Утверждаю

И.о. зав. каф. САУ

наименование

_____ Ланских Ю.В.

подпись Ф.И.О.

« 30 » июня 20 21 г.

ЗАДАНИЕ
на курсовой проект

по дисциплине «Глобальные сети»

полное название дисциплины

Студенту Работинскому А.А., обучающемуся по образовательной программе 09.03.02 – Информационные системы и технологии. Информационные системы и технологии управления технологическими процессами в промышленности

полное название образовательной программы

пятый заочная

курс обучения форма обучения

Тема курсовой работы (проекта): разработка сайта интернет-магазина типографии (вариант 16).

название темы курсового проекта

1. Исходные данные: Разрабатываемое веб-приложение должно строиться на фреймворке Laravel и декораторе Bootstrap.

2. Основные разделы:
   1. Описание процессов автоматизации в предметной области
   2. Проектирование модели данных, основных подсистем веб-приложения
   3. Реализация веб-приложения

3. График выполнения:
   1. Разработка проекта – 22.07.2021
   2. Описание процессов автоматизации, проектирование БД– 30.08.2021
   3. Проектирование веб-приложения – 30.09.2021
   4. Реализация веб-приложения – 24.10.2021

Представить выполненный курсовой проект на проверку не позднее: 09.11.21

Дата

Руководитель работы	_____	Земцов М.А.	30.06.21
	Подпись руководителя	Ф.И.О. руководителя	Дата
Задание принял	_____	Работинский А.А.	30.06.21
	Подпись обучающегося	Ф.И.О. обучающегося	Дата

Реферат

Работинский А.А. разработка сайта интернет-магазина типографии: ТПЖА.090302.438 ПЗ: Курсовой проект / ВятГУ, каф. САУ; рук. Земцов М.А. – Киров, 2021. ПЗ 48 с., 15 рис., 1 табл., 5 источ., 3 прил.

ВЕБ-ПРИЛОЖЕНИЕ, АРХИТЕКТУРА ИНФОРМАЦИОННОЙ СИСТЕМЫ, ТРЕБОВАНИЯ, PHP, MYSQL SERVER, САЙТ ИНТЕРНЕТ-МАГАЗИНА ТИПОГРАФИИ.

Объектом курсового проекта является сайт интернет-магазина типографии.

Цель работы – выполнение проектирования и реализации сайта интернет-магазина типографии.

В результате выполнения курсового проекта были описание процессов автоматизации в предметной области, проектирование и реализация базы данных. Было выполнено проектирование модели данных с использованием фреймворка Laravel и декоратора Bootstrap. Выполнена реализация сайта интернет-магазина типографии.

№ строки	Формат	Обозначение	Наименование	Количество листов	№ экз.	Примеч.
1			Документация общая			
2			Вновь разработанная			
3						
4	А4	ТПЖА 090302.438 ПЗ	Пояснительная записка	48		
5						
6						
7						
8						
9						
10						
11						
12						
13						
14						
15						
16						
17						
18						
19						
20						
21						
22						
23						
24						
25						
26						
27						
28						

Изм	Лист	№ докум.	Подпись	Дата	ТПЖА 090302.438 ДКП		
Разраб.		Работинский			РАЗРАБОТКА САЙТА ИНТЕРНЕТ-МАГАЗИНА ТИПОГРАФИИ (ВАРИАНТ 16)	Литер	Лист 1 / Листов 1
Провер.		Земцов					
Т. контр.						Кафедра САУ	
Н. контр.						Группа ИТб-5301-02-20	
Утверд.							

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ
БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ
ВЫСШЕГО ОБРАЗОВАНИЯ
«ВЯТСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»
ИНСТИТУТ МАТЕМАТИКИ И ИНФОРМАЦИОННЫХ СИСТЕМ
ФАКУЛЬТЕТ АВТОМАТИКИ И ВЫЧИСЛИТЕЛЬНОЙ ТЕХНИКИ
КАФЕДРА СИСТЕМ АВТОМАТИЗАЦИИ УПРАВЛЕНИЯ

РАЗРАБОТКА САЙТА ИНТЕРНЕТ-МАГАЗИНА ТИПОГРАФИИ (ВАРИАНТ 16)

Курсовой проект по дисциплине
«Глобальные сети»

Пояснительная записка
ТПЖА.090302.438 ПЗ

Выполнил: студент гр. ИТБ-5301-02-20 _____ / Работинский А.А. /
(подпись)

Руководитель: _____ / Земцов М.А. /
(подпись)

Курсовой проект защищен с оценкой «_____» «___»_____ 2021 г.

Комиссия _____ /_____/
(подпись)
_____ /_____/
(подпись)
_____ /_____/
(подпись)

Киров 2021

Содержание

Введение

Решение постоянно возникающих задач, таких как расширение бизнеса, привлечение новых клиентов, вывод на рынок новой продукции подталкивает компании к развитию своего бизнеса. Одним из актуальных направлений развития бизнеса в наше время все больше становится развитие коммерческой деятельности в виде своего собственного интернет-магазина. Интернет-магазины существенно уменьшают издержки производителя, сэкономив на содержании обычного магазина, расширяют рынки сбыта, так же расширяют возможности покупателя: покупать любой товар в любое время в любой стране в любом городе в любое время суток в любое время года. Также сильной стороной интернет-магазина можно считать его компактность: необходимо наличие одного лишь склада, в зависимости от размера штата сотрудников – офиса, при этом отпадает надобность аренды помещения для размещения торговой точки. Плюсом использования интернет-магазина является немногочисленность рабочего персонала: необходим бухгалтер, при необходимости водитель-грузчик, человек, отвечающий за добавление контента и товара в интернет-магазин, также некоторые из этих обязанностей может брать на себя и сам директор по мере возможности. Главной же причиной, побуждающей человека заняться коммерческой деятельностью в сети интернет, являются низкий стартовый капитал и относительно невысокие риски. Всё это дает интернет-магазину большое преимущество перед обычными магазинами.

Актуальность сайта интернет-магазина типографии продиктована демонстрацией товаров для широкой массы пользователей, что в свою очередь может привести к расширению бизнеса.

Целью данного курсового проекта является проектирование и реализация сайта интернет-магазина типографии. С помощью сайта планируется привлечь новых покупателей, расширить рынок продаж.

1 Описание процессов автоматизации в предметной области

В начале разработки идет описание необходимой предметной области по данному заданию с использованием обычного языка, не привязываясь к какой-либо модели представления данных.

1.1 Описание предметной области

В курсовом проекте требуется разработать и реализовать сайт интернет-магазина. На сайте должна содержаться вся необходимая информация о предлагаемых товарах, отображаться товары дня, контакты, информация о компании. Также на сайте интернет-магазина должна быть информация о компании, и она не должна быть избыточна.

Пользователь может купить как один товар, так и несколько товаров, которые может оплачивать банковской картой. Выбрать можно будет из нескольких каталогов, в которых содержаться товары. В корзине отображаются все товары, которые желает купить пользователь. Чтобы приобрести товары необходимо зарегистрироваться на сайте. После регистрации становится доступна покупка выбранных товаров. Для того, чтобы пользователь мог оформить заказ, необходимо добавить товар в корзину. В корзину можно добавлять неограниченное количество товаров. Помимо этого, в корзине можно изменить количество товаров, удалять или добавлять товары. Далее следует перейти к оформлению заказа – на странице оформления заказа пользователь должен выбрать способ доставки и оплаты, и при необходимости отредактировать информацию о доставке (ФИО, адрес доставки, почтовый индекс) и нажать кнопку "оформить заказ". После оформления появится сообщение об успешном оформлении заказа, корзина очистится, и можно будет перейти в личный кабинет в раздел "Мои заказы" для отслеживания статуса заказа. Заказы могут оформлять только авторизованные пользователи.

Эта информация должна храниться в БД и использоваться для ежемесячного анализа приобретённых товаров, введения разовых акций на определённые

товары, дополнительное приобретение товаров, во избежание дефицита. Или наоборот отмена приобретения определённых товаров при отсутствии спроса на них. Одним из важных моментов будет являться процесс принятии решений о рентабельности сайта интернет-магазина и возможном расширении существующего – вывод дополнительного товара, внедрение собственной доставки и т.д. или открытии сайта для других городов.

1.2. Определение и решение задач бизнеса

В настоящее время любая система автоматизации деятельности может решать одну из 2 ключевых задач бизнеса:

– автоматизация повторяющихся по времени действий;

– автоматизация сбора, анализа и представления информации для принятия решения.

В рамках этого курсового проекта решение требует 2 ключевая задача бизнеса, а именно автоматизация сбора, анализа и представления информации для принятия решения.

Основная задача разрабатываемого сайта интернет-магазина типографии – предоставить пользователям возможность ознакомиться с товаром не выходя из дома.

Помимо этого, есть список требований к сайту, к которому относятся:

– надежная защита информации с помощью парольного разграничения доступа к сайту;

– минимальные действия пользователей для приобретения товаров;

– минимальная загруженность сайта для улучшения зрительного восприятия;

– структурированная информация;

– грамотность, уникальность.

1.3 Выводы к разделу 1

В представленном разделе произведен анализ предметной области и определена основная задача разработки сайта интернет-магазина типографии. Кроме того, приведено подробное описание функционирования сайта с указанием возможностей для покупателей.

2. Проектирование веб-приложения

В данном разделе курсового проекта рассматривается структура проекта, в котором описывается бизнес-логика веб-приложения интернет-магазина.

2.1 Проектирование модели данных, основные подсистемы веб-приложения

Интернет-магазин типографии разработан на скриптовом языке PHP (версия 8.0) с применением фреймворка Laravel и с использованием портативной серверной платформы и программной среды для веб-разработчиков open-Server 5.4.0.

Open Server Panel — это портативная программная среда, созданная специально для веб-разработчиков с учётом их рекомендаций и пожеланий. Программный комплекс включает в себя набор серверного программного обеспечения, а также управляющую утилиту, которая обладает мощными возможностями по администрированию и настройке всех доступных компонентов.

OSPanel широко используется с целью разработки, отладки и тестирования веб-проектов, а также для предоставления веб-сервисов в локальных сетях.

Одними из основных преимуществ будут следующие:

– незаметная работа в трее Windows;

– быстрые старт и остановка;

– автостарт сервера при запуске программы;

– несколько режимов управления доменами;

– монтирование виртуального диска;

– поддержка управления через командную строку;

– поддержка профилей настроек;

– удобный просмотр логов всех компонентов;

– переключение HTTP, MySQL и PHP модулей;

– подробная и понятная документация;

– быстрый доступ к шаблонам конфигурации;

– мультиязычный интерфейс;

– автозапуск программ по списку.

А также одними из особенностей среды будут следующие:

– не требует установки (портативность);

– одновременная работа с Denwer, Xampp и т.д.;

– работа на локальном/сетевом/внешнем IP;

– поддержка SSL без всякой дополнительной настройки;

– создание домена путем создания обычной папки;

– поддержка кириллических доменов;

– поддержка алиасов (доменных указателей);

– защита сервера от внешнего доступа;

– punycode конвертер доменных имён;

– набор популярных сторонних расширений PHP

– планировщик заданий (cron);

– создание локального поддомена без потери видимости основного домена в сети интернет.

Они являются ключевыми элементами в объектно-ориентированном моделировании.

2.2 Диаграмма компонентов

Проектируемая диаграмма компонентов предназначена для отображения структурных компонентов веб-приложения и связей между ними. В качестве компонентов рассматриваются только информационные объекты.

Диаграмма компонентов – это статическая структурная диаграмма, которая показывает разбиение программной системы на структурные компоненты и связи (зависимости) между компонентами. В качестве физических компонентов могут выступать файлы, библиотеки, модули, исполняемые файлы, пакеты и т. п.

На рисунке 1 представлены диаграммы компонентов.

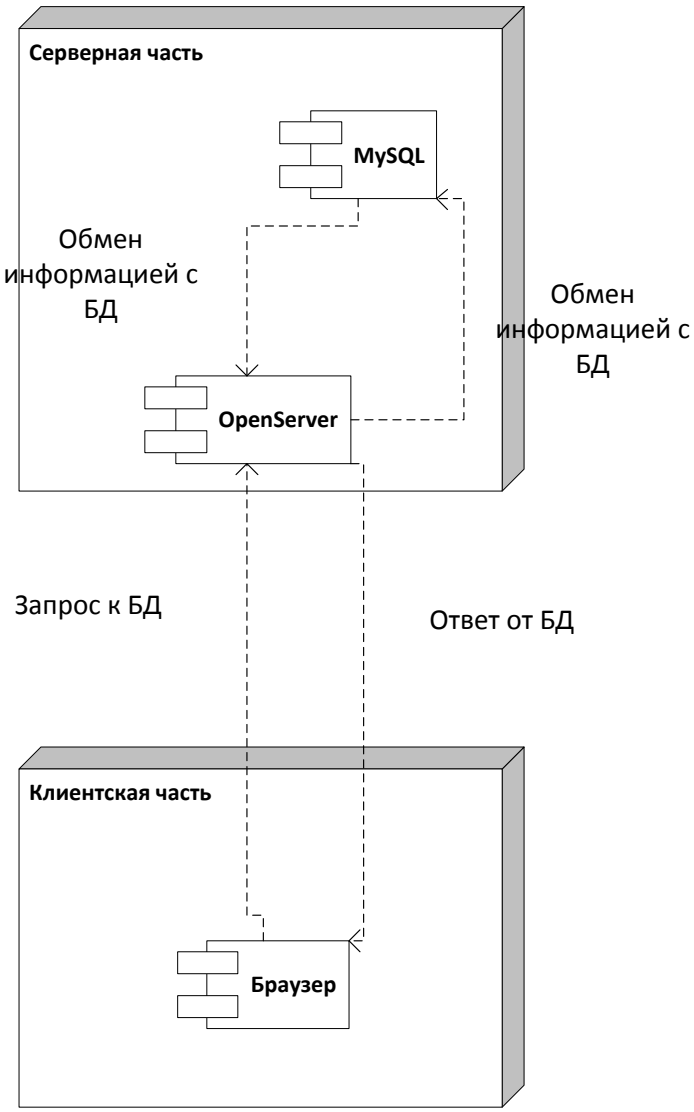

Рисунок 1 – Диаграмма компонентов

Исходя из диаграммы видно, что клиент обращается к серверу приложений, запрашивает данные из БД, сервер приложений обращается к БД MySQL. В свою очередь БД MySQL возвращает запрашиваемые данные, сервер приложений их обрабатывает и возвращает клиенту.

2.3 Диаграмма вариантов использования

Диаграмма вариантов использования – это диаграмма, отражающая отношения между актерами и прецедентами и являющаяся составной частью модели прецедентов. Как видно из рисунка 2, на ней есть 2 актера и 7 прецедентов. Некоторые прецеденты актеров «Пользователь» и «Администратор» также включают в себя другие прецеденты.

Рисунок 2 – Диаграмма вариантов использования

2.4 Проектирование БД веб-приложения

На данном этапе необходимо описать все сущности, их организацию, а также их логическую взаимосвязь, основываясь на реляционной модели представления данных. Сущности в реляционной модели данных представляются в виде таблиц, каждая таблица содержит атрибуты и записи. Каждая запись должна быть уникальна. Для однозначной идентификации записи в таблице существуют первичные ключи. Первичный ключ – это поле или набор полей в таблице, однозначно идентифицирующие запись (экземпляра объекта). Важным условием первичного ключа, является его уникальность в таблице.

Для поддержания логических связей и целостности данных, называемых ссылочной целостностью, используются внешние ключи. Внешний ключ – это ключ, используемый для объединения двух таблиц или более таблиц. Внешний ключ может состоять как из одного атрибута, так и быть составным, и состоять из нескольких атрибутов. Внешний ключ подчинённой таблицы, всегда ссылается на первичный ключ главной таблицы.

Существует три вида связей отношений (таблиц):

– связь «один-к-одному»;

– связь «один-к-многим»;

– связь «многие-ко-многим».

Связь «один-к-одному» устанавливается в случаях, когда одна строка главной таблицы в связана только с одной строкой подчиненной таблицы.

Связь «один-ко-многим» имеет место быть, когда одной записи родительской таблицы соответствует несколько записей дочерней таблицы. Так же такую связь называют «многие-к-одному».

Отношение «многие-ко-многим» имеет место, когда:

– записи в родительской таблице может соответствовать больше одной записи в дочерней таблице;

– записи в дочерней таблице может соответствовать больше одной записи в родительской таблице.

2.5 Описание сущностей

Согласно предметной области, возможно выделить следующие сущности БД:

– «бренды;

– «категории»;

– «позиции заказа»;

– «пользователи»;

– «слайдер»;

– «товары»;

– «заказы»;

– «роли»;

– «товары дня»;

– «атрибуты товара»;

– «значения атрибутов»;

– «способы оплаты»;

– «способы доставки».

Сущность «Пользователи» отражает список всех зарегистрированных пользователей сайта. В этой сущности содержаться сведения о их фамилии,

имени, отчества, адреса доставки, почтового индекса, электронной почте, пароле, дате регистрации, дате редактирования. Данная сущность содержит атрибут «Номер_пользователя», который является первичным ключом определяющий связь «один-к-многим» к сущности «Заказы». Также данная сущность содержит атрибут «Номер_роли», который является внешним ключом определяющий связь «один-к-многим» от сущности «Роли». Данная сущность содержит следующие атрибуты:

- «номер_пользователя»;
- «номер_роли»;
- «фамилия»;
- «имя»;
- «отчество»;
- «адрес доставки»;
- «почтовый индекс»;
- «электронную почту»;
- «пароль»;
- «дата_регистрации»;
- «дата_редактирования».

Первичный ключ «Номер_пользователя» определяет связь «один-к-многим» с сущностью «Заказы»; «многие-к-одному» с сущностью «Роли».

Сущность «Роли» содержит сведения о роле пользователя в веб-приложении, дате создания и дате редактирования. «Номер_роли», является первичным ключом атрибута.

Имеет следующие атрибуты:

- «номер_роли»;
- «название»;
- «дата_создания»;
- «дата_редактирования».

Первичный ключ «Номер_роли» определяет связь «один-к-многим» с сущностью «Пользователи».

Сущность «Заказы» описывает все оформленные заказы, их сумму, дату оформления, дату редактирования. Атрибут «Номер_заказа» является первичным ключом. Также данная сущность содержит атрибуты «Номер_пользователя», «Ном_способа_опл», «Ном_способа_дост» которые являются внешними ключами, определяющими связь «один-к-многим» от сущностей «Пользователи», «Способы оплаты», «Способы доставки» соответственно.

Имеет следующие атрибуты:

– «номер_заказа»;

– «номер_пользователя»;

– «номер_способа_оплаты»;

– «номер_способа_доставки»;

– «сумма_заказа»;

– «дата_оформления»;

– «дата_редактирования».

Сущность «Способы доставки» описывает наименование доставки, описание доставки, цену доставки, дату создания доставки, дату редактирования. Атрибут «Номер» является первичным ключом.

Имеет следующие атрибуты:

– «наименование»;

– «описание»;

– «цена_доставки»;

– «дата_создания»;

– «дата_редактирования».

Первичный ключ «Номер» определяет связь «один-к-многим» с сущностью «Заказы».

Сущность «Способ оплаты» описывает наименование оплаты, содержит само описание, дату создания, дату редактирования. Атрибут «Номер» является первичным ключом. Содержит следующие атрибуты:

– «номер»;

– «наименование»;

– «описание»;

– «дата_создания»;

– «дата_редактирования».

Первичный ключ «Номер» определяет связь «один-к-многим» к сущности «Заказы».

Сущность «Позиции заказа» описывает информацию о каждой позиции в заказе, цену, количество, дату оформления и дату редактирования. Первичным ключом является атрибут «Номер_позиции». Атрибуты «Номер_зазказа» и «Номер_товара» определяют связь «Многие-к-одному» к сущности «Заказы» и к сущности «Товары» соответственно. Сущность содержит следующие атрибуты:

– «номер_позиции»;

– «номер_заказа»;

– «номер_товара»;

– «цена»;

– «количество»;

– «дата_оформления»;

– «дата_редактирования».

Сущность «Бренды» описывает информацию о наименовании брендов, дате создания и дате редактирования. Первичным ключом является атрибут «Номер_бренда», который определяет связь «Один-ко-многим» к сущности «Товары». Сущность содержит следующие атрибуты:

– «номер_бренда»;

– «наименование»;

– «дата_создания»;

– «дата_редактирования».

Сущность «Товары» описывает информацию о всех товарах, находящихся на сайте, их цене, кратком и полном описании, количеству, артикулу, наличию на складе, дате создания, изображению, дате редактирования. Атрибут «Номер_товара» является первичным ключом и определяет связь «один-к-многим» к сущностям «Слайдер», «Товары_дня», «Атрибуты товаров». Содержит «Товары» содержит следующие атрибуты:

– «номер_товара»;

– «номер_категории»;

– «номер_бренда»;

– «наименование товара»;

– «короткое описание»;

– «полное описание»;

– «цена»;

– «артикул»;

– «наличие на складе»;

– «количество»;

– «изображение»;

– «дата создания»;

– «дата редактирования».

Сущность «Категории» описывает информацию о существующих наименованиях товаров, дате создания и дате редактирования информации. Атрибут «Номер_категории» является первичным ключом и определяет связь «один-к-многим» к сущности «Товары». Сущность содержит следующие атрибуты:

– «номер_категории»;

– «наименование»;

– «дата_создания»;

– «дата_редактирования».

Сущность «Слайдер» описывает информацию о товаре, заголовок, описание, дате создания и дате редактирования информации. Атрибут «Номер» является первичным ключом. Атрибут «Ном_товара» является внешним ключом и определяет связь «Многие-к-одному» к сущности «Товары». Сущность содержит следующие атрибуты:

– «номер»;

– «ном_товара»;

– «заголовок»;

– «описание»;

– «изображение»;

– «дата_создания»;

– «дата_редактирования».

Сущность «Товары_дня» описывает информацию о товарах, которые имеют выгодное предложение в определенный день, содержат заголовок, описание, изображение, дату создания и дату редактирования. Атрибут «Номер» является первичным ключом. Атрибут «Ном_товара» является внешним ключом и определяет связь «Многие-к-одному» к сущности «Товары».

Сущность содержит следующие атрибуты:

– «номер»;

– «номер_товара»;

– «заголовок»;

– «описание»;

– «изображение»;

– «дата_создания»;

– «дата_редактирования».

Сущность «атрибуты_товаров» описывает информацию о название товара, дате создание и дате редактирования. Атрибут «Номер_атрибута» является первичным ключом и определяет связь «один-к-многим» к сущности «Значение атрибутов». Атрибут «Ном_товара» является внешним ключом и определяет связь «Многие-к-одному» к сущности «Товары».

Сущность содержит следующие атрибуты:

– «ном_атрибута»;

– «ном_товара»;

– «название»;

– «дата_создания»;

– «дата_редактирования».

Сущность «Значения атрибутов» описывает информацию о значении атрибутов, дате создание и дате редактирования. Атрибут «Номер_значения» является первичным ключом. Атрибут «Ном_атрибута» является внешним ключом и определяет связь «Многие-к-одному» к сущности «Атрибуты товара».

Сущность содержит следующие атрибуты:

– «ном_значения»;

– «ном_атрибута»;

– «значение»;

– «дата_создания»;

– «дата_редактирования».

2.6 Проектирование логической структуры базы данных методом сущность-связь

На этапе проектирование логической структуры базы данных создаются подробные модели представлений данных предметной области. Средством моделирования предметной области на этапе является модель «сущность-связь».

В ней моделирование структуры данных предметной области базируется на использовании графических средств. В наглядном виде они представляют связи между сущностями.

Основными понятиями являются:

– сущность;

– атрибут;

– связь.

Сущность представляет собой объект, информация о котором хранится в БД. Сущность содержит экземпляры, отличающиеся друг от друга значениями атрибутов, атрибуты так же однозначно идентифицируют экземпляр. Атрибут – это свойство сущности. Атрибут, который уникальным образом идентифицирует экземпляры сущности, называется первичным ключом. Может быть составной ключ, представляющий комбинацию нескольких атрибутов. Связь сущностей представляет взаимодействие между сущностями. Она характеризуется мощностью, которая показывает, сколько сущностей участвует в связи. Проектируемая база данных в конечном счете должна быть приведена к 3 форме нормализации база данных. 3 нормальная форма означает что отношение удовлетворяет 2 нормальной форме и каждый не ключевой атрибут должен находится в не транзитивной зависимости от первичного ключа. Логическая модель базы данных согласно предметной области представлена рисунке 3.

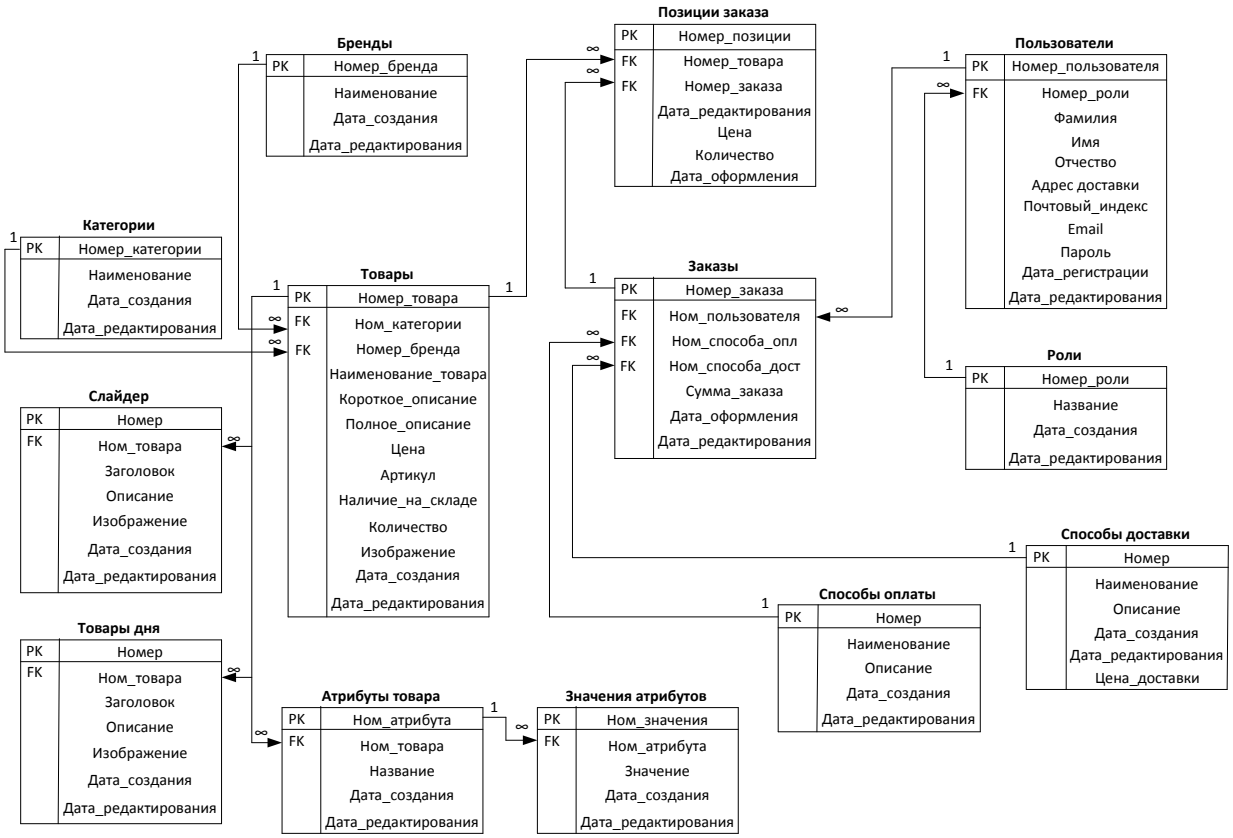

Рисунок 3 – Диаграмма логической модели базы данных

2.7 Описание этапа проектирования физической модели

На данном этапе база данных разрабатывается в среде, выбранной СУБД, в данном случае СУБД является MySQL Server. Производится разработка таблиц с описанием типов хранимых данных, закладывается размер хранимых данных, закладывается возможность или невозможность хранения пустых полей. Диаграмма физической модели базы данных представлена на рисунке 4.

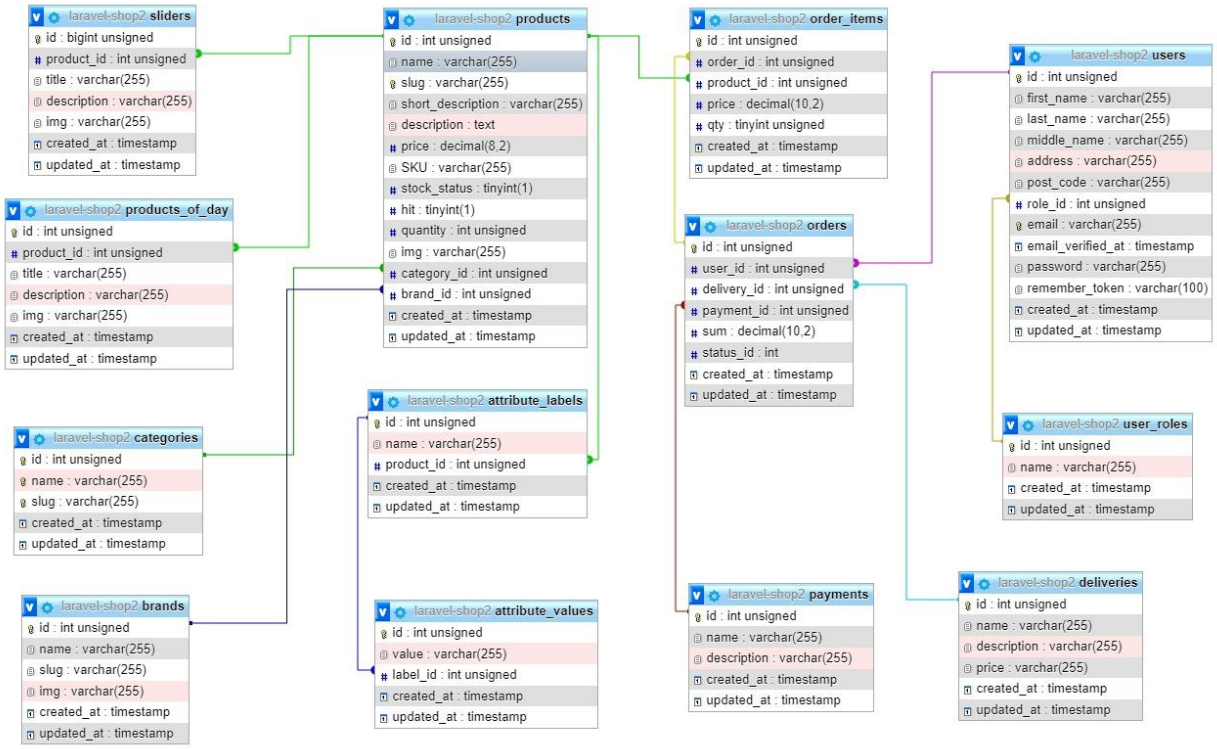

Рисунок 4 – Диаграмма физической модели БД

2.8 Разграничение прав пользователей

Для ввода, изменения, удаления, добавления данных, а также внесения изменений в структуру таблиц и базы данных, необходимо разграничить права пользователей. Это разграничение необходимо прежде всего с целью обеспечения безопасности хранимой информации.

Для поставленных задач были формализованы требования к архитектуре веб-приложения:

– минимизация функций на клиенте;

– масштабируемость;

– безопасность;

– централизованное управление.

Были рассмотрены следующие архитектуры информационных систем (ИС):

– файл-серверная;

– клиент-серверная;

– клиент-серверная с сервером приложений.

В ходе сравнительного анализа была выбрана трёхуровневая клиент-серверная архитектура с сервером приложения. Данная архитектура обладает следующими свойствами:

– минимальное количество функций на стороне клиентского приложения;

– передача минимально-необходимого объёма трафика между клиентом и сервером-приложений;

– минимальные затраты на наращивание функциональности и обновления ПО за счёт централизации сервера-приложения и сервера баз данных;

– клиентский слой не имеет доступа к слою данных и имеет ограниченный доступ к серверу приложений.

Так как данные свойства удовлетворяли заявленным, была выбрана именно эта архитектура веб-приложения. Модель обработки клиентских запросов в клиент-серверной архитектуре с сервером приложений системы управления базой данных (СУБД) представлена на рисунке 5.

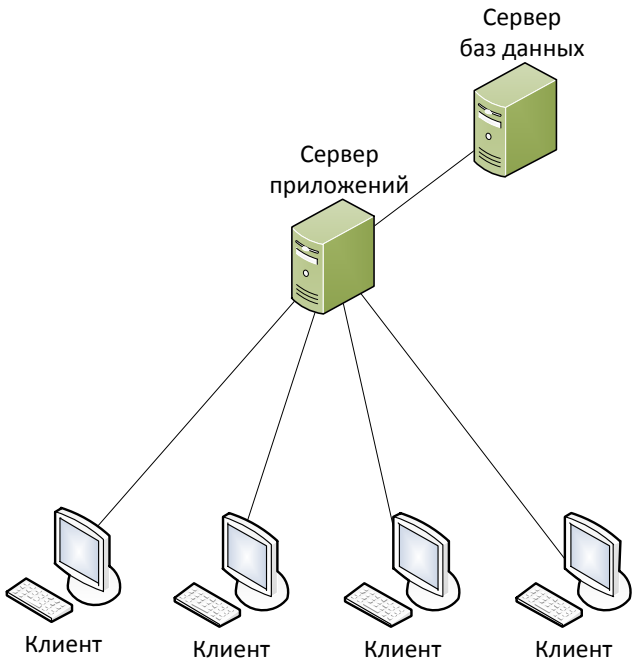

Рисунок 5 – Модель обработки клиентских запросов СУБД

СУБД располагается на сервере вместе с БД и осуществляет доступ к БД непосредственно, в монопольном режиме.

Трёхуровневая клиент-серверная архитектура с сервером приложений состоит из следующих компонентов:

– клиент;

– сервер приложений;

– слой данных.

Клиент – это компонент, предоставляемый конечному пользователю. Этот компонент не имеет прямых связей с базой данных, не нагружен основной бизнес-логикой и не хранит состояние приложения. На этот уровень выносится только простейшая бизнес-логика:

– интерфейс;

– проверка вводимых значений на допустимость и соответствие формату;

– несложные операции с данными (сортировка, группировка, подсчёт значений).

Сервер приложений – на нем располагается основная часть бизнес-логики. Вне этого слоя остаются только фрагменты, экспортируемые на клиента, а также элементы логики, погруженные в СУБД (хранимые процедуры и триггеры). Серверы приложений проектируются таким образом, чтобы добавление к ним дополнительных экземпляров обеспечивало горизонтальное масштабирование производительности программного комплекса и не требовало внесения изменений в программный код приложения.

В My SQL Server существуют два уровня распределения ролей пользователей: роли уровня сервера и роли уровня базы данных.

В рамках данного курсового проекта достаточно разграничить права пользователей на уровне базы данных. Для доступа к БД необходимо выделить 2 логических пользователя:

1. Администратор;
2. Пользователь.

Права доступа для пользователей к таблицам БД приведены в таблице 1

Таблица 1 – Права доступа пользователей к таблицам БД

Пользователь	Администратор	Пользователь
Пользователи	Select	Select, update, insert
Товары	Все права доступа (select, up date, insert, delete	Select, insert
Категории		Select
Заказы		Select, insert
Роли		Select
Способы доставки и оплаты		Select
Позиции заказа		Select, insert
Бренды		Select
Слайдер		Select
Товары дня		Select
Атрибуты товаров		Select
Значения атрибутов		Select

Администратор не может редактировать данные пользователей. Пользователи самостоятельно могут изменять свои личные данные через личный кабинет.

2.9 Описание конфигурации используемых подсистем фреймворка

В ходе проектирования курсового проекта был использован пакет фреймворка Laravel Breeze. Laravel Breeze – это минимальная и простая реализация всего функционала аутентификации Laravel, включая вход в систему, регистрацию, сброс пароля, подтверждение адреса электронной почты и пароля. Слой «View» комплекта Laravel Breeze по умолчанию состоит из простых шаблонов Blade, стилизованных с помощью Tailwind CSS. Breeze является прекрасной отправной точкой для создания нового приложения Laravel.

Процесс установки начинается с создания нового приложения Laravel, настройки базы данных и запуском миграции базы данных:

php artisan migrate

После этого Laravel Breeze устанавливается с помощью Composer:

composer require laravel/breeze --dev

После того, как Composer установил пакет Laravel Breeze, необходимо запустить команду breeze:install Artisan. Эта команда опубликует для приложения шаблоны, маршруты, контроллеры и другие ресурсы аутентификации. Laravel Breeze опубликует весь свой код в приложении, чтобы был полный контроль, а также обзор всего функционала и его реализации. После установки Breeze компилируем исходники. Все маршруты Breeze определены в файле routes/auth.php. Для доступа файла стилей приложения необходимы следующие команды:

php artisan breeze:install

npm install

npm run dev

php artisan migrate

Пользовательский интерфейс интернет-магазина типографии состоит из двух частей:

– клиентская;

– административная.

В клиентской части интернет-магазина были реализованы одни из следующих функциональных элементов:

– каталоги товаров;

– регистрация пользователей;

– авторизация пользователей;

– корзина товаров;

– оформления заказа, его оплата;

В административной части интернет-магазина были реализованы одни из следующих функциональных элементов:

– изменение атрибутов товаров;

– изменение категорий;

– управление заказами.

2.10 Выводы к разделу 2

В настоящем разделе курсового проекта были определены логическая и физическая модели БД, были определены роли, права пользователей, интерфейсные части интернет-магазина.

3. Реализация веб-приложения.

Не мало важным этапом разработки сайта интернет-магазина является реализация сайта. Насколько качественно было проведено проектирование, настолько качественно будет произведена разработка и реализация сайта.

3.1 Описание выбора языка программирования

Фреймворк Laravel использует архитектурную модель MVC (Model, View, Controller).

В свою очередь концепция MVC – это конструкционный шаблон, который описывает способ построения структуры веб-приложения, сферы ответственности и взаимодействие каждой из частей в данной структуре.

Идея MVC очень проста – нужно четко разделять ответственность за различное функционирование в приложении:

– Model – отвечает за обработку данных и логику приложения;

– View – отвечает за представление данных пользователю в любом поддерживаемом формате;

– Controller – отвечает за обработку запросов пользователя и вызов соответствующих ресурсов.

Для каждого контроллера реализованы классы-модели, в которых описаны свойства и методы, которые содержат наиболее важную часть логики приложения.

Все данные, которые обрабатывают методы контроллеров передаются в представление (View) в виде массива. В представлении отображаются все обработанные запросы, но не обрабатывает введенные пользователем данные. Представление отвечает за получение данных из модели. Представление напоминает обычную страницу html, в котором могут быть определены все стандартные элементы разметки, могут подключаться стили, скрипты.

3.2 Декоратор Bootstrap

Декоратор Bootstrap используется в разрабатываемом курсовом проекте для верстки адаптивного дизайна сайта. В настоящее время фреймворк Bootstrap довольно популярен, так как он позволяет верстать сайты намного быстрее, чем это можно сделать, используя CSS и JavaScript.

Декоратор Bootstrap – это открытый и бесплатный HTML, CSS и JS фреймворк, который используется веб-разработчиками. С его помощью работают по всему миру не только независимые разработчиками, но иногда и целые компании. На Bootstrap создано очень много различных сайтов.

Основными инструментами данного фреймворка Bootstrap являются:

– сетки – заранее заданные размеры колонок, которые можно сразу же использовать;

– шаблоны – фиксированный или резиновый шаблон документа;

– типографика – описания шрифтов, определение некоторых классов для шрифтов, таких как код, цитаты;

– медиа – предоставляет некоторое управление изображениями и видео;

– таблицы – средства оформления таблиц, (в том числе добавление функциональности сортировки);

– формы – классы для оформления форм и некоторых событий, происходящих с ними;

– навигация – классы оформления для панелей, вкладок, перехода по страницам, меню и панели инструментов;

– алерты – оформление диалоговых окон, подсказок и всплывающих окон.

К преимуществам декоратора Bootstrap можно отнести:

– высокая скорость-создания качественной адаптивной вёрстки даже начинающими веб-разработчиками;

– кроссбраузерность и кроссплатформенность (корректное отображение и работа сайта во всех поддерживаемых этим фреймворком браузерах и операционных системах);

– наличие большого количество готовых хорошо продуманных компонентов, протестированных на различных устройствах;

– возможность настройки под свой проект, достигается посредством изменения SCSS переменных и использования Bootstrap миксинов (можно изменить количество колонок, цвета, радиуса скруглений углов элементов, отступы между колонками и многое другое);

– низкий порог вхождения – для работы с фреймворком не обязательно иметь глубокие знания по HTML, CSS, JavaScript и jQuery (достаточно знать только основы вышеперечисленных технологий);

– наличие хорошо продуманного дизайна компонентов и согласованности (все компоненты выполнены в едином стиле);

– наличие огромного количества различной информации по фреймворку (статей, видеоматериалов, ответов на многие вопросы).

К одним из недостатков декоратора Bootstrap можно отнести:

– небольшие трудности при работе с методологиями;

– отсутствие обратной совместимости между версиями фреймворка;

– полная зависимость в js компонентах от jquery.

В итоге, использование фреймворка Bootstrap в курсовом проекте позволит увеличить скорость разработки, а также поможет создать интересный дизайн.

3.3 Описание реализации веб-приложения

Установка Laravel происходит через composer: в командной строке выполнить команду:

composer create-project laravel/laravel projectname,

где projectname – название проекта.

После установки идет процесс настройки конфигурации приложения, а именно – настройка подключения к базе данных. Для этой настройки открываем файл .env и редактируем его:

DB_CONNECTION=mysql

DB_HOST=127.0.0.1

DB_PORT=3306

DB_DATABASE=dbname

DB_USERNAME=username

DB_PASSWORD=password

где dbname – название базы данных

username – имя пользователя

password – пароль пользователя

Далее выполняем миграцию таблиц базы данных. Сделать это можно выполнив следующую команду:

php artisan migrate

Сами файлы миграций хранятся в директории database. После установки Laravel и завершения настроек конфигурации можно открыть сайт через браузер. Каркас главной страницы находится в файле resources\views\welcome.blade.php. Чтобы начать разработку собственных страниц необходимо создать класс контроллера. Для этого в директории app\http\controllers создаем файл контроллера. Название файла должно начинаться с заглавной буквы и должно содержать слово "Controller", например, ProductController.php. В файле ProductController.php содержится описание класса. Название класса должно соответствовать названию файла и должно быть унаследовано от Controller:

class ProductController extends Controller.

Также контроллеры можно создавать через командную строку:

php artisan make: controller ProductController,

в классе ProductController описать метод show()

```
class ProductController extends Controller {
        public function show()
        {
                return view('product.index');
        }
}
```

После создания контроллера необходимо создать вид (представление) контроллера. Для этого создаем в директории resources\views\product файл product.blade.php. В файле product.blade.php пишем html разметку страницы. Для того чтобы созданная страница отображалась в браузере необходимо прописать маршрут в файле routes\web.php.

Route('/product', 'ProductController@show');

Для отображения результата набираем в адресной строке браузера http://myshop/.

Также выделим настройку главной страницы

Главная страница имеет следующий маршрут:

Route::get('/', 'MainController@index')->name('home');

контроллером которого является MainController:

```
class MainController extends Controller
{
    public function index()
    {
        $popularProducts = Product::where('hit', '1')->get();
```

```
        return view('welcome', compact('popularProducts'));
    }
}
```

В методе index() определим переменную $popularProducts – для получения товаров, которые являются хитами (популярными). Логику получения данных из таблицы базы данных предоставляет модель Product. ORM Eloquent в свою очередь предоставляет готовые методы запросов к базе данных.

Все переменные передаются в вид главной страницы в виде массива:

```
return view('welcome', compact('popularProducts'));
```

Перейдем в вид главной страницы welcome.blade.php и в цикле foreach получим список популярных товаров:

```
@foreach($popularProducts as $product)
    <div class="product-default-single border-around">
        <img src="{{ asset('storage/'.$product->img) }}">
        <a href="#">{{ $product->name }}</a>
    </div>
@endforeach
```

Таким образом, на главной странице отображаются все популярные товары (в том числе изображения и наименования товаров).

3.4 Результат выполненной работы

На рисунке 6 представлена форма регистрации пользователей в системе.

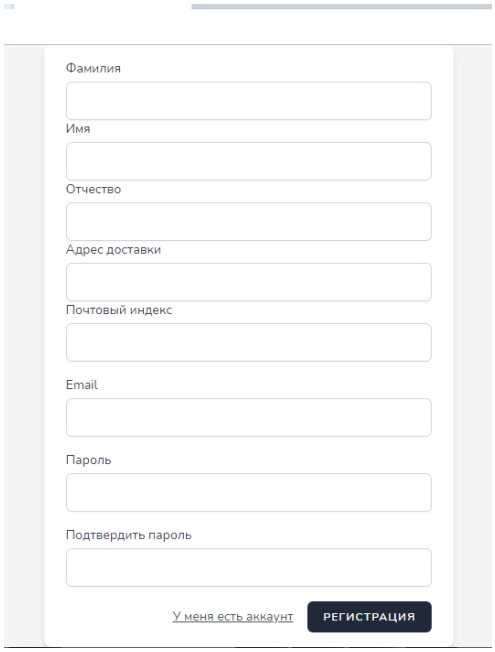

Рисунок 6 – Форма регистрации пользователей

На рисунке 7 представлена форма авторизации пользователей.

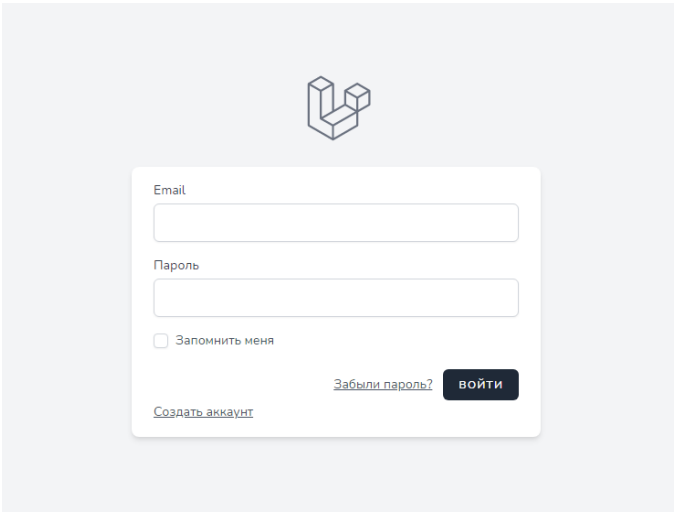

Рисунок 7 – Форма авторизации пользователей

На рисунке 8 представлена панель администратора.

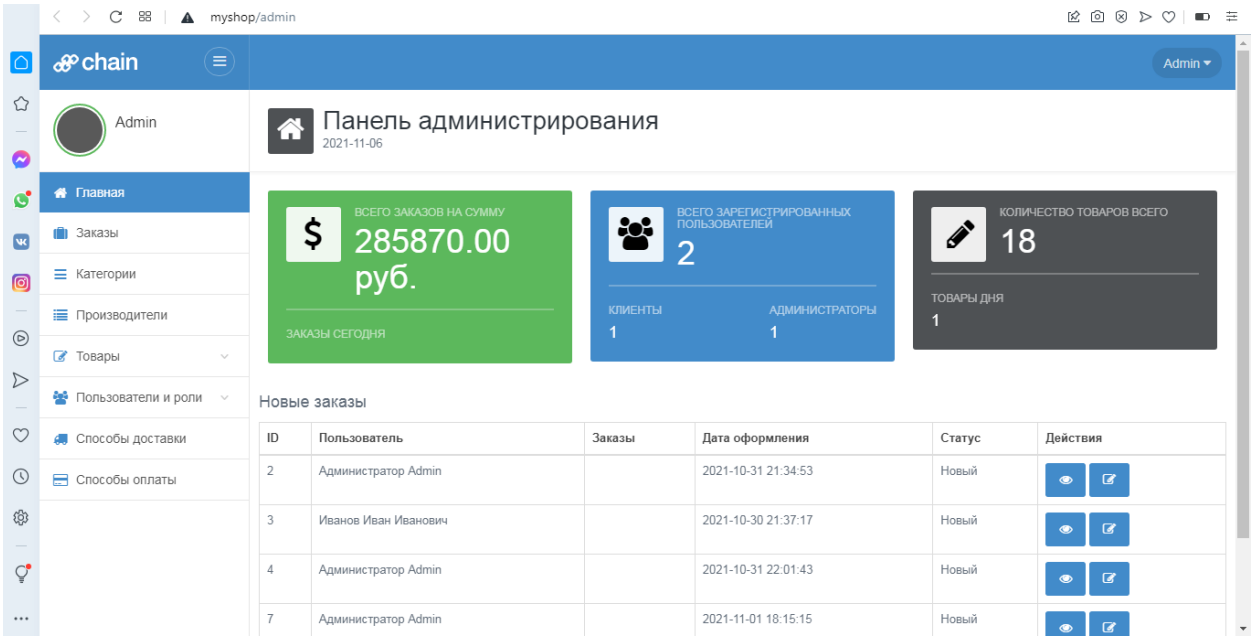

Рисунок 8 – панель администратора

На рисунке 9 представлены способы доставки товаров.

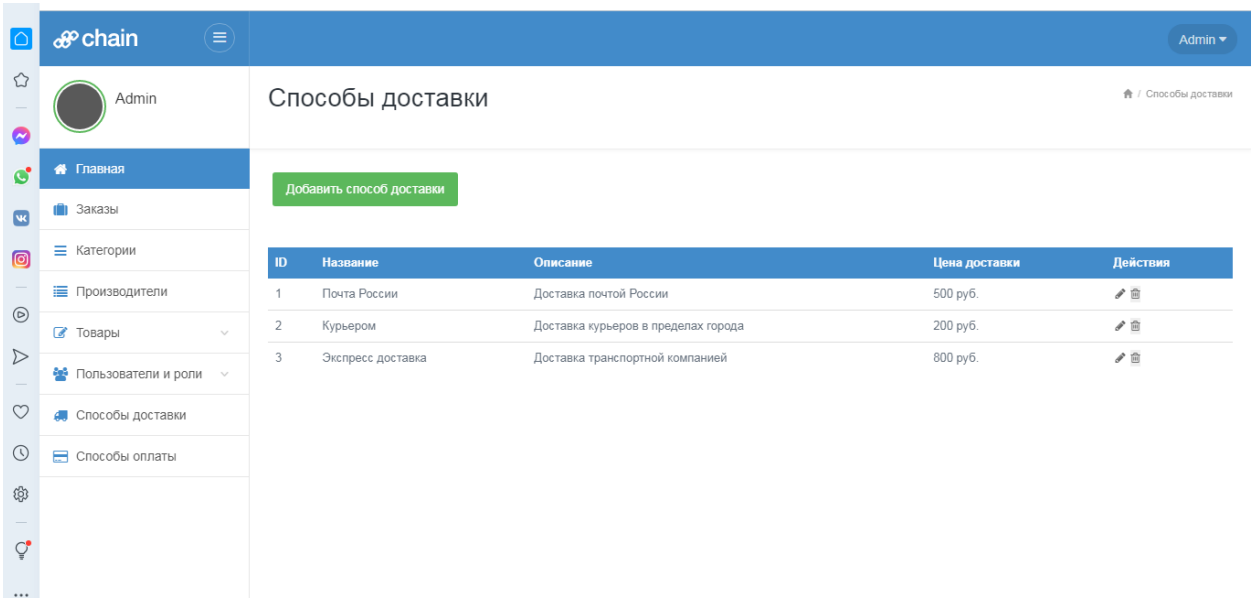

Рисунок 9 – Способы доставки товаров

На рисунке 10 представлены товары «Бренда№1».

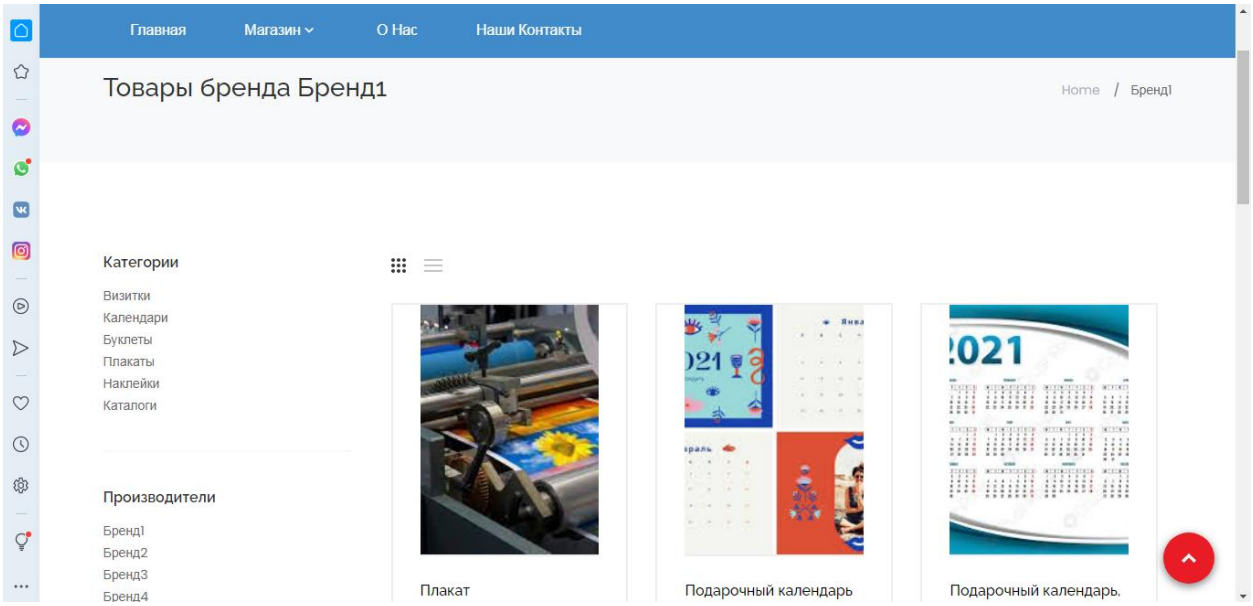

Рисунок 10 – Товары «Бренда№1»

На рисунке 11 представлены товары категории «Плакаты».

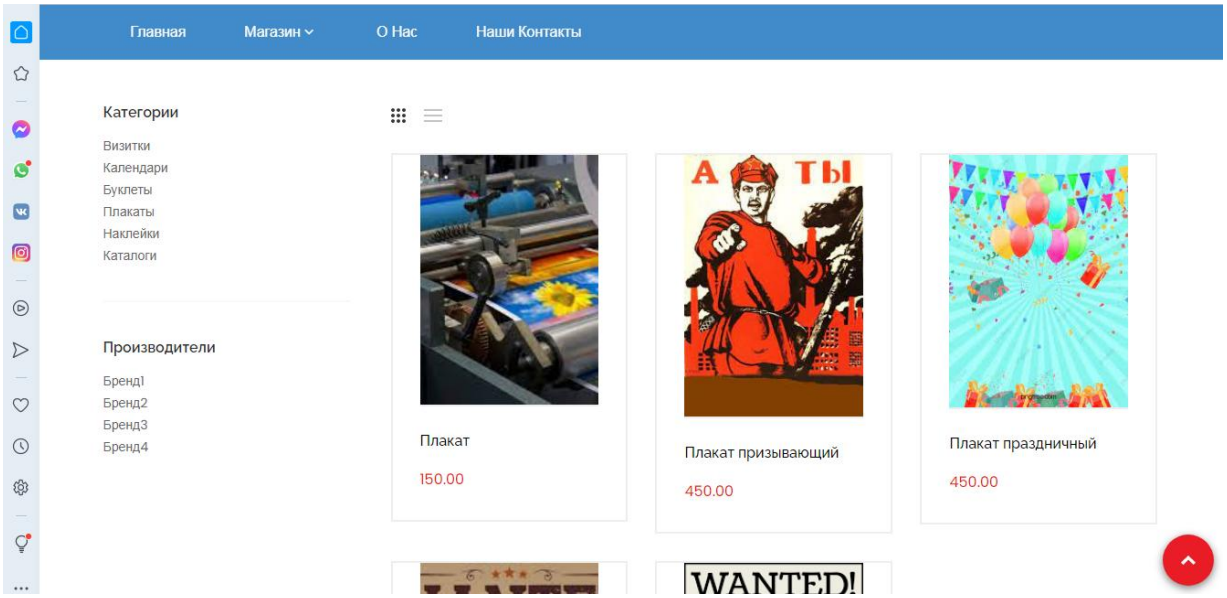

Рисунок 11 – Товары категории «Плакаты»

На рисунке 12 представлены товары категории «Плакаты»

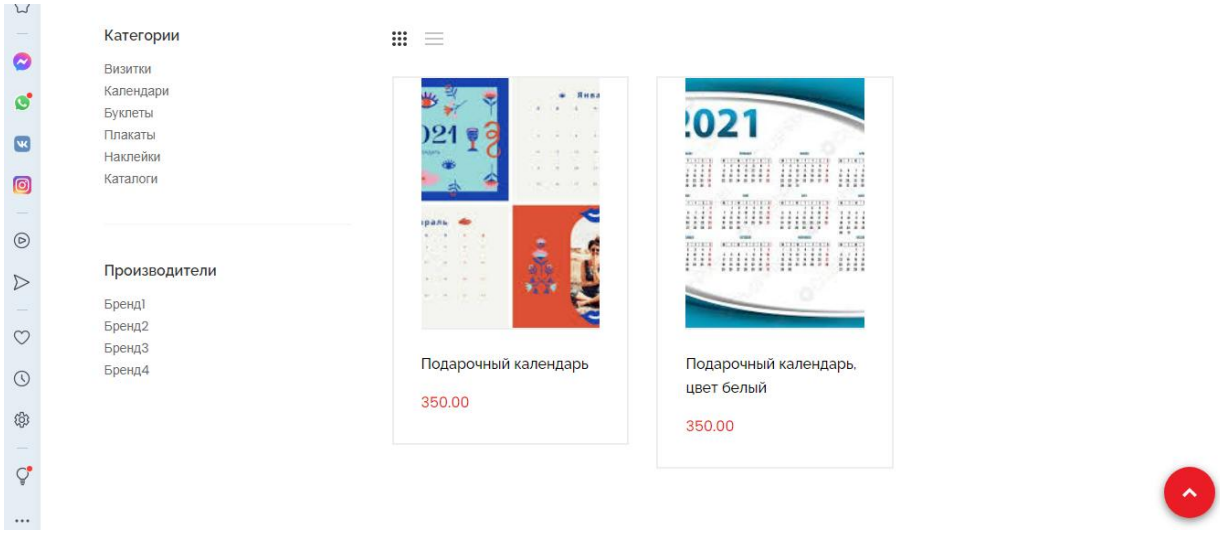

Рисунок 12 – Товары категории календари

На рисунке 13 представлена ситуация, когда введен неверный логин или пароль.

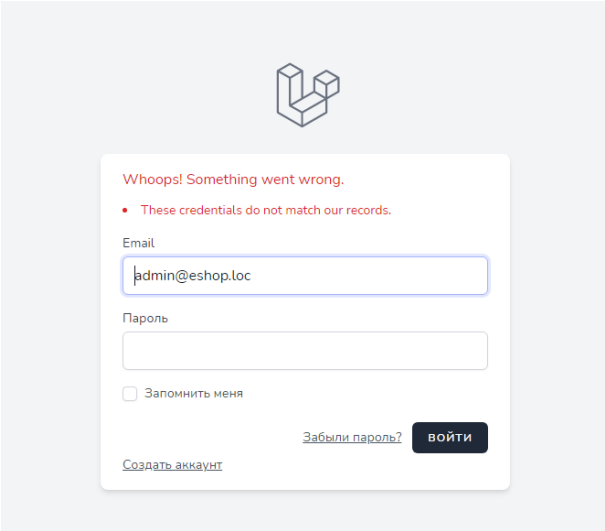

Рисунок 13 – Ошибка при неправильно введённом логине или пароле

На рисунке 14 представлен пример добавления товара в корзину.

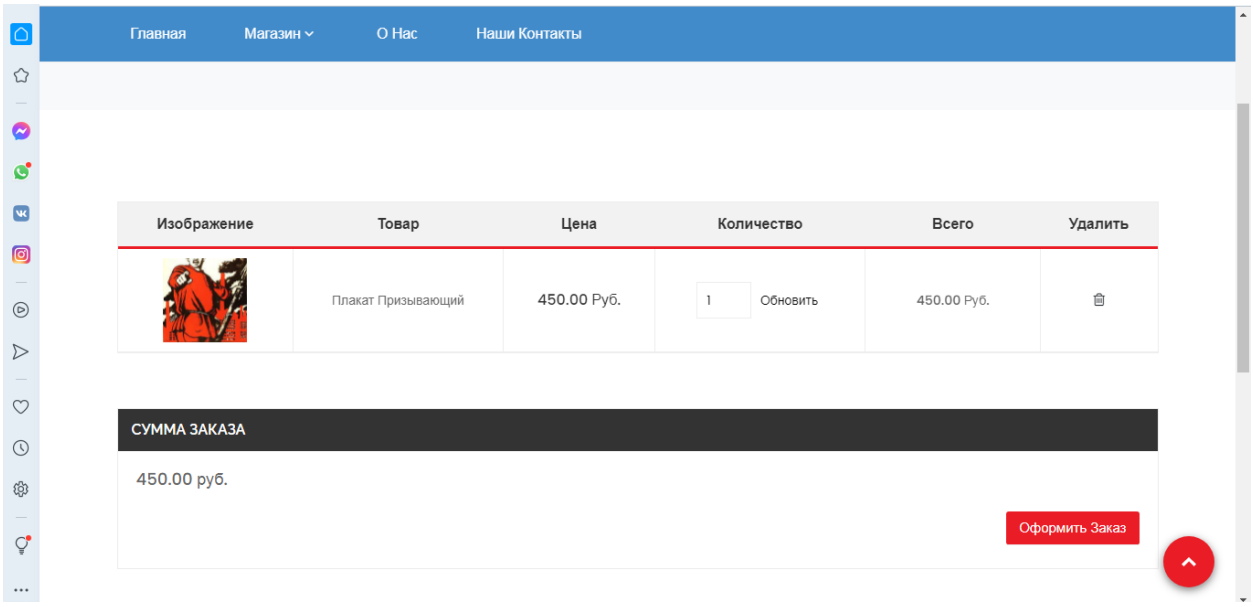

Рисунок 14 – Пример добавления товара в корзину.

На рисунке 15 представлена форма оформления заказа.

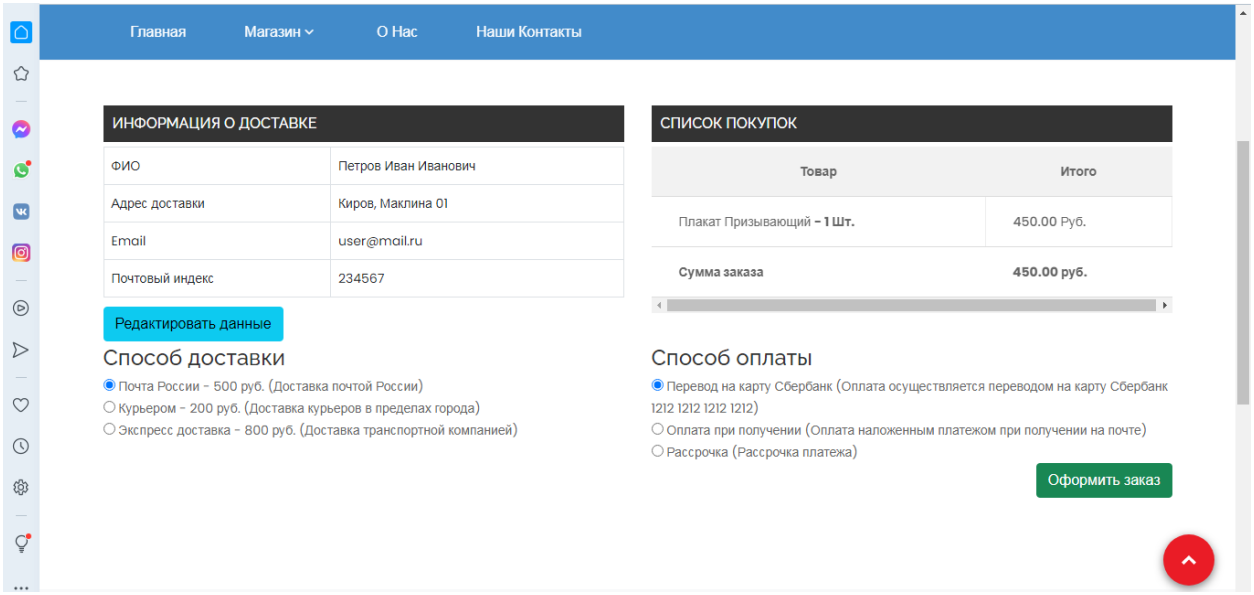

Рисунок 15 – Оформление заказа

В итоге реализованный интернет-магазин обладает минимальным набором функций, которые необходимы для функционирования и привлечения потенциальных клиентов.

3.5 Выводы к разделу 3

В настоящем разделе курсового проекта был описан процесс реализации сайта, а также представлены экранные формы реализованного интернет-магазина.

Заключение

В рамках курсового проекта была поставлена задача разработать сайт интернет-магазина типографии, ознакомиться с фреймворком Laravel используя декоратор bootstrap. Разработанный и реализованный сайт позволяет просматривать различные категории товаров, товары дня, оформлять заказы, оплачивать их.

Требования, которые изначально были определены, и, которыми должна обладать разрабатываемое веб-приложение, были выполнены.

При создании web-приложения использовалось большое количество средств для разработки. К ним относятся:

– язык PHP;

– фреймворк Laravel;

– портативная серверная платформа и программная среда для веб-разработчиков open-Server 5.4.0;

– декоратор Bootstrap.

Все они в совокупности обеспечивают функционирование веб-приложения и предоставляют возможность создания адаптивного дизайна, который пригоден для демонстрации.

Кроме того, в курсовом проекте представлены экранные формы разработанного веб-приложения. Также представлены части исходного кода.

Таким образом, в результате работы над данным курсовым проектом было разработано web-приложение, которое предназначено для просмотра товаров типографии, обладающее минимальным набором функций, а именно просмотр каталога товаров, выбор товаров, приобретение товаров, доставка товаров.

В дальнейшем возможно применение фреймворка laravel в проекте ВКР.

Приложение А

(обязательное)

Скрипт создания базы данных

```
CREATE TABLE `categories` (
  `id` int UNSIGNED NOT NULL,
  `name` varchar(255) COLLATE utf8mb4_unicode_ci NOT NULL,
  `slug` varchar(255) COLLATE utf8mb4_unicode_ci NOT NULL,
  `created_at` timestamp NULL DEFAULT NULL,
  `updated_at` timestamp NULL DEFAULT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_unicode_ci;


CREATE TABLE `products` (
  `id` int UNSIGNED NOT NULL,
  `name` varchar(255) COLLATE utf8mb4_unicode_ci NOT NULL,
  `slug` varchar(255) COLLATE utf8mb4_unicode_ci NOT NULL,
  `short_description` varchar(255) COLLATE utf8mb4_unicode_ci DEFAULT NULL,
  `description` text COLLATE utf8mb4_unicode_ci,
  `price` decimal(8,2) NOT NULL,
  `SKU` varchar(255) COLLATE utf8mb4_unicode_ci NOT NULL,
  `stock_status` tinyint(1) NOT NULL DEFAULT '1',
  `hit` tinyint(1) NOT NULL DEFAULT '0',
  `quantity` int UNSIGNED NOT NULL DEFAULT '10',
  `img` varchar(255) COLLATE utf8mb4_unicode_ci DEFAULT NULL,
  `category_id` int UNSIGNED DEFAULT NULL,
  `created_at` timestamp NULL DEFAULT NULL,
  `updated_at` timestamp NULL DEFAULT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_unicode_ci;

CREATE TABLE `attribute_labels` (
  `id` int UNSIGNED NOT NULL,
  `name` varchar(255) COLLATE utf8mb4_general_ci NOT NULL,
  `product_id` int UNSIGNED NOT NULL,
  `created_at` timestamp NULL DEFAULT NULL,
  `updated_at` timestamp NULL DEFAULT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_general_ci;


CREATE TABLE `attribute_values` (
  `id` int UNSIGNED NOT NULL,
  `value` varchar(255) COLLATE utf8mb4_general_ci NOT NULL,
  `label_id` int UNSIGNED NOT NULL,
  `created_at` timestamp NULL DEFAULT NULL,
```

```
  `updated_at` timestamp NULL DEFAULT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_general_ci;

CREATE TABLE `payments` (
  `id` int UNSIGNED NOT NULL,
  `name` varchar(255) COLLATE utf8mb4_unicode_ci NOT NULL,
  `description` varchar(255) COLLATE utf8mb4_unicode_ci NOT NULL,
  `created_at` timestamp NULL DEFAULT NULL,
  `updated_at` timestamp NULL DEFAULT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_unicode_ci;


CREATE TABLE `deliveries` (
  `id` int UNSIGNED NOT NULL,
  `name` varchar(255) COLLATE utf8mb4_unicode_ci NOT NULL,
  `description` varchar(255) COLLATE utf8mb4_unicode_ci DEFAULT NULL,
  `price` varchar(255) COLLATE utf8mb4_unicode_ci DEFAULT NULL,
  `created_at` timestamp NULL DEFAULT NULL,
  `updated_at` timestamp NULL DEFAULT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_unicode_ci;


CREATE TABLE `orders` (
  `id` int UNSIGNED NOT NULL,
  `user_id` int UNSIGNED NOT NULL,
  `status_id` int UNSIGNED NOT NULL DEFAULT '1',
  `delivery_id` int UNSIGNED NOT NULL,
  `payment_id` int UNSIGNED NOT NULL,
  `sum` varchar(255) COLLATE utf8mb4_unicode_ci NOT NULL,
  `created_at` timestamp NULL DEFAULT NULL,
  `updated_at` timestamp NULL DEFAULT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_unicode_ci;


CREATE TABLE `order_items` (
  `id` int UNSIGNED NOT NULL,
  `order_id` int UNSIGNED NOT NULL,
  `product_id` int UNSIGNED NOT NULL,
  `price` decimal(10,2) NOT NULL,
  `qty` tinyint UNSIGNED NOT NULL DEFAULT '1',
  `created_at` timestamp NULL DEFAULT NULL,
  `updated_at` timestamp NULL DEFAULT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_unicode_ci;

CREATE TABLE `sliders` (
  `id` bigint UNSIGNED NOT NULL,
  `product_id` int UNSIGNED NOT NULL,
  `title` varchar(255) COLLATE utf8mb4_unicode_ci NOT NULL,
```

```
  `description` varchar(255) COLLATE utf8mb4_unicode_ci DEFAULT NULL,
  `img` varchar(255) COLLATE utf8mb4_unicode_ci NOT NULL,
  `created_at` timestamp NULL DEFAULT NULL,
  `updated_at` timestamp NULL DEFAULT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_unicode_ci;

CREATE TABLE `products_of_day` (
  `id` bigint UNSIGNED NOT NULL,
  `product_id` int UNSIGNED NOT NULL,
  `title` varchar(255) COLLATE utf8mb4_unicode_ci NOT NULL,
  `description` varchar(255) COLLATE utf8mb4_unicode_ci DEFAULT NULL,
  `img` varchar(255) COLLATE utf8mb4_unicode_ci NOT NULL,
  `created_at` timestamp NULL DEFAULT NULL,
  `updated_at` timestamp NULL DEFAULT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_unicode_ci;

CREATE TABLE `users` (
  `id` int UNSIGNED NOT NULL,
  `email` varchar(255) COLLATE utf8mb4_unicode_ci NOT NULL,
  `email_verified_at` timestamp NULL DEFAULT NULL,
  `password` varchar(255) COLLATE utf8mb4_unicode_ci NOT NULL,
  `remember_token` varchar(100) COLLATE utf8mb4_unicode_ci DEFAULT NULL,
  `role_id` smallint UNSIGNED DEFAULT NULL,
  `created_at` timestamp NULL DEFAULT NULL,
  `updated_at` timestamp NULL DEFAULT NULL,
  `first_name` varchar(255) COLLATE utf8mb4_unicode_ci NOT NULL,
  `last_name` varchar(255) COLLATE utf8mb4_unicode_ci NOT NULL,
  `middle_name` varchar(255) COLLATE utf8mb4_unicode_ci DEFAULT NULL,
  `address` varchar(255) COLLATE utf8mb4_unicode_ci NOT NULL,
  `post_code` varchar(255) COLLATE utf8mb4_unicode_ci NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_unicode_ci;
```

Приложение Б

(обязательное)

Листинг программного кода

Листинг 1 – Класс контроллера CategoryController для вывода товаров категории

```
class CategoryController extends Controller {
    public function show($slug) {
        $category = Category::where('slug', $slug)->first();
        $category_id = $category->id;
        $category_name = $category->name;
        $productList = Product::where('category_id', $category_id)->paginate(8);
        $categoryList = Category::all();
        $brandList = Brand::all();
        return view('category.product-category', compact('productList', 'categoryList', 'brandList', 'category_name'));
    }
}
```

Листинг 2 – Класс контроллера ProductController для вывода информации о товаре

```
class ProductController extends Controller {
    public function detail($slug) {
        $product = Product::where('slug', $slug)->first();
        $popular = Product::where('hit', 'on')->inRandomOrder()->limit(4)->get();
        $related = Product::where('category_id', $product->category_id)->inRandomOrder()->limit(10)->get();
        $attributeList = AttributeLabel::where('product_id', $product->id)->get();
        return view('product.detail', compact('product', 'popular', 'related', 'attributeList'));
    }
}
```

Листинг 3 – класс контроллера CartController для описания методов добавления, изменения, просмотра и удаления товаров из корзины

```
class CartController extends Controller
{
    public function index() {
        $categoryList = Category::all();
        return view('cart.index', compact('categoryList'));
    }
    public function store(Request $request) {
        Cart::add($request->id, $request->name, $request->qty, $request->price)->associate('App\Models\Product');
        return redirect()->route('cart')->with('success_message', 'Товар добавлен в корзину');
    }
    public function update(Request $request) {
        Cart::update($request->cartId, $request->qty);
```

```
        return redirect()->route('cart')->with('success_message', 'Количество товаров изменено');
    }
    public function destroy($id) {
        Cart::remove($id);
        return back()->with('success_message', 'Товар удален из корзины');
    }
}
```

Листинг 4 – класс контроллера CheckoutController для описания метода оформления заказа

```
class CheckoutController extends Controller {
    public function store(Request $request){
        $order = Order::create($request->all());
        foreach(\Cart::content() as $product) {
            $order->items()->create([
                'product_id' => $product->id,
                'price' => $product->price,
                'qty' => $product->qty,
            ]);
        }
        \Cart::destroy();
        return redirect()->route('cart')->with('success_message', 'Спасибо, заказ оформлен');
    }
}
```

Листинг 5 – класс контроллера OrderController для описания методов управления заказами

```
class OrderController extends Controller {
    public function index() {
        $orderList = Order::all();
        return view('admin.order.index', compact('orderList'));
    }
    public function show($id) {
        $order = Order::find($id);
        return view('admin.order.show', compact('order'));
    }
    public function edit($id) {
        $order = Order::find($id);
        $deliveryList = Delivery::all();
        $paymentList = Payment::all();
        return view('admin.order.edit', compact('order',  'deliveryList', 'paymentList'));
    }
    public function update(Request $request, $id) {
        $order = Order::find($id);
        $data = $request->all();
        $order->update($data);
        $request->session()->flash('success', 'Изменения сохранены');
        return redirect()->route('order.index');
    }
```

```
    public function destroy($id) {
        Order::destroy($id);
        return redirect()->route('order.index')->with('success', 'Заказ удален');
    }
}
```

Листинг 6 – класс контроллера CategoryController – для описания методов управления категориями

```
class CategoryController extends Controller
{
    public function index() {
        $categoryList = Category::all();
        return view('admin.categories.index', compact('categoryList'));
    }
    public function create() {
        return view('admin.categories.create');
    }
    public function store(Request $request) {
        $request->validate(['name' => 'required']);
        Category::create($request->all());
        $request->session()->flash('success', 'Категория добавлена');
        return redirect()->route('categories.index');
    }
    public function edit($id) {
        $category = Category::find($id);
        return view('admin.categories.edit', compact('category'));
    }
    public function update(Request $request, $id) {
        $request->validate([
            'name' => 'required'
        ]);
        $category = Category::find($id);
        $category->slug = null;
        $category->update($request->all());
        $request->session()->flash('success', 'Изменения сохранены');
        return redirect()->route('categories.index');
    }
    public function destroy($id) {
        Category::destroy($id);
        return redirect()->route('categories.index')->with('success', 'Категория удалена');
    }
}
```

Листинг 7 – класс контроллера ProductController для описания методов управления товарами

```
class ProductController extends Controller
{
    public function index() {
        $productList = Product::paginate(20);
```

```
        return view('admin.product.index', compact('productList'));
    }
    public function create() {
        $categoryList = Category::all();
        $brandList = Brand::all();
        return view('admin.product.create', compact('categoryList', 'brandList'));
    }
    public function store(Request $request) {
        $request->validate([
            'name' => 'required',
            'price' => 'required',
            'SKU' => 'required',
                                'quantity' => 'integer',
            'category_id' => 'required|integer',
            'img' => 'required|image|mimes:jpeg,png,jpg,gif,svg|max:2048',
        ]);
        $data = $request->all();
        $data['hit'] = isset($data['hit']) ? 1 : 0;
        $data['stock_status'] = isset($data['stock_status']) ? 1 : 0;
        $data['brand_id'] = $data['brand_id'] == 0 ? null : $data['brand_id'];
        $data['img'] = Product::uploadImage($request);

        Product::create($data);
        $request->session()->flash('success', 'Товар добавлен');
        return redirect()->route('product.index');
    }
    public function edit($id) {
        $product = Product::find($id);
        $categoryList = Category::all();
        $brandList = Brand::all();
        return view('admin.product.edit', compact('product', 'categoryList', 'brandList'));
    }
    public function update(Request $request, $id) {
        $request->validate([
            'name' => 'required',
            'price' => 'required',
            'SKU' => 'required',
            'category_id' => 'required|integer',
            'img' => 'image|mimes:jpeg,png,jpg,gif,svg|max:2048',
        ]);
        $product = Product::find($id);
        $product->slug = null;
        $data = $request->all();
        $data['hit'] = isset($data['hit']) ? 1 : 0;
        $data['stock_status'] = isset($data['stock_status']) ? 1 : 0;
        $data['brand_id'] = $data['brand_id'] == 0 ? null : $data['brand_id'];
        if (isset($data['img'])) {
            $data['img'] = Product::uploadImage($request);
```

```
        }
        $product->update($data);
        $request->session()->flash('success', 'Изменения сохранены');
        return redirect()->route('product.index');
    }
    public function destroy($id) {
        Product::destroy($id);
        return redirect()->route('product.index')->with('success', 'Товар удален');
    }
}
```

Приложение В

(справочное)

Библиографический список

1. Астахова И.Ф. СУБД. Язык SQL в примерах и задачах [Электронный ресурс] / И.Ф. Астахова. – М.: Изд-во ФИЗМАТЛИТ, 2009. – URL: https://avidreaders.ru/book/subd-yazyk-sql-v-primerah-i.html (дата обращения: 01.11.2021).

2. Архитектура web-приложений [Электронный ресурс]. URL: https://frontend-park-mailru.firebaseapp.com/slides/s5/ (дата обращения: 07.11.2021).

3. СТП ВятГУ 102-2004. Общие требования к оформлению и представлению курсовых проектов и работ /Киров: - Изд-во ВятГУ, 2004. – 26 с.

4. Начало работы с Bootstrap [Электронный ресурс]. URL: https://getbootstrap.com/ (дата обращения: 08.11.2021).

5. Начало работы с Laravel [Электронный ресурс]. URL: https://laravel.ru/ (дата обращения: 08.11.2021).