

Министерство науки и высшего образования РФ
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Самарский государственный технический университет»

Институт	Автоматики и информационных технологий
Кафедра	Вычислительная техника

УТВЕРЖДАЮ

И.о. зав. кафедрой	Чуваков А.В.
	(подпись) (ФИО)

«__» _____ 2023 г.

Выпускная квалификационная работа

Обучающегося Зюзина Михаила Алексеевича, ИАИТ, 4 курса, 9 группы
(фамилия, имя, отчество, факультет, курс, группа)

09.03.04 – «Программная инженерия», профиль «Программная инженерия»
(код, направление подготовки (специальности), направленность (профиль) образования)

На тему: Разработка сайта продуктового магазина
(полное наименование темы в соответствии с приказом об утверждении тем ВКР)

Руководитель работы доцент, к.т.н., доцент Ефимушкина Н.В.
(должность, ученая степень, звание, подпись, дата, фамилия, инициалы)

Консультант профессор, д.т.н., профессор Яговкин Н.Г.
(должность, ученая степень, звание, подпись, дата, фамилия, инициалы)

Консультант доцент, к.э.н., доцент Сафронов Е.Г.
(должность, ученая степень, звание, подпись, дата, фамилия, инициалы)

Нормоконтролер доцент, к.т.н., доцент Камальдинова З.Ф.
(подпись, дата, фамилия, инициалы)

Студент _____ Зюзин М.А.
(подпись, дата, фамилия, инициалы)

Самара 2023 г.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ	2
1 ТИПЫ САЙТОВ И СРЕДСТВА ИХ РАЗРАБОТКИ	5
1.1 Классификация сайтов	5
1.2 Средства разработки сайтов	8
1.3 Структуры сайтов	14
1.4 Выводы	16
2 ПРОЕКТИРОВАНИЕ САЙТА ИНТЕРНЕТ-МАГАЗИНА	17
2.1 Анализ предметной области	17
2.2 Разработка серверной части сайта	18
2.3 Разработка пользовательской части	33
2.4 Подсистема анализа продаж	39
2.5 Руководство администратора	42
3 ЭКОНОМИЧЕСКАЯ ЧАСТЬ	47
3.1 Описание продукта	47
3.2 Оценка затрат на разработку программного продукта	47
4 ОХРАНА ТРУДА	47
4.1 Анализ и проектирование производственной среды	47
4.2 Проектирование рабочего места	47
4.3 Выбор оборудования	47
4.4 Проектирование схемы подключения оборудования	47
ЗАКЛЮЧЕНИЕ	47
Список используемой литературы	49
Графический материал	66

ВВЕДЕНИЕ

В современном мире, невозможно представить компанию, которая не имеет своего сайта. Сайт предоставляет компании очень много преимуществ, таких как:

- Большой охват клиентов;
- Доступность информации о компании;
- Упрощение связи между клиентом и администратором;
- Быстрое обновление информации о товарах, услугах и новостях компании;
- Доступность каталога товаров;
- Репутационная составляющая.

Люди по всему земному шару проводят огромное количество времени в интернете. В глобальной сети сайт или интернет-страница могут быть представлены разными способами. Это определяется задачами, для которых он создается. Для упорядочения информации о сайтах применяется их классификация по целому ряду признаков, наиболее распространенными из которых являются следующие: тип решаемых задач, охват, реализуемые технологии, размер, структура.

По первому признаку выделяют три класса: информационные (новостные, со статьями, блоги), социальные (социальные сети, форумы, сайты знакомств) и веб-сервисы (почтовые, хостинги, онлайн-инструменты). По охвату пользователей сайты бывают ориентированы на большой и малый охваты. По реализуемым технологиям, выделяют три класса: статические, динамические, смешанные. По размеру все сайты делятся на одностраничные и многостраничные. По структуре сайты разделяются на: линейные, иерархические, паутинные, гибридные.

Любой сайт состоит из пользовательской и серверной части (frond-end и back-end). На пользовательской, расположены текст, кнопки, панели,

изображение, видео. За логику и функционирование сайта отвечает back-end, который скрыт от пользователя.

Реализация Front-end части происходит с помощью таких языков как: HTML, CSS, Javascript. Для реализации Back-end части, используются языки: PHP, C#, Ruby, Java.

Основой Back-end является база данных – содержащая основную информацию из конкретной предметной области. Наибольшее распространение получили реляционные базы.

Для управления этими базами, используется система управления базами данных (СУБД). Наиболее популярными СУБД в настоящее время являются: Oracle, MySQL, PostgreSQL, Microsoft SQL Server, Microsoft Access.

Целью выпускной квалификационной работы является разработка сайта продуктового магазина. Для достижения этой цели, необходимо решить следующие задачи:

- 1) определить целевое назначение и функции сайта;
- 2) спроектировать структуру сайта и базы данных;
- 3) выбрать средства разработки;
- 4) разработать Back-end часть;
- 5) разработать Front-end часть.

Исходными данными для решения поставленных задач являются:

- 1) номенклатура товаров магазина;
- 2) информация о каждом товаре;

Результатами должны быть:

- 1) база данных, содержащая информацию о следующих объектах:
 - товарах,
 - услугах,
 - статистике продаж.
- 2) приложение, обеспечивающее реализацию следующих функций:

- предоставление информации о товарах и акциях, проводимых магазином,
- выполнение онлайн-заказа,
- сбор статистики продаж по видам товаров и периодам работы.

После тестирования заказчиком сайт предполагается использовать в одной из торговых организаций города Самары.

1 ТИПЫ САЙТОВ И СРЕДСТВА ИХ РАЗРАБОТКИ

1.1 Классификация сайтов

«Сайт – это информационная единица в интернете, ресурс из веб-страниц, которые объединены общей темой и связаны друг с другом с помощью ссылок. Он регистрируется на одно юридическое или физическое лицо и обязательно привязан к конкретному домену, являющемуся его адресом. Сайт может состоять как из одной, так и из огромного количества страниц. Каждая страница – это текстовый файл или их набор, написанный на специальном языке разметки или программирования (HTML, PHP, CSS и пр.). После загрузки на компьютер файлы обрабатываются в браузере. В конечном итоге пользователь видит загруженную страницу сайта. В настоящее время существует широкое разнообразие сайтов, которые могут быть разбиты на классы по следующим признакам» [5].

По типу решаемых задач сайты принято делить на следующие классы, изображенные на рисунке 1.1.

- информационные;
- социальные;
- веб-сервисы.

По информации, предоставляемой сайтами, они делятся на:

- новостные, которые информируют о событиях, произошедших в социальной жизни человека;
- со статьями, содержащие большое количество текстов, связанных определенной тематикой;
- для ведения блогов, позволяющие комментировать и участвовать в дискуссиях.



Рисунок 1.1 - Классификация сайтов по типу решаемых задач

Среди социальных сайтов выделяют следующие классы:

- социальные сети, предназначены для общения пользователей;
- форумы – площадки, созданные для обсуждения определенных тематик;
- сайты знакомств.

Веб-сервисы подразделяются на следующие классы:

- почтовые сервисы, обеспечивающие общение между пользователями с помощью текстов или файлов;
- хостинги – места на серверах, предназначенные для размещения сайтов в интернете;
- онлайн-инструменты, представляющие пользователям сетевые ресурсы.

По размеру, бывают двух типов:

- одностраничные – состоящие из одной страницы;
- многостраничные – содержащие в себе множество страниц.

По реализуемым технологиям сайты делятся на следующие классы:

- статические, которые имеют постоянную структуру и содержание и поставляются в готовом виде;
- динамические, содержащие изменяемые страницы. Они генерируются автоматически, и адаптируются к интересам каждого пользователя;
- смешанные, объединяющие в себе динамические и статические страницы;
- флеш-сайты – устаревшая технология, созданные на основе программы Adobe Flash;

По охвату пользователей сайты разделяются на две категории:

- предназначенные для небольшого количества, пользователей. При проектировании таких сайтов, не учитывается возможность появления большого количество пользователей;
- с большим охватом, учитывающие возможность посещения в одно и тоже время большого количества пользователей, что значительно увеличивает нагрузку на аппаратуру.

Коммерческие подразделяются на следующие классы:

- интернет-магазины – предназначены для продажи товаров или услуг. Имеют сложный функционал, и содержат каталог с описанием продуктов, выбор различных способов оплаты и корзину;
- сайты услуг – содержат сведения о услугах предоставляемыми определенными компаниями и их контактными данными;
- корпоративные порталы – ресурсы, предназначенные для предоставления информации о компании, её услугами или товарами.

Таким образом, существует широкое разнообразие сайтов, которые различаются по целому ряду признаков: назначению, типам решаемых задач, охвату пользователей, используемым технологиям и размерам. Наиболее распространенными являются сайты, направленные на получение прибыли – интернет-магазины.

1.2 Средства разработки сайтов

Общепринятая структура сайта включает в себя две основные составляющие: пользовательскую (frond-end) и серверную (back-end).

Основной задачей первой является привлечение и удержание пользователей с помощью соответствующего дизайна. При разработке этой части решаются следующие задачи:

- а) Проектирование дизайна;
- б) Верстка;

- c) Связь с back-end частью.

Для создания дизайна используются различные средства для создания дизайна:

- a) Готовые компоненты интерфейса, которые потом конвертируются в код,
- b) Языки программирования высокого уровня,
- c) Специальные конструкторы сайтов.

Первый и третий способы не обеспечивают возможности тонкой настройки компонентов, а второй позволяет создавать сайты любого типа с любым функционалом. Для решения поставленных в работе задач целесообразно использовать именно этот способ.

Основными языками программирования для веб-разработки являются:

- a) «HTML - стандартизированный язык гипертекстовой разметки документов для просмотра страниц в браузере. Веб-браузеры получают HTML документ с сервера по протоколам HTTP/HTTPS или открывают с локального диска, этот документ реализует пользовательский интерфейс» [6].
- b) «CSS - формальный язык описания внешнего вида документа (веб-страницы), написанного с использованием языка разметки. Он может применяться к любым XML-документам, например, к SVG или XUL» [7].
- c) «JavaScript поддерживает объектно-ориентированный, императивный и функциональный стили. Является реализацией спецификации ECMAScript. JavaScript обычно используется как встраиваемый язык для программного доступа к объектам приложений. Наиболее широкое применение он находит в браузерах как язык сценариев для придания интерактивности веб-страницам» [8].

«Для CSS и JavaScript существует множество фреймворков. Наиболее популярным из них является Bootstrap 4. Это - открытый и бесплатный HTML,

CSS и JS фреймворк, который используется веб-разработчиками для быстрой вёрстки адаптивных дизайнов сайтов и веб-приложений» [9].

Связь с серверной частью обычно обеспечивается путем обмена Json файлами. Json-файл - текстовый формат обмена данными, основанный на JavaScript.

Как и многие другие текстовые форматы, JSON легко читается людьми.

Back-end включает в себя внутреннюю часть сайта и сервера. Основные процессы, которые реализует back-end:

- a) Получение данных с клиентской стороны;
- b) Обработка их сервером;
- c) Возвращение обработанных данных обратно пользователю.

«Наиболее популярной архитектурой распределенных систем, таких как World Wide Web, в настоящее время является REST (Representational state transfer). Каждая единица информации в ней однозначно определяется глобальным идентификатором, таким как URL. Каждая URL, в свою очередь, имеет строго заданный формат» [10]. Для отправки и принятия URL-запросов, в большинстве сайтов используется протокол HTTP.

«Этот протокол лежит в основе обмена данными в Интернете и позволяет получать различные ресурсы, например, HTML-документы. HTTP является протоколом клиент-серверного взаимодействия, и инициирует запросов к серверу самим получателем, обычно веб-браузером. Полученный документ может состоять из различных частей, например, отдельно полученного текста, описания структуры документа, изображений, видеофайлов, скриптов и многого другого» [11].

Наиболее популярными языками для разработки серверной части сайтов являются следующие.

- a) «PHP — распространённый язык программирования общего назначения с открытым исходным кодом. Он специально сконструирован для веб-разработок и его код может внедряться непосредственно в HTML» [12].

б) «С# и ASP.NET. С# - один из наиболее востребованных языков программирования. Фреймворк ASP.NET разработан для С# и платформы .NET. Он позволяет создавать веб приложения» [13].

с) «Ruby и Ruby on Rails, динамический, рефлексивный, интерпретируемый высокоуровневый язык программирования. Он обеспечивает независимость от операционной системы реализацией многопоточности, динамическую типизацию, сборщик мусора и многие другие возможности» [14].

д) «Java — универсальный объектно-ориентированный язык, который часто используется для веб-разработки. Программы на Java транслируются в байт-код, который затем выполняется виртуальной машиной Java» [15]. Реализация back-end логики на языке Java, обеспечивает множество преимуществ по сравнению с другими языками, а фреймворк Spring, расширяет его возможности, особенно в веб разработке.

Основная информация, необходимая для работы сайта, в крупных проектах хранится в базах данных. Для управления этими базами используются СУБД:

Баз данных [БД] – набор постоянно хранимой информации или данных, которые хранятся в электронном виде и используются системами программных продуктов. Наиболее распространенной архитектурой баз данных является реляционная. Базы данных с реляционной структурой совмещают в себе преимущества иерархических и объектно-ориентированных структур, благодаря чему, является универсальной архитектурой, подходящей под большинство задач. Пример такой архитектуры изображен на рисунке 1.2

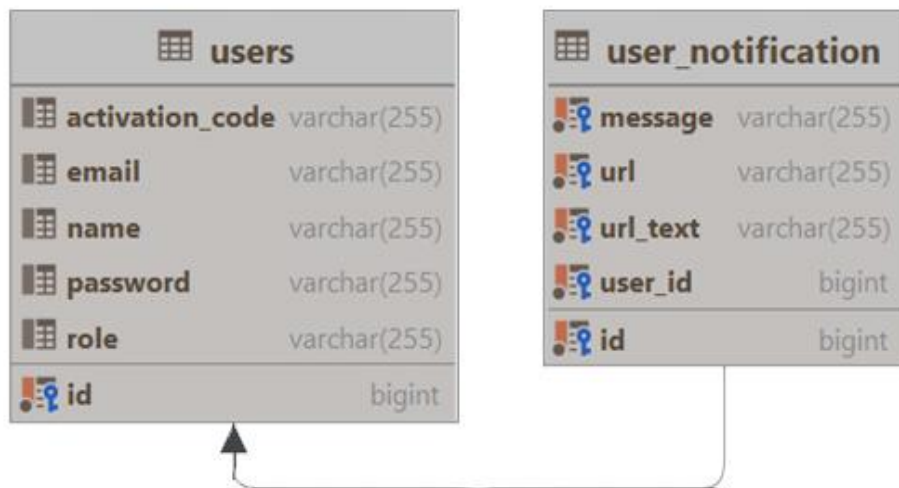


Рисунок 1.2 - Реляционная структура базы данных

Работа с БД обеспечивает система управления базами данных (СУБД). Наиболее популярными СУБД в настоящее время являются:

- a) Oracle – популярный у разработчиков продукт, который способен обрабатывать данные больших объемов, поддерживает язык SQL;
- b) MySQL, который использует стандартную форму SQL, имеет интуитивно понятный интерфейс и позволяет хранить большое число записей в таблицах. Он обладает высокой скоростью, поддерживает большинство ОС и гарантирует безопасность данных;
- c) PostgreSQL – масштабируемая, реляционная база данных, работающая на большинстве ОС, она имеет большое количество функций;
- d) Microsoft SQL Server – популярная СУБД, которая применяется только для ОС Windows, не имеет графического интерфейса, но также поддерживает SQL запросы;
- e) Microsoft Access - NoSQL СУБД, которая сочетает в себе реляционное ядро базы данных Microsoft Jet и имеет простейший интерфейс.
- f) Наиболее перспективной для разработки веб-сайтов представляется PostgreSQL.

Для создания приложений веб-сайтов наиболее популярными являются следующие IDE (Интегрированные среды разработки) на Java:

а) IntelliJ IDEA – это интеллектуальная среда, учитывающая контекст. Она предназначена для разработки разнообразных приложений на Java и других языках JVM. Кроме того, IntelliJ IDEA Ultimate помогает в разработке веб-приложений: она предлагает эффективные встроенные инструменты, поддержку JavaScript и связанных с ним технологий, а также расширенную поддержку таких популярных фреймворков, как Spring и Spring Boot. Бесплатные плагины позволяют дополнительно расширить возможности IntelliJ IDEA и использовать ее для работы с другими языками программирования, в том числе Go, Python, SQL, Ruby и PHP. [16].

б) «Eclipse – свободная интегрированная среда разработки модульных кроссплатформенных приложений. Развивается и поддерживается Eclipse Foundation» [17].

Одним из главных преимуществ IntelliJ IDEA перед Eclipse, является ultimate версия, которая поставляется на коммерческой основе, с возможностью получения бесплатных учебных лицензий и имеет встроенные инструменты для разработки веб-приложений

Таким образом, сайты традиционно используют две основные составляющие: пользовательскую и серверную. Для их разработки используются различные средства. Так, пользовательская часть может быть реализована на JavaScript с фреймворком Bootstrap 4. Серверная, обычно, должна иметь наиболее популярную архитектуру распределенных систем REST. В качестве среды разработки наиболее популярной является IntelliJ IDEA для языка Java с фреймворком Spring Web MVC. Наиболее перспективной для разработки веб-сайтов считается СУБД PostgreSQL.

1.3 Структуры сайтов

Структура сайта — это логическая связка страниц, расположение конкретных элементов дизайна, которые должны следовать стандартам разработки. В настоящее время наибольшее распространение получили следующие структуры:

- a) Линейная,
- b) Иерархическая,
- c) Паутинная,
- d) Гибридная.

Линейная – самая простая структура, в которой все связи между страницами последовательные. Общий вид такой структуры представлен на рисунке 1.3. Такая схема применяется в презентациях и портфолио. В ней, страницы представляются цепочкой. Описываемая схема не удобна для перехода между страницами. Основное достоинство рассматриваемого сайтов такого вида состоит в простоте.



Рисунок 1.3 - Линейная структура сайта

Иерархия – структура, в которой связи между страницами выполнены в форме дерева. Она приведена на рисунке 1.4. При такой структуре, каждому разделу отводится отдельная ветвь.

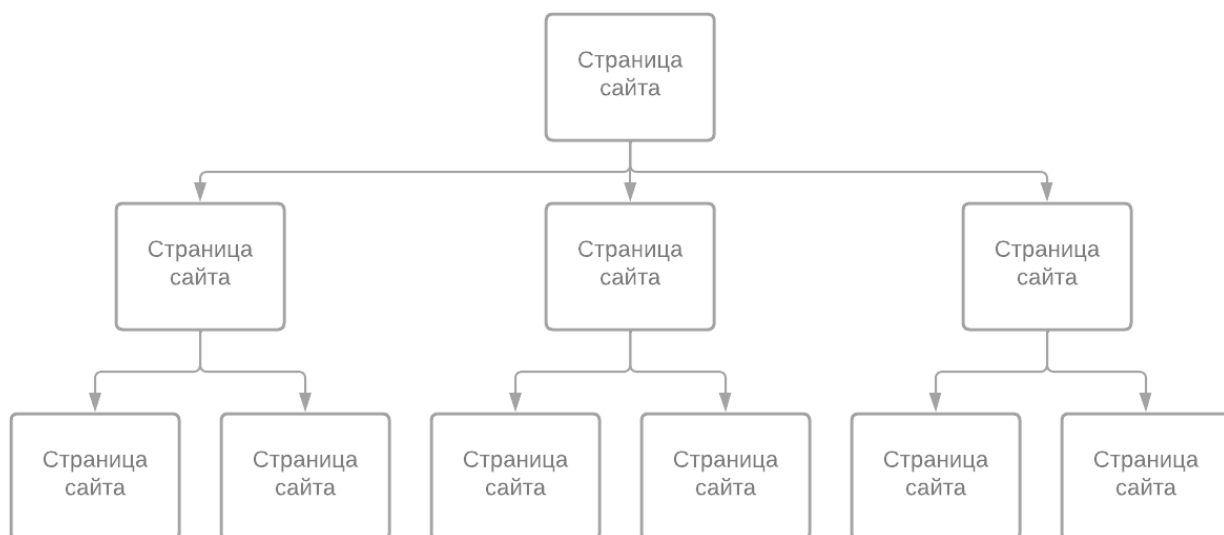


Рисунок 1.4 - Иерархичная структура сайта

Паутина – структура, в которой каждая страница связана со всеми остальными, как показано на рисунке 2.3. Такая схема часто применяется на информационных порталах. Она имеет сложные логические связи.

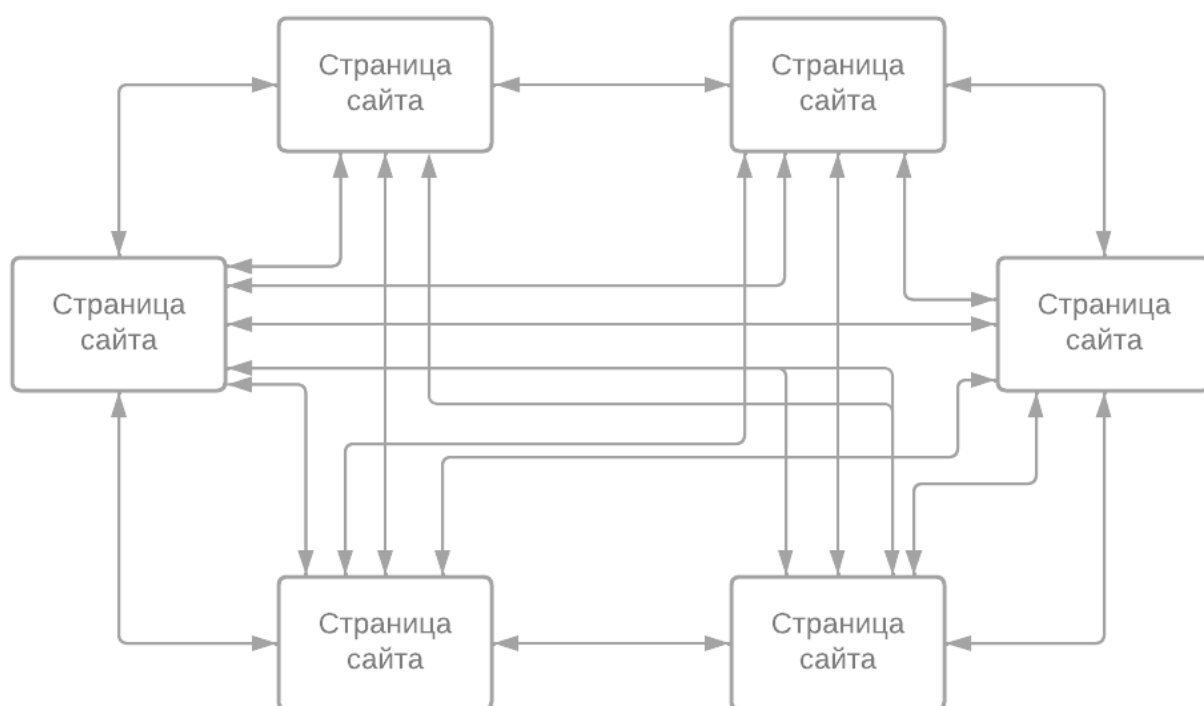


Рисунок 1.5- - Паутинная структура сайта

Приведенные структуры являются базовыми и редко используются в чистом виде. В современных сайтах обычно применяют их комбинацию. Такой вид

структуры называется гибридной. Именно она применяется при проектировании большинства сайтов.

Таким образом, существует три основных типа структур сайтов: линейная, иерархическая и паутинная, которые в чистом виде обычно не применяются, наиболее распространение получила гибридная структура сайта, которая является сочетанием основных. Она позволяет реализовать сайт любой сложности, с любыми возможными связями между страницами.

1.4 Выводы

Существует широкое разнообразие сайтов, которые различаются по целому ряду признаков: назначению, типам решаемых задач, охвату пользователей, используемым технологиям и размерам. Наиболее распространенными являются сайты, обеспечивающие продажу товаров или предоставление услуг – интернет-магазины.

Разработка сайта разделяется две основные составляющие: пользовательскую и серверную. Для их разработки используются различные средства. Так, пользовательскую часть целесообразно реализовать на JavaScript с фреймворком Bootstrap 4. Серверная часть должна иметь наиболее популярную архитектуру распределенных систем REST. В качестве среды разработки предлагается использовать IntelliJ IDEA для Java с фреймворком Spring Web MVC. Наиболее перспективной для разработки веб-сайтов признана PostgreSQL. Для обеспечения наибольшего комфорта пользователей, структура сайта должна быть гибридной.

2 ПРОЕКТИРОВАНИЕ САЙТА ИНТЕРНЕТ-МАГАЗИНА

2.1 Анализ предметной области

«Клиенты магазина традиционно делятся на две категории: посетители и покупатели. Первые только рассматривают товары, а вторые – приобретают. Основной задачей магазина, является перевод посетителей в покупатели. Для ее решения необходимо проанализировать основные признаки и психологию покупателя. По данным портала RBC, на 2021 год, типичный покупатель продуктового интернет-магазина обладает следующими свойствами» [18]:

- a) Женщина (по статистике количество женщин, совершающих покупки в интернете, преобладает над количеством мужчин);
- b) Приблизительный возраст потенциального покупателя составляет 25-34 года (на эту возрастную категорию пришлось 26,6% покупателей);
- c) Средний доход составляет меньше 20 тыс. руб.;
- d) Место проживания покупателей – населенные пункты с числом жителей меньше 100 тыс. (на покупки из таких городов приходится 35,8%);
- e) Большинство покупательниц – замужем (63,3%) и не имеют детей (40,2%);
- f) Они посещают интернет-магазины с мобильных устройств, в промежутке от 6 до 10 часов вечера.

Проанализировав этот портрет покупателя, можно сделать вывод о том, что магазин должен предлагать продукты первой необходимости, а также выдерживать большой трафик в вечерние часы.

«По данным Роспотребнадзора, в перечень продуктов первой необходимости входят следующие товары» [19]:

- зеленые овощи;
- яблоки;
- рис;

- бобовые;
- орехи;
- рыба;
- кисломолочные продукты;
- крупы;
- мясные продукты: свинина, говядина, баранина, куриное мясо;
- куриные яйца;
- изделия из пшеничной муки;
- чай;
- макаронные изделия;
- картофель;
- репчатый лук;
- белокочанная капуста;
- морковь.

Следующие продукты, не относятся к категории первой необходимости, поэтому они могут отсутствовать в номенклатуре товаров:

- мюсли;
- фитнес-батончики;
- йогурты с наполнителями;
- обезжиренные молочные продукты;
- соки и нектары в коробках;
- соевые продукты.

2.2 Разработка серверной части сайта

Серверная часть написана на языке Java, с использованием следующих библиотек: Spring Boot, Spring Web, Spring MVC, Spring Security, Spring Thymleaf, Lombok, Mapstruct, Spring mail Hibernate, Flyway.

При проектировании сайта, в первую очередь необходимо выявить допустимые действия пользователя и администратора. Анализ предметной области показал, что доступ к сайту могут получить три типа акторов:

- 1) администратор;
- 2) зарегистрированный покупатель;
- 3) незарегистрированный клиент.

Диаграмма вариантов использования для этих акторов приведена на рисунке 2.1.

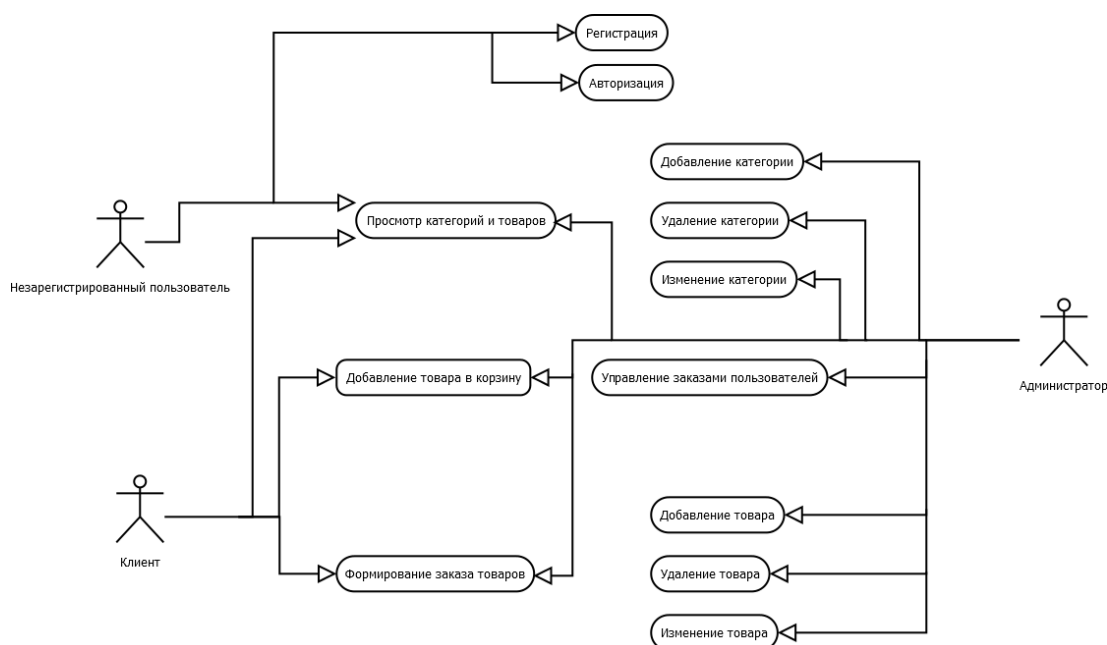


Рисунок 2.1 – Диаграмма вариантов использования

Пользователи, которые не прошли авторизацию имеют самый низкий уровень доступа. Они могут посещать только главную страницу, страницу с товарами и окно регистрации.

Роль – клиент присваивается покупателю. Он имеет доступ к личной корзине, возможности добавления продукта в корзину, формирования заказа, а также отслеживания состояния заказов.

Роль, которой доступен наибольший функционал сайта – администратор. Ему разрешается добавлять, удалять или редактировать карточки с товарами,

получать информацию о заказах всех пользователей, а также изменять их статусы.

Проектирование серверной части разделяется на три основных этапа:

- 1) проектирование базы данных;
- 2) выбор архитектуры;
- 3) выбор схемы разделения данных.

Основной составляющей серверной части является база данных, в которой должна храниться информация о товарах, покупателях и статистике покупок. Она должна содержать следующие основные сущности:

- 1) пользователь, которым имеет электронный адрес, имя, пароль и указатель на уровень доступа к интернет-магазину;
- 2) продукт, у которого есть название, описание, цена и изображение;
- 3) заказ, в котором хранятся сведения об адресе, общей сумме заказа, статусе, продуктах, которые в него входят, их количестве, и цене;
- 4) корзина, в которой хранится информация о товарах, которые добавил каждый пользователь в нее.

Кроме того, для хранения дополнительной информации используются следующие вспомогательные сущности:

- 1) отношение продукта к категории – данная сущность позволяет связать друг с другом сущности продукта и категории;
- 2) отношение продуктов к корзине – сущность, позволяющая объединить сущности продукта и корзины;
- 3) детали заказа – сущность, в которой хранятся подробности о заказе;
- 4) скидки – сущность, хранящая информацию о цене продукта со скидкой;
- 5) уведомления пользователей – сущность, содержащая в себе уведомления пользователей;

- 6) отзыв – который хранит информацию о отзыве товару, который оставил пользователь;
- 7) статистика посещений – сущность, в которую записывается частота посещений пользователями товаров;
- 8) статистика покупок – сущность, в которую записываются покупки, совершаемые пользователями;
- 9) статистика частоты добавления в корзину – сущность, в которую записывается информация, о добавлении товаров в корзину.

Для проектирования базы данных построена модель сущность-связь, которая приведена на рисунке 2.2. В модели используются следующие обозначения сущностей:

- 1) Покупатель – users;
- 2) Уведомления – user_notification;
- 3) Продукт – products;
- 4) Отношение продукта к категории – product_category;
- 5) Отношение продукта к корзине – buckets_product;
- 6) Отзыв о продукте – product_review;
- 7) Категория – categories;
- 8) Скидка – discount;
- 9) Заказ – orders;
- 10) Детали заказа – orders_details;
- 11) Корзина – bucket;
- 12) Статистика посещений – visit_stats;
- 13) Статистика покупок – buy_stats;
- 14) Статистика частоты добавления в корзину – frequency_add_to_cart_stats.

Покупатель (User) связан со следующими сущностями:

- 1) Product_review – один ко многим;
- 2) User_notification – один ко многим;

- 3) Bucket – один к одному.

Продукт (Products) связан с сущностями:

- 1) Product_review – один ко многим;
- 2) Bucket – многие ко многим через таблицу bucket_products;
- 3) Category – многие ко многим, через таблицу product_category;
- 4) Orders_details – один к одному;
- 5) Discount – один к одному;
- 6) Visit_stats – один к одному;
- 7) Buy_stats – один к одному;
- 8) Frequency_add_to_cart_stats – один к одному.

Заказ (Order) связан со следующими сущностями:

- 1) Order_details – один ко многим;
- 2) Users – один к одному.

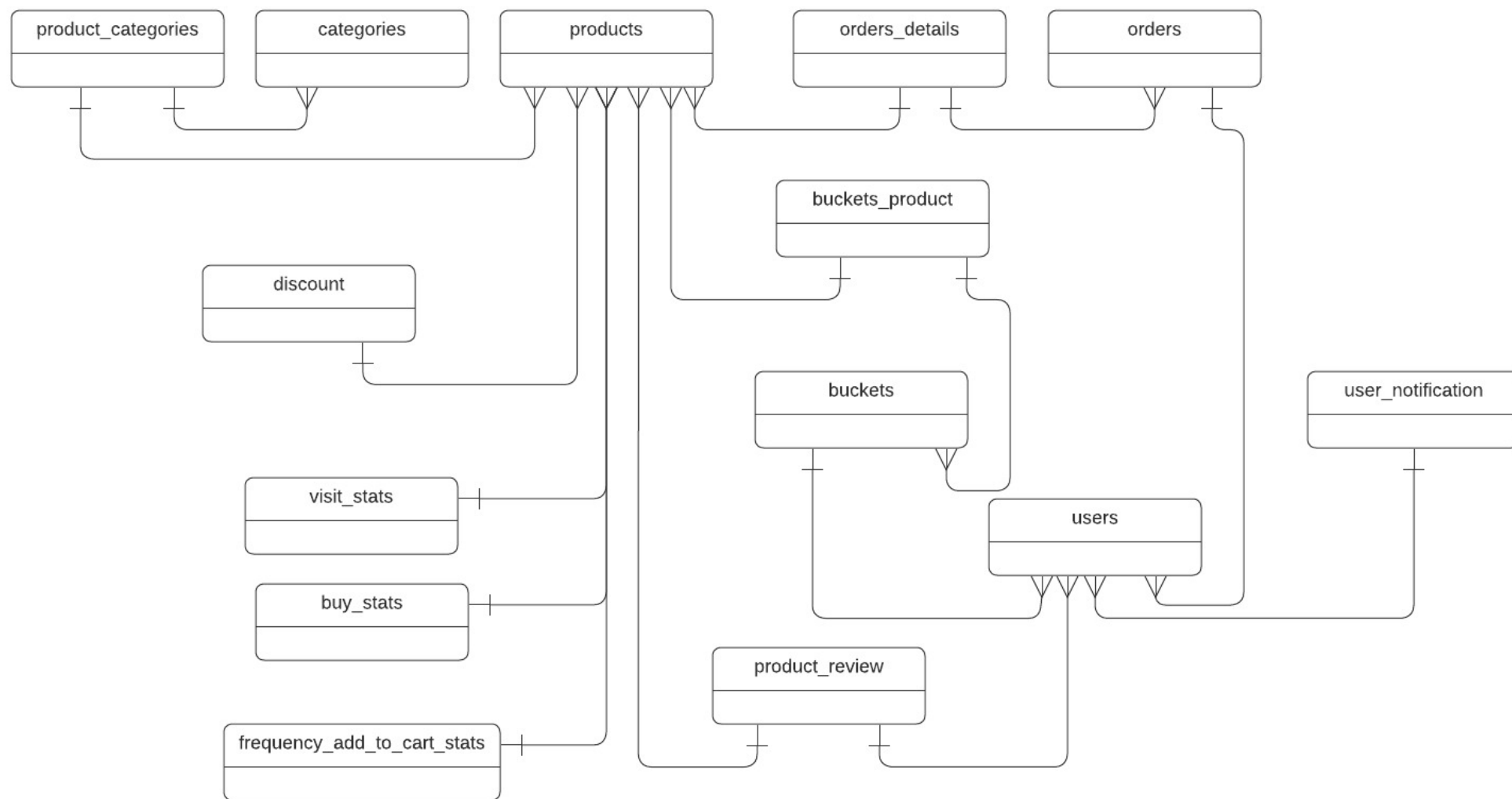


Рисунок 2.2 - Er-диаграмма проектируемой базы данных

По рисунку 2.2 построена даталогическая модель базы данных. В ней используются следующие таблицы:

- 1) Покупатель;
- 2) Уведомления покупателя;
- 3) Продукт;
- 4) Отношение продукта к категории;
- 5) Отношение продукта к корзине;
- 6) Отзыв о продукте;
- 7) Категория;
- 8) Скидка;
- 9) Заказ;
- 10) Детали заказа;
- 11) Корзина;
- 12) Статистика посещений;
- 13) Статистика покупок;
- 14) Статистика частоты добавления в корзину.

Содержимое таблиц приведено в табл. 2.1-2.14:

Таблица 2.1 - Пользователь (User)

Наименование	Содержание	Тип
Id	Уникальный идентификатор.	BigInt
Activation_code	Код активации для подтверждения владения пользователем указанной при регистрации электронный почты.	Varchar (255)
Email	Электронный адрес пользователя, который в дальнейшем будет использоваться для отправки уведомлений.	Varchar (255)

Name	Имя пользователя, который у него будет в профиле, а также, будет отображаться в отзывах на продукты.	Varchar (255)
Password	Пароль, под которым пользователь будет заходить в свой аккаунт.	Varchar (255)
Role	Роль пользователя на сайте.	Varchar (255)

Таблица 2.2 - Уведомление пользователя (User_notification)

Наименование	Содержание	Тип
Id	Уникальный идентификатор.	BigInt
Message	Сообщение, содержащее реакцию на действие пользователя.	Varchar (255)
url	Поле url для перехода на страницу.	Varchar (255)
url_text	Текст, по клику на который будет осуществлен переход на новую страницу по ссылке из поля url.	Varchar (255)
User_id	Уникальный идентификатор пользователя, которому адресовано сообщение.	Varchar (255)

Таблица 2.3 - Продукты (Products)

Наименование	Содержание	Тип
Id	Уникальный идентификатор.	BigInt
Description	Описание товара.	Varchar (255)
Image	Url, который указывает на местоположение картинки.	Varchar (255)

Price	Цена продукта.	double
Title	Название продукта.	Varchar (255)

Таблица 2.4 - Скидка (Discount)

Наименование	Содержание	Тип
Id	Уникальный идентификатор.	BigInt
Discount_price	Цена продукта со скидкой.	double
Product_id	Уникальный идентификатор продукта.	Varchar (255)

Таблица 2.5 - Отзыв о продукте (Product_review)

Наименование	Содержание	Тип
Id	Уникальный идентификатор.	BigInt
Review	Отзыв пользователя.	Varchar (255)
Stars	Количество звезд, которые пользователь поставил товару.	Integer
Product_id	Уникальный идентификатор продукта, на который пользователь оставил отзыв.	BigInt
User_id	Уникальный идентификатор пользователя, который оставил отзыв.	BigInt

Таблица 2.6 - Категории (Categories)

Наименование	Содержание	Тип
Id	Уникальный идентификатор.	BigInt
title	Наименование категории.	Varchar (255)

Таблица 2.7 - Отношение продукта к категории (Products_Categories)

Наименование	Содержание	Тип
Product_id	Уникальный идентификатор продукта	BigInt
Category_id	Уникальный идентификатор категории	BigInt

Таблица 2.8 - Заказы (Orders)

Наименование	Содержание	Тип
Id	Уникальный идентификатор.	BigInt
Address	Адрес доставки заказа.	Varchar (255)
Created	Дата создания заказа.	timestamp
Status	Статус заказа.	Varchar (255)
Sum	Общая сумма заказа.	Double
Updated	Дата обновления статуса заказа.	Timestamp
User_id	Уникальный идентификатор пользователя, который совершил заказ.	BigInt

Таблица 2.9 - Детали заказа (OrderDetails)

Наименование	Содержание	Тип
Id	Уникальный идентификатор.	BigInt
Amount	Количество продукта, в единицах.	integer
Price	Цена с учетом количества.	Double
Product_id	Уникальный идентификатор продукта.	BigInt
Order_details_id	Уникальный идентификатор заказа.	BigInt

Таблица 2.10 - Корзина (Buckets)

Наименование	Содержание	Тип
Id	Уникальный идентификатор.	BigInt
User_id	Уникальный идентификатор пользователя корзины.	BigInt

Таблица 2.11 - Отношение продукта к корзине (Buckets_product)

Наименование	Содержание	Тип
Bucket_id	Уникальный идентификатор корзины.	BigInt
Product_id	Уникальный идентификатор продукта, который находится в корзине.	BigInt

Таблица 2.12 - Статистика посещений (Visit_stats)

Наименование	Содержание	Тип
Id	Уникальный идентификатор.	BigInt
Created	Дата создания.	Timestamp
Product_id	Уникальный идентификатор продукта.	BigInt

Таблица 2.13 - Статистика совершения покупок (Buy_stats)

Наименование	Содержание	Тип
Id	Уникальный идентификатор.	BigInt
Amount	Количество продукта.	Integer
Created	Дата создания.	Timestamp
Product_id	Уникальный идентификатор продукта.	BigInt

Таблица 2.14 - Статистика добавления в корзину (Frequency_add_to_cart_stats)

Наименование	Содержание	Тип
--------------	------------	-----

Id	Уникальный идентификатор	BigInt
Created	Дата создания	Timestamp
Product_id	Уникальный идентификатор продукта	BigInt

Для интеграции базы данных была выбрана СУБД PostgreSQL, а для реализации JavaPersistence API - библиотека Hibernate, которая позволит сократить объемы низкоуровневого программирования при работе с базой. С помощью этой библиотеки, в приложении были созданы классы, представленные на рисунке 2.3. Каждый из них отражает конкретную сущность в базе данных. Для упрощения взаимодействия клиентской и серверной частей, в разрабатываемом сайте, используются два вида моделей: domain и dto.

Domain model отражает конкретные сущности БД (рисунок 2.3);

Dto model – модель данных, которая передается из одного слоя приложения в другой. В контексте разрабатываемого интернет-магазина, она используется для передачи информации из серверной части в клиентскую, и сокрытия данных из domain model (рисунок 2.4).

Так как domain и dto модели различаются, необходимо реализовать конвертацию моделей. Для этого используется библиотека MapStruct, которая позволяет проводить неявную трансформацию моделей, опираясь на аннотации Hibernate и SpringData, создавая отдельные интерфейсы с аннотацией @Mapper.

Для упрощения дальнейшей разработки, и уменьшения объема кода, была использована библиотека Lombok, благодаря которой, пустые конструкторы, конструкторы со всеми параметрами, геттеры и сеттеры заменяются аннотациями. Так же, эта библиотека позволяет использовать аннотацию @builder, благодаря которой при создании объектов класса появляется возможность оставлять некоторые поля пустыми, без необходимости создавать под такие операции отдельные конструкторы.

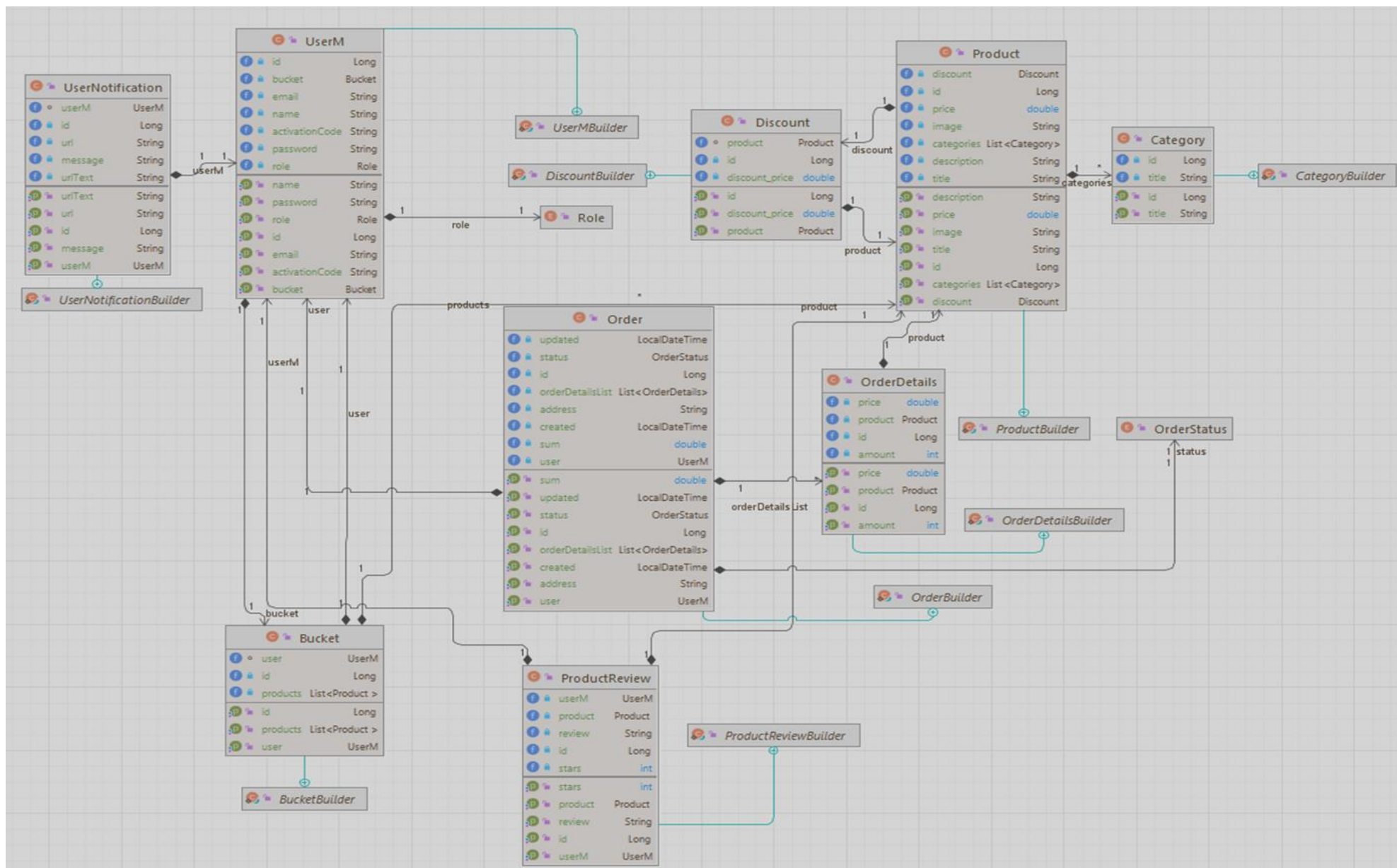


Рисунок 2.3 - Uml диаграмма domain моделей

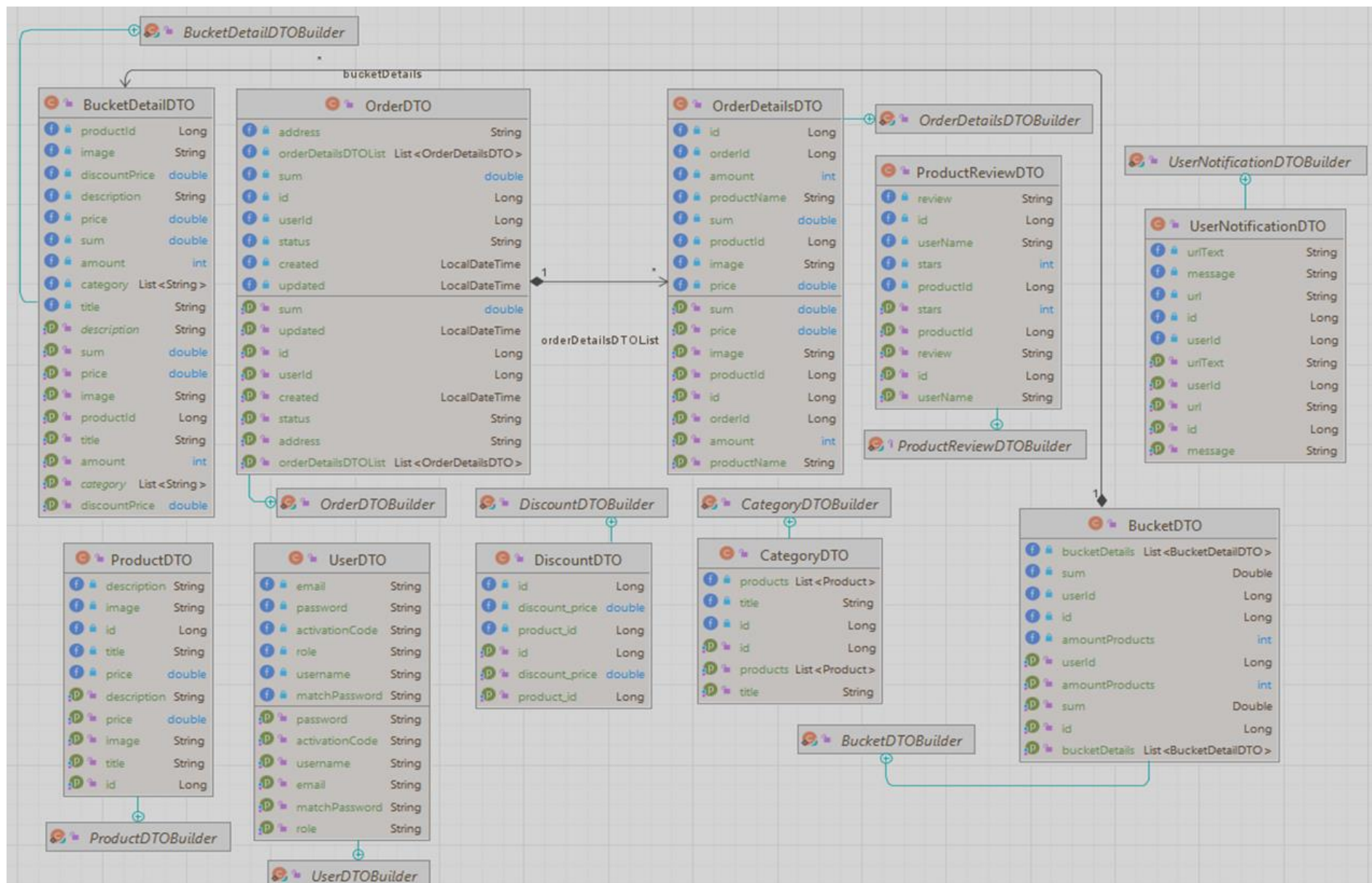


Рисунок 2.4 - Uml диаграмма dto моделей

Для проектирования интернет-магазина была выбрана REST архитектура. В ней, каждая единица информации задается url'ом. Используемые url-адреса приведены в Приложении.

В качестве схемы разделения данных в приложении, была выбрана схема Model-View-Controller (MVC). Она представляет функционирование сайта с помощью трех компонентов:

- a) Модель;
- b) Представление;
- c) Контроллер.

Модель предоставляет данные и методы работы с ними: запросы в базу данных, и проверку на корректность. Представление получает данные из модели и отправляет их пользователю. Оно не обрабатывает введенные данные пользователя. Контроллер обеспечивает связь между пользователем и системой.

В разрабатываемом сайте, существуют репозитории, связанные с соответствующими таблицами. Они перечислены в Приложении.

2.3 Разработка пользовательской части

Разработка пользовательской части выполнена на языках HTML, CSS и Javascript, с использованием фреймворка Bootstrap 4 – для CSS и библиотеки Gstatic для JavaScript. Bootstrap 4 позволяет реализовать адаптивную верстку, а также содержит большое количество пресетов, благодаря которым ускоряется проектирование дизайна сайта. Библиотека Gstatic используется для вывода стилизованных графиков, на основе данных, получаемых с сервера.

Для развертывания сайта необходимы следующие минимальные аппаратные требования к компьютеру:

- a) Тактовая частота процессора: не менее 1 ГГц;
- b) Оперативная память: не менее 256 МБ;
- c) Разрешение экрана: не менее 1024×768.

Поддерживаемые настольные операционные системы:

- a) Microsoft Windows 10, 32/64 бита;
- b) Microsoft Windows 11 Apple OS X 10.7 и более поздние;
- c) Ubuntu 16.04 и более поздние.

Устройства, с которых можно перейти на сайт: ПК, телефоны с операционными системами android (5.0 и более поздние) и IOS (7.0 и более поздние).

Поддерживаемые браузеры для операционных систем ПК:

- a) Google Chrome 28.0 и более поздние;
- b) Mozilla Firefox 47.0 и более поздние;
- c) Apple Safari 8.0 и более поздние;
- d) Opera 36.0 и более поздние;
- e) Safari.

Поддерживаемые браузеры для мобильных устройств на базе Android и IOS:

- a) Android Browser (для Android);
- b) Safari (для iOS);
- c) Яндекс.браузер;
- d) Google Chrome.

Главная страница представлена на рисунке 2.5. Она содержит контактные данные и название магазина.

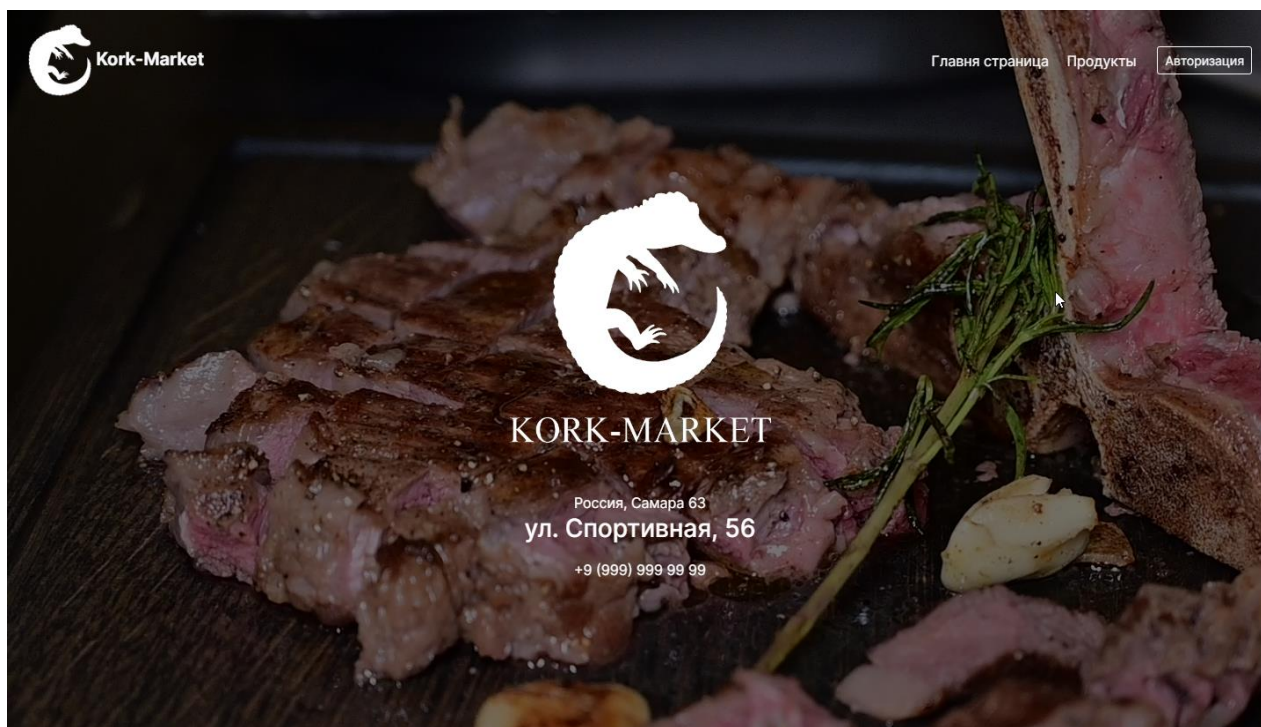


Рисунок 2.5 - Главная страница

Вверху страницы, находится контекстное меню, которое видоизменяется в зависимости от роли пользователя, а для незарегистрированного пользователя содержит два поля: главная страница и продукты. Меню для зарегистрированного пользователя дополнительно поля: корзина и заказы. В меню администратора содержатся добавлены пункты: пользователи, управление заказами и статистика.

С главной страницы незарегистрированный пользователь может попасть на страницу аутентификации. Её экранная форма представлена на рисунке 2.6.

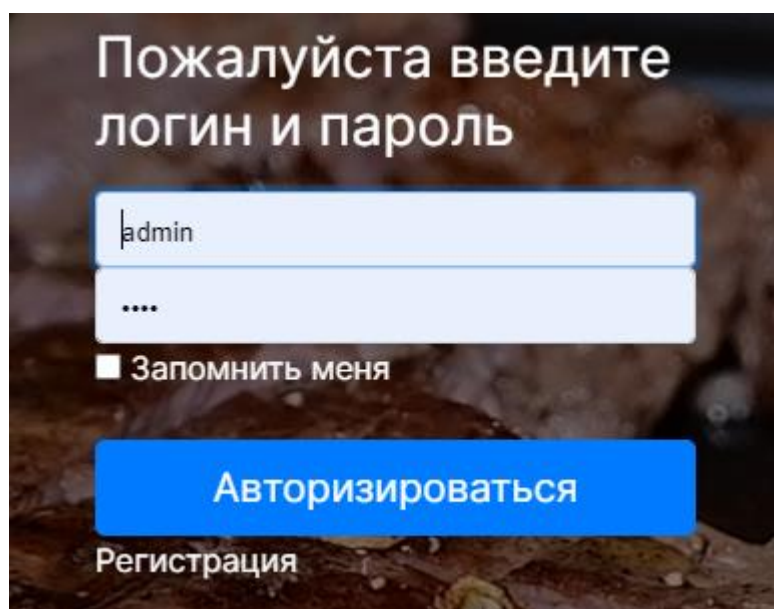


Рисунок 2.6 - Экран аутентификации

Далее, пользователь может перейти на экран регистрации, нажав на кнопку «Регистрация». Экранная форма страницы регистрации представлена на рисунке 2.7.

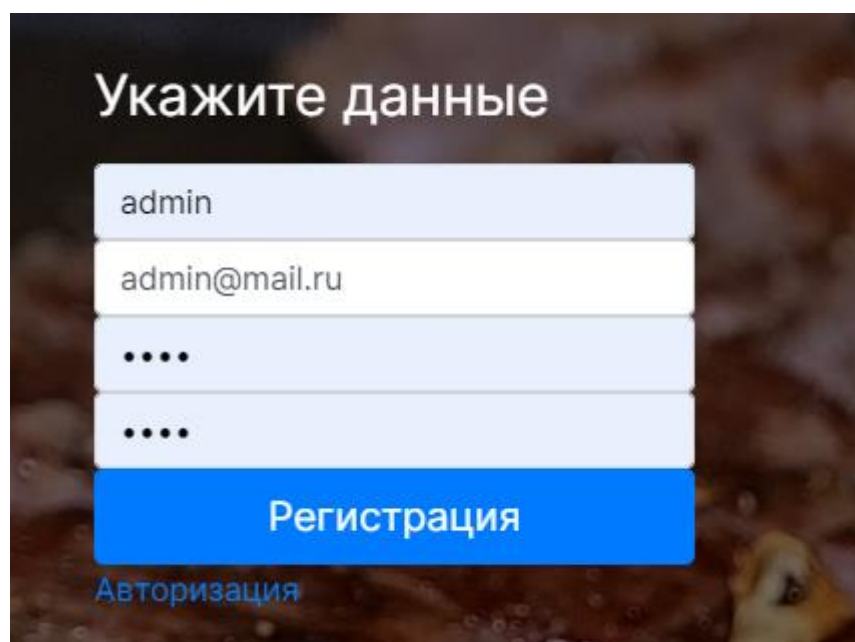


Рисунок 2.7 - Экран регистрации

После аутентификации пользователь возвращается на главную страницу. Теперь ему становится доступна расширенная версия сайта. Для добавления товара в корзину, необходимо нажать на кнопку «Добавить в корзину».

Экранная форма страницы с предлагаемыми товарами представлена на рисунке 2.8.

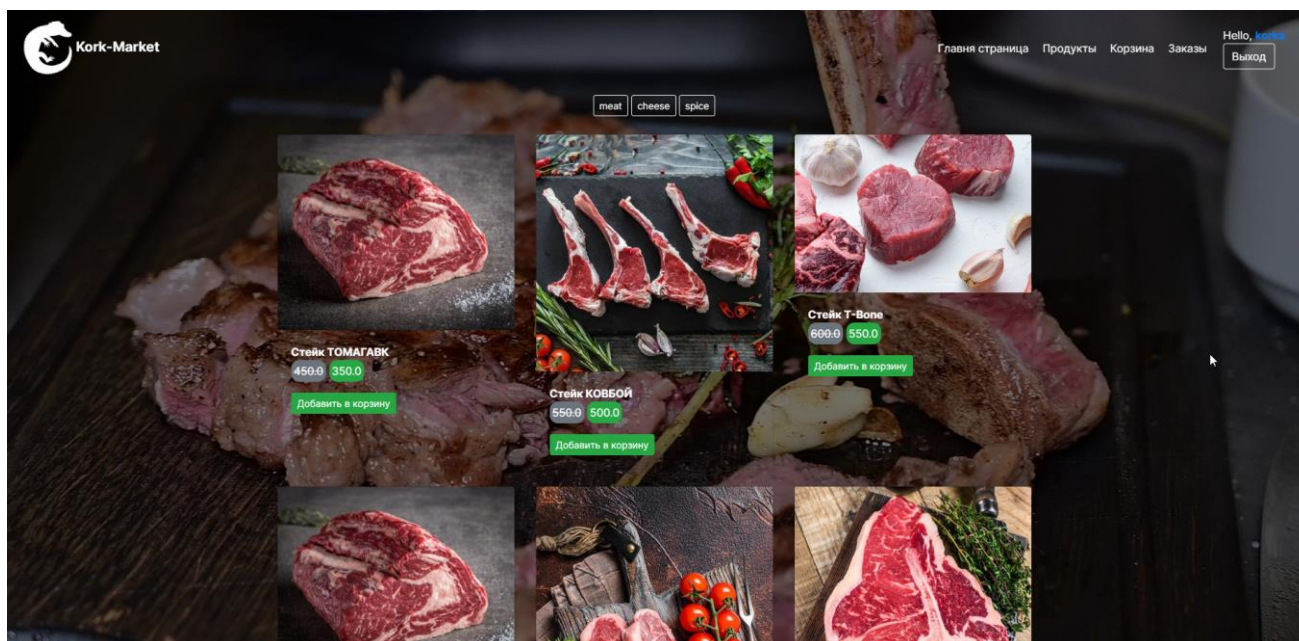


Рисунок 2.8 - Каталог продуктов для пользователя

При щелчке по названию товара выполняется переход на страницу с подробной информацией о нем. Эта страница представлена на рисунке 2.9.

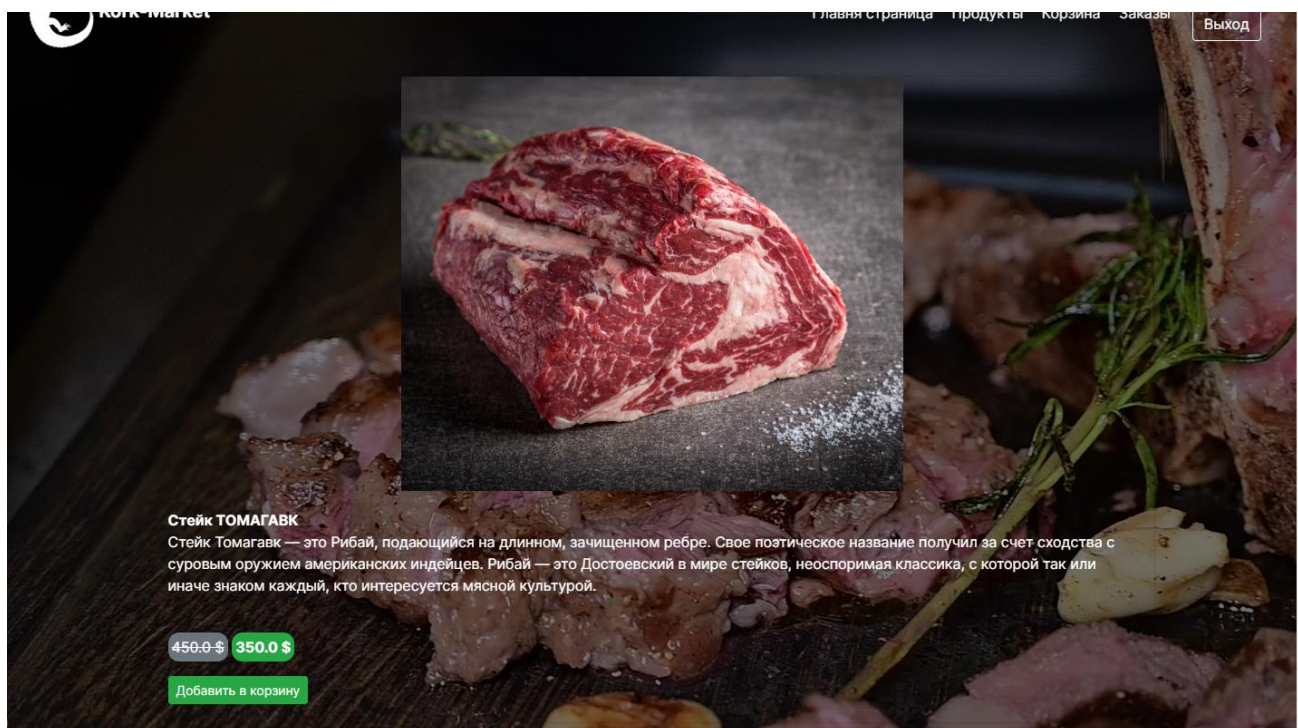


Рисунок. 2.9 - Подробная информация о товаре для пользователя

Внизу страницы находятся отзывы пользователей о товаре. Зарегистрированный пользователь, может оставить свой отзыв, если он не оставлял его ранее. Экранная форма представлена на рисунке 2.10.

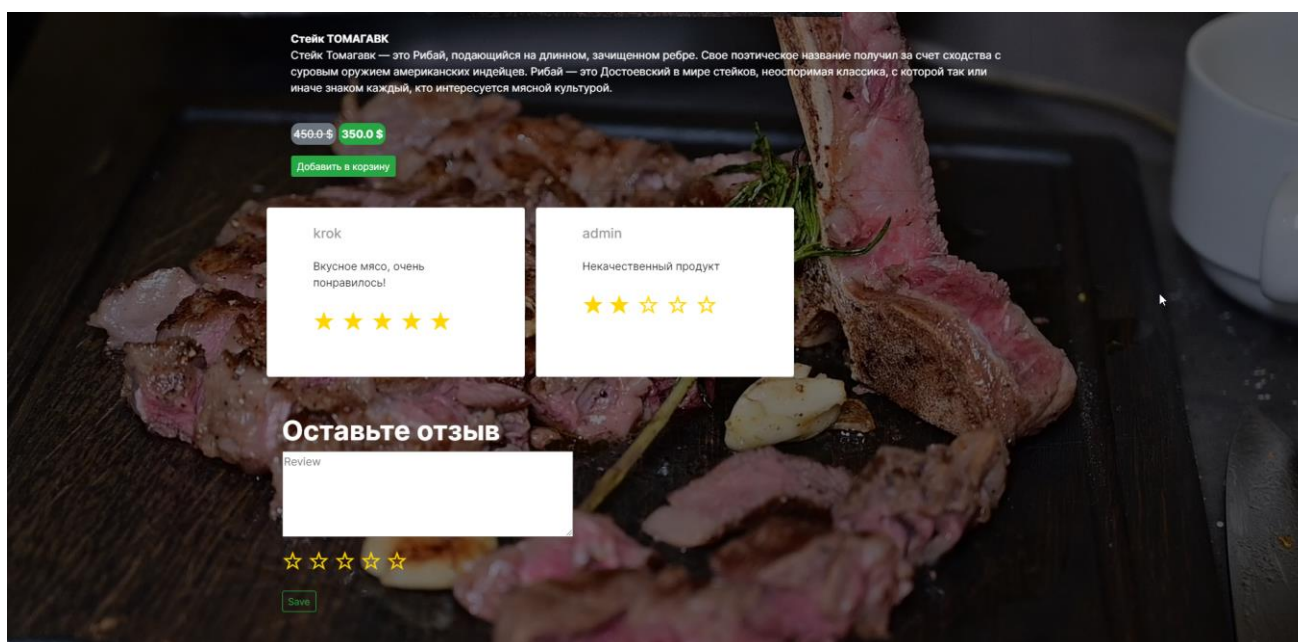


Рисунок. 2.10 - Отзывы пользователей и форма для создания нового отзыва

Для добавления товара в корзину, пользователю необходимо нажать на кнопку «Добавить в корзину». При этом, произойдет переход на страницу с категориями, а пользователь получит уведомление об успешном добавлении товара в корзину. Для просмотра содержимого корзины, пользователю необходимо нажать на кнопку «Корзина» в контекстном меню. Вид корзины пользователя представлен на рисунке 2.11. Она содержит информацию о сумме и списке всех товаров. Пользователь может изменить количество товара с помощью кнопок «+» и «-». Для полного удаления товара из корзины, необходимо нажать на кнопку «Удалить» или нажимать на кнопку «-», до тех пор, пока количества товаров не станет равным 0 и он перестанет выводиться в списке. Для очистки всей корзины, необходимо нажать на кнопку «Очистить корзину», а для формирования заказа – «Заказать».

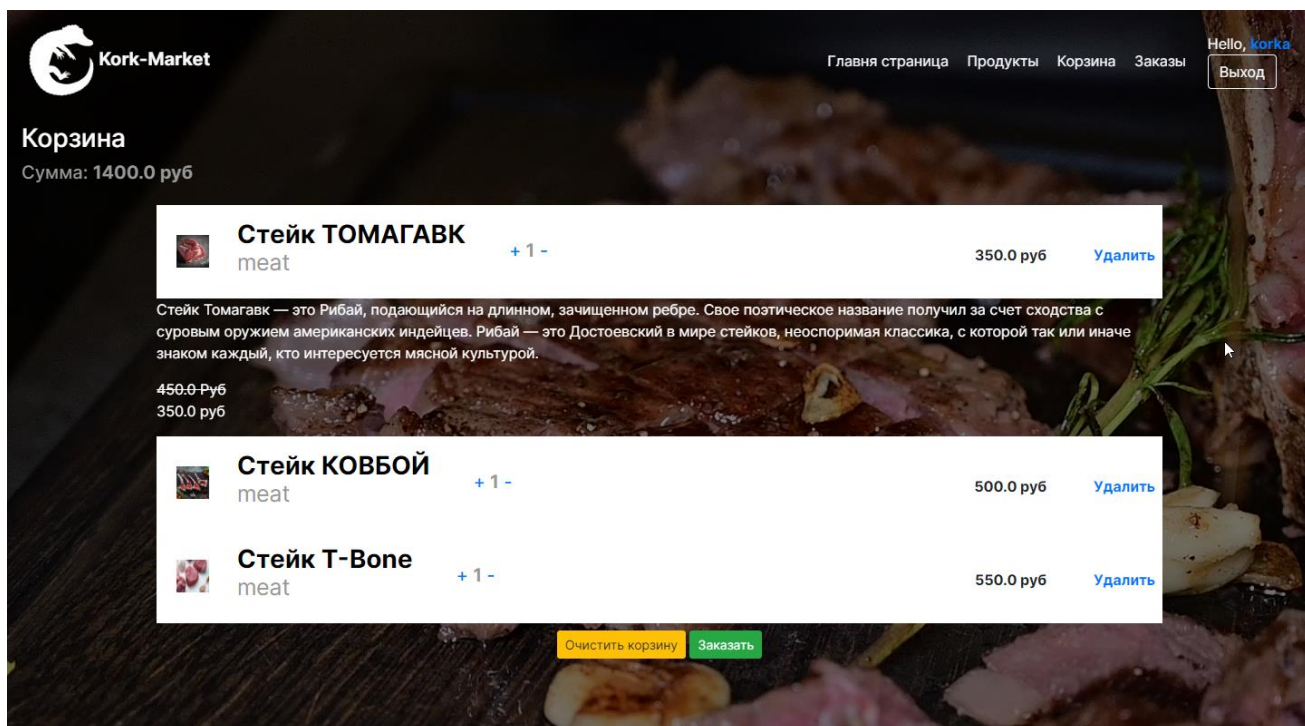


Рисунок 2.11 - Корзина пользователя

После нажатия на кнопку «Заказать» откроется окно для ввода адреса и проверки списка товаров. После указания адреса пользователю придет уведомление. Заказы пользователя представлены в виде списка, при нажатии на элемент которого, выводится дополнительная информация о товарах в этом заказе, их количестве и общей цене. Экранная форма представлена на рисунке 2.12.

Уведомления, поступающие пользователям, представлены в виде всплывающих окон в правом верхнем углу экрана. Их закрытие выполняется стандартным способом. Пример уведомления показан на рисунке 2.13.

При возникновении ошибки пользователь получит соответствующее сообщение.

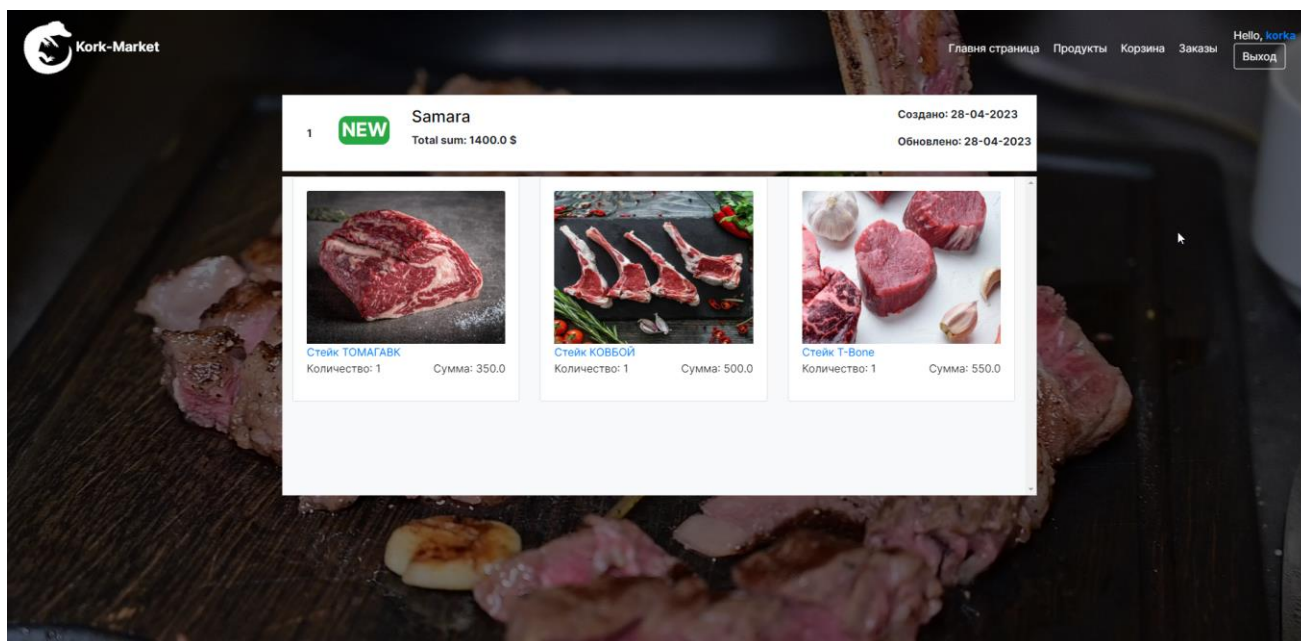


Рисунок 2.12 - Заказ пользователя

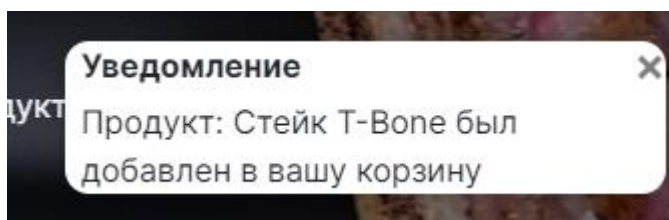


Рисунок 2.13 - Пример уведомления

2.4 Подсистема анализа продаж

Любому интернет-магазину для увеличения количества покупателей и прибыли необходимо проводить анализ своей работы. На основе полученных данных принимаются бизнес-решения и разрабатывается маркетинговая стратегия. Крупные интернет-магазины собирают следующую информацию:

- 1) посещаемости сайта;
- 2) частота покупок товара;
- 3) просмотры товарных страниц;
- 4) среднее время пребывания на сайте;
- 5) частота добавление товара в корзину;
- 6) средняя сумма заказа.

По этим данным менеджеры интернет-магазина смогут разработать стратегии дальнейшего развития сайта, выявить потребности покупателей, получить дополнительную информацию об активности пользователя на портале, разработать план по улучшению деятельности магазина, для увеличения трафика.

«Анализ продаж — это оценка эффективности реализации товаров по различным показателям, которые в совокупности отражают общую картину, а в динамике дают прогноз будущего развития. Изучать данные можно по всему ассортименту, отдельным категориям и товарам, по всей территории продаж и конкретным регионам» [21].

Анализ продаж разделяется на несколько этапов:

- 1) выбор инструментов для сбора данных и анализа;
- 2) пошаговая аналитика продаж;
- 3) определение показателей для оценки;
- 4) сравнение полученных данных с предыдущими периодами;
- 5) выводы, принятие тактических и стратегических решений.

Для анализа продаж на стороне клиента используются системы аналитики. Самые известные из них — Google Analytics и Яндекс.Метрика. С помощью этих систем, можно получить следующие данные:

- a) Популярные запросы,
- b) Определение популярных страниц, на которых число пользователей максимально
- c) Страницы, с которых пользователь покидает сайт;
- d) Устройства, с которых посещают сайт;
- e) Индексация страниц;
- f) Адаптация к устройствам различных видов;
- g) Баги, которые случались при визите покупателя.

В разрабатываемом проекте, для сбора данных со стороны клиента используется система аналитики GoogleAnalytic. Для подключения этой

системы необходимо на официальном сайте Google подтвердить владение домена, а на стороне клиента – подключить библиотеку Ganalytics.

Сбор информации со стороны сервера осуществляется с помощью записи о каждом индексируемом действии пользователя на сайте, которая отправляется в базу данных. На страницу с аналитическими данными выводится сгруппированная информация в виде графиков, которая прошла предобработку на сервере.

Было решено собирать следующие данные о работе интернет-магазина:

- количество посещений и покупок товаров,
- количество покупок товаров в разных категориях,
- частота добавления товара в корзину,
- частота покупки товара,
- средняя стоимость корзины.

На основе анализа данных о посещении страниц сайта и количестве покупок можно сделать выводы о частоте их показа в поисковиках определенных страниц, и отредактировать непопулярные страницы, чтобы увеличить шанс появления в поисковике. Информация о частоте посещения страниц с товарами, позволит выявить типы популярных товаров и разработать стратегию развития бизнеса.

Данные о популярных товарах и категориях позволяют выявить самые востребованные товары. Соотношение просмотров товаров и их покупки позволит оценить вклад визуального оформления сайта, карточек товара и отзывов. В результате, можно провести работы по улучшению дизайна и качества продукта.

Средняя стоимость заказа позволяет сделать вывод о финансовом уровне покупателей. Она упростит формирование цен на продукты и позволит разработать скидочные программы.

Данные о популярном времени для покупок, позволят определить время, когда сайт имеет наибольший трафик. И запланировать регулярные оптимизационные работы на сервере.

2.5 Руководство администратора

Для функционирования сайта необходимо оборудование со следующими техническими характеристиками:

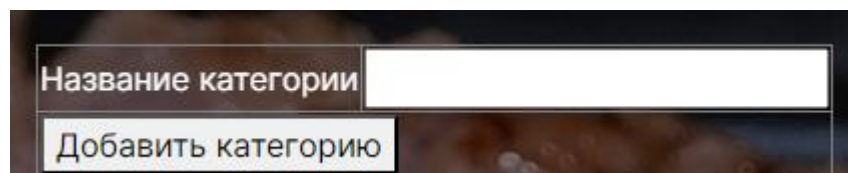
- двухъядерный процессор с частотой 2 ГГц или выше;
- оперативная память 1 Гб или выше;
- свободное место на диске (или SSD) от 1 Гб.

Программное обеспечение сервера должно соответствовать следующим характеристиками:

- ОС Ubuntu/Debian/Centos;
- веб-сервер Apache PHP (версия от 5.4 до 7.4);
- Postgresql.

Для публикации сайта на CMS администратору необходимо переместить собранный Maven jar файл в корневую папку сервера, и запустить командой: - jar online-store.jar. После этого нужно авторизоваться в СУБД Postgresql. Сайт начнет функционировать и индексироваться в поисковых системах. Для авторизации в роли администратора, в файле property необходимо указать логи и пароль.

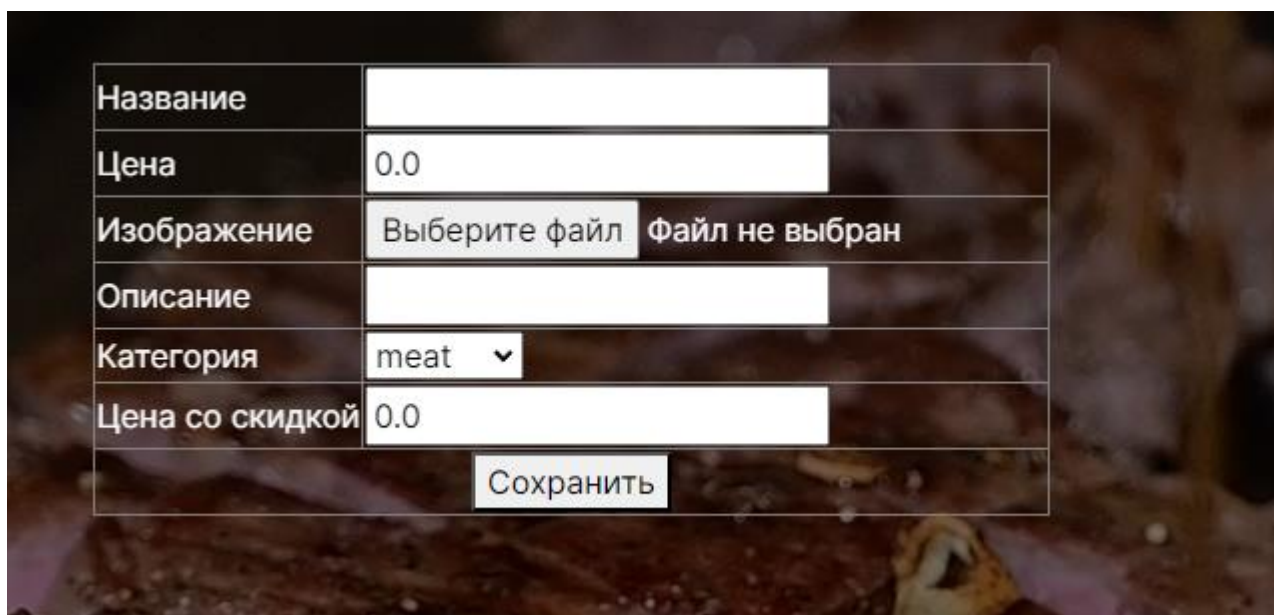
Функционал администратора значительно превышает пользовательский. У него имеется возможность создания новой категории. Для этого, на странице с категориями, необходимо нажать на кнопку «Создать категорию», после чего будет произведен переход на страницу содержащую соответствующую форму, представленную на рисунке 2.14.

A screenshot of a web form for creating a category. It features a text input field labeled "Название категории" (Category name) and a button labeled "Добавить категорию" (Add category).

Название категории	<input type="text"/>
<input type="button" value="Добавить категорию"/>	

Рисунок 2.14 - Создание категории

Администратор может вводить или редактировать информацию об имеющихся продуктах. Для создания этого необходимо на странице с категориями нажать кнопку «Добавить новый продукт» или «Изменить». Примеры таких страниц приведены на рисунках 2.15 - 2.16.

A screenshot of a web form for creating a product. The form contains several input fields: "Название" (Name), "Цена" (Price) with a value of 0.0, "Изображение" (Image) with a file selection button and "Файл не выбран" (File not selected), "Описание" (Description), "Категория" (Category) with a dropdown menu showing "meat", and "Цена со скидкой" (Price with discount) with a value of 0.0. A "Сохранить" (Save) button is at the bottom.

Название	<input type="text"/>
Цена	<input type="text" value="0.0"/>
Изображение	<input type="button" value="Выберите файл"/> Файл не выбран
Описание	<input type="text"/>
Категория	<input type="text" value="meat"/> ▼
Цена со скидкой	<input type="text" value="0.0"/>
<input type="button" value="Сохранить"/>	

Рисунок 2.15 - Создание продукта

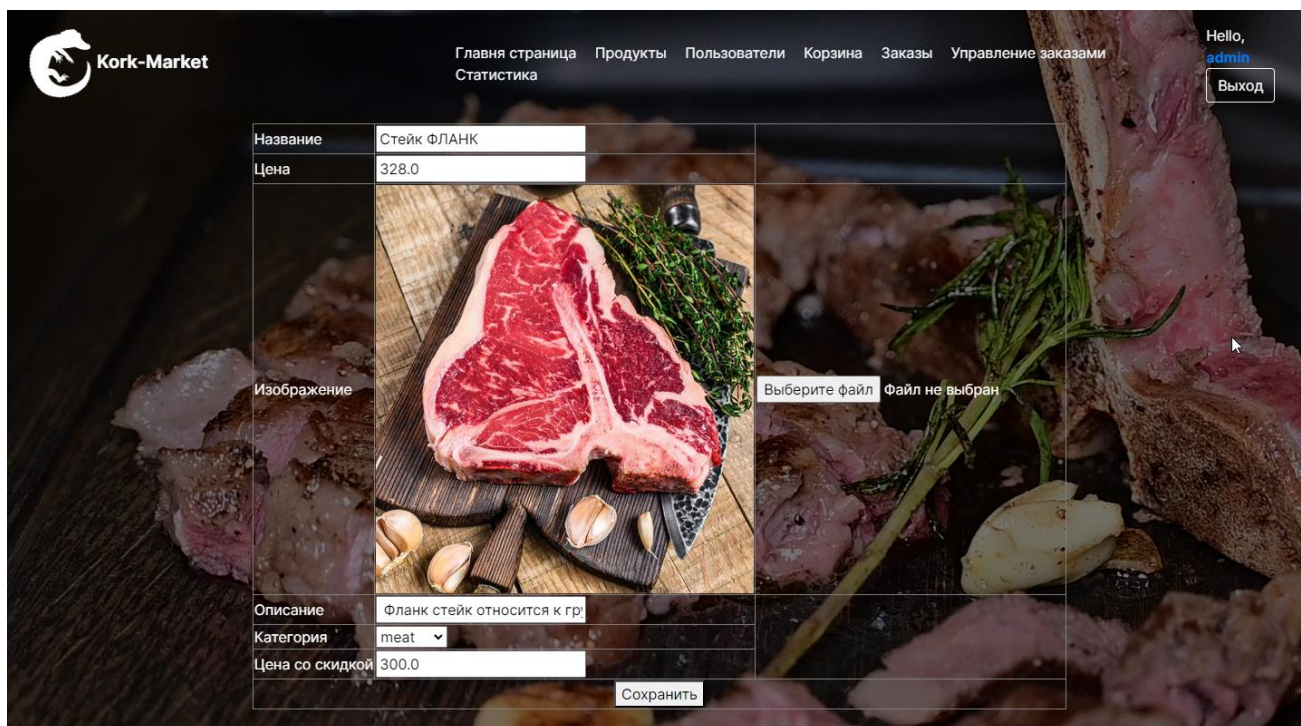


Рисунок 2.16 - Редактирование информации о продукте

Администратор имеет возможность изменять роли пользователей. Для этого, необходимо в меню выбрать вкладку users, после чего произойдет переход на страницу со списком пользователей. Далее, напротив одного из пользователей кликнуть на кнопку «Изменить», которая приведет на страницу с подробной информацией о пользователе. Примеры таких страниц представлены на рисунках 2.17 – 2.18.

Имя пользователя	Электронный адрес	Роль пользователя	
admin	mail@mail.ru	ADMIN	Изменить
korka	korka@mail.ru	CLIENT	Изменить
krok	kroka@mail.ru	MANAGER	Изменить

Рисунок 2.17 - Список пользователей

Name	admin
Email	mail@mail.ru
Role	ADMIN
Save	

Рисунок 2.18 - Подробная информация о пользователе

Администратор также может просматривать заказы пользователей, отсортированные по статусу, получать о них подробную информацию, и изменять их статус. Для просмотра страницы со списком заказов необходимо выбрать пункт меню по кнопке «Управление заказами». Пример экранной формы представлен на рисунке 2.19.

NEW APPROVED CANCELED PAID CLOSED COMPLETED

Номер	Адрес	Дата создания	Дата обновления	Статус	Сумма	
1	Молодогвардейская ул., 244, Самара,	2023-05-03T15:54:52.991480	2023-05-03T15:54:53.035461	NEW	1400.0	Edit

Рисунок 2.19 - Список заказов

Администратору доступна статистика, собираемая подсистемой анализа продаж. Демонстрация графического отображение статистических данных представлена на рисунках 2.20-2.21



Рисунок 2.20 - График статистики покупок и посещений

3 ЭКОНОМИЧЕСКАЯ ЧАСТЬ

3.1 Описание продукта

3.2 Оценка затрат на разработку программного продукта

4 ОХРАНА ТРУДА

4.1 Анализ и проектирование производственной среды

4.2 Проектирование рабочего места

4.3 Выбор оборудования

4.4 Проектирование схемы подключения оборудования

ЗАКЛЮЧЕНИЕ

В ВКР разработаны структура и программное обеспечение сайта продуктового магазина.

В ходе изучения предметной области был составлен портрет типичного покупателя и выявлены наиболее популярные продукты.

В процессе моделирования веб-сайта были разработаны и описаны: диаграмма вариантов использования, ER-диаграмма базы данных, UML диаграммы domain и dto моделей. На основе разработанных диаграмм была спроектирована база данных и серверные модели.

Сайт является корпоративным и имеет иерархическую структуру. Он содержит следующие основные подсистемы:

- 1) управление,
- 2) анализ продаж,
- 3) накопления и оформления заказа,
- 4) авторизации.

Сайт реализован с помощью следующих средств:

- клиентская часть выполнена при помощи фреймворка Thymleaf, языка разметки HTML 5, языка стилей CSS, фреймворка CSS - Bootstrap 4 и языка программирования Javascript;
- серверная часть выполнена при помощи языка Java с использованием фреймворка Spring Web MVC. Реализация базы данных выполнена с помощью СУБД PostgreSQL.

В результате был получен многостраничный интернет-магазин, выполняющий следующие функции:

- 1) вход в систему с разными уровнями доступа;
- 2) просмотр категорий товаров и отзывов;
- 3) подтверждение аккаунта по электронной почте;
- 4) формирование корзины;
- 5) оформление покупок товара;
- 6) оформление заказа и отслеживание его статуса;
- 7) хранение информации о пользователе, его корзине, заказах, продуктах, изображений продуктов и категорий.

Администратору магазина предоставляются следующие функции:

- 1) добавление и редактирование товаров и категорий
- 2) управление статусом заказов пользователей
- 3) просмотр информации о пользователе
- 4) сбор статистики

Разработанный сайт проходит опытную эксплуатацию и будет внедрен в одной из торговых организаций г. Самары.

Список используемой литературы

- 1 Выпускная квалификационная работа [Текст]: методические указания / И.В. Воронцов, Н.В. Ефимушкина, С.Ю. Леднева, С.П. Орлов, А.И. Пугачев. – Самара: Самар. гос. техн. ун-т, 2015. – 63 с.: ил.
- 2 Поцелуева Л.П. Методические указания по экономическому обоснованию дипломных проектов [Текст]: методические указания / Л.П. Поцелуева. – СамГТУ, 2007. – 23 с.
- 3 Безопасность труда при работе на персональных компьютерах [Текст]: методические указания к выполнению дипломного проекта / Сост. Л.А. Моссоулина, Е.В. Алекина. – Самара: Самар. гос. техн. ун-т, 2016. – 28 с.: ил..
- 4 Проектирование рабочего места оператора ПЭВМ [Текст]: методические указания. – СамГТУ, 2016. – 14 с.
- 5 Сайт – определение и виды [Электронный ресурс]. - URL: <https://xn--80aahvkuapc1be.xn--p1ai/raznoe/veb-sayt-sajt-opredelenie-i-vidy.html> Дата обращения: (25.03.23).
- 6 HTML [Электронный ресурс]. – URL: <https://ru.wikipedia.org/wiki/HTML> Дата обращения: (25.03.23).
- 7 CSS [Электронный ресурс]. – URL: <https://ru.wikipedia.org/wiki/CSS> Дата обращения: (25.03.23).
- 8 JavaScript [Электронный ресурс]. – URL: <https://ru.wikipedia.org/wiki/JavaScript> Дата обращения: (27.03.23).
- 9 Что такое Bootstrap и зачем он нужен? [Электронный ресурс]. – URL: <https://itchief.ru/bootstrap/introduction> Дата обращения: (27.03.23).
- 10 Архитектура REST [Электронный ресурс]. – URL: <https://habr.com/ru/post/38730/> Дата обращения: (28.03.23).
- 11 Обзор протокола HTTP [Электронный ресурс]. – URL: <https://developer.mozilla.org/ru/docs/Web/HTTP/Overview> Дата обращения: (28.03.23).

- 12 Что такое PHP? [Электронный ресурс]. – URL: <https://www.php.net/manual/ru/intro-what-is.php> Дата обращения: (28.03.23).
- 13 Язык программирования C#: краткая история, возможности и перспективы [Электронный ресурс]. – URL: <https://timeweb.com/ru/community/articles/что-такое-csharp> Дата обращения: (28.03.23).
- 14 Ruby [Электронный ресурс]. – URL: <https://ru.wikipedia.org/wiki/Ruby> Дата обращения: (28.03.23).
- 15 Язык программирования Java [Электронный ресурс]. – URL: <https://web-creator.ru/articles/java> Дата обращения: (5.04.23).
- 16 Что такое IntelliJ IDEA? [Электронный ресурс]. – URL: <https://www.jetbrains.com/ru-ru/idea/features/> Дата обращения: (05.04.23).
- 17 Eclipse [Электронный ресурс]. – URL: <https://ru.wikipedia.org/wiki/Eclipse>
- 18 Аналитики описали портрет типичного покупателя интернет-магазина [Электронный ресурс]. – URL: https://www.rbc.ru/technology_and_media/10/10/2021/61618e229a7947975cf67c04 Дата обращения: (05.04.23).
- 19 Полезные продукты: топ-10 главных героев ежедневного меню [Электронный ресурс]. – URL: <https://04.rospotrebnadzor.ru/index.php/press-center/healthy-lifestyle/13599-17112020.html> Дата обращения: (07.04.23).
- 20 Что такое Spring Framework? От внедрения зависимостей от Web MVC [Электронный ресурс]. – URL: <https://habr.com/ru/articles/490586/> Дата обращения: (07.04.23).
- 21 Три метода анализа продаж для роста дохода интернет-магазина [Электронный ресурс]. – URL: <https://www.cossa.ru/special/business-processes/295242/> Дата обращения: (08.04.23).

Url адреса:

- 1) /;
- 2) /authentication[post];
- 3) /error[get];
- 4) /forbidden[get];
- 5) /login[get];
- 6) /login-error[get];
- 7) /my/main[get];
- 8) /my/main[post];
- 9) /my/bucket[get];
- 10) /my/bucket/clear[get];
- 11) /my/bucket/{id}/decrease[get];
- 12) /my/bucket/{id}/increase[get];
- 13) /my/bucket/{id}/remove[get];
- 14) /product/{id}/bucket[get];
- 15) /product/{title}[get];
- 16) /review/new/{id}[get];
- 17) /notification/remove/{id}[get];
- 18) /registration[get];
- 19) /registration[post];
- 20) /registration/activate/{code}[get];
- 21) /order/new[get];
- 22) /order/new[post];
- 23) /order/orders[get];
- 24) /admin/category/new[get];
- 25) /admin/category/new[post];
- 26) /admin/category/{title}/remove[get];
- 27) /admin/product/edit/{title}[get];
- 28) /admin/product/new[get];
- 29) /admin/product/new[post];

```

30)    /admin/product/{id}/remove[get];
31)    /admin/stats[get];
32)    /admin/users[get];
33)    /admin/users/edit[post];
34)    /admin/users/orders[get];
35)    /admin/users/orders[post];
36)    /admin/users/{name}/edit[get].

```

Файл MailConfig.java:

```

@Configuration
public class MailConfig {
    @Bean
    public JavaMailSender getJavaMailSender() throws IOException {
        JavaMailSenderImpl mailSender = new JavaMailSenderImpl();
        mailSender.setHost("smtp.yandex.ru");
        mailSender.setPort(465);
        mailSender.setUsername("Kork2006@yandex.ru");
        mailSender.setPassword(readFile("C:\\mailPass.txt"));
        Properties props = mailSender.getJavaMailProperties();
        props.put("mail.transport.protocol", "smtp");
        props.put("mail.smtp.auth", "true");
        props.put("mail.smtp.starttls.enable", "true");
        props.put("mail.debug", "true");
        props.put("mail.smtp.ssl.enable", "true");
        return mailSender;
    }
    public static String readFile(String path) throws IOException
    {
        List<String> lines = Files.readAllLines(Path.of(path), StandardCharsets.UTF_8);
        return String.join(System.lineSeparator(), lines);
    }
}

```

Файл SecurityConfig.java:

```

@Configuration
@EnableWebSecurity
@EnableGlobalMethodSecurity(securedEnabled = true)
public class SecurityConfig extends WebSecurityConfigurerAdapter {
    private UserService userService;

    @Autowired
    public void setUserService(UserService userService) {
        this.userService = userService;
    }

    @Override
    protected void configure(AuthenticationManagerBuilder auth) throws Exception {
        auth.authenticationProvider(authenticationProvider());
    }

    @Basic
    private AuthenticationProvider authenticationProvider() {
        DaoAuthenticationProvider auth = new DaoAuthenticationProvider();
        auth.setUserDetailsService(userService);
        auth.setPasswordEncoder(passwordEncoder());
        return auth;
    }

    @Bean
    public PasswordEncoder passwordEncoder() {
        return new BCryptPasswordEncoder();
    }

    @Override
    protected void configure(HttpSecurity http) throws Exception {
        http.authorizeRequests()
            .antMatchers("/admin/**").hasAuthority("ADMIN")
            .antMatchers("/my/**").authenticated()
            .antMatchers("/order/**").authenticated()
            .antMatchers("/registration").not().fullyAuthenticated()
            .antMatchers("/activate/**").permitAll()
            .antMatchers("/category/*","/product/*").permitAll()
            .anyRequest().permitAll()
            .and()

```

```

.exceptionHandling().accessDeniedPage("/error")
.and()
.formLogin()
.loginPage("/login")
.failureUrl("/login-error")
.defaultSuccessUrl("/", true)
.permitAll()
.and()
    .rememberMe()
    .userDetailsService(userService)
.and()
.exceptionHandling().accessDeniedPage("/forbidden")
.and()
.logout().logoutRequestMatcher(new AntPathRequestMatcher("/logout"))
.logoutSuccessUrl("/").deleteCookies("JSESSIONID")
.invalidateHttpSession(true)
.and()
.csrf().disable();} }

```

Файл SecurityConfig.java:

```

@Configuration
@EnableWebMvc
public class WebConfig implements WebMvcConfigurer {
    @Override
    public void addResourceHandlers(ResourceHandlerRegistry registry) {
        if (!registry.hasMappingForPattern("/assets/**")) {
            registry.addResourceHandler("/assets/**").addResourceLocations("classpath:/assets/");
            registry.addResourceHandler("/img/**")
                .addResourceLocations("classpath:/static/img/");
        }
    }
}

```

Файл AdminController.java:

```

@Controller
@RequestMapping("/admin")
public class AdminController {

```

```

private final ProductService productService;
private final UserNotificationService userNotificationService;
private final CategoryService categoryService;
private final UserService userService;
private final OrderService orderService;
private final DiscountService discountService;
private final BucketService bucketService;
private final StatsService<VisitStats, VisitStatsDTO> visitStatsService;
private final StatsService<BuyStats, BuyStatsDTO> buyStatsService;
private final StatsService<FrequencyAddToCartStats, FrequencyAddToCartStatsDTO>
frequencyAddToCartStatsService;

private String username;
private Long orderId;
private String productTitle = "";

public AdminController(ProductService productService, UserNotificationService
userNotificationService, CategoryService categoryService, UserService userService,
OrderService orderService, DiscountService discountService, BucketService bucketService,
StatsService<VisitStats, VisitStatsDTO> visitStatsService, StatsService<BuyStats,
BuyStatsDTO> buyStatsService, StatsService<FrequencyAddToCartStats,
FrequencyAddToCartStatsDTO> frequencyAddToCartStatsService) {
    this.productService = productService;
    this.userNotificationService = userNotificationService;
    this.categoryService = categoryService;
    this.userService = userService;
    this.orderService = orderService;
    this.discountService = discountService;
    this.bucketService = bucketService;
    this.visitStatsService = visitStatsService;
    this.buyStatsService = buyStatsService;
    this.frequencyAddToCartStatsService = frequencyAddToCartStatsService;
}

@PostMapping("/product/new")
public String createNewProduct(
    Model model,
    ProductDTO productDTO,

```



```

        @RequestParam(name = "categories") String category,
        Principal principal,
        HttpServletRequest request,
        @RequestParam(name = "discount") String discount,
        @RequestParam("file") MultipartFile file
    ) {
        if (principal != null) {
            List<UserNotificationDTO> dtos =
userNotificationService.getNotificationsByUserName(principal.getName());
            model.addAttribute("notifications", dtos);
            model.addAttribute("notificationsCount", dtos.size());
        }
        if (productDTO.getTitle() == null
            || productDTO.getDescription() == null
            || productDTO.getPrice() <= 0
            || (file.isEmpty() && (productTitle.equals("") || productTitle.isEmpty())))
        ) {
            model.addAttribute("message", "All fields must be filled");
            model.addAttribute("product", productDTO);
            model.addAttribute("categories", categoryService.getAll());
            model.addAttribute("discount",
DiscountDTO.builder().discount_price(Double.parseDouble(discount)).build());
            return "productCreate";
        }
        if (Double.parseDouble(discount) >= productDTO.getPrice()) {
            model.addAttribute("message", "Discount must be less product price");
            model.addAttribute("product", productDTO);
            model.addAttribute("categories", categoryService.getAll());
            model.addAttribute("discount",
DiscountDTO.builder().discount_price(Double.parseDouble(discount)).build());
            return "productCreate";
        }
        if (productTitle.equals("") || productTitle.isEmpty()) {
            if (productService.save(productDTO, file, category, Double.valueOf(discount))) {
                model.addAttribute("product", productDTO);
            }
        }
    }
}

```

```

        model.addAttribute("categories", categoryService.getAll());
        model.addAttribute("discount",
DiscountDTO.builder().discount_price(Double.parseDouble(discount)).build());
        return "redirect:/category";
    } else {
        model.addAttribute("message", "Server error, could not save.");
        model.addAttribute("product", productDTO);
        model.addAttribute("categories", categoryService.getAll());
        model.addAttribute("discount",
DiscountDTO.builder().discount_price(Double.parseDouble(discount)).build());
        return "productCreate";
    }
} else {
    if (productService.changeName(productDTO, productTitle, category, file,
Double.valueOf(discount))) {
        model.addAttribute("product", productDTO);
        model.addAttribute("categories", categoryService.getAll());
        System.out.println("DISCOUNT: " + discount + " | " +
DiscountDTO.builder().discount_price(Double.parseDouble(discount)).build().toString());
        model.addAttribute("discount",
DiscountDTO.builder().discount_price(Double.parseDouble(discount)).build());
        return "redirect:/category";
    } else {
        model.addAttribute("errorMessage", "Product change error");
        return "error";
    }
}
}

@GetMapping("/product/new")
public String newProduct(Model model, Principal principal) {
    if (principal != null) {
        List<UserNotificationDTO> dtos =
userNotificationService.getNotificationsByUserName(principal.getName());
        model.addAttribute("notifications", dtos);
        model.addAttribute("notificationsCount", dtos.size());
    }
    if (categoryService.getAll() == null) {

```

```

        model.addAttribute("errorMessage", "Can't add product because there are no categories");
        return "error";
    }

    model.addAttribute("product", new ProductDTO());
    model.addAttribute("categories", categoryService.getAll());
    model.addAttribute("discount", new DiscountDTO());
    return "productCreate";
}

@GetMapping("/product/{id}/remove")
public String removeProduct(@PathVariable Long id, HttpServletRequest request, Model
model) {
    if (!bucketService.checkProductInBuckets(id)) {
        model.addAttribute("errorMessage", "This product is in another user's cart, cannot be
deleted");
        return "error";
    }
    if (productService.removeWithPhoto(id)) {
        return "redirect:/category";
    } else {
        model.addAttribute("errorMessage", "Error while deleting");
        return "error";
    }
}

@GetMapping("/product/edit/{title}")
public String editProductPage(Model model, @PathVariable String title, Principal principal) {
    if (principal != null) {
        List<UserNotificationDTO> dtos =
userNotificationService.getNotificationsByUserName(principal.getName());
        model.addAttribute("notifications", dtos);
        model.addAttribute("notificationsCount", dtos.size());
    }
    productTitle = title;
    ProductDTO dto = productService.getProductByName(title);
    DiscountDTO discountDTO = discountService.findDiscountByProductId(dto.getId());
    model.addAttribute("product", dto);
}

```

```

        model.addAttribute("categories", categoryService.getAll());
        model.addAttribute("discount", discountDTO);
        return "productCreate";
    }

    @GetMapping("/category/new")
    public String newCategoryPage(Model model, Principal principal) {
        if (principal != null) {
            List<UserNotificationDTO> dtos =
userNotificationService.getNotificationsByUserName(principal.getName());
            model.addAttribute("notifications", dtos);
            model.addAttribute("notificationsCount", dtos.size());
        }
        model.addAttribute("category", new CategoryDTO());
        return "categoryCreate";
    }

    @PostMapping("/category/new")
    public String createCategory(CategoryDTO categoryDTO, Model model) {
        if (categoryService.saveCategory(categoryDTO)) {
            return "redirect:/category";
        } else {
            model.addAttribute("errorMessage", "Error when save category");
            return "error";
        }
    }

    @GetMapping("category/{title}/remove")
    public String removeCategory(@PathVariable String title, Model model) {
        if (productService.removeProductsByCategoryName(title)
            && categoryService.removeCategoryByName(title)) {
            return "redirect:/category";
        } else {
            model.addAttribute("errorMessage", "Error while deleting");
            return "error";
        }
    }

    @GetMapping("/users")
    public String userList(Model model, Principal principal) {
        if (principal != null) {

```

```

        List<UserNotificationDTO> dtos =
userService.getNotificationsByUserName(principal.getName());
        model.addAttribute("notifications", dtos);
        model.addAttribute("notificationsCount", dtos.size());
    }
    List<String> roles = Stream.of(Role.values())
        .map(Enum::name).toList();
    model.addAttribute("roles", roles);
    model.addAttribute("users", userService.getAll());
    return "userList";
}
@PostMapping("/users/edit")
public String updateRole(@RequestParam(name = "roles") String role, Principal principal,
Model model) {
    if (role.equals(Role.ADMIN.name()) &&
!userService.findByName(principal.getName()).getRole().name().equals(Role.ADMIN.name()))
    {
        model.addAttribute("errorMessage", "You do not have enough rights to change the user
role to administrator");
        return "error";
    }
    userService.updateRole(username, role);
    return "redirect:/admin/users";
}
@GetMapping("/users/{name}/edit")
public String updateUserPage(@PathVariable String name, Model model, Principal principal)
{
    if (principal != null) {
        List<UserNotificationDTO> dtos =
userService.getNotificationsByUserName(principal.getName());
        model.addAttribute("notifications", dtos);
        model.addAttribute("notificationsCount", dtos.size());
    }
    UserM userM = userService.findByName(name);
    UserDTO dto = UserDTO.builder()

```

```

        .username(userM.getName())
        .email(userM.getEmail())
        .role(userM.getRole().name())
        .build();
username = name;
List<String> roles = Stream.of(Role.values())
        .map(Enum::name).toList();
model.addAttribute("roles", roles);
model.addAttribute("user", dto);
return "userEdit";
}
/**
 * Order
 */
@PostMapping("/users/orders")
public String orderManagementPageEdit(@RequestParam(name = "status") String status,
Model model, Principal principal) {
    orderService.updateOrderStatus(status, orderId);
    return "redirect:/admin/users/orders";
}
@GetMapping("/users/orders")
public String orderManagementPage(Model model, Principal principal) {
    if (principal != null) {
        List<UserNotificationDTO> dtos =
userNotificationService.getNotificationsByUserName(principal.getName());
        model.addAttribute("notifications", dtos);
        model.addAttribute("notificationsCount", dtos.size());
    }
    List<String> stats = Stream.of(OrderStatus.values())
        .map(Enum::name).toList();
    List<OrderDTO> orderDTOS = orderService.getAll();
    model.addAttribute("status", stats);
    model.addAttribute("orders", orderDTOS);
    return "orderManagement";
}

```

```

@GetMapping("/users/orders/{status}")
public String getOrdersByStatus(@PathVariable String status, Model model, Principal
principal) {
    if (principal != null) {
        List<UserNotificationDTO> dtos =
userNotificationService.getNotificationsByUserName(principal.getName());
        model.addAttribute("notifications", dtos);
        model.addAttribute("notificationsCount", dtos.size());
    }
    List<String> stats = Stream.of(OrderStatus.values())
        .map(Enum::name).toList();
    List<OrderDTO> dtos = orderService.getOrderByStatus(status);
    model.addAttribute("orders", dtos);
    model.addAttribute("status", stats);
    return "orderManagement";
}

@GetMapping("/users/orders/edit/{id}")
public String orderEditManagementPage(Model model, Principal principal, @PathVariable
Long id) {
    if (principal != null) {
        List<UserNotificationDTO> dtos =
userNotificationService.getNotificationsByUserName(principal.getName());
        model.addAttribute("notifications", dtos);
        model.addAttribute("notificationsCount", dtos.size());
    }
    OrderDTO dto = orderService.findOrderById(id);
    List<String> stats = Stream.of(OrderStatus.values())
        .map(Enum::name).toList();
    orderId = id;
    model.addAttribute("orderDTO", dto);
    model.addAttribute("status", stats);
    return "orderEditManagement";
}

/**
 * Statistics

```

```

*/
@GetMapping("/stats")
public String statisticsPage(Model model, Principal principal) {
    if (principal != null) {
        List<UserNotificationDTO> dtos =
userNotificationService.getNotificationsByUserName(principal.getName());
        model.addAttribute("notifications", dtos);
        model.addAttribute("notificationsCount", dtos.size());
    }
    String visitStatsJson = JSONValue.toJSONString(visitStatsService.collectStats());
    String buyStatsJson = JSONValue.toJSONString(buyStatsService.collectStats());
    String frequencyStatsJson =
JSONValue.toJSONString(frequencyAddToCartStatsService.collectStats());
    List<ProductDTO> productDTOList = productService.getAll();
    List<String> productsTitle = new ArrayList<>();
    for (ProductDTO p : productDTOList) {
        productsTitle.add(p.getTitle());
    }
    model.addAttribute("productsTitle", JSONValue.toJSONString(productsTitle));
    model.addAttribute("visitStatsJson", visitStatsJson.equals("{}") ? null : visitStatsJson);
    model.addAttribute("buyStatsJson", buyStatsJson.equals("{}") ? null : buyStatsJson);
    model.addAttribute("frequencyStatsJson", frequencyStatsJson.equals("{}") ? null :
frequencyStatsJson);

    model.addAttribute("products", productDTOList);
    return "statistics";
}
}

```

Файл BucketController.java

```

@Controller
public class BucketController {
    private final BucketService bucketService;
    private final UserService userService;
    private final ProductService productService;

```



```

private final UserNotificationService userNotificationService;

public BucketController(BucketService bucketService, UserService userService,
ProductService productService, UserNotificationService userNotificationService) {

    this.bucketService = bucketService;
    this.userService = userService;
    this.productService = productService;
    this.userNotificationService = userNotificationService;
}

@GetMapping("/my/bucket")
public String aboutBucket(Model model, Principal principal) {
    if (principal != null) {
        List<UserNotificationDTO> dtos =
userNotificationService.getNotificationsByUserName(principal.getName());
        model.addAttribute("notifications",dtos);
    }
    if (principal == null) {
        model.addAttribute("bucket", new BucketDTO());
    } else {
        BucketDTO bucketDTO = bucketService.getBucketByUser(principal.getName());
        System.out.println("\n\n"+bucketDTO);
        model.addAttribute("bucket", bucketDTO);
    }
    return "bucket";
}

@GetMapping("/my/bucket/{id}/remove")
public String removeFromBucket(Model model, Principal principal, @PathVariable Long id) {
    UserM userM = userService.findByName(principal.getName());
    if (userM == null) {
        throw new RuntimeException("User - " + principal.getName() + " not found");
    }
    if(productService.findProductById(id) == null){
        model.addAttribute("errorMessage", "Product not found");
        return "error";
    }
    BucketDTO bucketDTO =

```

```

bucketService.removeProduct(userM.getBucket(),id,principal.getName());
    model.addAttribute("bucket", bucketDTO);
    return "redirect:/my/bucket";
}

@GetMapping("/my/bucket/clear")
public String cleanBucket(Model model,Principal principal){
    UserM userM = userService.findByName(principal.getName());
    if (userM == null) {
        throw new RuntimeException("User - " + principal.getName() + " not found");
    }
    bucketService.clearBucket(userM.getBucket(),principal.getName());
    BucketDTO bucketDTO = bucketService.getBucketByUser(principal.getName());
    model.addAttribute("bucket", bucketDTO);
    return "redirect:/my/bucket";
}

@GetMapping("/my/bucket/{id}/increase")
public String increase(Model model, Principal principal, @PathVariable Long id){
    if(productService.findProductById(id) == null){
        model.addAttribute("errorMessage", "Product not found");
        return "error";
    }
    if (principal == null) {
        model.addAttribute("bucket", new BucketDTO());
    } else {
        bucketService.amountIncrease(userService.findByName(principal.getName()).getBucket(),id);
        BucketDTO bucketDTO = bucketService.getBucketByUser(principal.getName());
        model.addAttribute("bucket", bucketDTO);
    }
    return "redirect:/my/bucket";
}

@GetMapping("/my/bucket/{id}/decrease")
public String decrease(Model model, Principal principal, @PathVariable Long id){
    if(productService.findProductById(id) == null){
        model.addAttribute("errorMessage", "Product not found");
        return "error";
    }

```

```
    }  
    bucketService.amountDecrease(userService.findByName(principal.getName()).getBucket(),id);  
    BucketDTO bucketDTO = bucketService.getBucketByUser(principal.getName());  
    model.addAttribute("bucket", bucketDTO);  
    return "redirect:/my/bucket";  
    }  
}
```

Графический материал