
4. [Электронный ресурс] Models – Официальный сайт фреймворка Laravel. URL: https://laravel.com/docs/7.x/eloquent (дата обращения: 06.06.2024)
5. [Электронный ресурс] Views – Официальный сайт фреймворка Laravel. URL: https://laravel.com/docs/11.x/views (дата обращения: 06.06.2024)
6. [Электронный ресурс] Middleware – Официальный сайт фреймворка Laravel. URL: https://laravel.com/docs/7.x/middleware (дата обращения: 06.06.2024)
7. [Электронный ресурс] Blade шаблоны – Официальный сайт фреймворка Laravel. URL: https://laravel.com/docs/11.x/blade (дата обращения: 06.06.2024)
8. [Электронный ресурс] Validation – Официальный сайт фреймворка Laravel. URL: https://laravel.com/docs/11.x/validation (дата обращения: 06.06.2024)
9. [Электронный ресурс] Migrations – Официальный сайт фреймворка Laravel. URL: https://laravel.com/docs/11.x/migrations (дата обращения: 06.06.2024)
10. [Электронный ресурс] CSRF – Официальный сайт фреймворка Laravel. URL: https://laravel.com/docs/11.x/csrf (дата обращения: 06.06.2024)
11. [Электронный ресурс] Controllers – Официальный сайт фреймворка Laravel. URL: https://laravel.com/docs/11.x/controllers (дата обращения: 06.06.2024)
12. [Электронный ресурс] Официальный сайт фреймворка Django. URL: https://www.djangoproject.com/ (дата обращения: 06.06.2024)
13. [Электронный ресурс] Fontawesome – набор инструментов для шрифтов и значков. URL: https://fontawesome.com/ (дата обращения: 06.06.2024)
14. [Электронный ресурс] VueJS – Прогрессивный JavaScript-фреймворк. URL: https://vuejs.org/ (дата обращения: 06.06.2024)
15. [Электронный ресурс] Официальная документация пакета Orchid Platform для фреймворка Laravel. URL: https://orchid.software/ru/docs (дата обращения: 06.06.2024)