Частное учреждение

профессиональная образовательная организация

ТЕХНИКУМ «БИЗНЕС И ПРАВО»

ЗАЩИЩЕНО С ОЦЕНКОЙ

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(подпись и ФИО руководителя)

«\_\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 20\_\_\_\_г.

КУРСОВАЯ РАБОТА

|  |  |
| --- | --- |
| Тема | РАЗРАБОТКА ИНФОРМАЦИОННОЙ СИСТЕМЫ ДЛЯ ЦВЕТОЧНОГО МАГАЗИНА |
| По МДК 09.01 | Проектирование и разработка веб-приложений |
| Выполнил | Сурин А. Ю. |
| Курс, группа | 4 курс, группа ИС-4 |
| Специальность | 09.02.07 Информационные системы и программирование |
| Научный руководитель | Мартыненко В.А. |

г. Белореченск

2023 г.

СОДЕРЖАНИЕ

|  |  |
| --- | --- |
| Введение | 3 |
| 1. ОБЩИЕ СВЕДЕНИЯ | 5 |
| 1.1 Анализ предметной области | 5 |
| 1.2 Цели и задачи выполняемой системы | 5 |
| 1.3 Метод реализации процесса проектирования | 6 |
| 1.4 Требования к системе | 7 |
| 1.5 Обоснование проектных решений | 9 |
| 2. ПРОЕКТНАЯ ЧАСТЬ | 25 |
| 2.1 Техническое задание | 25 |
| 2.2 Установка и настройка среды разработки | 27 |
| 2.3 Руководства пользователя | 29 |
| 2.4 Руководство администратора | 31 |
| ЗАКЛЮЧЕНИЕ | 34 |
| СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ | 36 |
| ПРИЛОЖЕНИЯ |  |

ВВЕДЕНИЕ

Актуальность исследования – в современном электронном бизнесе конкуренция становится все более ожесточенной, особенно в сегменте онлайн-торговли цветами. Развитие информационной системы для интернет-магазина цветов является ключевым стратегическим моментом для успешного преодоления вызовов насыщенного рынка. Способность предоставлять широкий ассортимент качественных цветочных композиций, облегчать процесс заказа и обеспечивать персонализированный опыт для каждого клиента становится фундаментальным в привлечении и удержании клиентской базы.

В сфере электронной коммерции цветов создание персонализированного опыта играет важную роль в удержании клиентов. Использование данных о предыдущих покупках и истории посещений позволяет создавать индивидуальные предложения, рекомендации товаров и предоставлять специальные акции, что способствует повышению удовлетворенности клиентов и повторным покупкам.

Объект исследования – информационная система интернет-магазина цветов.

Предмет исследования – предметом исследования являются автоматизация процессов размещения, редактирования, удаления и поиска товаров для клиента, а также возможности авторизации и регистрации для просмотра истории заказов, формирования раздела избранных товаров, корзины товаров и оставления отзывов.

Цель исследования – рассмотреть и установить повышение эффективности процессов поиска, добавления, редактирования, удаления, товаров компании за счет разработки и внедрения информационной системы интернет-магазина, а также обеспечение удобства авторизации и регистрации, для дальнейшего пользования сайтом клиента.

Задачи исследования:

* изучить особенности функционирования интернет-магазина цветов;
* рассмотреть ряд существующих разработок для решения поставленной цели;
* выбрать метод реализации работы автоматизированных процессов информационной системы;
* разработать саму информационную систему интернет-магазина;
* разработать информационную систему;
* протестировать разработанную информационную систему интернет-магазина;

Методы исследования – разработка интернет-магазина включает в себя изучение функционала отзывов, категорий товаров, товаров, личных кабинетов, и обработки заказов. Методы исследования включают в себя также анализ различных конкурентов платформ интернет-магазинов отзывов, размещения заказов, личных кабинетов, систем обработки заказов и управления товарами в корзине и избранном для оптимизации удобного функциональности интернет-магазина.

1. ОБЩИЕ СВЕДЕНИЯ

1.1 Анализ предметной области

Предлагаемая система ориентирована на разработку интернет-магазина цветов, предоставляющего удобный и интуитивно понятный интерфейс для администраторов и модераторов. Главная цель - обеспечить возможность создания категорий и управления товарами внутри этих категорий. Администраторы будут иметь возможность управлять содержимым магазина, создавать категории, добавлять и редактировать информацию о продуктах, а также управлять разделом отзывов.

Система будет спроектирована с упором на простоту использования и надежность, обеспечивая администраторам удобное и эффективное управление контентом магазина. Целью является создание интуитивно понятного интерфейса, который позволит осуществлять управление категориями, товарами и отзывами без лишних трудностей. Такой подход позволит создать надежную платформу для продажи цветов и обеспечит комфортное взаимодействие с информацией, что способствует эффективной работе администраторов интернет-магазина.

1.2 Цели и задачи выполняемой системы

Основными целями разрабатываемой системы интернет-магазина является:

* Управление категориями: разработка удобного интерфейса для администраторов, обеспечивающего легкое управление категориями и возможность быстрого добавления, редактирования и удаления объектов;
* Обновление ассортимента: предоставление администраторам возможности оперативно удалять и обновлять товары в категориях, обеспечивая постоянное обновление и разнообразие предложений;
* Поиск товаров: реализация удобного поиска, обеспечивающего эффективную навигацию по сайту и быстрое нахождение нужных товаров для пользователей и отдельно для администраторов;
* Интерфейс для пользователей: предоставление пользователю удобного и привлекательного интерфейса для взаимодействия с разделами магазина;

Задачами информационной системы являются:

* Разработка удобного интерфейса для администраторов, позволяющего легко работать с товарами, их категориями, пользователями и отзывами;
* Предоставление администратором возможности легко и эффективно управлять пользователями, обеспечивая возможность в добавлении, редактировании и удалении пользователе, а также специальных возможностей таких как сброс пароля;
* Обеспечение стабильной и бесперебойной работы интернет-магазина для обеспечения комфортного пользования всеми пользователями. Важно обеспечить страницы различными проверками от неподключенной базы данных или сервера;

1.3 Методы реализации процесса проектирования

В ходе проведения работ по разработке интернет-магазина цветов автоматизируются процессы работы с товарами, категориями, пользователями и отзывами, будет выполняться проверка на корректность данных, с возможностью поиска, редактирования и удаления.

При внедрении системы будет достигнуто:

* добавление товара;
* редактирование товара;
* удаление товара;
* поиск товаров;
* добавление категорий;
* редактирование категорий;
* удаление категорий;
* добавление отзывов;
* управление не одобренными отзывами;
* управление одобренными отзывами;
* управление отображение отзывов на главной странице;
* создание пользователей;
* редактирование пользователь;
* удаление пользователей;
* поиск пользователей;

Объекты автоматизации характеризуются:

* обеспечивать максимальную работоспособность системы;
* быть надежной и технологичной;
* обслуживание должно проводиться минимальным количеством персонала;

1.4 Требования к системе

Система должная предполагать наличие следующих модулей, представленных на рисунке 1:

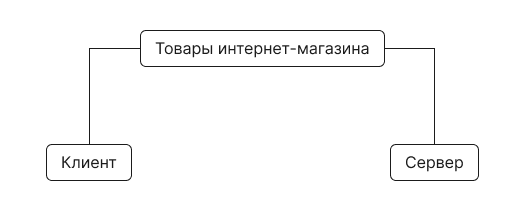


Рисунок 1 – Модули, которые должны быть в программе

Описание модулей разрабатываемой системы:

* app – содержит основные компоненты приложения, включая контроллеры, модели, и другие классы, отвечающие за бизнес-логику приложения;
* bootstrap: здесь находятся файлы для инициализации приложения, включая загрузку настроек и автозагрузку классов;
* config: содержит файлы конфигурации приложения, включая настройки базы данных, сервисы и другие параметры;
* database: место для миграций базы данных, сидов (seeders) и моделей данных;
* node\_modules, vendor: дополнительные зависимости для работы;
* public: публичная часть приложения, здесь хранятся статические файлы, доступные извне (изображения, CSS, JavaScript);
* resources: содержит шаблоны представлений, нескомпилированные ресурсы (например, SCSS, JavaScript), миграции и другие ресурсы для разработки;
* routes: здесь находятся файлы маршрутизации, определяющие URL-адреса и соответствующие им действия;
* storage: этот каталог содержит сгенерированные файлы, такие как журналы, временные файлы и загруженные пользовательские файлы;
* tests: место для тестовых скриптов и файлов, используемых для автоматизированного тестирования приложения;

Модули разрабатываемой системы, представленные на рисунке 2:

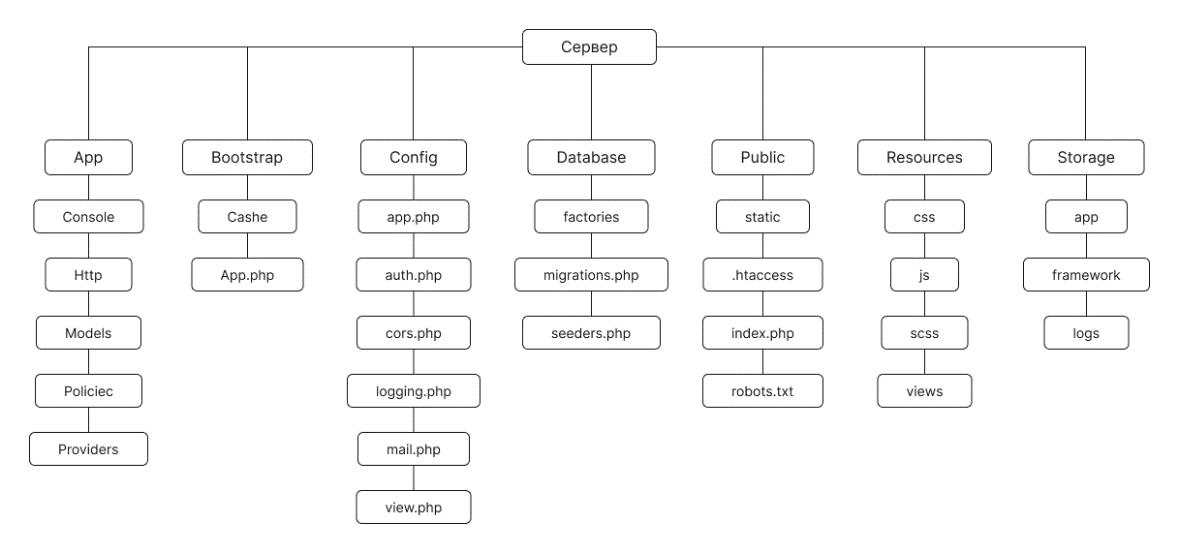


Рисунок 2 – Модули разрабатываемой системы

1.5 Обоснование проектных решений

На данном этапе необходимо описать примерную архитектуру разрабатываемой системы. Для того, чтобы сделать систему интернет-магазина цветов, мы будем использовать клиент-серверную архитектуру. В клиент-серверной архитектуре имеется три звена:

1. Представление данных — на стороне клиента;
2. Прикладной компонент — на выделенном сервере приложения, здесь происходит вся бизнес-логика, которая в дальнейшем будет взаимодействовать с клиентом;
3. Управление ресурсами — сервер БД (базу данных), который и представляет запрашиваемые данные;

Для данного проекта выбрана эта архитектура, так как у нее есть ряд преимуществ перед другими архитектурами:

1. Высокую степень гибкости и масштабируемости;
2. Высокую безопасность;
3. Высокую производительность (т. к. задачи распределены между Серверами);

MySQL — это мощная система управления базами данных, которая помогает хранить и организовывать информацию о нашем цветочном магазине. Наша база данных содержит информацию о цветах, букетах, клиентах и заказах. От списка красивых букетов для различных событий до данных о клиентах все это хранится и обрабатывается в нашей базе данных. MySQL обеспечивает эффективное хранение и доступ к информации, помогая нам управлять нашим магазином более эффективно и организованно.

На рисунке 3 представлена структура клиент–серверной архитектуры:

Изображение выглядит как текст, диаграмма, снимок экрана, Шрифт

Автоматически созданное описание

Рисунок 3 – Клиент–серверная архитектура

Требования к системе разработаны в соответствии со стандартом качества программного обеспечения ISO 9126:2001, описывающим многоуровневую модель характеристик качества и соответствующий им набор атрибутов.

Требования к интернет-магазину цветов представлены в таблице 1:

Таблица 1 – Требования к системе

|  |  |  |  |
| --- | --- | --- | --- |
| Атрибуты | Метки | | Требования |
| Удобство | | | |
| Простота использования | Среднее время, необходимое  пользователю, чтобы найти  необходимый элемент в программе | | Поиск отдельного элемента интерфейса не должен занимать более 1 секунды |
| Привлекательность | Соответствие интерфейса  требованиям | | 99% элементов должны быть работоспособны для  пользователя |
| Обучаемость | Показатель, затрачиваемый  пользователями на обучение  работе | | Пользователь должен изучить работу ПО за первые 3-5 минут  использования |
| Производительность | | | |
| Временная эффективность | Время выполнения компонента программы | - Приложение должно  обслуживать любую функцию не дольше 1 сек  (не включая задержки в сети); | |
| Эффективность использования ресурсов | Объемы ресурсов требуемых для выполнения задач | - Постоянная память,  используемая программой – не более 10МБ;  - Приложение должно потреблять не более 3МБ  памяти на каждый неактивный сеанс с  пользователем; | |
| Удобство установки | Легкость установки | Необходима установка:  XAMPP, MySQL, PhpMyAdmin | |

Продолжение таблицы 1

|  |  |  |
| --- | --- | --- |
| 1 | 2 | 3 |
| Переносимость | | |
| Способность к сосуществованию | Способность ПО сосуществовать с другими  программами в общем окружении | Программа не должна  занимать большие объемы памяти устройства (максимум 3МБ) |
| Удобство замены другого ПО данным | Совместимость структур данных | - Система должна быть  разработана по общим  принципам и не требовать навыков от пользователя;  - Должна поддерживать  внедрение новых версий;  - Система должна быть применима вместо других  программных систем для решения тех же задач в определенном окружении; |

Требования к численности и квалификации персонала системы и режиму его работы:

1. В системе существуют 2 группы пользователей: пользователи и администраторы. Пользователи могут просматривать категории товаров и товары на страницах сайта, а также имеют возможность оставлять отзыв будучи авторизованным.
2. Администратор выполняет следующие функции: просмотра, занесение данных, редактирование, удаление данных, и поиск. Данный функционал относится ко всем сущностям, которые будут существовать в проекте.
3. Пользователи системы должны обладать базовыми навыками взаимодействия с компьютером и использования веб-ресурсов. Опыт работы с персональными компьютерами или устройствами, доступ к интернету и уверенное понимание основных принципов работы веб-интерфейсов будет полезным для комфортного использования системы.

В таблице 2 представлены требования к надежности.

Таблица 2 – Требования к надёжности

|  |  |  |
| --- | --- | --- |
| Атрибуты | Метрики | Требования |
| Надёжность | | |

Продолжение таблицы 2

|  |  |  |
| --- | --- | --- |
| Устойчивость к отказу | Коэффициент аварийных  отказов | - Не менее 90% ошибок  в программе должны  обрабатываться без  экстренного завершения;  - Обеспечение бесперебойного питание  активного сетевого  оборудования; |
| Способность к восстановлению | Отношение количества числа успешного восстановление работы системы к ее аварийному завершению за определённый промежуток времени | - Восстановление  работоспособности  системы не должно превышать 5 минут; |
| Уровень зрелости | Среднее время работы без сбоев | - Система должна  работать 24 часа в сутки;  - Система должна оповещать пользователя при некорректных действиях в системе;  - Данные, которые вводит пользователь должны сохраняться; |

Средняя доступность приложения должна составлять не менее 99%

Среднее время между сбоями — это среднее время, за которое компонент или модуль может выполнять свои функции без перерыва.

Измеряется от начала работы до момента следующего сбоя.

Среднее время работы без сбоев должно составлять не менее 1500 часов.

В таблице 3 представлен перечень и критерии отказов для каждой функции, по которой задаются требования по надежности.

Таблица 3 – Перечень и критерии отказов для каждой функции

|  |  |
| --- | --- |
| Функции | Критерии отказа |
| Отображение данных | Отсутствие соединения с БД |
| Взаимодействие с товарами | Отсутствие соединения с БД |
| Поиск | Отсутствие соединения с БД |
| Добавление | Отсутствие соединения с БД |
| Редактирование | Отсутствие соединения с БД |
| Удаление | Отсутствие соединения с БД |
| Авторизация и регистрация | Отсутствие соединения с БД |

Требования к функциям (ВИ), выполняемым приложением описывается с помощью CASE средств:

Сценарий:

1. Система отображает форму для входа в систему.
2. Пользователь вводит необходимые данные для входа в систему.
3. Пользователь нажимает на кнопку входа «Войти».
4. Система сверяет данные, введённые пользователем с данными в БД.

Расширение: в случае некорректности введенных данных или отсутствия соединения с сервером при нажатии на кнопку «Войти», выведется сообщение с описанием ошибки.

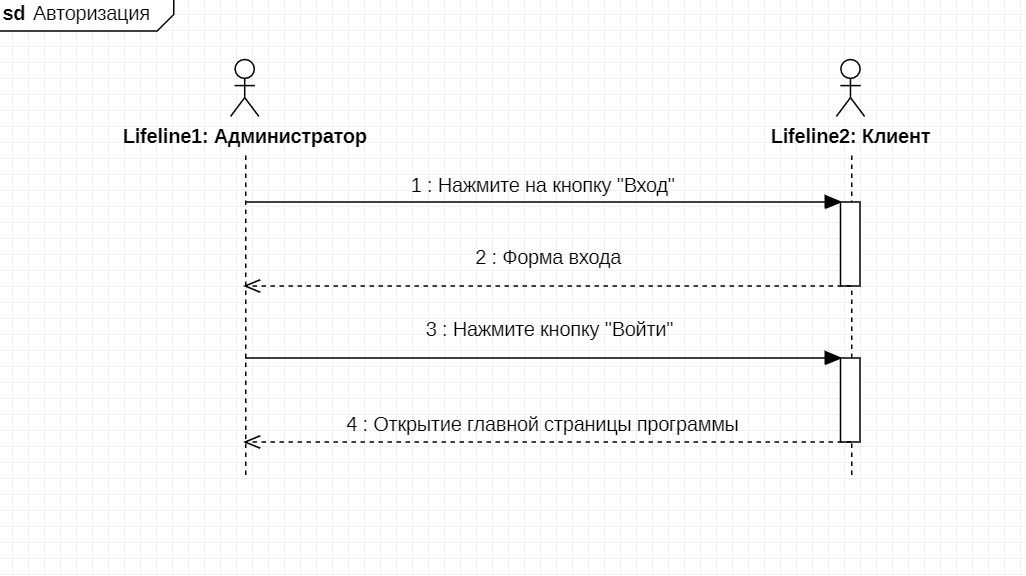


Рисунок 4 - Диаграмма последовательности «Вход пользователя»

Имя: Удаление данных.

Описание: администратор может просмотреть списки пользователей, категорий товаров, товаров магазина и отзывы, и при неактуальности данных может удалить их с DB.

Сценарий:

1. Администратор выбирает необходимые данные.
2. Администратор нажимает кнопку для удаления данных.
3. Сервер соединяется с DB.
4. Проверяется соединение с DB.
5. Сервер получает сообщение, что соединение установлено.
6. Сервер отправляет запрос на удаление данных.
7. DB передает серверу обновленный список данных.
8. Клиент получает ответ от сервера.
9. Администратор видит таблицу в приложении с обновленными данными.

Расширения: при отсутствии соединения с сервером, клиент выдаст ошибку администратору.

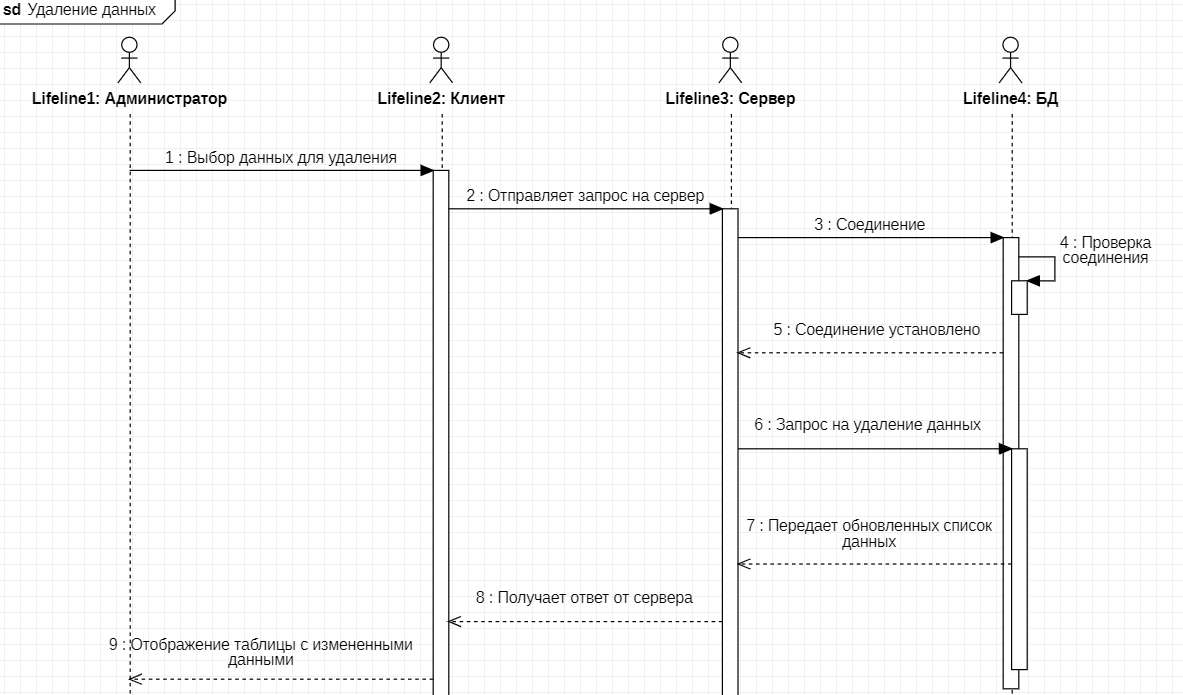


Рисунок 5 - Диаграмма последовательности «Удаление данных»

Имя: Добавление данных.

Описание: администратор может просмотреть списки пользователей, категорий товаров и товаров магазина, и имеет возможность добавить новые записи в DB.

Сценарий:

1. Администратор нажимает кнопку «Добавить»
2. Клиент отправляет запрос на сервер.
3. Сервер соединяется с DB.
4. Проверяется соединение с DB.
5. Сервер получает сообщение, что соединение установлено.
6. Сервер отправляет клиенту ответ.
7. Клиент передает администратору форму для заполнения данных.
8. Администратор заполняет форму.
9. Клиент отправляет запрос на сервер.
10. Сервер передает данные в БД.
11. БД передает сообщение серверу, что данные успешно сохранены.
12. Сервер отправляет ответ клиенту.
13. Добавляется запись в интерфейс приложения.

Расширения: при отсутствии соединения с сервером, клиент выдаст ошибку. Также при отправке данных с некорректностью данных сервер выдаст ошибка, и передаст ее клиенту чтоб ее вывести администратору.

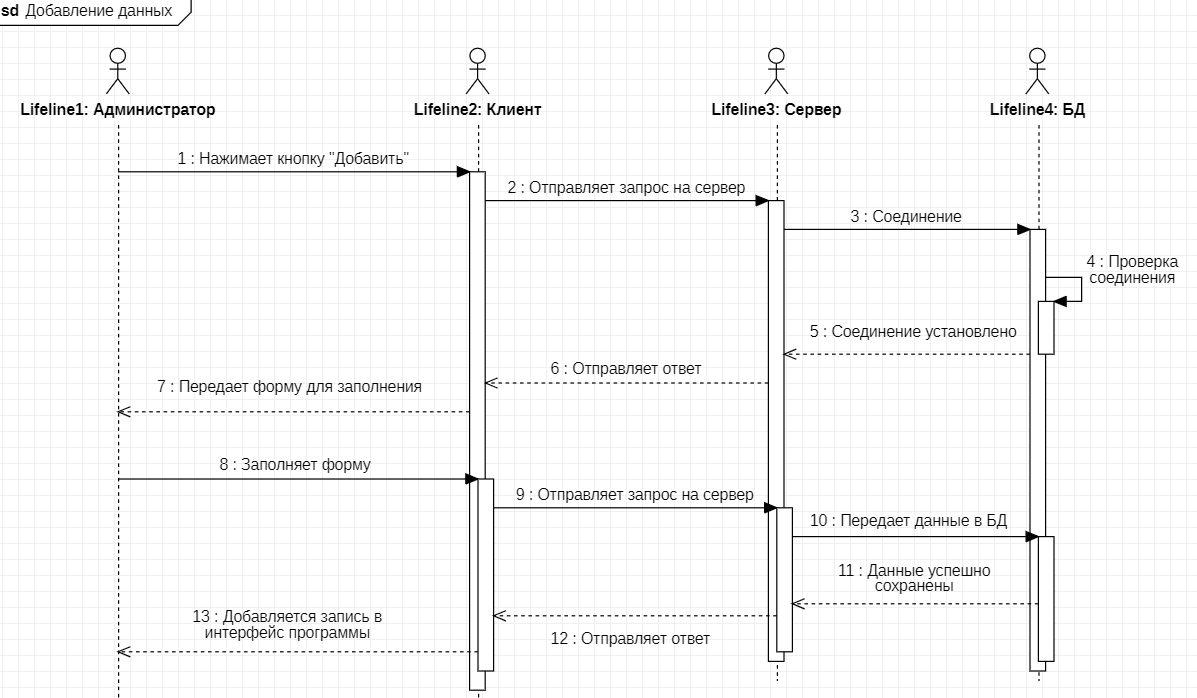


Рисунок 6 - Диаграмма последовательности «Добавление данных»

Имя: Редактирование данных.

Описание: администратор может просмотреть списки фирм, вакансий, специалистов и заявок, и имеет возможность изменить записи в DB.

Сценарий:

1. Администратор нажимает кнопку «Редактировать»
2. Клиент отправляет запрос на сервер.
3. Сервер соединяется с DB.
4. Проверяется соединение с DB.
5. Сервер получает сообщение, что соединение установлено.
6. Сервер отправляет клиенту ответ.
7. Клиент передает администратору форму для заполнения данных.
8. Администратор заполняет форму.
9. Клиент отправляет запрос на сервер.
10. Сервер передает данные в БД.
11. БД передает серверу сообщение, что данные успешно сохранены.
12. Сервер отправляет ответ клиенту.
13. Обновляется запись в интерфейсе приложения, которая редактировалась.

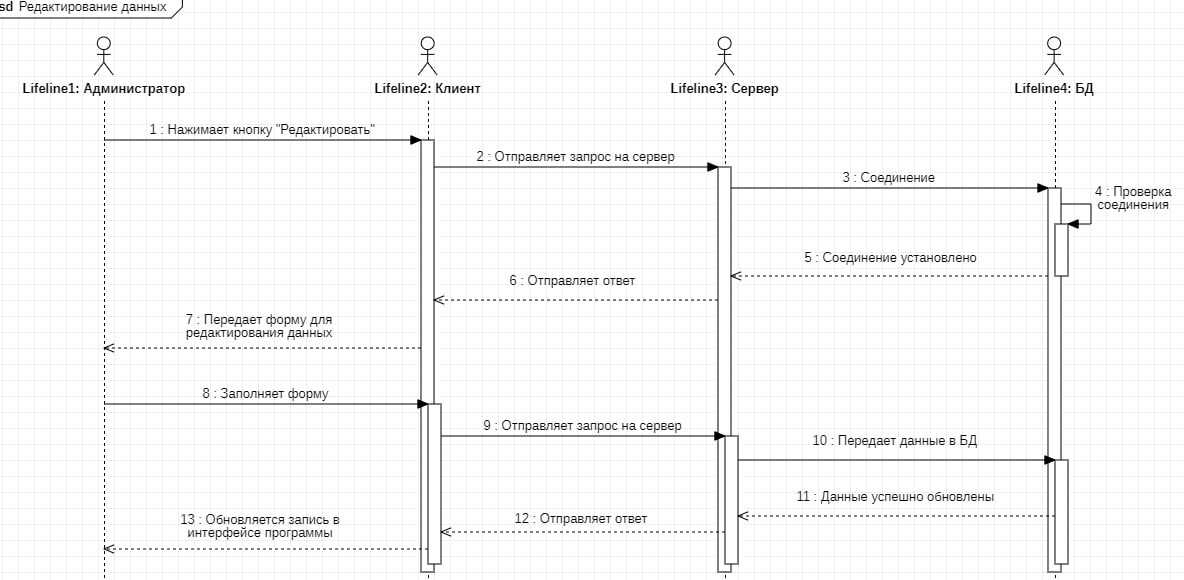


Рисунок 7 - Диаграмма последовательности «Редактирование данных»

Имя: Поиск данных.

Описание: для удобства пользования администратор может осуществить поиск по DB, с помощью ключевых слов или признаков.

Сценарий:

1. Администратор пишет в строку поиска данные, ожидая ответа от сервера.
2. Клиент отправляет запрос на сервер.
3. Сервер соединяется с DB.
4. Проверяется соединение с DB.
5. Сервер получает сообщение, что соединение установлено.
6. Сервер отправляет DB запрос на поиск данных.
7. DB отправляет серверу список данных.
8. Сервер отправляет клиенту список данных.
9. Клиент отображает результат поиска в виде списка данных.

Расширения: при отсутствии соединения с сервером, клиент выдаст ошибку администратору.

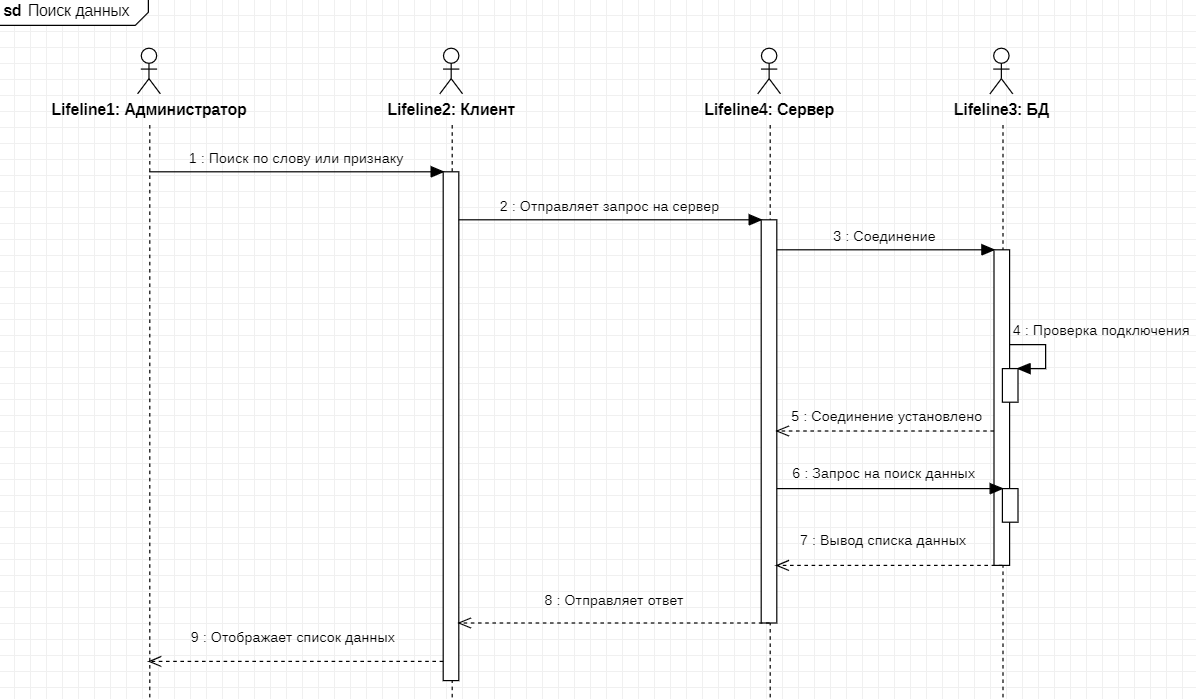


Рисунок 8 - Диаграмма последовательности “Поиск данных”

Для нормального функционирования системы веб-приложения на базе PHP Laravel требуется обеспечить соответствующую инфраструктуру:

* наличие сервера для хостинга приложения;
* персональные устройства специалистов для работы с приложением;
* надежное интернет-соединение для доступа к приложению;
* программа XAMPP для локальной разработки и тестирования приложения;
* обязательное создание резервных копий информации для ее сохранности и возможности восстановления при необходимости;
* требование к наличию сервера базы данных MySQL для хранения данных приложения;

При анализе предметной области разрабатываемой системы были выделены следующие сущности, на основе которых будет разрабатываться база данных системы:

* Пользователи;
* Категории товаров;
* Товар;
* Изображения товара;
* Описания товара;
* Отзывы;

База данных информационной системы должна состоять из следующих связанных таблиц:

1. Users - содержащая в себе информацию о пользователях системы;
2. Categories - содержащая в себе информацию о категории товара;
3. Products - содержащая в себе информацию о товарах магазина;
4. ProductImages - содержащая в себе информацию о привязанных изображениях к определенному товару;
5. ProductCompositions - содержащая в себе информацию о привязанных элементов состава товара к определенному товару;
6. Reviews – содержит в себе информацию об отзывах пользователей;

Данные о пользователях представлены в таблице Users:

Таблица 4 – Данные о пользователях

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Имя поля | Тип | Размер | Комментарий | Ограничение |
| id | BIGINT | 20 | Идентификатор пользователя | Первичный ключ |
| name | VARCHAR | 255 | Имя пользователя | NOT NULL |
| email | VARCHAR | 255 | Почта пользователя | NOT NULL |
| password | VARCHAR | 255 | Хешированный пароль пользователя | NOT NULL |
| role | ENUM | 255 | Роль пользователя | NOT NULL |

Данные о категориях должны размещаться в таблице Categories

Таблица 5 – Данные о категориях товаров

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Имя поля | Тип | Размер | Комментарий | Ограничение |
| id | BIGINT | 20 | Идентификатор категории | Первичный ключ |
| name | VARCHAR | 255 | Имя категории | NOT NULL |
| subtitle | VARCHAR | 255 | Подзаголовок категории | NOT NULL |
| order\_index | INT | 20 | Индекс и порядок категории | NOT NULL |

Данные о товарах должны размещаться в таблице Products

Таблица 5 – Данные товарах

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Имя поля | Тип | Размер | Комментарий | Ограничение |
| id | BIGINT | 20 | Идентификатор  товара | Первичный ключ |
| name | VARCHAR | 255 | Название товара | NOT NULL |
| category\_id | BIGINT | 20 | Идентификатор категории | NOT NULL |
| article | INT | 20 | Уникальный артикул товара | NOT NULL |
| price | INT | 20 | Цена товара | NOT NULL |

Данные о изображениях товара должны размещаться в таблице ProductImages

Таблица 6 – Данные о изображениях товара

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Имя поля | Тип | Размер | Комментарий | Ограничение |
| id | BIGINT | 20 | Идентификатор  файла | Первичный ключ |
| name | VARCHAR | 255 | Уникальное имя файла | NOT NULL |
| path | BIGINT | 20 | Путь к файлу | NOT NULL |
| product\_id | BIGINT | 20 | Идентификатор продукта | NOT NULL |

Данные о элементах состава товара должны размещаться в таблице ProductCompositions

Таблица 7 – Данные о элементах состава товара

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Имя поля | Тип | Размер | Комментарий | Ограничение |
| id | BIGINT | 20 | Идентификатор элемента состава | Первичный ключ |

Продолжение таблицы 7

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| name | VARCHAR | 255 | Имя элемента состава | NOT NULL |
| quantity | INT | 20 | Количество элемента состава | NOT NULL |
| product\_id | BIGINT | 20 | Идентификатор продукта | NOT NULL |

Данные об отзывах магазина должны располагаться в таблице Reviews

Таблица 7 – Данные о элементах состава товара

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Имя поля | Тип | Размер | Комментарий | Ограничение |
| id | BIGINT | 20 | Идентификатор  отзыва | Первичный ключ |
| name | VARCHAR | 255 | Имя пользователя | NOT NULL |
| email | VARCHAR | 255 | Почта пользователя | NOT NULL |
| rating | INT | 20 | Рейтинг отзыва | NOT NULL |
| comment | TEXT | - | Текст отзыва | NOT NULL |
| user\_id | BIGINT | 20 | Идентификатор пользователя | NOT NULL |
| status | BOOLEAN | - | Статус отзыва | NOT NULL |
| display\_on\_homepage | BOOLEAN | - | Отображение на главной странице | NOT NULL |

На рисунке 9 представлен интерфейс добавления товара в БД.

В данном окне указывается название товара, его стоимость, выбор категории (к какой категории относится товар), а также фотографии и состав товара.

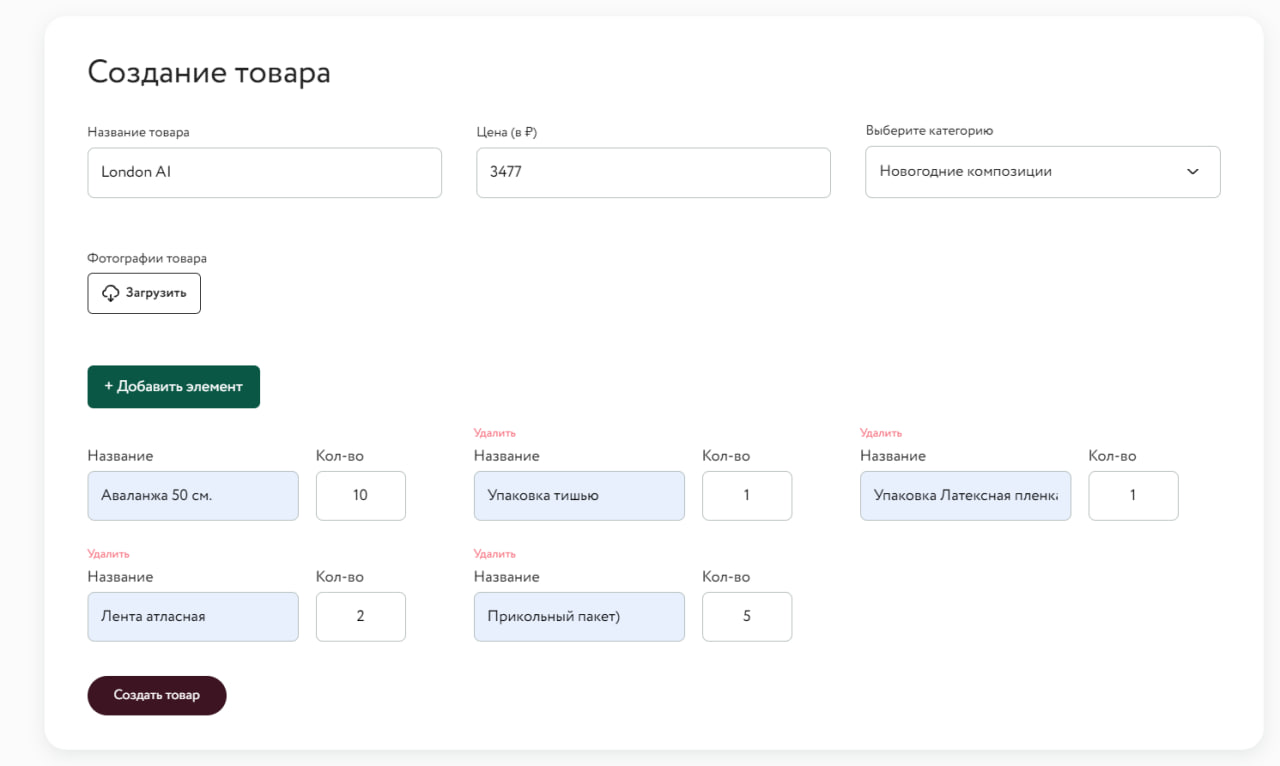


Рисунок 9 – Интерфейс добавления товара в БД

На рисунке 10 представлен интерфейс просмотра товаров определенной категории в панели администратора.

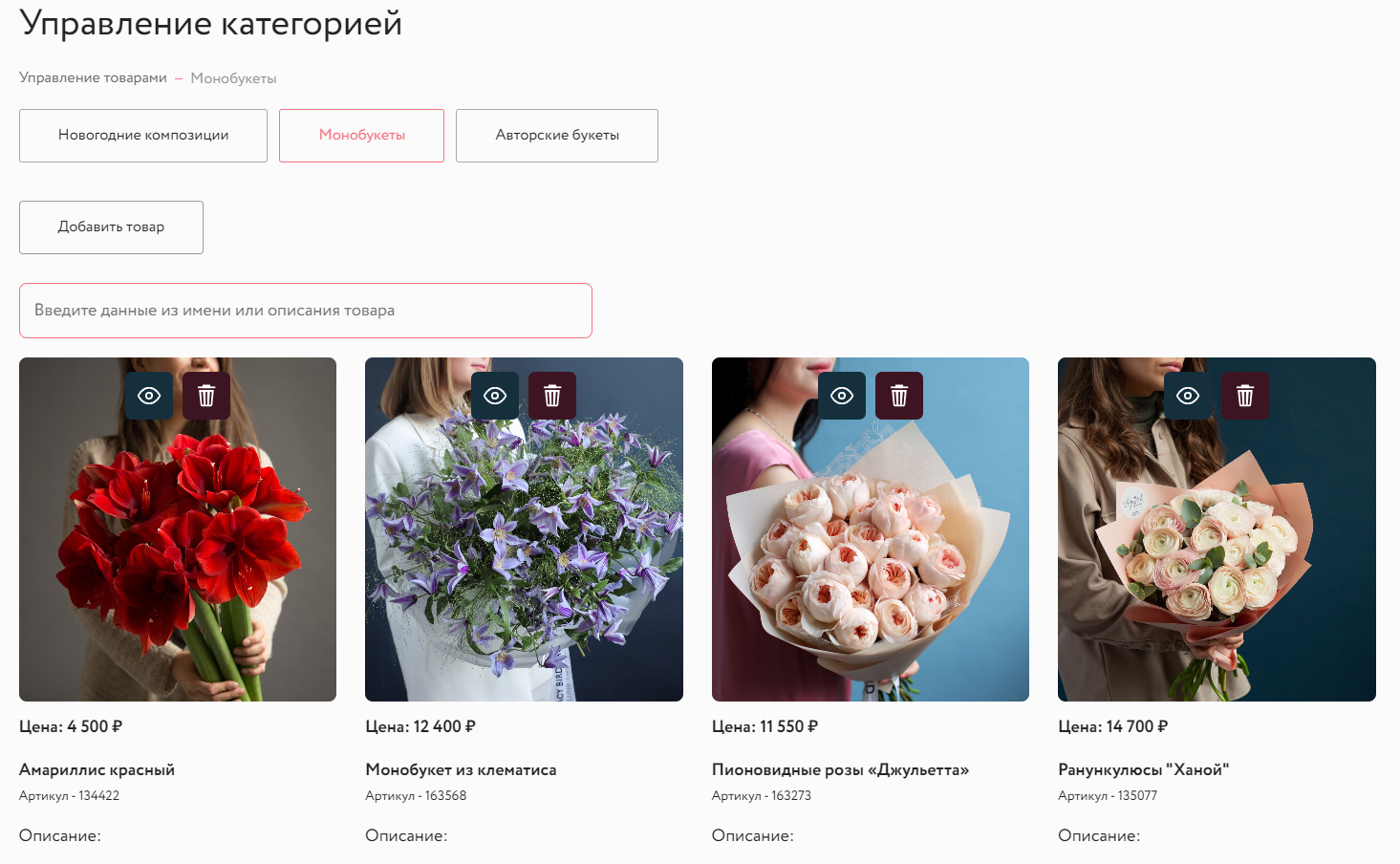


Рисунок 10 – Список товаров определенной категории

На рисунке 11 предоставлен интерфейс удаления товара.

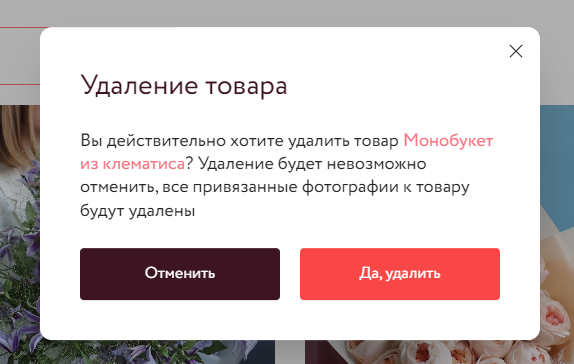


Рисунок 11 – Окно удаления товара

На рисунке 12 предоставлен интерфейс редактирования категории.

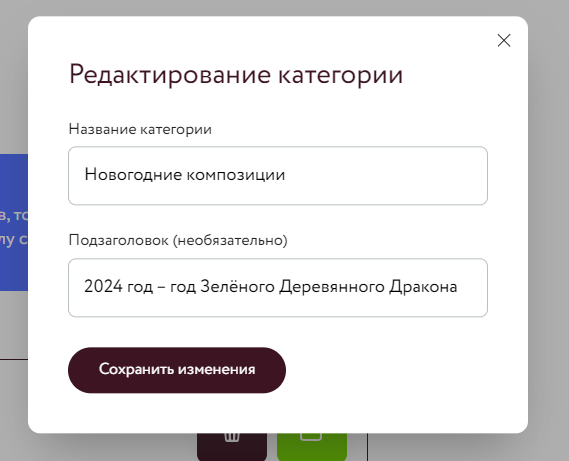


Рисунок 12 – Окно редактирования категории

На рисунке 13 предоставлен интерфейс для работы с неодобренными отзывами.

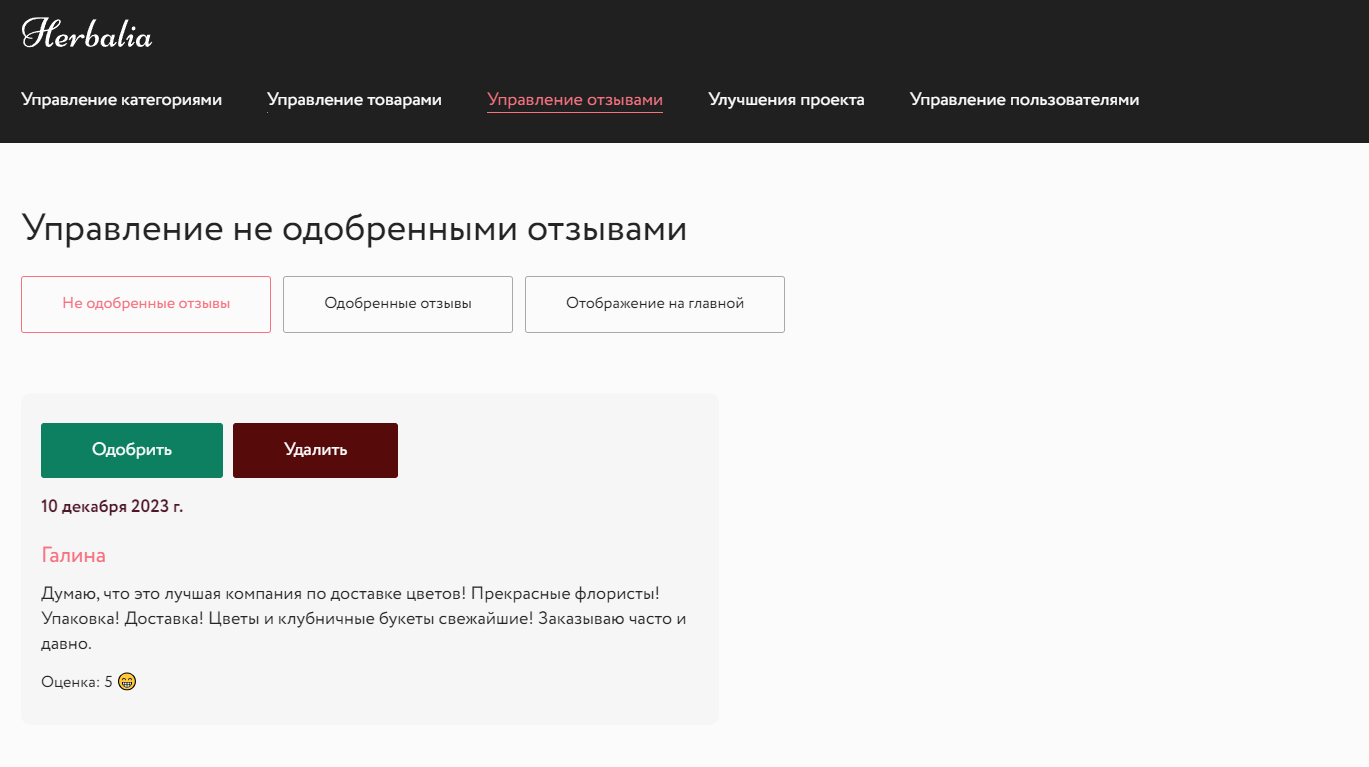


Рисунок 13 – Интерфейс управления неодобренными отзывами

На рисунке 14 предоставлен интерфейс для работы с одобренными отзывами.

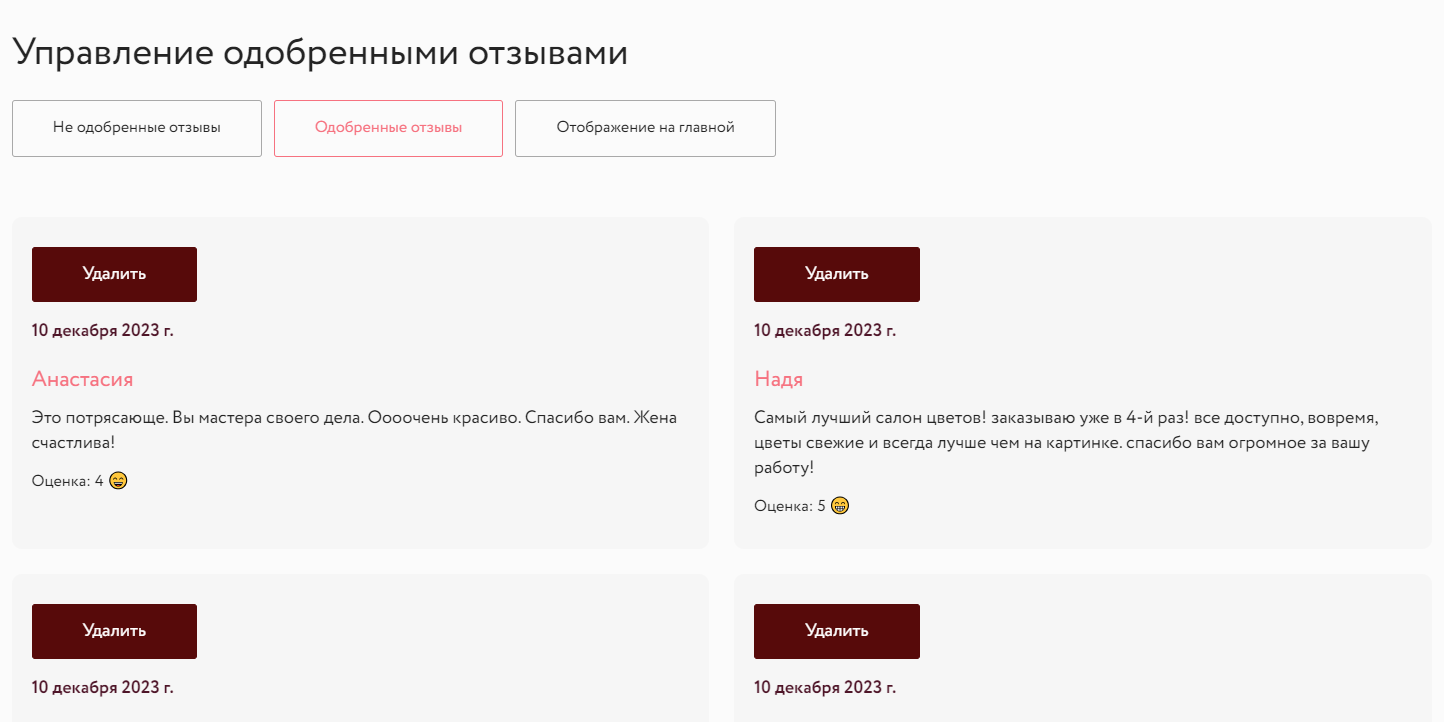


Рисунок 14 – Интерфейс управления одобренными отзывами

На рисунке 15 предоставлен интерфейс для удаления отзыва.

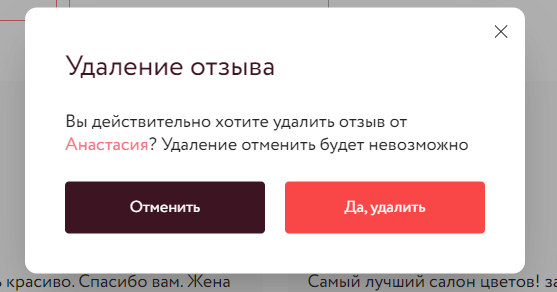


Рисунок 15 – Интерфейс для удаления отзыва

На рисунке 16 предоставлен интерфейс для управления отображением отзывов на главной странице

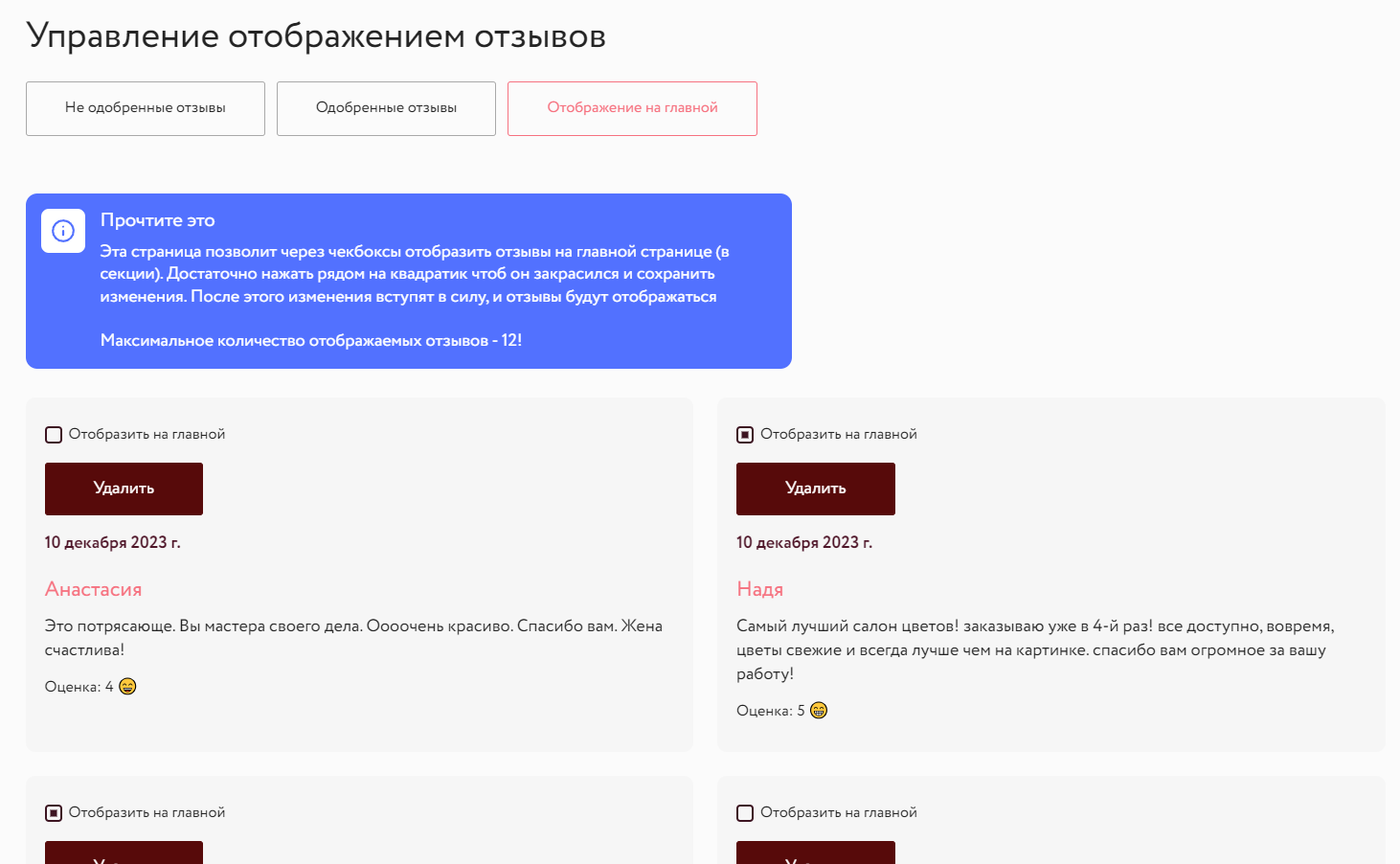


Рисунок 16 – Интерфейс для отображения отзывов на главной странице

На рисунке 17 предоставлен интерфейс для управления пользователями

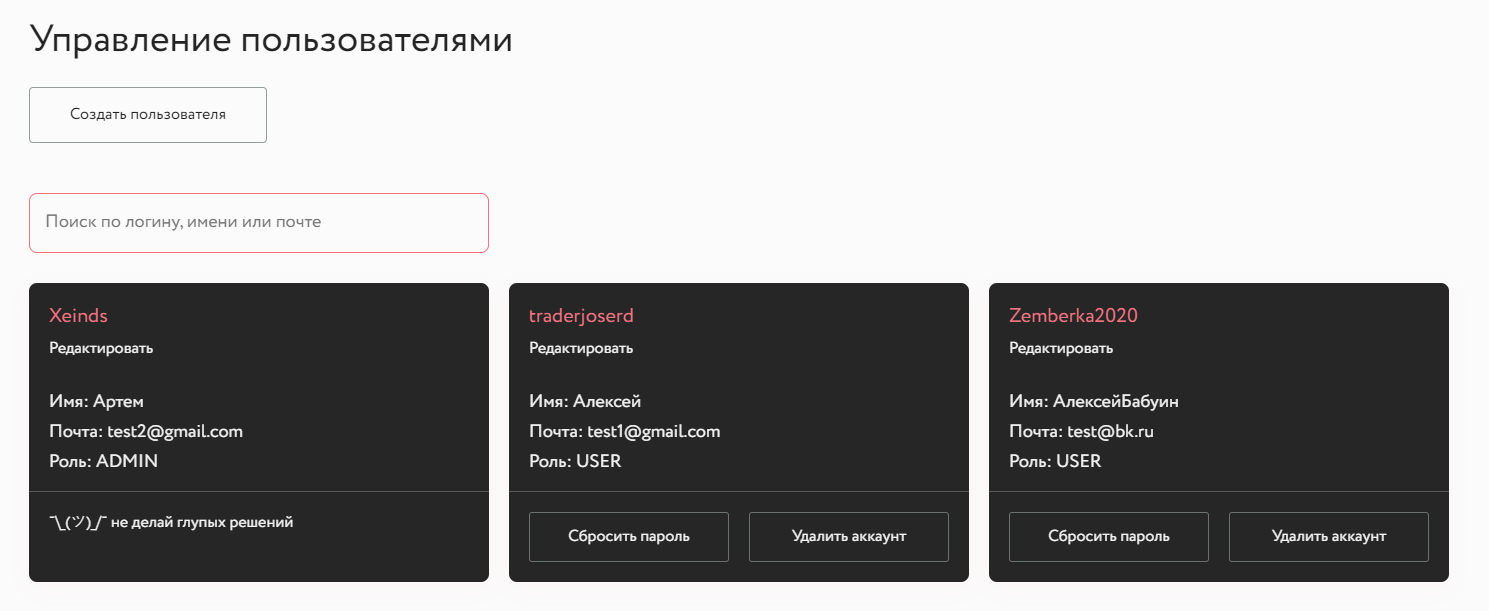


Рисунок 17 – Интерфейс для управления пользователями

На рисунке 18 предоставлен интерфейс для удаления пользователя

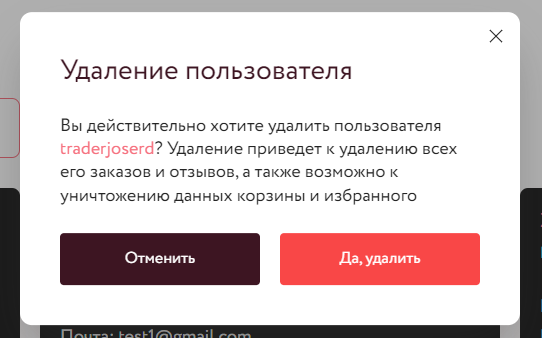


Рисунок 18 – Интерфейс для удаления пользователя

На рисунке 19 предоставлен интерфейс для удаления пользователя

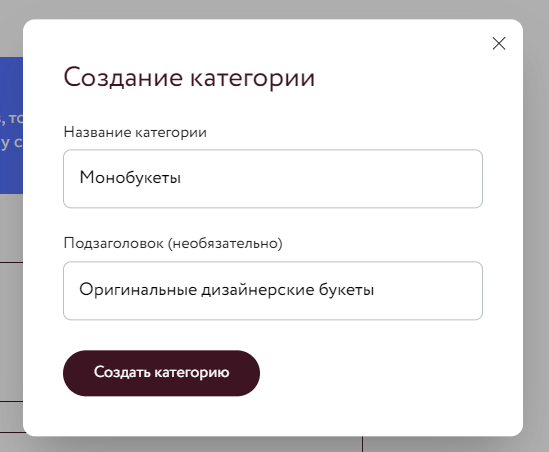


Рисунок 19 – Интерфейс для удаления пользователя

На рисунке 20 предоставлен интерфейс для редактирования существующей категории товаров

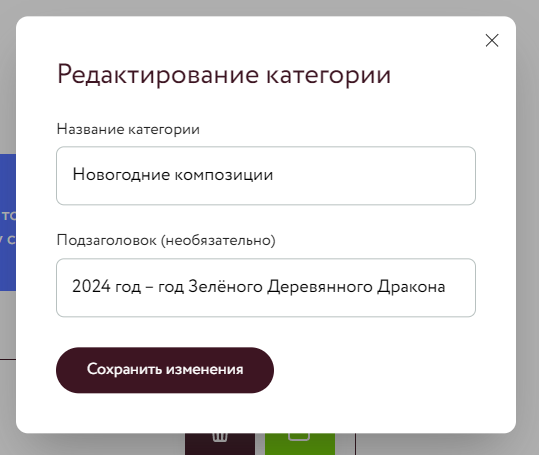


Рисунок 20 – Интерфейс для редактирования категории товаров

На рисунке 20 предоставлен интерфейс для создания пользователя

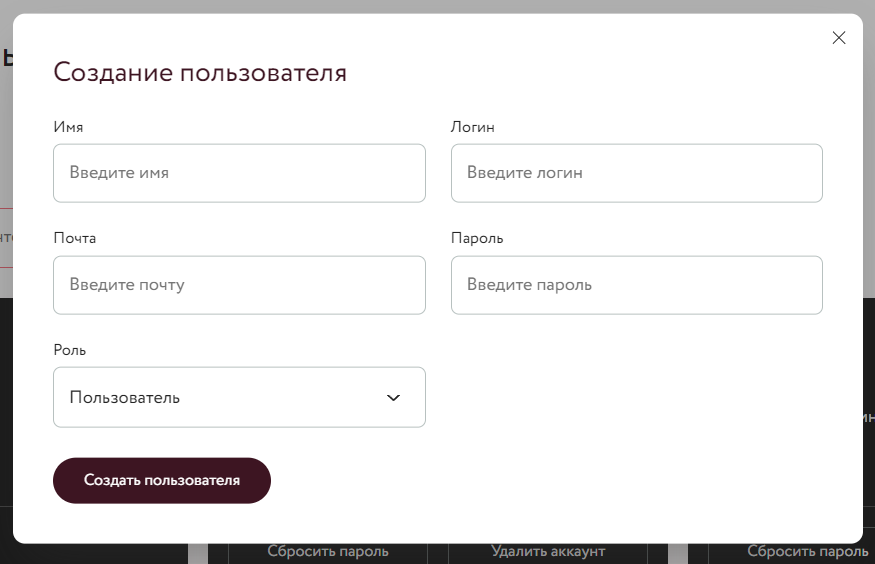


Рисунок 20 – Интерфейс для создания пользователя

2. ПРОЕКТНАЯ ЧАСТЬ

2.1 Техническое задание

Электронный каталог цветочного магазина предназначен для предоставления клиентам возможности просмотра и приобретения цветов и товаров для садоводства через веб-платформу. Целью разработки является создание удобной и интуитивно понятной онлайн-платформы для клиентов и администраторов магазина.

Разработанный электронный каталог цветочного магазина стремится создать гармоничное взаимодействие между клиентами и администраторами, предоставляя не только удобство приобретения товаров и просмотра информации, но и возможность эффективного управления контентом и обратной связи.

Из функциональных требований исходит следующие пункты.

Управление товарами:

* Создание, редактирование и удаление информации о товарах (название, описание, цена, изображения);
* Возможность добавления категорий к товарам для удобства поиска;

Управление отзывами:

* Добавление отзывов о товарах и сервисе в целом со стороны пользователей;
* Полноценная возможность управления отзывами (одобрение, удаление оставленных отзывов, а также отображение части отзывов на главной странице);

Управление пользователями:

* Авторизация и аутентификация пользователей;
* Возможность просмотра информации пользователей администраторами;
* Возможность добавлять, редактировать и удалять пользователей от имени администратора;

Интерфейс для пользователей:

* Просмотр каталога товаров с возможностью детального ознакомления и добавления отзывов;

Нефункциональные требования играют важную роль в обеспечении эффективной работы и безопасности электронного каталога цветочного магазина. Интерфейс должен быть интуитивно понятным для всех пользователей и адаптивным для корректного отображения на различных устройствах. Безопасность приложения несет важное значение и предполагает реализацию системы аутентификации и авторизации для администраторов и пользователей, а также защиту от уязвимостей веб-приложения, включая SQL-инъекции и XSS-атаки. Производительность включает оптимизацию загрузки изображений и данных для быстрого отображения страниц каталога, а также минимизацию времени ответа сервера на запросы клиентов.

Нефункциональные требования выполняются для обеспечения безопасности, производительности и удобства использования электронного каталога цветочного магазина:

Интерфейс:

* Интуитивно понятный интерфейс для всех категорий пользователей;
* Адаптивный дизайн для корректного отображения на различных устройствах;

Безопасность:

* Реализация системы аутентификации и авторизации для администраторов и пользователей;
* Защита от уязвимостей веб-приложения, таких как SQL-инъекции, CSRF-атак и XSS-атаки;

Производительность:

* Оптимизация загрузки изображений и данных для быстрого отображения страниц каталога;
* Минимизация времени ответа сервера на запросы клиентов;

Инновационные технологии, безупречная функциональность и надежная безопасность — вот то, что делает этот электронный каталог цветочного магазина неповторимым. Позвольте погрузиться в детали технических требований, определяющих основу его функционирования.

Технические требования составляют основу для создания надежной и инновационной платформы каталога цветочного магазина:

Фреймворк и язык программирования:

* Использование PHP фреймворка Laravel для бэкенда приложения;
* Использование Blade шаблонизатора для генерации HTML-страниц;
* Vanilla JS для добавления интерактивности на клиентской стороне;

Хранение данных:

* Использование базы данных для хранения информации о товарах, отзывах и пользователях;
* Возможно использование MySQL или другой совместимой СУБД;

Тестирование и безопасность:

* Регулярное тестирование приложения для выявления и исправления ошибок;
* Реализация мер безопасности для защиты данных пользователей;

2.2 Установка и настройка среды разработки

Для начала выберем хостинг, который поддерживает PHP и базы данных MySQL (или другую, поддерживаемую Laravel). Некоторые известные хостинги, такие как Bluehost, SiteGround и DigitalOcean, предоставляют простые способы установки Laravel.

Далее нужно зайти в панель управления вашего хостинга. Обычно, после входа в аккаунт, мы попадаем на главную страницу хостинга или панель управления, где можно управлять нашим веб-сайтами и настройками.

Некоторые хостинги предоставляют «Мастер установки приложений» или «Установку одним кликом». Выберем Laravel из списка доступных приложений и создаем новый проект. Этот метод установки обычно упрощает процесс создания проекта, автоматически формируя начальную структуру приложения и предоставляя необходимые зависимости. По завершению установки, рекомендуется проверить корректность установки и настроек, чтобы гарантировать правильное функционирование приложения

Разработанный электронный каталог цветочного магазина стремится создать гармоничное взаимодействие между клиентами и администраторами, предоставляя не только удобство приобретения товаров и просмотра информации, но и возможность эффективного управления контентом и обратной связи.

Если хостинг не предоставляет установку Laravel, то можно загрузить файлы проекта на сервер с помощью FTP или SSH шлюз. Также нужно создать директорию для нашего проекта, например, public\_html/mylaravelapp, и загрузить туда файлы проекта.

После требуется создать новую базу данных MySQL через панель управления хостингом. Обычно, в разделе «Базы данных» есть возможность создать новую базу данных, указать ей имя, пользователя и пароль для доступа.

Также нам нужно настроить подключение к базе данных. Найдя файл env в корне проекта и внесите изменения в параметры подключения к базе данных: DB\_CONNECTION, DB\_HOST, DB\_PORT, DB\_DATABASE, DB\_USERNAME, и DB\_PASSWORD.

Настройка маршрутизации в Laravel включает в себя определение URL-адресов, которые будут обрабатываться вашим приложением, и их направление на соответствующие контроллеры и действия. Это позволяет корректно отображать содержимое и обеспечивать функциональность приложения в зависимости от URL-адресов, по которым пользователи перемещаются.

После успешной установки и корректной настройки маршрутизации, сайт становится доступным для посещения, обеспечивая пользователям функциональность и удобство в использовании. Если всё сделано правильно, мы увидим тот самый контент проекта, который мы загрузили. После всех вышеописанных этапов сайт должен работать корректно, и на него могут заходить живые люди и пользоваться им.

2.3 Руководство пользователя

В приложении имеется возможность просматривать и оставлять отзывы. Созданием отзывов могут заниматься все авторизованные пользователи системы! Чтобы добавить свой отзыв о магазине или товаре, то нужно перейти из шапки сайта в вкладку «Отзывы», по нажатию которой будет предоставлен следующий интерфейс на рисунке 23.

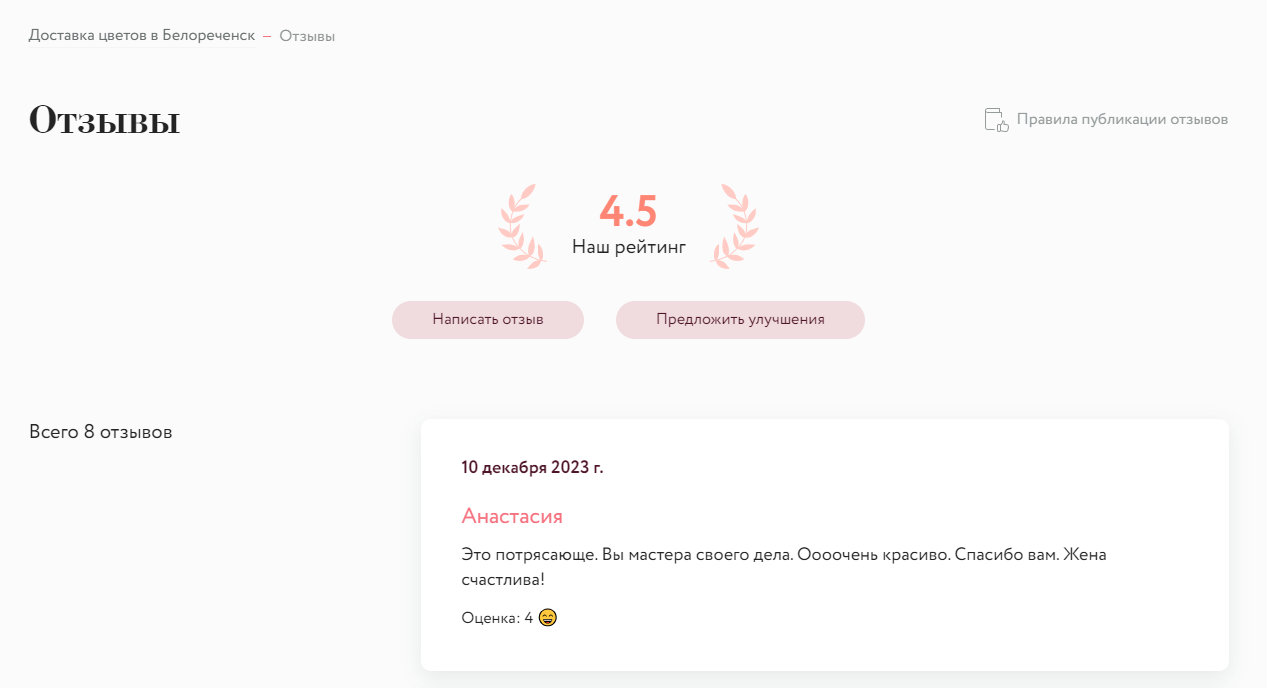


Рисунок 22 – Интерфейс для страницы отзывов

На данной странице можно увидеть все отзывы людей, которые были одобрены администратором. Также имеется ссылка на написание отзыва. По нажатию которой откроется страница с формой отправки отзыва.

На страницу написания отзыва может попасть лишь авторизованный пользователь. Если гость приложения попытается зайти на эту страницу, то его автоматически перенаправит на страницу авторизации. Таким образом мы защищаем приложение от спам-атак в виде отзывов о магазине.

Если пользователь в форме ввел некорректно имя, почту то форма не отправится в БД, она уведомит пользователя об ошибках. В случае исправления ошибок и повторной попытки отправить отзыв – он успешно добавиться в БД, а пользователю высветиться окно о том, что его отзыв находиться на модерации, и будет одобрен в ближайшее время.

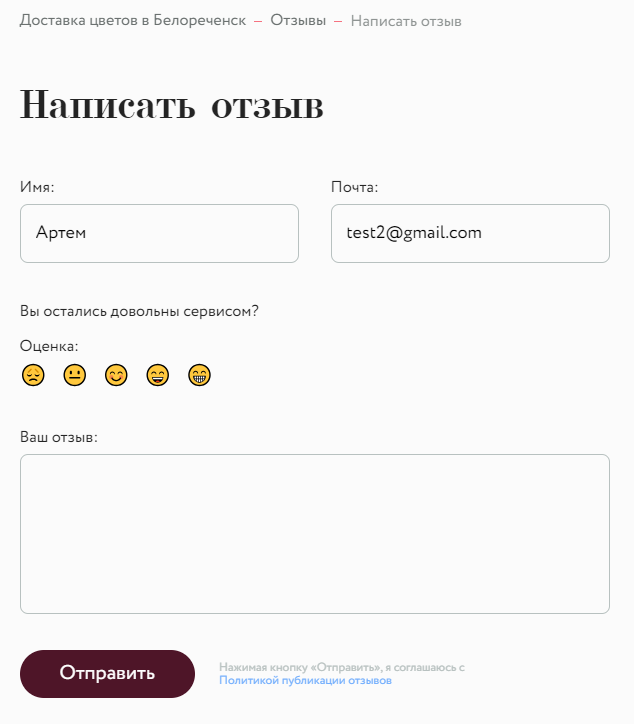


Рисунок 24 – Страница с формой отправки отзыва

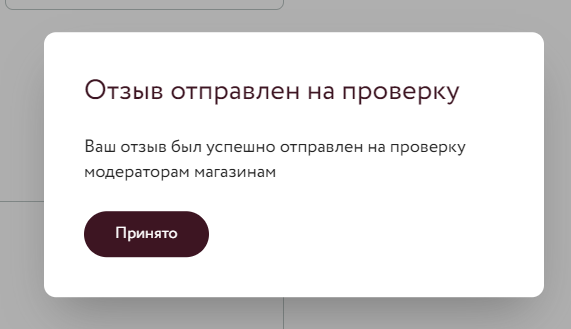


Рисунок 25 – Успешная отправка созданного отзыва пользователем

2.4 Руководство администратора

В программе имеется функция добавления данных. Добавлять информацию можно в следующие таблицы – категории, товары, и пользователи. Чтобы вызвать окно или страницу добавления достаточно будет нажать кнопку «Добавить» в определенном экране программы. Когда была нажат кнопка, то откроется интерфейс, в котором будут доступны поля для заполнения новыми данными.

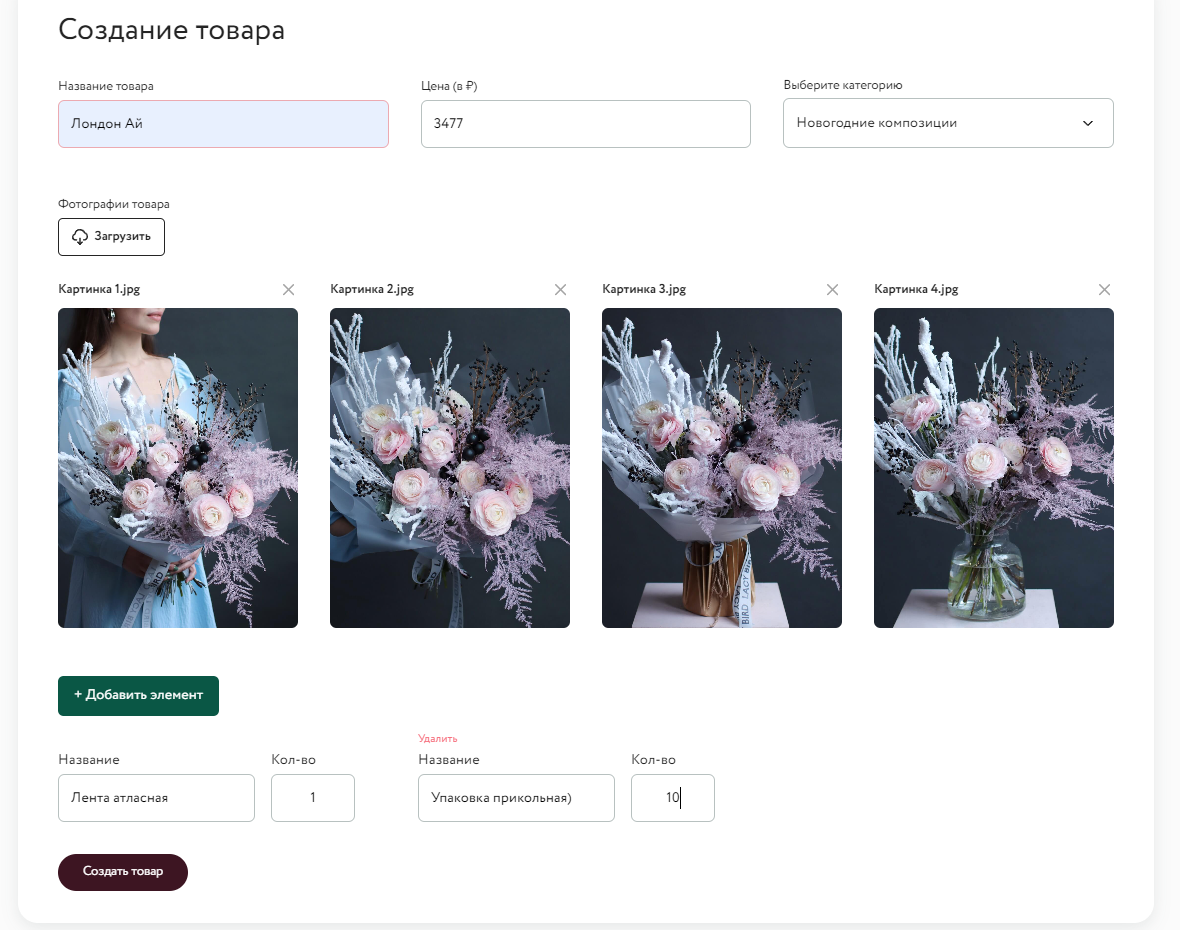


Рисунок 26 – Руководство администратора для добавления данных

Дальше идет заполнение данных администратором. Но стоит учитывать, что если были введены некорректные данные, то в итоге будут выведены свои ошибки для каждого поля, в которых будет описано что не так заполнено в поле. После того как будет нажата кнопка «Сохранить» в модальном окне, то после успешного создания объекта сущности перенаправит на общую страницу администрирования (где выведены все объекты какой-либо сущности). И далее запись, которая добавилась в БД, отобразится в интерфейсе программы.

В программе имеется функция редактирования данных. Редактировать информацию можно в следующих таблицах – категории, и пользователи. Чтобы вызвать интерфейс редактирования нужно сначала определиться какой объект сущности мы будем изменять.

Изображение выглядит как текст, снимок экрана, Шрифт, программное обеспечение

Автоматически созданное описание

Рисунок 27 – Руководство администратора для редактирования данных

Когда мы нажали на кнопку, то откроется сам интерфейс редактирования в модальном окне или же на отдельной странице редактирования, и будут заполнены поля значениями, взятыми из определенной записи.

Дальше уже идет заполнение новыми данными под старые. Стоит учитывать, что если были введены некорректные данные, то в итоге будут выведены свои ошибки для каждого поля, в которых будет описано что не так заполнено в поле, и потребуется изменить данные чтобы они стали корректными, иначе данные не отправятся на сервер.

После того как была нажата кнопка «Сохранить», изменения вступят в силу. Администратор на клиенте увидит перенаправление на другую страницу (в случае успеха), и далее запись, которая изменилась, обновится в интерфейсе программы.

В программе имеется функция удаления существующих данных. Удалять информацию можно из следующих таблицах – категории, товары, пользователи и отзывы. Чтобы вызвать удаление достаточно будет нажать на кнопку удаления у определенной записи, и откроется всплывающее окно с подтверждением удаления данных.

Изображение выглядит как текст, цветок, визитная карточка, снимок экрана

Автоматически созданное описание

Рисунок 28 – Руководство администратора для удаления данных

Если была нажата кнопка «Да, удалить», то удалится запись. После этого обновится список данных в интерфейсе приложения, где удаленная запись не будет уже отображаться.

Если же была нажата кнопка «Нет», то запись не будет удалена из базы данных, и закроется всплывающее окно подтверждения удаления записи.

ЗАКЛЮЧЕНИЕ

Разработка интернет-магазина цветов на базе PHP Laravel с шаблонизатором Blade позволила создать функциональный и эстетичный проект. Используя современные инструменты разработки, включая фреймворк Laravel, достигнута гибкость и масштабируемость приложения. Целью проекта было создание удобного пространства для выбора и заказа цветов, обеспечивая пользователям уникальный и легкий опыт покупки.

Использование фреймворка Laravel значительно облегчило расширение функциональности магазина с ростом его популярности. Масштабируемость этого фреймворка позволила без проблем внедрять новые возможности и обрабатывать увеличивающееся количество запросов.

Одной из ключевых задач было обеспечение безопасности, интуитивной навигации и удобства в управлении контентом для администраторов. Используя Blade шаблоны в Laravel, удалось создать интуитивно понятный интерфейс для администраторов и пользователей, обеспечив безопасную и удобную среду для работы с данными о клиентах, категориях товаров и самих товаров.

Инфраструктура на основе PHP Laravel и шаблонизатора Blade позволила быстро реализовать функционал добавления, редактирования и удаления товаров, а также создать эффективную систему поиска, что значительно облегчило процесс поиска нужного товара.

Задачи, описанные в начале исследовательской работе решены, а именно создать трёхуровневую архитектуру приложения, быстрый доступ к базе данных, добавлению товара, просмотра, редактирования и удаления любого товара, а также управление другим контентом на сайте.

В результате выполнения курсовой, был создан динамический сайт для внесения и хранения там данных о товарах. Для его разработки применялось актуальное программное обеспечение, без помощи которого не было бы возможности добиться поставленной цели. Это программное обеспечение включает в себя:

* язык программирования PHP;
* среда разработки для работы с языком программирования PHP – PhpStorm;
* фреймворк PHP Laravel;
* веб-сервер XAMPP;
* реляционная система управления БД MySQL;
* приложение PhpMyAdmin;

Этот проект стал хорошим примером использования современных технологий разработки в области интернет-магазинов, обеспечивая пользователей качественным и интуитивно понятным интерфейсом, а администраторов — легким управлением контентом. Всё это было возможно благодаря гибкости и функциональности PHP Laravel и Blade шаблонизатора, которые позволили создать удивительный интернет-магазин цветов, удовлетворяющий потребности как пользователей, так и администраторов.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Вандюк, Джон К. CMS Drupal. Руководство по разработке системы управления сайтом / Вандюк, Джон К., Мэтт Вестгейт, - М.: Вильямс, 2019. - 400 c.
2. Гаевский, А.Ю. 100% самоучитель. Создание Web-страниц и Web-сайтов. HTML и JavaScript / А.Ю. Гаевский, В.А. Романовский. - М.: Триумф, 2020. - 464 c.
3. Дакетт, Джон HTML и CSS. Разработка и дизайн веб-сайтов / Джон Дакетт. - Москва: Наука, 2019. - 480 c.
4. Дронов, В. JavaScript в Web-дизайне / В. Дронов. - М.: СПб: БХВ, 2019. - 880 c.
5. Дронов, В. PHP, MySQL и Dreamweaver. Разработка интерактивных Web-сайтов / В. Дронов. - М.: БХВ-Петербург, 2019. - 480 c.
6. Дронов, В.А. PHP 5/6, MySQL 5/6 и Dreamweaver CS4. Разработка интерактивных Web-сайтов / В. А. Дронов. - М.: БХВ-Петербург, 2019. - 820 c.
7. Дронов, Владимир JavaScript и AJAX в Web-дизайне / Владимир Дронов. - Москва: Высшая школа, 2019. - 736 c.
8. Дронов, Владимир PHP, MySQL и Dreamweaver MX 2004. Разработка интерактивных Web-сайтов / Владимир Дронов. - М.: "БХВ-Петербург", 2019. - 448 c.
9. Колисниченко, Д. PHP 5/6 и MySQL 6. Разработка Web-приложений / Д. Колисниченко. - М.: БХВ-Петербург, 2021. - 560 c.
10. Колисниченко, Д.Н. PHP 5/6 и MySQL 6. Разработка Web-приложений / Д.Н. Колисниченко. - М.: БХВ-Петербург, 2021. - 364 c.
11. Кузнецов PHP. Практика создания Web-сайтов / Кузнецов, М.В. и. - М.: БХВ-Петербург, 2020. - 895 c.
12. Кузнецов, М. PHP 5. Практика создания Web-сайтов / М. Кузнецов, И. Симдянов, С. Голышев. - М.: БХВ-Петербург, 2020. - 960 c.
13. Леонтьев, Борис PHP 5.0 для начинающих, или как создать динамический WEB-сайт / Борис Леонтьев. - М.: Новый издательский дом, 2019. - 176 c.
14. Петюшкин, Алексей HTML в Web-дизайне / Алексей Петюшкин. - Москва: Машиностроение, 20. - 400 c.
15. Шкрыль, А. PHP — это просто. Программируем для Web-сайта / А. Шкрыль. - М.: БХВ-Петербург, 2021. - 368 c.

# **ПРИЛОЖЕНИЕ**

Kernel.php

<?php  
  
namespace App\Http;  
  
use App\Http\Middleware\AdminMiddleware;  
use App\Http\Middleware\Authenticate;  
use App\Http\Middleware\CheckDatabaseConnection;  
use Illuminate\Foundation\Http\Kernel as HttpKernel;  
  
class Kernel extends HttpKernel  
{  
 protected $routeMiddleware = [  
 'admin' => AdminMiddleware::class,  
 'checkdb' => CheckDatabaseConnection::class,  
 'auth' => Authenticate::class,  
 ];  
  
 protected $middleware = [  
 // \App\Http\Middleware\TrustHosts::class,  
 \App\Http\Middleware\TrustProxies::class,  
 \Illuminate\Http\Middleware\HandleCors::class,  
 \App\Http\Middleware\PreventRequestsDuringMaintenance::class,  
 \Illuminate\Foundation\Http\Middleware\ValidatePostSize::class,  
 \App\Http\Middleware\TrimStrings::class,  
 \Illuminate\Foundation\Http\Middleware\ConvertEmptyStringsToNull::class,  
 ];  
  
 protected $middlewareGroups = [  
 'web' => [  
 \App\Http\Middleware\EncryptCookies::class,  
 \Illuminate\Cookie\Middleware\AddQueuedCookiesToResponse::class,  
 \Illuminate\Session\Middleware\StartSession::class,  
 \Illuminate\View\Middleware\ShareErrorsFromSession::class,  
 \App\Http\Middleware\VerifyCsrfToken::class,  
 \Illuminate\Routing\Middleware\SubstituteBindings::class,  
 ],  
  
 'api' => [  
 // \Laravel\Sanctum\Http\Middleware\EnsureFrontendRequestsAreStateful::class,  
 \Illuminate\Routing\Middleware\ThrottleRequests::class . ':api',  
 \Illuminate\Routing\Middleware\SubstituteBindings::class,  
 ],  
 ];  
  
 protected $middlewareAliases = [  
 'auth' => Authenticate::class,  
 'auth.basic' => \Illuminate\Auth\Middleware\AuthenticateWithBasicAuth::class,  
 'auth.session' => \Illuminate\Session\Middleware\AuthenticateSession::class,  
 'cache.headers' => \Illuminate\Http\Middleware\SetCacheHeaders::class,  
 'can' => \Illuminate\Auth\Middleware\Authorize::class,  
 'guest' => \App\Http\Middleware\RedirectIfAuthenticated::class,  
 'password.confirm' => \Illuminate\Auth\Middleware\RequirePassword::class,  
 'precognitive' => \Illuminate\Foundation\Http\Middleware\HandlePrecognitiveRequests::class,  
 'signed' => \App\Http\Middleware\ValidateSignature::class,  
 'throttle' => \Illuminate\Routing\Middleware\ThrottleRequests::class,  
 'verified' => \Illuminate\Auth\Middleware\EnsureEmailIsVerified::class,  
 ];  
}

StoreUserRequest.php

<?php

namespace App\Http\Requests;

use Illuminate\Foundation\Http\FormRequest;

class StoreUserRequest extends FormRequest

{

    public function authorize(): bool

    {

        return auth()->user()->role === 'ADMIN';

    }

    public function rules(): array

    {

        return [

            'login' => 'required|string|min:5|max:60|unique:users',

            'name' => 'required|string|min:2|max:50',

            'email' => 'required|email|unique:users,email',

            'password' => 'required|string|min:8|max:60',

            'role' => 'required|in:ADMIN,USER',

        ];

    }

}

UpdateUserRequest.php

<?php

namespace App\Http\Requests;

use Illuminate\Foundation\Http\FormRequest;

class UpdateUserRequest extends FormRequest

{

    public function authorize(): bool

    {

        return auth()->user()->role === 'ADMIN';

    }

    public function rules(): array

    {

        return [

            'login' => 'required|string|min:5|max:60|unique:users,login,' . $this->user->id,

            'name' => 'required|string|min:2|max:50',

            'email' => 'required|email|unique:users,email,' . $this->user->id,

            'password' => 'nullable|string|min:8|max:60',

            'role' => 'required|in:ADMIN,USER',

        ];

    }

}

AdminMiddleware.php

<?php  
namespace App\Http\Middleware;  
use Closure;  
use Illuminate\Http\Request;  
class AdminMiddleware  
{  
 public function handle(Request $request, Closure $next)  
 {  
 if (!$request->user() || $request->user()->role !== 'ADMIN') {  
 return redirect('/403\_error');  
 }  
 return $next($request);  
 }  
}

CheckDatabaseConnection.php

<?php  
namespace App\Http\Middleware;  
use Closure;  
use Illuminate\Database\QueryException;  
use Illuminate\Http\Request;  
use Illuminate\Support\Facades\DB;  
use Symfony\Component\HttpFoundation\Response;  
class CheckDatabaseConnection  
{  
 public function handle(Request $request, Closure $next): Response  
 {  
 try {  
 DB::connection()->getPdo();  
 } catch (\Exception) {  
 return redirect('/503\_error');  
 }  
 return $next($request);  
 }  
}

RouteServiceProvider.php

<?php

namespace App\Providers;

use Illuminate\Cache\RateLimiting\Limit;

use Illuminate\Foundation\Support\Providers\RouteServiceProvider as ServiceProvider;

use Illuminate\Http\Request;

use Illuminate\Support\Facades\RateLimiter;

use Illuminate\Support\Facades\Route;

class RouteServiceProvider extends ServiceProvider

{

    public const HOME = '/home';

    public function boot(): void

    {

        RateLimiter::for('api', function (Request $request) {

            return Limit::perMinute(60)->by($request->user()?->id ?: $request->ip());

        });

        $this->routes(function () {

            Route::middleware('api')

                ->prefix('api')

                ->group(base\_path('routes/api.php'));

            Route::middleware('web')

                ->group(base\_path('routes/web.php'));

            Route::middleware('web')

                ->group(base\_path('routes/home.php'));

            Route::middleware('web')

                ->group(base\_path('routes/auth/auth.php'));

            Route::middleware('web')

                ->group(base\_path('routes/admin/category.php'));

            Route::middleware('web')

                ->group(base\_path('routes/admin/products.php'));

            Route::middleware('web')

                ->group(base\_path('routes/reviews.php'));

            Route::middleware('web')

                ->group(base\_path('routes/admin/reviews.php'));

            Route::middleware('web')

                ->group(base\_path('routes/admin/users.php'));

        });

    }

}

User.php

<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Factories\HasFactory;

use Illuminate\Database\Eloquent\Relations\HasMany;

use Illuminate\Foundation\Auth\User as Authenticatable;

class User extends Authenticatable

{

    use HasFactory;

    protected $fillable = [

        'login',

        'name',

        'email',

        'password',

        'role',

    ];

    public function reviews(): HasMany

    {

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

    }

}

Review.php

<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Factories\HasFactory;

use Illuminate\Database\Eloquent\Model;

use Illuminate\Database\Eloquent\Relations\BelongsTo;

class Review extends Model

{

    use HasFactory;

    protected $fillable = [

        'name',

        'email',

        'rating',

        'comment',

        'status',

        'display\_on\_homepage',

    ];

    public function user(): BelongsTo

    {

        return $this->belongsTo(User::class);

    }

}

ProductImage.php

<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Factories\HasFactory;

use Illuminate\Database\Eloquent\Model;

use Illuminate\Database\Eloquent\Relations\BelongsTo;

class ProductImage extends Model

{

    use HasFactory;

    protected $fillable = [

        'name',

        'path'

    ];

    public function product(): BelongsTo

    {

        return $this->belongsTo(Product::class);

    }

}

ProductComposition.php

<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Factories\HasFactory;

use Illuminate\Database\Eloquent\Model;

use Illuminate\Database\Eloquent\Relations\BelongsTo;

class ProductComposition extends Model

{

    use HasFactory;

    protected $fillable = [

        'product\_id',

    ];

    public function product(): BelongsTo

    {

        return $this->belongsTo(Product::class);

    }

}

Product.php

<?php

namespace App\Models;

use App\Http\Controllers\Admin\ProductController;

use Illuminate\Database\Eloquent\Factories\HasFactory;

use Illuminate\Database\Eloquent\Model;

use Illuminate\Database\Eloquent\Relations\BelongsTo;

use Illuminate\Database\Eloquent\Relations\HasMany;

class Product extends Model

{

    use HasFactory;

    protected $fillable = [

        'name',

        'price',

        'article',

        'category\_id'

    ];

    protected static function boot(): void

    {

        parent::boot();

        static::deleting(function ($product) {

            $product->images()->delete();

        });

    }

    public function category(): BelongsTo

    {

        return $this->belongsTo(Category::class);

    }

    public function images(): HasMany

    {

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

    }

    public function composition(): HasMany

    {

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

    }

}

Category.php

<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Factories\HasFactory;

use Illuminate\Database\Eloquent\Model;

use Illuminate\Database\Eloquent\Relations\HasMany;

class Category extends Model

{

    use HasFactory;

    protected $fillable = [

        'name',

        'subtitle',

        'order\_index'

    ];

    public function products(): HasMany

    {

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

    }

}

ReviewController.php

<?php

namespace App\Http\Controllers;

use App\Models\Review;

use App\Models\User;

use Illuminate\Contracts\Foundation\Application;

use Illuminate\Contracts\View\Factory;

use Illuminate\Contracts\View\View;

use Illuminate\Http\JsonResponse;

use Illuminate\Http\RedirectResponse;

use Illuminate\Http\Request;

use Illuminate\Support\Facades\Auth;

use Illuminate\Support\Facades\Response;

class ReviewController extends Controller

{

    public function index(): View|Application|Factory

    {

        $reviews = Review::where('status', true)->get();

        $totalRating = 0;

        $reviewCount = count($reviews);

        foreach ($reviews as $review) {

            $totalRating += $review->rating;

        }

        $averageRating = $reviewCount > 0 ? round($totalRating / $reviewCount, 1) : 0;

        return view('reviews', compact('reviews', 'averageRating'));

    }

    public function create(): Application|Factory|View|RedirectResponse

    {

        if (!Auth::check()) {

            return redirect()->route('auth');

        }

        return view('reviews-form');

    }

    public function store(Request $request): JsonResponse

    {

        if (Auth::check()) {

            $validatedData = $request->validate([

                'name' => 'required|string|max:50',

                'email' => 'required|email|max:80',

                'rating' => 'required|integer|min:1|max:5',

                'comment' => 'required|string|max:2000',

            ]);

            $review = new Review($validatedData);

            Auth::user()->reviews()->save($review);

            return response()->json(['message' => 'Отзыв был успешно добавлен'], 200);

        } else {

            return response()->json(['message' => 'Пользователь не авторизован'], 403);

        }

    }

}

HomeController.php

<?php

namespace App\Http\Controllers;

use App\Models\Category;

use App\Models\Product;

use App\Models\Review;

use Illuminate\Contracts\View\Factory;

use Illuminate\Contracts\View\View;

use Illuminate\Foundation\Application;

class HomeController extends Controller

{

    public function index(): View|Application|Factory|\Illuminate\Contracts\Foundation\Application

    {

        $categories = Category::orderBy('order\_index')->get();

        $categoryProducts = [];

        $approvedReviews = Review::where('display\_on\_homepage', true)->get();

        foreach ($categories as $category) {

            $categoryProducts[$category->id] = Product::where('category\_id', $category->id)->limit(6)->get();

        }

        return view('index', compact('categories', 'categoryProducts', 'approvedReviews'));

    }

}

Controller.php

<?php

namespace App\Http\Controllers;

use Illuminate\Foundation\Auth\Access\AuthorizesRequests;

use Illuminate\Foundation\Validation\ValidatesRequests;

use Illuminate\Routing\Controller as BaseController;

class Controller extends BaseController

{

    use AuthorizesRequests, ValidatesRequests;

}

RegisterController.php

<?php

namespace App\Http\Controllers\Auth;

use App\Http\Controllers\Controller;

use App\Models\User;

use Exception;

use Illuminate\Contracts\View\Factory;

use Illuminate\Contracts\View\View;

use Illuminate\Foundation\Application;

use Illuminate\Http\RedirectResponse;

use Illuminate\Http\Request;

use Illuminate\Routing\Redirector;

use Illuminate\Support\Facades\Auth;

use Illuminate\Support\Facades\DB;

use Illuminate\Support\Facades\Hash;

class RegisterController extends Controller

{

    public function register(Request $request): Application|Redirector|RedirectResponse|\Illuminate\Contracts\Foundation\Application

    {

        DB::beginTransaction();

        try {

            $validatedData = $request->validate([

                'login' => 'required|string|min:5|max:60|unique:users|regex:/^[a-zA-Z0-9\_]+$/',

                'name' => 'required|string|min:2|max:50|regex:/^[A-Za-zА-Яа-яЁё\-]+$/u',

                'email' => 'required|email|unique:users,email',

                'password' => 'required|string|min:8|max:60',

            ]);

            User::create([

                'login' => $validatedData['login'],

                'name' => $validatedData['name'],

                'email' => $validatedData['email'],

                'password' => Hash::make($validatedData['password']),

                'role' => 'USER',

            ]);

            DB::commit();

            return redirect('login')->with('success', 'Регистрация прошла успешно!');

        } catch (Exception) {

            DB::rollBack();

            return redirect('/503\_error');

        }

    }

    public function showRegisterForm(): Factory|Application|View|Redirector|\Illuminate\Contracts\Foundation\Application|RedirectResponse

    {

        if (Auth::check()) {

            return redirect('/');

        }

        return view('register');

    }

}

AuthController.php

<?php

namespace App\Http\Controllers\Auth;

use App\Http\Controllers\Controller;

use Exception;

use Illuminate\Contracts\View\Factory;

use Illuminate\Contracts\View\View;

use Illuminate\Foundation\Application;

use Illuminate\Http\RedirectResponse;

use Illuminate\Http\Request;

use Illuminate\Routing\Redirector;

use Illuminate\Support\Facades\Auth;

class AuthController extends Controller

{

    public function login(Request $request): Application|Redirector|RedirectResponse|\Illuminate\Contracts\Foundation\Application

    {

        $credentials = $request->only('login', 'password');

        try {

            if (Auth::attempt($credentials)) {

                return redirect('/');

            } else {

                return redirect()->back()->withErrors([

                    'login' => 'Неверно введен логин или пароль',

                ]);

            }

        } catch (Exception) {

            return redirect('/503\_error');

        }

    }

    public function showLoginForm(): Factory|Application|View|Redirector|\Illuminate\Contracts\Foundation\Application|RedirectResponse

    {

        if (Auth::check()) {

            return redirect('/');

        }

        return view('login');

    }

    public function logout(Request $request): Application|Redirector|RedirectResponse|\Illuminate\Contracts\Foundation\Application

    {

        Auth::logout();

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

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

        return redirect('/login');

    }

}

AdminReviewController.php

<?php

namespace App\Http\Controllers\Admin;

use App\Http\Controllers\Controller;

use App\Models\Review;

use Illuminate\Contracts\View\Factory;

use Illuminate\Contracts\View\View;

use Illuminate\Foundation\Application;

use Illuminate\Http\JsonResponse;

use Illuminate\Http\Request;

use Illuminate\Support\Facades\Response;

class AdminReviewController extends Controller

{

    public function reviewsForApproval(): View|Application|Factory|\Illuminate\Contracts\Foundation\Application

    {

        $reviewsForApproval = Review::where('status', false)->get();

        return view('admin.reviews.reviews\_for\_approval', ['reviewsForApproval' => $reviewsForApproval]);

    }

    public function approvedReviews(): View|Application|Factory|\Illuminate\Contracts\Foundation\Application

    {

        $approvedReviews = Review::where('status', true)->get();

        return view('admin.reviews.approved\_reviews', ['approvedReviews' => $approvedReviews]);

    }

    public function approvedReviewsHomepage(): View|Application|Factory|\Illuminate\Contracts\Foundation\Application

    {

        $approvedReviews = Review::where('status', true)->get();

        return view('admin.reviews.approved\_reviews\_homepage', ['approvedReviews' => $approvedReviews]);

    }

    public function updateDisplayOnHomepage(Request $request): JsonResponse

    {

        $checkedReviews = $request->input('checkedReviews');

        Review::whereIn('id', $checkedReviews)->update(['display\_on\_homepage' => true]);

        $uncheckedReviews = Review::whereNotIn('id', $checkedReviews)->pluck('id');

        Review::whereIn('id', $uncheckedReviews)->update(['display\_on\_homepage' => false]);

        return response()->json(['message' => 'Статусы отзывов для главной страницы обновлены']);

    }

    public function approveReview(Request $request, $id): JsonResponse

    {

        $review = Review::find($id);

        if (!$review) {

            return Response::json(['message' => 'Отзыв не найден'], 404);

        }

        $review->status = true;

        $review->save();

        return Response::json(['message' => 'Отзыв одобрен']);

    }

    public function deleteReview(Request $request, $id): JsonResponse

    {

        $review = Review::find($id);

        if (!$review) {

            return Response::json(['message' => 'Отзыв не найден'], 404);

        }

        $review->delete();

        return Response::json(['message' => 'Отзыв удален']);

    }

}

CategoryController.php

<?php

namespace App\Http\Controllers\Admin;

use App\Http\Controllers\Controller;

use App\Models\Category;

use Illuminate\Contracts\View\Factory;

use Illuminate\Contracts\View\View;

use Illuminate\Foundation\Application;

use Illuminate\Http\JsonResponse;

use Illuminate\Http\Request;

class CategoryController extends Controller

{

    public function index(): View|Application|Factory|\Illuminate\Contracts\Foundation\Application

    {

        $categories = Category::orderBy('order\_index')->get();

        return view('admin.manage\_categories', compact('categories'));

    }

    public function getAllCategories(): array

    {

        return Category::all()->pluck('name', 'id')->toArray();

    }

    public function store(Request $request): JsonResponse

    {

        $validatedData = $request->validate([

            'name' => 'required|unique:categories|max:250',

            'subtitle' => 'nullable|max:250',

        ]);

        $maxOrderIndex = Category::max('order\_index');

        $category = Category::create([

            'name' => $validatedData['name'],

            'subtitle' => $validatedData['subtitle'],

            'order\_index' => $maxOrderIndex !== null ? $maxOrderIndex + 1 : 1,

        ]);

        return response()->json([

            'category' => $category,

            'message' => 'Категория успешно создана'

        ], 201);

    }

    public function checkUniqueName(Request $request): JsonResponse

    {

        $name = $request->input('name');

        $category = Category::where('name', $name)->first();

        if ($category && $category->id != $request->input('id')) {

            return response()->json([

                'unique' => false,

            ]);

        }

        return response()->json([

            'unique' => true

        ]);

    }

    public function update(Request $request, Category $category): JsonResponse

    {

        $validatedData = $request->validate([

            'name' => 'required|max:250|unique:categories,name,' . $category->getKey() . ',id',

            'subtitle' => 'nullable|max:250'

        ]);

        $category->update($validatedData);

        return response()->json([

            'category' => $category,

            'message' => 'Категория успешно обновлена'

        ], 200);

    }

    public function updateCategoryOrder(Request $request): JsonResponse

    {

        try {

            $categoriesOrder = $request->input('categoriesItems');

            foreach ($categoriesOrder as $index => $categoryId) {

                Category::where('id', $categoryId)->update(['order\_index' => $index + 1]);

            }

            return response()->json(['message' => 'Порядок категорий успешно обновлен'], 200);

        } catch (\Exception $e) {

            return response()->json(['error' => $e->getMessage()], 500);

        }

    }

    public function destroy(Category $category): JsonResponse

    {

        $category->delete();

        return response()->json(['message' => 'Категория успешно удалена'], 201);

    }

}

ProductController.php

<?php

namespace App\Http\Controllers\Admin;

use App\Http\Controllers\Controller;

use App\Models\Category;

use App\Models\Product;

use App\Models\ProductComposition;

use Illuminate\Support\Facades\File;

use Illuminate\Contracts\View\Factory;

use Illuminate\Contracts\View\View;

use Illuminate\Foundation\Application;

use Illuminate\Http\JsonResponse;

use Illuminate\Http\RedirectResponse;

use Illuminate\Http\Request;

use Illuminate\Support\Facades\Session;

use Illuminate\Support\Facades\Storage;

class ProductController extends Controller

{

    public function index(): View|Application|Factory|\Illuminate\Contracts\Foundation\Application

    {

        $categories = Category::orderBy('order\_index')->get();

        $products = Product::all();

        return view('admin.manage\_products', compact('categories', 'products'));

    }

    public function create(CategoryController $categoryController): View|Application|Factory|\Illuminate\Contracts\Foundation\Application

    {

        $categories = $categoryController->getAllCategories();

        Session::put('previous\_url', url()->previous());

        return view('admin.add\_product', compact('categories'));

    }

    public function store(Request $request): RedirectResponse

    {

        $validatedData = $request->validate([

            'name' => 'required|max:240',

            'price' => 'required|numeric|min:1700|max:1000000',

            'photos.\*' => 'required|image',

            'category\_id' => 'required|exists:categories,id'

        ]);

        $article = mt\_rand(101000, 189999);

        $product = Product::create([

            'name' => $validatedData['name'],

            'price' => $validatedData['price'],

            'article' => $article,

            'category\_id' => $validatedData['category\_id'],

        ]);

        foreach ($request->file('photos') as $photo) {

            if ($photo->isValid()) {

                $imageName = time() . '\_' . $photo->getClientOriginalName();

                $photo->storeAs('public/photos', $imageName);

                $product->images()->create([

                    'name' => $imageName,

                    'path' => 'photos/' . $imageName,

                ]);

            } else {

                return back()->with('error', 'Некорректный файл изображения');

            }

        }

        if ($request->has('composition')) {

            $compositionData = json\_decode($request->input('composition'), true);

            if ($compositionData) {

                $dataToInsert = [];

                foreach ($compositionData as $data) {

                    $dataToInsert[] = [

                        'name' => $data['name'],

                        'quantity' => $data['quantity'],

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

                    ];

                }

                ProductComposition::insert($dataToInsert);

            }

        }

        return redirect(Session::get('previous\_url'));

    }

    public function showByCategory($category\_id): View|Application|Factory|\Illuminate\Contracts\Foundation\Application

    {

        Session::put('previous\_url', url()->current());

        $products = Product::where('category\_id', $category\_id)->get();

        $categories = Category::orderBy('order\_index')->get();

        $currentCategory = Category::find($category\_id);

        return view('admin.category\_page', compact('products', 'categories', 'currentCategory'));

    }

    public function show(string $id): View|Application|Factory|\Illuminate\Contracts\Foundation\Application

    {

        $product = Product::findOrFail($id);

        $currentCategory = $product->category;

        $currentComposition = $product->composition;

        return view('product\_detail', compact('product', 'currentCategory', 'currentComposition'));

    }

    public function destroy(string $id): JsonResponse

    {

        $product = Product::findOrFail($id);

        if ($product) {

            foreach ($product->images as $image) {

                Storage::delete('public/photos/' . $image->name);

            }

            $product->delete();

            return response()->json(['message' => 'Продукт был успешно удален']);

        } else {

            return response()->json(['message' => 'Продукт не был найден'], 404);

        }

    }

    public function getAllProductsCategory($category\_id): JsonResponse

    {

        $products = Product::with('images')

            ->where('category\_id', $category\_id)

            ->get();

        $formattedProducts = $products->map(function ($product) {

            $imagePath = null;

            if ($product->images->isNotEmpty()) {

                $imagePath = $product->images->first()->path;

            }

            return [

                'id' => $product->id,

                'name' => $product->name,

                'price' => $product->price,

                'description' => $product->description,

                'article' => $product->article,

                'category\_id' => $product->category\_id,

                'image\_path' => $imagePath,

            ];

        });

        return response()->json(['products' => $formattedProducts]);

    }

    public function searchProductsInCategory(Request $request, $category\_id): JsonResponse

    {

        $query = $request->input('query');

        $productsQuery = Product::with('images')

            ->where('category\_id', $category\_id)

            ->where(function ($queryBuilder) use ($query) {

                $queryBuilder->where('name', 'like', "%$query%")

                    ->orWhere('description', 'like', "%$query%");

            });

        $products = $query ? $productsQuery->get() : collect([]);

        $formattedProducts = $products->map(function ($product) {

            $imagePath = null;

            if ($product->images->isNotEmpty()) {

                $imagePath = $product->images->first()->path;

            }

            return [

                'id' => $product->id,

                'name' => $product->name,

                'price' => $product->price,

                'description' => $product->description,

                'article' => $product->article,

                'category\_id' => $product->category\_id,

                'image\_path' => $imagePath,

            ];

        });

        return response()->json(['products' => $formattedProducts]);

    }

}

UserController.php

<?php

namespace App\Http\Controllers\Admin;

use App\Http\Controllers\Controller;

use App\Http\Requests\StoreUserRequest;

use App\Http\Requests\UpdateUserRequest;

use App\Models\User;

use Illuminate\Contracts\View\Factory;

use Illuminate\Contracts\View\View;

use Illuminate\Foundation\Application;

use Illuminate\Http\JsonResponse;

use Illuminate\Http\RedirectResponse;

use Illuminate\Http\Request;

use Illuminate\Support\Facades\App;

use Illuminate\Support\Facades\Hash;

use Illuminate\Support\Facades\Log;

class UserController extends Controller

{

    public function index(): View|Application|Factory|\Illuminate\Contracts\Foundation\Application

    {

        $users = User::query();

        $roles = User::distinct()->pluck('role');

        $users = $users->get();

        return view('admin.manage\_users', compact('users', 'roles'));

    }

    public function getAllUsers(): JsonResponse

    {

        $users = User::all();

        return response()->json(['users' => $users]);

    }

    public function search(Request $request): JsonResponse

    {

        $query = $request->input('query');

        $usersQuery = User::query()

            ->where('login', 'like', "%$query%")

            ->orWhere('name', 'like', "%$query%")

            ->orWhere('email', 'like', "%$query%");

        $users = $query ? $usersQuery->get() : collect([]);

        return response()->json(['users' => $users]);

    }

    public function store(StoreUserRequest $request): JsonResponse

    {

        $validatedData = $request->validate([

            'login' => 'required|string|min:5|max:60|unique:users|regex:/^[a-zA-Z0-9\_]+$/',

            'name' => 'required|string|min:2|max:50|regex:/^[A-Za-zА-Яа-яЁё\-]+$/u',

            'email' => 'required|email|unique:users,email',

            'password' => 'required|string|min:8|max:60',

        ]);

        $validatedData['password'] = Hash::make($validatedData['password']);

        $validatedData['role'] = $request->input('role', 'USER');

        $user = User::create($validatedData);

        return response()->json(['message' => 'Пользователь успешно добавлен', 'user' => $user]);

    }

    public function edit(User $user): View|Factory|Application

    {

        return view('admin.edit\_user', compact('user'));

    }

    public function update(UpdateUserRequest $request, User $user): RedirectResponse

    {

        $validatedData = $request->validated();

        $user->update($validatedData);

        return redirect()->route('admin.users.index')->with('success', 'Пользователь успешно обновлен');

    }

    public function destroy(User $user): JsonResponse

    {

        $user->delete();

        return response()->json(['message' => 'Пользователь успешно удален']);

    }

}

Index.blade.php

@include('fragments/head', ['title' => 'Доставка цветов | Купить букет цветов с доставкой в Белореченск'])

<body class="body">

<header class="header header--main">

    @include('fragments/header\_top')

    <div class="header\_\_content">

        <h1 class="header\_\_title">

            Загляните в Herbalia <br>

            Таинственный мир цветов

        </h1>

        <a class="header\_\_link" href="/catalog">Перейти в каталог</a>

    </div>

</header>

<section class="products indent">

    <div class="container">

        <div class="products\_\_header">

            <h2 class="products\_\_title title">Доставка цветов в любую точку</h2>

            <ul class="products\_\_benefits" aria-label="Наши преимущества">

                <li class="products\_\_benefits-item">Срочная доставка. Бесплатно</li>

                <li class="products\_\_benefits-item">Гарантия свежести</li>

                <li class="products\_\_benefits-item">Фото букета перед доставкой</li>

                <li class="products\_\_benefits-item">Бесплатная открытка</li>

            </ul>

        </div>

        <ul class="products\_\_items">

            @foreach($categories as $category)

                @if(!$categoryProducts[$category->id]->isEmpty())

                    <li class="products-row\_\_header">

                        <h3 class="products-row\_\_title">

                            {{ $category->name }}

                        </h3>

                        <span class="products-row\_\_extra-wrapper">

                    <span class="products-row\_\_subtitle">{{ $category->subtitle }}</span>

                    <span class="products-row\_\_extra">

                        <span class="products\_\_row-header\_\_extra-item products\_\_row-header\_\_extra-item--type-price">

                            от {{ number\_format($categoryProducts[$category->id]->min('price'), 0, '.', ' ') }} ₽

                        </span>

                        <span class="products\_\_row-header\_\_extra-item products\_\_row-header\_\_extra-item--type-count">

                            {{ $categoryProducts[$category->id]->count() }} шт

                        </span>

                    </span>

                </span>

                        <ul class="products\_\_list">

                            @foreach($categoryProducts[$category->id]->take(7) as $product)

                                <li class="products\_\_card">

                                    @foreach($product->images as $image)

                                        @if($loop->first)

                                            <a class="products\_\_link"

                                               href="{{ route('product.show', ['id' => $product->id]) }}">

                                                <img class="products\_\_img"

                                                     src="{{ asset('storage/' . $image->path) }}" height="360"

                                                     alt="{{ $product->name }}">

                                            </a>

                                        @endif

                                    @endforeach

                                    <div class="products\_\_head">

                                        <a class="products\_\_name"

                                           href="{{ route('product.show', ['id' => $product->id]) }}">{{ $product->name }}</a>

                                        <button class="products\_\_like" type="button"></button>

                                    </div>

                                    <p class="products\_\_delivery">

                                        Бесплатная доставка <span>с 10:00 до 23:00</span>

                                    </p>

                                    <span class="products\_\_price">{{ number\_format($product->price, 0, '.', ' ') }} ₽</span>

                                    <button class="products\_\_order" type="button">Заказать</button>

                                </li>

                            @endforeach

                            <li class="products\_\_card">

                                <a class="products\_\_more" href="{{ route('contacts')}}">

                                    <div class="products\_\_circle">

                                        <svg width="45" height="34" viewBox="0 0 18 15" fill="none"

                                             xmlns="http://www.w3.org/2000/svg">

                                            <path d="M1 7.5H17M17 7.5L11 1M17 7.5L11 14" stroke="#3d1522"

                                                  stroke-width="1.2"

                                                  stroke-linecap="round" stroke-linejoin="round"/>

                                        </svg>

                                    </div>

                                    Показать больше

                                </a>

                            </li>

                        </ul>

                    </li>

                @endif

            @endforeach

        </ul>

    </div>

</section>

@include('home/top\_reviews')

@include('home/popular\_questions')

@include('home/original')

@include('home/possibilities')

@include('home/delivery')

@include('home/guarantees')

@include('fragments/contacts')

@include('fragments/footer')

@include('fragments/modals\_header')

@vite(['resources/js/app.js'])

@vite(['resources/js/main.js'])

</body>

Login.blade.php

@include('fragments/head', ['title' => 'Авторизация'])

<body class="body">

<section class="auth">

    <div class="container">

        <div class="auth\_\_content">

            <a class="auth\_\_logo logo" href="/">

                <img class="logo" width="226" height="52" src="{{asset('static/images/icons/auth-logo.svg')}}"

                     alt="логотип">

            </a>

            <a class="auth\_\_back" href="/">

                Вернуться на главную

            </a>

            <h2 class="auth\_\_title">Вход в профиль</h2>

            <p class="auth\_\_text">Вы увидите историю заказов, будете иметь возможность оставлять отзывы и т.д.</p>

            <form method="POST" action="{{ route('login') }}">

                @csrf

                <span class="error @if($errors->has('login')) error--active @endif" id="authError">

                    {{ $errors->first('login') ?? 'Неверно введен логин или пароль' }}

                </span>

                <ul class="auth\_\_list">

                    <li class="auth\_\_item">

                        <label class="label" for="login">Логин</label>

                        <input class="input" name="login" id="login" type="text" placeholder="Введите логин"

                               value="{{ old('login') }}" required>

                    </li>

                    <li class="auth\_\_item">

                        <label class="label" for="password">Пароль</label>

                        <input class="input" id="password" type="password" placeholder="Введите пароль" name="password"

                               value="{{ old('password') }}" required>

                    </li>

                </ul>

                <span class="auth\_\_link">

                    Нет аккаунта? <a href="{{route('register')}}">Тогда зарегистрируйтесь</a>

                </span>

                <button class="auth\_\_btn" id="loginBtn" type="submit">Войти</button>

            </form>

        </div>

        <div class="auth\_\_decor"></div>

    </div>

</section>

<script src="https://cdnjs.cloudflare.com/ajax/libs/jquery/3.7.1/jquery.min.js"

        integrity="sha512-v2CJ7UaYy4JwqLDIrZUI/4hqeoQieOmAZNXBeQyjo21dadnwR+8ZaIJVT8EE2iyI61OV8e6M8PP2/4hpQINQ/g=="

        crossorigin="anonymous" referrerpolicy="no-referrer"></script>

</body>

Register.blade.php

@include('fragments/head', ['title' => 'Регистрация'])

<body class="body">

<section class="auth">

    <div class="container">

        <div class="auth\_\_content">

            <a class="auth\_\_logo logo" href="/">

                <img class="logo" width="226" height="52" src="{{asset('static/images/icons/auth-logo.svg')}}"

                     alt="логотип">

            </a>

            <a class="auth\_\_back" href="/">

                Вернуться на главную

            </a>

            <h2 class="auth\_\_title">Регистрация профиля</h2>

            <form method="POST" action="{{route('register')}}">

                @csrf

                <ul class="auth\_\_list">

                    <li class="auth\_\_item">

                        <label class="label" for="login">Логин</label>

                        <span class="error" id="loginError">Логин не должен иметь запрещенные символы</span>

                        <span class="error" id="loginCheckError">Логин уже используется</span>

                        <span class="error" id="loginLengthError">Минимальное количество символов - 5</span>

                        <span class="error" id="loginMaxError">Максимальное количество символов - 60</span>

                        <input class="input" id="login" type="text" name="login" placeholder="Введите ваш логин">

                    </li>

                    <li class="auth\_\_item">

                        <label class="label" for="name">Имя</label>

                        <span class="error" id="nameMinError">Мин. количество символов - 2</span>

                        <span class="error" id="nameMaxError">Макс. количество символов - 50</span>

                        <span class="error" id="nameError">Имя не должно содержать запрещенные символы</span>

                        <input class="input" id="name" type="text" name="name" placeholder="Введите ваше имя"

                               value="{{ old('name') }}">

                    </li>

                    <li class="auth\_\_item">

                        <label class="label" for="email">Email</label>

                        <span class="error" id="emailError">Почта уже используется</span>

                        <span class="error" id="emailErrorParameters">Почта не соответствует параметрам</span>

                        <span class="error" id="emailLengthError">Макс. количество символов - 80</span>

                        <input class="input" id="email" type="email" name="email" placeholder="Введите ваш email"

                               value="{{ old('email') }}">

                    </li>

                    <li class="auth\_\_item">

                        <label class="label" for="password">Пароль</label>

                        <span class="error" id="passwordError">Пароль не должен иметь кириллицу</span>

                        <span class="error" id="passwordLengthError">Минимальное количество символов - 8</span>

                        <span class="error" id="passwordMaxError">Максимальное количество символов - 60</span>

                        <input class="input" id="password" type="password" placeholder="Введите пароль" name="password">

                    </li>

                </ul>

                <label class="auth\_\_checkbox custom-checkbox">

                    <input class="custom-checkbox\_\_input" type="checkbox" checked name="" value="">

                    <span class="checkmark"></span>

                    Я согласен с <a href="{{route('agreement')}}">пользовательским соглашением</a>

                </label>

                <span class="auth\_\_link">

                    Есть аккаунт? <a href="{{route('login')}}">Войти</a>

                </span>

                <button class="auth\_\_btn" id="registration-btn" type="submit">Зарегистрироваться</button>

            </form>

        </div>

        <div class="auth\_\_decor"></div>

    </div>

</section>

<script src="https://cdnjs.cloudflare.com/ajax/libs/jquery/3.7.1/jquery.min.js"

        integrity="sha512-v2CJ7UaYy4JwqLDIrZUI/4hqeoQieOmAZNXBeQyjo21dadnwR+8ZaIJVT8EE2iyI61OV8e6M8PP2/4hpQINQ/g=="

        crossorigin="anonymous" referrerpolicy="no-referrer"></script>

@vite(['resources/js/register.js'])

</body>

Reviews.blade.php

@php use Carbon\Carbon; @endphp

@include('fragments/head', ['title' => 'Отзывы клиентов интернет магазина Herbalia'])

<body class="body">

<header class="header header--not-main">

    @include('fragments/header\_top')

</header>

<section class="reviews static indent indent--breadcrumbs indent--footer">

    <div class="container">

        <ul class="breadcrumbs">

            <li class="breadcrumbs\_\_item">

                <a class="breadcrumbs\_\_link breadcrumbs\_\_link--nav" href="{{route('index')}}">

                    <span>

                        Доставка цветов в Белореченск

                    </span>

                </a>

            </li>

            <li class="breadcrumbs\_\_item">

                <span class="breadcrumbs\_\_link active">Отзывы</span>

            </li>

        </ul>

        <div class="reviews\_\_top">

            <h1 class="static\_\_title title">Отзывы</h1>

            <button class="reviews\_\_rules" id="btnRules">

                Правила публикации отзывов

            </button>

        </div>

        <div class="reviews\_\_rating">

            <img class="reviews\_\_img" src="{{asset('static/images/reviews-right.svg')}}"

                 alt="декор отзывы">

            <div class="reviews\_\_rating-points">

                <p class="reviews\_\_amount">{{$averageRating}}</p>

                <h6 class="reviews\_\_label">Наш рейтинг</h6>

            </div>

            <img class="reviews\_\_img reviews\_\_img--right" src="{{asset('static/images/reviews-right.svg')}}"

                 alt="декор отзывы">

        </div>

        <div class="reviews\_\_actions">

            @auth

                <a class="reviews\_\_actions-link" href="{{route('reviews-form')}}">Написать отзыв</a>

                <a class="reviews\_\_actions-link" href="">Предложить улучшения</a>

            @endauth

            @guest

                <p class="reviews\_\_actions-text">

                    Вы не можете оставлять отзыв

                    <br>

                    Чтоб сделать это - <a href="{{route('login')}}">войдите</a> или <a href="{{route('register')}}">зарегистрируйтесь</a>

                </p>

            @endguest

        </div>

        <div class="reviews\_\_inner @if(count($reviews) === 0)reviews\_\_inner--empty @endif">

            @if(count($reviews) === 0)

                <p>Пока отзывов нет ¯\\_(ツ)\_/¯</p>

            @else

                <div class="reviews\_\_left">

                <span class="reviews\_\_quantity">

                    @if(count($reviews) === 0)

                        Пока нет отзывов

                    @elseif(count($reviews) === 1)

                        Всего 1 отзыв

                    @elseif((count($reviews) % 10 === 2 || count($reviews) % 10 === 3 || count($reviews) % 10 === 4) && (count($reviews) < 10 || count($reviews) > 20))

                        Всего {{ count($reviews) }} отзыва

                    @else

                        Всего {{ count($reviews) }} отзывов

                    @endif

                </span>

                </div>

                <div class="reviews\_\_content">

                    <ul class="reviews\_\_list">

                        @foreach($reviews as $review)

                            <li class="reviews\_\_item">

                                <time class="reviews\_\_date">

                                    {{Carbon::parse($review->created\_at)->locale('ru')->isoFormat('D MMMM YYYY г.')}}

                                </time>

                                <span class="reviews\_\_name">{{$review->name}}</span>

                                <p class="reviews\_\_text">{{$review->comment}}</p>

                                <span class="reviews\_\_rate">

                                    Оценка: {{ $review->rating }}

                                    @if ($review->rating == 5)

                                        😁

                                    @elseif ($review->rating == 4)

                                        😄

                                    @elseif ($review->rating == 3)

                                        😊

                                    @elseif ($review->rating == 2)

                                        😐

                                    @else

                                        😞

                                    @endif

                                </span>

                            </li>

                        @endforeach

                    </ul>

                </div>

            @endif

        </div>

    </div>

</section>

<div class="modal modal--reviews" id="modalRules">

    <button class="modal\_\_close" id="modalCloseRules" type="button"></button>

    <h3 class="modal\_\_title reviews\_\_title">Правила публикации отзывов</h3>

    <div class="modal\_\_content">

        <p>

            Мы всегда рады вашим отзывам и предложениям по улучшению.

            <br>

            Вместе мы делаем отличный, удобный и надёжный сервис для Всех!

        </p>

        <p>

            Ниже мы хотели бы отметить основные правила публикации отзывов:

            <br>

            1. Отзывы от наших клиентов (дарителей и получателей);

            <br>

            2. Объективные плюсы и минусы;

            <br>

            3. По выполненным заказам;

        </p>

        <p>

            Если у вас рекламация, то мы рассчитываем на объективность с обоих сторон.

        </p>

        <p>

            Помните, что лучше сразу написать нам или позвонить.

            <br>

            И мы оперативно решим любую возникшую проблему.

        </p>

    </div>

</div>

@include('fragments/footer')

@vite(['resources/js/reviews/main.js?type=module'])

@vite(['resources/js/app.js'])

</body>

Reviews-form.blade.php

@include('fragments/head', ['title' => 'Написать отзыв'])

<body class="body">

<header class="header header--not-main">

    @include('fragments/header\_top')

</header>

<section class="reviews static indent indent--breadcrumbs indent--footer">

    <div class="container">

        <ul class="breadcrumbs">

            <li class="breadcrumbs\_\_item">

                <a class="breadcrumbs\_\_link breadcrumbs\_\_link--nav" href="{{route('index')}}">

                    <span>

                        Доставка цветов в Белореченск

                    </span>

                </a>

            </li>

            <li class="breadcrumbs\_\_item">

                <a class="breadcrumbs\_\_link breadcrumbs\_\_link--nav" href="{{route('reviews')}}">

                    <span>

                        Отзывы

                    </span>

                </a>

            </li>

            <li class="breadcrumbs\_\_item">

                <span class="breadcrumbs\_\_link active">Написать отзыв</span>

            </li>

        </ul>

        <h1 class="static\_\_title title">Написать отзыв</h1>

        <form class="reviews-form" action="" method="POST">

            @csrf

            <div class="reviews-form\_\_items">

                <div class="reviews-form\_\_item">

                    <label class="label" for="name">Имя:</label>

                    <span class="error" id="nameError">Имя не должно содержать запрещенные символы</span>

                    <span class="error" id="nameMinError">Мин. количество символов - 2</span>

                    <span class="error" id="nameMaxError">Макс. количество символов - 50</span>

                    <input class="input" type="text" id="name" value="{{old('name') ?? auth()->user()->name}}"

                           name="name">

                </div>

                <div class="reviews-form\_\_item">

                    <label class="label" for="email">Почта:</label>

                    <span class="error" id="emailCheckError">Почта уже используется</span>

                    <span class="error" id="emailErrorParameters">Почта не соответствует параметрам</span>

                    <span class="error" id="emailLengthError">Макс. количество символов - 80</span>

                    <input class="input" type="email" id="email" value="{{old('email') ?? auth()->user()->email}}"

                           name="email">

                </div>

            </div>

            <div class="reviews-form\_\_rating">

                <span>Вы остались довольны сервисом?</span>

                <label class="label" for="rating">Оценка:</label>

                <input type="radio" id="rating1" name="rating" value="1">

                <label class="reviews-form\_\_emoji" for="rating1">😞</label>

                <input type="radio" id="rating2" name="rating" value="2">

                <label class="reviews-form\_\_emoji" for="rating2">😐</label>

                <input type="radio" id="rating3" name="rating" value="3">

                <label class="reviews-form\_\_emoji" for="rating3">😊</label>

                <input type="radio" id="rating4" name="rating" value="4">

                <label class="reviews-form\_\_emoji" for="rating4">😄</label>

                <input type="radio" id="rating5" name="rating" value="5" checked>

                <label class="reviews-form\_\_emoji" for="rating5">😁</label>

            </div>

            <div class="reviews-form\_\_comment">

                <label class="label" for="comment">Ваш отзыв:</label>

                <textarea class="reviews-form\_\_textarea input input--textarea" id="comment" name="comment"

                          required></textarea>

            </div>

            <div class="reviews-form\_\_accept">

                <button class="reviews-form\_\_btn" type="submit">Отправить</button>

                <p class="reviews-form\_\_link">

                    Нажимая кнопку «Отправить», я соглашаюсь с

                    <a href="{{asset(route('agreement'))}}" target="\_blank">Политикой публикации отзывов</a>

                </p>

            </div>

        </form>

    </div>

</section>

<div class="modal" id="modalComplete">

    <h3 class="modal\_\_title reviews\_\_title">Отзыв отправлен на проверку</h3>

    <div class="modal\_\_content">

        <p>

            Ваш отзыв был успешно отправлен на проверку модераторам магазинам

        </p>

        <a class="modal\_\_btn modal\_\_btn--complete" href="{{route('reviews')}}">Принято</a>

    </div>

</div>

@include('fragments/footer')

@vite(['resources/js/reviews/add-review.js?type=module'])

@vite(['resources/js/app.js'])

</body>

Add\_product.blade.php

@include('fragments/head', ['title' => 'Создание товара'])

<body class="body">

<header class="admin\_\_header">

    @include('fragments/header\_admin')

</header>

<section class="admin">

    <div class="container container--admin">

        <form class="admin-add\_\_form" action="{{route('admin.products.store')}}" method="POST"

              enctype="multipart/form-data">

            <input type="hidden" name="selected\_photos" id="selectedPhotos">

            <input type="hidden" name="composition" id="hiddenComposition">

            <h1 class="admin\_\_title admin\_\_title--bottom">Создание товара</h1>

            @csrf

            <div class="admin-add\_\_content">

                <ul class="admin-add\_\_items">

                    <li class="admin-add\_\_row">

                        <label class="label" for="name">Название товара</label>

                        <span class="error" id="errorName">Макс. количество символов - 240</span>

                        <input class="input" id="name" name="name" type="text" required placeholder="Введите название">

                    </li>

                    <li class="admin-add\_\_row">

                        <label class="label" for="price">Цена (в ₽)</label>

                        <span class="error" id="errorMinPrice">Мин. цена товара - 1 700₽</span>

                        <span class="error" id="errorMaxPrice">Макс. цена товара - 1 000 000₽</span>

                        <input class="input" id="price" name="price" type="number" required placeholder="Введите цену">

                    </li>

                    <li class="admin-add\_\_row">

                        <label class="label" for="photo">Выберите категорию</label>

                        <select class="select" name="category\_id" id="category\_id">

                            @foreach($categories as $id => $name)

                                <option value="{{$id}}">{{$name}}</option>

                            @endforeach

                        </select>

                    </li>

                </ul>

                <div class="admin-add\_\_files">

                    <label class="label" for="photo">Фотографии товара</label>

                    <span class="error error\_\_files" id="photosError">

                        Ваш файл имеет неверный формат. Пожалуйста, выберите изображение в формате PNG, WebP, JPEG или JPG.

                    </span>

                    <span class="error error\_\_files" id="photosLimitMinError">

                        Минимальное количество фото - 1

                    </span>

                    <span class="error error\_\_files" id="photosLimitMaxError">

                        Максимальное количество фото - 4

                    </span>

                    <input type="file" name="photos[]" id="fileInput" multiple style="display: none;"

                           accept="image/png, image/webp, image/jpeg, image/jpg">

                    <label class="custom-file\_\_upload" for="fileInput">

                        Загрузить

                    </label>

                    <ul class="custom-file\_\_items" id="uploadedImages"></ul>

                </div>

                <div class="admin-add\_\_composition">

                    <button class="admin-add\_\_plus" type="button">+ Добавить элемент</button>

                    <ul class="admin-add\_\_composition-list" id="compositionContainer">

                        <li class="admin-add\_\_row" id="firstRow">

                            <div class="admin-add\_\_errors">

                                <span class="error">Макс. количество символов - 240</span>

                                <span class="error">Мин. число - 1</span>

                                <span class="error">Макс. число - 1000</span>

                            </div>

                            <div class="admin-add\_\_row-content">

                                <div class="admin-add\_\_column">

                                    <label for="elementName\_1">Название</label>

                                    <input class="input" type="text" id="elementName\_1" placeholder="Элемент состава"

                                           required>

                                </div>

                                <div class="admin-add\_\_column admin-add\_\_column--quantity">

                                    <label for="elementQuantity\_1">Кол-во</label>

                                    <input class="admin-add\_\_quantity input" type="number" id="elementQuantity\_1"

                                           value="1" required placeholder="Кол-во">

                                </div>

                            </div>

                        </li>

                    </ul>

                </div>

            </div>

            <button class="modal\_\_btn" type="submit" id="submitButton">Создать товар</button>

        </form>

    </div>

</section>

@vite(['resources/js/products/add-product.js?type=module'])

@vite(['resources/js/components/custom-select.js'])

</body>

Category\_page.blade.php

@include('fragments/head', ['title' => $currentCategory ? $currentCategory->name : ''])

<body class="body">

<header class="admin\_\_header">

    @include('fragments/header\_admin')

</header>

<section class="admin">

    <div class="container container--admin">

        <div class="admin\_\_wrapper">

            <h1 class="admin\_\_title">Управление категорией</h1>

            <ul class="admin\_\_breadcrumbs breadcrumbs">

                <li class="breadcrumbs\_\_item">

                    <a class="breadcrumbs\_\_link breadcrumbs\_\_link--nav" href="{{route('admin.products.index')}}">

                        <span>Управление товарами</span>

                    </a>

                </li>

                <li class="breadcrumbs\_\_item">

                    <span class="breadcrumbs\_\_link active">{{$currentCategory->name}}</span>

                </li>

            </ul>

            <ul class="admin-products\_\_buttons">

                @foreach($categories as $category)

                    <li class="admin-products\_\_item">

                        <a class="admin-products\_\_link @if($category->id === $currentCategory->id)admin-products\_\_link--active @endif"

                           href="{{route('admin.products.category', ['category\_id' => $category->id])}}">{{$category->name}}

                        </a>

                    </li>

                @endforeach

            </ul>

            @if(!$products->isEmpty())

                <a class="admin\_\_btn admin\_\_btn--categories" href="{{route('admin.products.create')}}">

                    Добавить товар

                </a>

                <form class="admin-products\_\_search"

                      action="{{ route('admin.products.category.search', ['category\_id' => $currentCategory->id]) }}"

                      method="GET"

                      id="productSearchForm"

                      data-search-url="{{ route('admin.products.category.search', ['category\_id' => $currentCategory->id]) }}">

                    <input type="text" name="query" placeholder="Введите данные из имени или описания товара">

                </form>

            @endif

            @if($products->isEmpty())

                <div class="admin-products\_\_empty">

                    <img class="admin-products\_\_decor" src="{{asset('static/images/icons/flowers.svg')}}" width="120"

                         alt="декор">

                    <span class="admin-products\_\_title">В категории пока пусто ¯\\_(ツ)\_/¯</span>

                    <p class="admin-products\_\_text">Товары еще не были созданы, но вы можете создать их</p>

                    <a class="admin-products\_\_btn admin\_\_btn admin\_\_btn--categories"

                       href="{{route('admin.products.create')}}">Добавить товар</a>

                </div>

            @else

                <ul class="admin-products\_\_list" id="productsListContainer">

                    @foreach($products as $product)

                        <li class="admin-products\_\_card" data-product-id="{{$product->id}}">

                            <div class="admin-products\_\_head">

                                <div class="admin-products\_\_actions">

                                    <a class="admin-products\_\_action admin-products\_\_action--view"

                                       href="{{route('product.show', ['id' => $product->id])}}" target="\_blank">

                                    </a>

                                    <button class="admin-products\_\_action admin-products\_\_action--delete"

                                            data-product-id="{{$product->id}}"></button>

                                </div>

                                @foreach($product->images as $image)

                                    @if($loop->first)

                                        <img class="admin-products\_\_img" src="{{ asset('storage/' . $image->path) }}"

                                             height="360" alt="{{$product->name}}" data-image-path="{{$image->path}}">

                                    @endif

                                @endforeach

                            </div>

                            <p class="admin-products\_\_price">

                                Цена: {{ number\_format($product->price, 0, '.', ' ') }} ₽

                            </p>

                            <h3 class="admin-products\_\_title">{{$product->name}}</h3>

                            <p class="admin-products\_\_article">Артикул - {{$product->article}}</p>

                            <p>Описание: <br>{{$product->description}}</p>

                        </li>

                    @endforeach

                </ul>

            @endif

        </div>

    </div>

</section>

<div class="modal" id="modalDeleteProduct">

    <button class="modal\_\_close" id="modalCloseDeleteProduct" type="button"></button>

    <h3 class="modal\_\_title">Удаление товара</h3>

    <p class="modal\_\_text">

        Вы действительно хотите удалить товар <span id="nameDeleteProduct"></span>? Удаление будет

        невозможно отменить, все привязанные фотографии к товару будут удалены

    </p>

    <div class="modal\_\_buttons">

        <button class="modal\_\_btn modal\_\_btn--cancel" id="cancelDeleteProduct" type="button">Отменить</button>

        <button class="modal\_\_btn modal\_\_btn--confirm" id="confirmDeleteProduct" type="submit">Да, удалить</button>

    </div>

</div>

@vite(['resources/js/products/products.js?type=module'])

</body>

Edit\_user.blade.php

@include('fragments/head', ['title' => 'Редактирование пользователя'])

<body class="body">

<header class="admin\_\_header">

    @include('fragments/header\_admin')

</header>

<section class="admin">

    <div class="container container--admin">

        <div class="admin\_\_wrapper">

            <h1 class="admin\_\_title">Редактирование пользователя</h1>

            <form id="updateForm" action="{{ route('admin.users.update', ['user' => $user]) }}" method="POST">

                @csrf

                @method('PUT')

                <div class="admin-edit\_\_list">

                    <div class="admin-edit\_\_item">

                        <label for="login">Логин</label>

                        <span class="error" id="loginCheckError">Логин уже используется</span>

                        <span class="error" id="loginError">Логин не должен иметь запрещенные символы</span>

                        <span class="error" id="loginLengthError">Минимальное количество символов - 5</span>

                        <span class="error" id="loginMaxError">Максимальное количество символов - 60</span>

                        <input class="input" type="text" id="login" name="login" value="{{$user->login}}" required>

                    </div>

                    <div class="admin-edit\_\_item">

                        <label for="name">Имя</label>

                        <span class="error" id="nameError">Имя не должно содержать запрещенные символы</span>

                        <span class="error" id="nameMinError">Мин. количество символов - 2</span>

                        <span class="error" id="nameMaxError">Макс. количество символов - 50</span>

                        <input class="input" type="text" id="name" name="name" value="{{$user->name}}" required>

                    </div>

                    <div class="admin-edit\_\_item">

                        <label for="email">Почта</label>

                        <span class="error" id="emailCheckError">Почта уже используется</span>

                        <span class="error" id="emailErrorParameters">Почта не соответствует параметрам</span>

                        <span class="error" id="emailLengthError">Макс. количество символов - 80</span>

                        <input class="input" type="email" id="email" name="email" value="{{$user->email}}" required>

                    </div>

                    <div class="admin-edit\_\_item">

                        <label for="role">Роль</label>

                        <select class="select" id="role" name="role">

                            <option value="USER" {{ $user->role === 'USER' ? 'selected' : '' }}>Пользователь</option>

                            <option value="ADMIN" {{ $user->role === 'ADMIN' ? 'selected' : '' }}>Администратор</option>

                        </select>

                    </div>

                </div>

                <button class="modal\_\_btn" type="submit" id="updateButton">Обновить</button>

            </form>

        </div>

    </div>

</section>

</body>

@vite(['resources/js/components/custom-select.js'])

@vite(['resources/js/user/edit.js?type=module'])

@vite(['resources/js/user.js?type=module'])

Manage\_categories.blade.php

@include('fragments/head', ['title' => 'Управление категориями'])

<body class="body">

<header class="admin\_\_header">

    @include('fragments/header\_admin')

</header>

<section class="admin">

    <div class="container container--admin">

        <div class="admin\_\_wrapper">

            <h1 class="admin\_\_title">Управление категориями</h1>

            <button class="admin\_\_btn" id="addBtnCategory" type="button">Добавить категорию</button>

            @if($categories->isEmpty())

                <p class="admin-category\_\_info">Категорий еще нет, но вы можете создать их</p>

            @else

                <div class="admin-warning">

                <span class="admin-warning\_\_icon">

                    <img width="26" height="26" src="{{asset('static/images/icons/info.svg')}}" alt="предупреждение">

                </span>

                    <div>

                        <span class="admin-warning\_\_title">Прочтите это</span>

                        Если вы хотите изменить порядок элементов, то просто перетаскивайте их.

                        Изменения вступят в силу сразу после одного перетаскивания

                    </div>

                </div>

                <span class="admin-category\_\_info">Ваши категории:</span>

                <ul class="admin-category\_\_list" id="sortableList">

                    @foreach($categories as $category)

                        <li class="admin-category\_\_item" data-category-id="{{$category->id}}">

                            <div class="admin-category\_\_content">

                                <span class="admin-category\_\_index">Номер в порядке - {{$category->order\_index}}</span>

                                <h3 class="admin-category\_\_name">{{ $category->name }}</h3>

                                @if($category->subtitle)

                                    <p class="admin-category\_\_text">{{ $category->subtitle }}</p>

                                @endif

                            </div>

                            <div class="admin-category\_\_actions">

                                <button class="admin-category\_\_action admin-category\_\_delete"

                                        data-category-id="{{$category->id}}"></button>

                                <button class="admin-category\_\_action admin-category\_\_edit"

                                        data-category-id="{{$category->id}}"

                                        data-category-name="{{$category->name}}"

                                        data-category-subtitle="{{$category->subtitle}}">

                                </button>

                            </div>

                        </li>

                    @endforeach

                    @endif

                </ul>

        </div>

    </div>

</section>

<div class="modal" id="modalAddCategory" data-modal="modalAddCategory">

    <button class="modal\_\_close" id="modalCloseAddCategory" type="button"></button>

    <h3 class="modal\_\_title" id="modalTitle">Создание категории</h3>

    <form action="{{ route('admin.categories.store') }}" id="categoryForm" method="POST">

        @csrf

        <ul class="modal\_\_list">

            <li class="modal\_\_item">

                <label class="label" for="name">Название категории</label>

                <span class="error" id="nameUniqueError">Имя должно быть уникальным</span>

                <span class="error" id="nameMaxError">Максимальное количество символов в имени - 250</span>

                <input class="input" type="text" id="name" name="name" required>

            </li>

            <li class="modal\_\_item">

                <label class="label" for="subtitle">Подзаголовок (необязательно)</label>

                <span class="error" id="subtitleMaxError">Максимальное количество символов в подзаголовке - 250</span>

                <input class="input" type="text" id="subtitle" name="subtitle">

            </li>

        </ul>

        <button class="modal\_\_btn" type="submit">Создать категорию</button>

    </form>

</div>

<div class="modal" id="modalEditCategory" data-modal="modalEditCategory">

    <button class="modal\_\_close" id="modalCloseEditCategory" type="button"></button>

    <h3 class="modal\_\_title" id="modalTitle">Редактирование категории</h3>

    <form action="{{ route('admin.categories.store') }}" method="POST">

        <span class="error" id="nameUniqueErrorEdit">Имя должно быть уникальным</span>

        <span class="error" id="nameMaxErrorEdit">Максимальное количество символов - 250</span>

        @csrf

        <ul class="modal\_\_list">

            <li class="modal\_\_item">

                <label class="label" for="nameEdit">Название категории</label>

                <input class="input" type="text" id="nameEdit" name="nameEdit" required>

            </li>

            <li class="modal\_\_item">

                <label class="label" for="subtitleEdit">Подзаголовок (необязательно)</label>

                <span class="error"

                      id="subtitleMaxErrorEdit">Максимальное количество символов в подзаголовке - 250</span>

                <input class="input" type="text" id="subtitleEdit" name="subtitleEdit">

            </li>

        </ul>

        <button class="modal\_\_btn" type="button" id="editBtn">Сохранить изменения</button>

    </form>

</div>

<div class="modal" id="modalDeleteCategory" data-modal="modalDeleteCategory">

    <button class="modal\_\_close" id="modalCloseDeleteCategory" type="button"></button>

    <h3 class="modal\_\_title">Удаление категории</h3>

    <p class="modal\_\_text">

        Вы действительно хотите удалить категорию <span></span>? Ее удаление приведет к удалению

        всех товаров, принадлежащих к этой категории

    </p>

    <div class="modal\_\_buttons">

        <button class="modal\_\_btn modal\_\_btn--cancel" id="cancelDeleteCategory" type="button">Отменить</button>

        <button class="modal\_\_btn modal\_\_btn--confirm" id="confirmDeleteCategory" type="submit">Да, удалить</button>

    </div>

</div>

@vite(['resources/js/categories/categories.js?type=module'])

@vite(['resources/js/categories/dragging-categories.js'])

</body>

Manage\_products.blade.php

@include('fragments/head', ['title' => 'Управление товарами'])

<body class="body">

<header class="admin\_\_header">

    @include('fragments/header\_admin')

</header>

<section class="admin">

    <div class="container container--admin">

        <div class="admin\_\_wrapper">

            <h1 class="admin\_\_title">Управление товарами</h1>

            <div class="admin-warning admin-warning--green">

                <span class="admin-warning\_\_icon">

                    <img width="26" height="26" src="{{asset('static/images/icons/check-circle.svg')}}"

                         alt="предупреждение">

                </span>

                <div>

                    Чтобы управлять товарами - выберите одну из категорий для дальнейшей работы

                </div>

            </div>

            @if($categories->isEmpty())

                <div class="admin-products\_\_empty">

                    <img class="admin-products\_\_decor" src="{{asset('static/images/icons/flowers.svg')}}" width="120"

                         alt="декор">

                    <span class="admin-products\_\_title">В управлении пока пусто ¯\\_(ツ)\_/¯</span>

                    <p class="admin-products\_\_text">Категорий еще нет, но вы можете создать их</p>

                    <a class="admin-products\_\_btn admin\_\_btn admin\_\_btn--categories"

                       href="{{route('admin.categories.index')}}">Добавить категорию</a>

                </div>

            @endif

            <ul class="admin-products\_\_buttons">

                @foreach($categories as $category)

                    <li class="admin-products\_\_item">

                        <a class="admin-products\_\_link"

                           href="{{route('admin.products.category', ['category\_id' => $category->id])}}">{{$category->name}}

                        </a>

                    </li>

                @endforeach

            </ul>

        </div>

    </div>

</section>

</body>

Manage\_users.blade.php

@include('fragments/head', ['title' => 'Управление пользователями'])

<body class="body">

<header class="admin\_\_header">

    @include('fragments/header\_admin')

</header>

<section class="admin">

    <div class="container container--admin">

        <div class="admin\_\_wrapper">

            <h1 class="admin\_\_title">Управление пользователями</h1>

            <button class="admin-users\_\_btn" id="createUserBtn" type="button">Создать пользователя</button>

            <form class="admin-users\_\_search" action="#" method="GET" id="searchForm">

                <input class="input" type="text" id="query" name="query" placeholder="Поиск по логину, имени или почте">

            </form>

            <ul class="admin-users\_\_list" id="userListContainer">

                @foreach($users as $user)

                    <li class="admin-users\_\_item" data-user-id="{{$user->id}}">

                        <p class="admin-users\_\_login"><span>{{$user->login}}</span></p>

                        <a class="admin-users\_\_edit" href="{{route('admin.users.edit', ['user' => $user])}}">

                            Редактировать

                        </a>

                        <div class="admin-users\_\_content">

                            <p>Имя: {{$user->name}}</p>

                            <p>Почта: {{$user->email}}</p>

                            <p>Роль: {{$user->role}}</p>

                        </div>

                        <div class="admin-users\_\_actions">

                            @if($user->id !== auth()->id())

                                <button class="admin-users\_\_action admin-users\_\_action--reset" type="button">Сбросить

                                    пароль

                                </button>

                                <button class="admin-users\_\_action admin-users\_\_action--delete" type="button"

                                        data-user-id="{{$user->id}}">

                                    Удалить аккаунт

                                </button>

                            @else

                                <p class="admin-users\_\_action admin-users\_\_action--long">¯\\_(ツ)\_/¯ не делай глупых

                                    решений</p>

                            @endif

                        </div>

                    </li>

                @endforeach

            </ul>

        </div>

    </div>

</section>

</body>

<div class="modal modal--long" id="modalAddUser">

    <button class="modal\_\_close" id="modalCloseAddUser" type="button"></button>

    <h3 class="modal\_\_title">Создание пользователя</h3>

    <form action="{{ route('admin.users.store') }}" id="userForm" method="POST">

        @csrf

        <ul class="modal\_\_list modal\_\_list--grid">

            <li class="modal\_\_item">

                <label class="label" for="name">Имя</label>

                <span class="error" id="nameError">Имя не должно содержать запрещенные символы</span>

                <span class="error" id="nameMinError">Мин. количество символов - 2</span>

                <span class="error" id="nameMaxError">Макс. количество символов - 50</span>

                <input class="input" type="text" id="name" name="name" required placeholder="Введите имя">

            </li>

            <li class="modal\_\_item">

                <label class="label" for="login">Логин</label>

                <span class="error" id="loginCheckError">Логин уже используется</span>

                <span class="error" id="loginError">Логин не должен иметь запрещенные символы</span>

                <span class="error" id="loginLengthError">Минимальное количество символов - 5</span>

                <span class="error" id="loginMaxError">Максимальное количество символов - 60</span>

                <input class="input" type="text" id="login" name="login" required placeholder="Введите логин">

            </li>

            <li class="modal\_\_item">

                <label class="label" for="email">Почта</label>

                <span class="error" id="emailCheckError">Почта уже используется</span>

                <span class="error" id="emailErrorParameters">Почта не соответствует параметрам</span>

                <span class="error" id="emailLengthError">Макс. количество символов - 80</span>

                <input class="input" type="email" id="email" name="email" required placeholder="Введите почту">

            </li>

            <li class="modal\_\_item">

                <label class="label" for="password">Пароль</label>

                <span class="error" id="passwordError">Пароль не должен иметь кириллицу</span>

                <span class="error" id="passwordLengthError">Минимальное количество символов - 8</span>

                <span class="error" id="passwordMaxError">Максимальное количество символов - 60</span>

                <input class="input" type="password" id="password" name="password" required

                       placeholder="Введите пароль">

            </li>

            <li class="modal\_\_item">

                <label class="label" for="password">Роль</label>

                <select class="select" id="role" name="role">

                    <option value="USER" {{ old('role') === 'USER' ? 'selected' : '' }}>Пользователь</option>

                    <option value="ADMIN" {{ old('role') === 'ADMIN' ? 'selected' : '' }}>Администратор</option>

                </select>

            </li>

        </ul>

        <button class="modal\_\_btn" id="createBtnUser" type="submit">Создать пользователя</button>

    </form>

</div>

<div class="modal" id="modalDeleteUser">

    <button class="modal\_\_close" id="modalCloseDeleteUser" type="button"></button>

    <h3 class="modal\_\_title">Удаление пользователя</h3>

    <p class="modal\_\_text">

        Вы действительно хотите удалить пользователя <span id="loginDeleteUser"></span>? Удаление приведет к удалению

        всех его заказов и отзывов, а также возможно к уничтожению данных корзины и избранного

    </p>

    <div class="modal\_\_buttons">

        <button class="modal\_\_btn modal\_\_btn--cancel" id="modalCancelDeleteUser" type="button">Отменить</button>

        <button class="modal\_\_btn modal\_\_btn--confirm" id="confirmDeleteUser" type="submit">Да, удалить</button>

    </div>

</div>

@vite(['resources/js/components/custom-select.js'])

@vite(['resources/js/user.js?type=module'])

Approved\_reviews\_homepage.blade.php

@php use Carbon\Carbon; @endphp

@include('fragments/head', ['title' => 'Отображение отзывов на главной'])

<body class="body">

<header class="admin\_\_header">

    @include('fragments/header\_admin')

</header>

<section class="admin admin-reviews">

    <div class="container container--admin">

        <div class="admin\_\_wrapper">

            <h1 class="admin\_\_title">Управление отображением отзывов</h1>

            <ul class="admin-reviews\_\_buttons">

                <li>

                    <a class="admin-reviews\_\_link {{ request()->routeIs('admin.reviews.index') ? 'admin-reviews\_\_link--active' : '' }}"

                       href="{{route('admin.reviews.index')}}">

                        Не одобренные отзывы

                    </a>

                </li>

                <li>

                    <a class="admin-reviews\_\_link {{ request()->routeIs('admin.reviews.approved') ? 'admin-reviews\_\_link--active' : '' }}"

                       href="{{route('admin.reviews.approved')}}">

                        Одобренные отзывы

                    </a>

                </li>

                <li>

                    <a class="admin-reviews\_\_link {{ request()->routeIs('admin.reviews.home') ? 'admin-reviews\_\_link--active' : '' }}"

                       href="{{route('admin.reviews.home')}}">

                        Отображение на главной

                    </a>

                </li>

            </ul>

            @if($approvedReviews->isEmpty())

                <p>Тут пока еще нет одобренных отзывов</p>

            @else

                <div class="admin-warning admin-warning--long">

                <span class="admin-warning\_\_icon">

                    <img width="26" height="26" src="{{asset('static/images/icons/info.svg')}}" alt="предупреждение">

                </span>

                    <div>

                        <span class="admin-warning\_\_title">Прочтите это</span>

                        Эта страница позволит через чекбоксы отобразить отзывы на главной странице (в секции).

                        Достаточно нажать рядом на квадратик чтоб он закрасился и сохранить изменения.

                        После этого изменения вступят в силу, и отзывы будут отображаться

                        <br>

                        <br>

                        Максимальное количество отображаемых отзывов - 12!

                    </div>

                </div>

                <div class="admin-reviews\_\_content">

                    <ul class="admin-reviews\_\_list admin-reviews\_\_list--sticky">

                        @foreach($approvedReviews  as $review)

                            <li class="admin-reviews\_\_item reviews\_\_item" data-review-id="{{$review->id}}">

                                <label class="auth\_\_checkbox custom-checkbox">

                                    <input class="custom-checkbox\_\_input review-checkbox" type="checkbox" name="" value=""

                                           data-review-id="{{$review->id}}" @if($review->display\_on\_homepage) checked @endif>

                                    <span class="checkmark"></span>

                                    Отобразить на главной

                                </label>

                                <div class="admin-reviews\_\_control">

                                    <button class="admin-reviews\_\_btn delete-btn"

                                            data-review-id="{{$review->id}}" data-review-name="{{$review->name}}">

                                        Удалить

                                    </button>

                                </div>

                                <time class="reviews\_\_date">

                                    {{ Carbon::parse($review->created\_at)->locale('ru')->isoFormat('D MMMM YYYY г.') }}

                                </time>

                                <span class="reviews\_\_name">{{$review->name}}</span>

                                <p class="reviews\_\_text">{{$review->comment}}</p>

                                <span class="reviews\_\_rate">

                            Оценка: {{ $review->rating }}

                                    @if ($review->rating == 5)

                                        😁

                                    @elseif ($review->rating == 4)

                                        😄

                                    @elseif ($review->rating == 3)

                                        😊

                                    @elseif ($review->rating == 2)

                                        😐

                                    @else

                                        😞

                                    @endif

                        </span>

                            </li>

                        @endforeach

                    </ul>

                    <button class="admin-reviews\_\_save modal\_\_btn">Сохранить изменения</button>

                </div>

            @endif

        </div>

    </div>

</section>

<div class="modal" id="modalDeleteReview">

    <button class="modal\_\_close" id="modalCloseDeleteReview" type="button"></button>

    <h3 class="modal\_\_title">Удаление отзыва</h3>

    <p class="modal\_\_text">

        Вы действительно хотите удалить отзыв от <span id="reviewUserName"></span>? Удаление отменить будет невозможно

    </p>

    <div class="modal\_\_buttons">

        <button class="modal\_\_btn modal\_\_btn--cancel" id="cancelDeleteReview" type="button">Отменить</button>

        <button class="modal\_\_btn modal\_\_btn--confirm" id="confirmDeleteReview" type="submit">Да, удалить</button>

    </div>

</div>

@vite(['resources/js/reviews/manage.js?type=module'])

@vite(['resources/js/reviews/display-reviews-home.js'])

</body>

Approved\_reviews.blade.php

@php use Carbon\Carbon; @endphp

@include('fragments/head', ['title' => 'Управление отзывами'])

<body class="body">

<header class="admin\_\_header">

    @include('fragments/header\_admin')

</header>

<section class="admin admin-reviews">

    <div class="container container--admin">

        <div class="admin\_\_wrapper">

            <h1 class="admin\_\_title">Управление одобренными отзывами</h1>

            <ul class="admin-reviews\_\_buttons">

                <li>

                    <a class="admin-reviews\_\_link {{ request()->routeIs('admin.reviews.index') ? 'admin-reviews\_\_link--active' : '' }}"

                       href="{{route('admin.reviews.index')}}">

                        Не одобренные отзывы

                    </a>

                </li>

                <li>

                    <a class="admin-reviews\_\_link {{ request()->routeIs('admin.reviews.approved') ? 'admin-reviews\_\_link--active' : '' }}"

                       href="{{route('admin.reviews.approved')}}">

                        Одобренные отзывы

                    </a>

                </li>

                <li>

                    <a class="admin-reviews\_\_link {{ request()->routeIs('admin.reviews.home') ? 'admin-reviews\_\_link--active' : '' }}"

                       href="{{route('admin.reviews.home')}}">

                        Отображение на главной

                    </a>

                </li>

            </ul>

            @if($approvedReviews->isEmpty())

                <p>Тут пока еще нет одобренных отзывов</p>

            @else

                <ul class="admin-reviews\_\_list">

                    @foreach($approvedReviews  as $review)

                        <li class="admin-reviews\_\_item reviews\_\_item" data-review-id="{{$review->id}}">

                            <div class="admin-reviews\_\_control">

                                <button class="admin-reviews\_\_btn delete-btn"

                                        data-review-id="{{$review->id}}" data-review-name="{{$review->name}}">

                                    Удалить

                                </button>

                            </div>

                            <time class="reviews\_\_date">

                                {{ Carbon::parse($review->created\_at)->locale('ru')->isoFormat('D MMMM YYYY г.') }}

                            </time>

                            <span class="reviews\_\_name">{{$review->name}}</span>

                            <p class="reviews\_\_text">{{$review->comment}}</p>

                            <span class="reviews\_\_rate">

                            Оценка: {{ $review->rating }}

                                @if ($review->rating == 5)

                                    😁

                                @elseif ($review->rating == 4)

                                    😄

                                @elseif ($review->rating == 3)

                                    😊

                                @elseif ($review->rating == 2)

                                    😐

                                @else

                                    😞

                                @endif

                        </span>

                        </li>

                    @endforeach

                </ul>

            @endif

        </div>

    </div>

</section>

<div class="modal" id="modalDeleteReview">

    <button class="modal\_\_close" id="modalCloseDeleteReview" type="button"></button>

    <h3 class="modal\_\_title">Удаление отзыва</h3>

    <p class="modal\_\_text">

        Вы действительно хотите удалить отзыв от <span id="reviewUserName"></span>? Удаление отменить будет невозможно

    </p>

    <div class="modal\_\_buttons">

        <button class="modal\_\_btn modal\_\_btn--cancel" id="cancelDeleteReview" type="button">Отменить</button>

        <button class="modal\_\_btn modal\_\_btn--confirm" id="confirmDeleteReview" type="submit">Да, удалить</button>

    </div>

</div>

@vite(['resources/js/reviews/manage.js?type=module'])

</body>

Reviews\_for\_approval.blade.php

@php use Carbon\Carbon; @endphp

@include('fragments/head', ['title' => 'Управление отзывами'])

<body class="body">

<header class="admin\_\_header">

    @include('fragments/header\_admin')

</header>

<section class="admin admin-reviews">

    <div class="container container--admin">

        <div class="admin\_\_wrapper">

            <h1 class="admin\_\_title">Управление не одобренными отзывами</h1>

            <ul class="admin-reviews\_\_buttons">

                <li>

                    <a class="admin-reviews\_\_link {{ request()->routeIs('admin.reviews.index') ? 'admin-reviews\_\_link--active' : '' }}"

                       href="{{route('admin.reviews.index')}}">

                        Не одобренные отзывы

                    </a>

                </li>

                <li>

                    <a class="admin-reviews\_\_link {{ request()->routeIs('admin.reviews.approved') ? 'admin-reviews\_\_link--active' : '' }}"

                       href="{{route('admin.reviews.approved')}}">

                        Одобренные отзывы

                    </a>

                </li>

                <li>

                    <a class="admin-reviews\_\_link {{ request()->routeIs('admin.reviews.home') ? 'admin-reviews\_\_link--active' : '' }}"

                       href="{{route('admin.reviews.home')}}">

                        Отображение на главной

                    </a>

                </li>

            </ul>

            <ul class="admin-reviews\_\_list">

                @foreach($reviewsForApproval as $review)

                    <li class="admin-reviews\_\_item reviews\_\_item" data-review-id="{{$review->id}}">

                        <div class="admin-reviews\_\_control">

                            <button class="admin-reviews\_\_btn approve-btn" data-review-id="{{$review->id}}">Одобрить

                            </button>

                            <button class="admin-reviews\_\_btn delete-btn"

                                    data-review-id="{{$review->id}}" data-review-name="{{$review->name}}">

                                Удалить

                            </button>

                        </div>

                        <time class="reviews\_\_date">

                            {{ Carbon::parse($review->created\_at)->locale('ru')->isoFormat('D MMMM YYYY г.') }}

                        </time>

                        <span class="reviews\_\_name">{{$review->name}}</span>

                        <p class="reviews\_\_text">{{$review->comment}}</p>

                        <span class="reviews\_\_rate">

                            Оценка: {{ $review->rating }}

                            @if ($review->rating == 5)

                                😁

                            @elseif ($review->rating == 4)

                                😄

                            @elseif ($review->rating == 3)

                                😊

                            @elseif ($review->rating == 2)

                                😐

                            @else

                                😞

                            @endif

                        </span>

                    </li>

                @endforeach

            </ul>

        </div>

    </div>

</section>

<div class="modal" id="modalDeleteReview">

    <button class="modal\_\_close" id="modalCloseDeleteReview" type="button"></button>

    <h3 class="modal\_\_title">Удаление отзыва</h3>

    <p class="modal\_\_text">

        Вы действительно хотите удалить отзыв от <span id="reviewUserName"></span>? Удаление отменить будет невозможно

    </p>

    <div class="modal\_\_buttons">

        <button class="modal\_\_btn modal\_\_btn--cancel" id="cancelDeleteReview" type="button">Отменить</button>

        <button class="modal\_\_btn modal\_\_btn--confirm" id="confirmDeleteReview" type="submit">Да, удалить</button>

    </div>

</div>

@vite(['resources/js/reviews/manage.js?type=module'])

</body>