Министерство образования Республики Беларусь

Учреждение образования «Белорусский государственный университет   
информатики и радиоэлектроники»

Факультет инженерно-экономический

Кафедра экономической информатики

Дисциплина «Распределенные системы обработки информации»

|  |  |
| --- | --- |
|  | «К ЗАЩИТЕ ДОПУСТИТЬ» |
|  | Руководитель курсового проекта  ассистент кафедры экономической информатики  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ А. П. Лыщик |
|  | \_\_\_.\_\_\_\_.2023 |

**ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

к курсовому проекту

на тему:

**«Автоматизированная система управления заказами интернет-магазина обуви»**

|  |  |
| --- | --- |
|  | Выполнил студент группы 074002  Кофанова Мария Сергеевна  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (подпись студента) |
|  | Курсовой проект представлен на проверку \_\_\_.\_\_\_\_.2023  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (подпись студента) |

Минск 2023

**РЕФЕРАТ**

Кофанова, М. С. Автоматизированная система управления заказами в интернет-магазине обуви / М. С. Кофанова. – Минск: БГУИР, 2023 – 61 с.

Пояснительная записка 59 с., 42 рис., 8 источников, 3 приложения

Ключевые слова: система, автоматизированная, контроль, обувь, разработка, информационная, модель, схема, моделирование, диаграмма, анализ, тестирование, пользователь, администратор.

Объект исследования: интернет-магазин обуви.

Предмет исследования: способы и пути повышения качества управления заказами в интернет-магазине обуви.

Цель курсового проекта: совершенствование управления заказами в интернет-магазине путем создания автоматизированной системы управления заказами в интернет-магазине обуви.

Методология проведения работы: в процессе разработки автоматизированной системы использованы методы анализа, систематизации, классификации, обобщения данных, качественно-количественные методы обработки данных, функциональный анализ процессов, принципа построения баз данных, моделирование системы с помощью UML-диаграмм.

Результаты работы: проведен теоретический анализ, а также моделирование деятельности по управлению заказами в интернет-магазине. Выполнено проектирование и конструирование автоматизированной системы управления заказами в интернет-магазине обуви. Осуществлено тестирование и проверка работоспособности системы. Предоставлена инструкция по развертыванию автоматизированной системы, а также сквозной тестовый пример реализации некоторых вариантов использования.

Область применения результатов: услуги по продаже обуви в интернете.

**СОДЕРЖАНИЕ**

[Перечень условных обозначений, символов и терминов 7](#_Toc134563528)

[Введение 8](#_Toc134563529)

[1 Анализ и моделирование предметной области автоматизированной системы 9](#_Toc134563530)

[1.1 Описание деятельности интернет-магазина обуви 9](#_Toc134563531)

[1.2 Разработка функциональной модели предметной области 11](#_Toc134563532)

[1.3 Анализ требований к разрабатываемому программному средству. Спецификация функциональных требований 16](#_Toc134563533)

[1.4 Разработка информационной модели предметной области 17](#_Toc134563534)

[1.5 UML-модели представления программного средства и их описание 20](#_Toc134563535)

[1.5.1 Диаграмма последовательности 20](#_Toc134563536)

[1.5.2 Диаграмма состояний 21](#_Toc134563537)

[1.5.3 Диаграмма компонентов 22](#_Toc134563538)

[1.5.4 Диаграмма развертывания 23](#_Toc134563539)

[1.5.5 Диаграмма классов 24](#_Toc134563540)

[2 Проектирование и конструирование автоматизированной системы управления заказами в интернет-магазине обуви 28](#_Toc134563541)

[2.1 Постановка задачи 28](#_Toc134563542)

[2.2 Обоснование выбора компонентов и технологий для реализации программного средства 29](#_Toc134563543)

[2.3 Архитектурные решения 31](#_Toc134563544)

[2.4 Описание алгоритмов, реализующих ключевую бизнес-логику разрабатываемого программного средства 33](#_Toc134563545)

[2.5 Проектирование пользовательского интерфейса 34](#_Toc134563546)

[2.6 Методы и средства, используемые для обеспечения безопасности данных 35](#_Toc134563547)

[3 Тестирование и проверка работоспособности программного средства 37](#_Toc134563548)

[4 Руководство по развертыванию и использованию программного средства 39](#_Toc134563549)

[Заключение 50](#_Toc134563550)

[Список использованных источников 51](#_Toc134563551)

[Приложение А (обязательное) Отчет о проверке на заимствования в системе «Антиплагиат» 52](#_Toc134563552)

[Приложение Б (обязательное) Листинг кода программы 53](#_Toc134563558)

[Приложение В (обязательное) Листинг скрипта генерации базы данных 59](#_Toc134563561)

# **ПЕРЕЧЕНЬ УСЛОВНЫХ ОБОЗНАЧЕНИЙ, СИМВОЛОВ И ТЕРМИНОВ**

СУБД – Система управления базами данных

HTTP – *Hypertext Transfer Protocol* – протокол передачи данных в сети

IDEF0 – *Integrated DEFinition 0* – методология функционального моделирования

Java EE – *Java Enterprise Edition* – набор спецификаций, описывающий архитектуру для средних и крупных предприятий

JPA – *Java Persistence API* – Java API, предоставляющий механизм взаимодействия с базой данных посредством технологии ORM

JTA – *Java Transaction API* – Java API, предоставляющий механизм управления транзакциями в Java EE

JWT – *JSON Web Token* – стандарт создания токенов доступа в формате JSON

MVC – *Model View Controller* – архитектурный шаблон построения приложений, при котором логика разделена на модель, представление и контроллер

Spring

Boot – это среда на основе *Java* с открытым исходным кодом, используемая для создания микросервиса.

SQL – *Structured Query Language –* декларативный язык взаимодействия с реляционной базой данных

UML – *Unified Modeling Language –* графический язык визуализации, специфицирования, конструирования систем

# **ВВЕДЕНИЕ**

Интернет-магазины обуви становятся все более популярными в Беларуси. Количество онлайн-покупателей растет с каждым годом. Поэтому существует необходимость создания эффективной системы управления заказами, которая удовлетворяла бы потребности покупателей, облегчала обработку заказов и улучшала обслуживание клиентов.

Такая система может решить ряд проблем управления заказами. Во-первых, она поможет автоматизировать процесс принятия заказа, сократить время обработки заказа и снизить риск ошибки персонала. Во-вторых, системы управления заказами позволяют легко отслеживать статус заказов и информировать клиентов о текущем состоянии их заказов. Это также способствует удобному управлению заказами и запасами, и повышает эффективность работы интернет-магазина.

*Цель курсового проекта* – разработка и внедрение автоматизированной системы управления заказами для интернет-магазина обуви.

Для достижения поставленной цели необходимо решить следующие задачи:

* проанализировать требования и потребности системы управления заказами для интернет-магазина обуви;
* выбрать подходящую технологию для внедрения системы;
* разработать архитектуру системы;
* реализовать основные функциональные модули;
* протестировать работу системы.

*Объектом* исследования является автоматизированная система управления заказами, которая должна облегчить процесс управления заказами и повысить эффективность работы интернет-магазина обуви.

*Предметом* исследования курсового проекта является процесс управления заказами интернет-магазина обуви.

Таким образом, данный курсовой проект является важным для развития интернет-магазинов по продаже обуви в Республике Беларусь. Разработка автоматизированной системы управления заказами для интернет-магазинов позволит упростить рабочие процессы персонала, сократить время обработки заказов и повысить качество обслуживания покупателей.

# **1 АНАЛИЗ И МОДЕЛИРОВАНИЕ ПРЕДМЕТНОЙ ОБЛАСТИ АВТОМАТИЗИРОВАННОЙ СИСТЕМЫ**

## **1.1 Описание деятельности интернет-магазина обуви**

Продажа обуви в интернет-магазине весьма распространенная услуга. Сейчас можно найти много интернет-магазинов по продаже любой обуви, от фирменных кроссовок до летних босоножек.

Процесс продажи обуви в интернет-магазине прост, но существенно отличается от традиционной розничной торговли. Вместо того чтобы предлагать свои товары в физических магазинах, интернет-магазины используют веб-платформы для привлечения и удержания клиентов.

Первый этап процесса продаж обуви в интернет-магазине – это создание виртуального каталога товаров. Каталог должен быть четким и удобным для использования, чтобы клиенты могли легко найти нужный товар. В каталоге должно быть предоставлено достаточно информации о товаре: его описание, включая размер обуви, материал из которого она изготовлена, цветовая гамма обуви, сезон и т.д.; характеристики; фотографии и отзывы клиентов.

Следующим этапом является привлечение клиентов на сайт интернет-магазина обуви. Для этого могут использоваться различные методы, такие как контекстная реклама, социальные сети, электронная почта и другие методы онлайн-маркетинга. Важно, чтобы весь трафик, привлеченный на сайт, был качественным и заинтересованным в продуктах магазина.

Когда клиент находит подходящий товар, он может добавить его в корзину и перейти к оформлению заказа. Важно предоставить клиентам различные способы оплаты, чтобы они могли выбрать наиболее удобный для себя. Кроме того, интернет-магазин должен обеспечить быструю и надежную доставку товара.

После получения заказа клиент может оставить отзыв о своем опыте покупки в интернет-магазине обуви. Положительные отзывы могут привлечь новых клиентов и улучшить репутацию магазина в глазах посетителей сайта.

Электронная торговля, особенно в последние годы, стала все более популярной. Благодаря интернет-магазинам люди могут покупать товары со всего мира, не выходя из дома. Однако, чтобы быть успешным в электронной торговле, необходимо постоянно улучшать качество сервиса и учитывать потребности клиентов.

Чтобы продажи в обувном интернет-магазине шли успешно следует соблюдать следующее:

1. Хорошее качество фотографий обуви:

Чем более высококачественные и детальные фотографии продукта, тем более уверенно будут чувствовать себя покупатели. Лучшие изображения позволяют более точно передавать визуальные характеристики продукта, такие как цвет, текстура, форма и размер. Дополнительно, можно предоставить возможность клиенту оценить товар с помощью 3D-моделей или видео.

1. Подробное описание обуви:

Описания позиций обуви должны быть полными, точными и содержать всю важную информацию, такую как размер, материалы, производитель, страна происхождения и т.д. Также можно использовать привлекательный язык и рассказать о преимуществах и уникальных свойствах той или иной модели.

1. Широкий ассортимент:

Чем больше позиций в обувном интернет-магазине, тем больше шансов на то, что клиент найдет именно то, что искал. Не нужно ограничиваться одной маркой или типом обуви, а нужно расширить ассортимент до различных стилей, материалов и цветов.

1. Удобная навигация по сайту:

Сайт должен быть интуитивно понятным и легким в использовании. Нужно разделить товары на категории, использовать удобный поиск по сайту, обеспечить возможность сортировки товаров по разным параметрам, таким как цена, бренд, размер и т.д.

1. Оценка конкурентов:

Важно изучить своих конкурентов и определить, что они делают хорошо и что можно улучшить в своем магазине. Нужно обратите внимание на их цены, ассортимент, качество товаров, способы доставки и обслуживания клиентов. Это поможет выработать стратегию, которая выделит интернет-магазин на фоне других и привлечет больше покупателей.

1. Выгодные условия покупки:

Важно рассмотреть возможность предоставления скидок на товары, акции, подарки при покупке или бесплатную доставку. Эти условия могут стать решающим фактором при выборе магазина для покупки обуви.

## **1.2 Разработка функциональной модели предметной области**

Методология IDEF0 предписывает построение иерархической системы диаграмм – единичных описаний фрагментов системы. Сначала проводится описание системы в целом и ее взаимодействия с окружающим миром (контекстная диаграмма), после чего проводится функциональная декомпозиция – система разбивается на подсистемы, и каждая подсистема описывается отдельно (диаграммы декомпозиции). Затем каждая подсистема разбивается на более мелкие и так далее до достижения нужной степени подробности.

Каждая IDEF0-диаграмма содержит блоки и дуги. Блоки изображают функции моделируемой системы. Дуги связывают блоки вместе и отображают взаимодействия и взаимосвязи между ними.

Функциональные блоки (работы) на диаграммах изображаются прямоугольниками, означающими поименованные процессы, функции или задачи, которые происходят в течение определенного времени и имеют распознаваемые результаты. Имя работы должно быть выражено отглагольным существительным, обозначающим действие.

Рассмотрим процесс продажи обуви в интернет-магазине для более детального разбора предметной области.

В качестве средства моделирования используем нотацию IDEF0.

На рисунке 1.1 представлена контекстная диаграмма верхнего уровня. Входными данными являются информация о клиенте и каталог товаров.

В процессе организации экспорта продукции входные данные преобразуются в оплату заказа, товарный чек и доставленный смартфон.

Управление процессом определяется законодательством и данными о поставщиках.

Ресурсами, необходимыми для выполнения процесса являются:

* администратор;
* пользователь;
* ПО.



Рисунок 1.1 – Контекстная диаграмма верхнего уровня

Диаграмма верхнего уровня декомпозируется на 3 блока: «Просмотр списка товаров», «Добавление товара в корзину» и «Оформление заказа». Декомпозиция представлена на рисунке 1.2.

Рисунок 1.2 – Декомпозиция контекстной диаграммы верхнего уровня

Произведём декомпозицию блока «Добавление в корзину», изображенную на рисунке 1.3, который делится на следующие подпроцессы: «Фильтрация обуви по категории», «Просмотр подробной информации о товаре» и «Выбор товара».

Рисунок 1.3 – Декомпозиция блока «Добавление товара в корзину»

Произведём декомпозицию блока «Фильтрация обуви по категории», изображенную на рисунке 1.4, который делится на следующие подпроцессы: «Выбор категории из списка», «Выполнение поиска» и «Вывод списка обуви в категории».

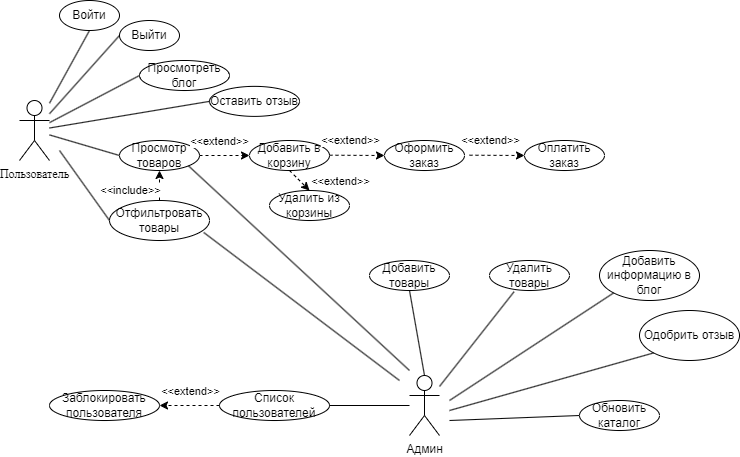


Рисунок 1.4 – Декомпозиция блока «Фильтрация обуви по категории»

В результате создания модели процесса продажи обуви и её детализации была более глубоко проработана предметная область.

Для построения данных диаграмм была использована программа AllFusion Process Modeler.

## **1.3 Анализ требований к разрабатываемому программному средству. Спецификация функциональных требований**

****Рисунок 1.5 – Диаграмма вариантов использования для интернет-магазина обуви

На диаграмме, приведённой выше отражены функции, которые доступны клиенту, администратору, менеджеру, партнёру и всем пользователям.

Администратору доступно:

* добавление/удаление товаров;
* просмотр и фильтрация товаров;
* блокировка пользователя;
* просмотр списка пользователей;
* обновление каталога;
* добавление информации в блог;
* одобрение отзывов на размещение.

Клиенту доступны следующие функции:

* вход/выход из личного кабинета;
* просмотр и фильтрация товаров;
* добавление/удаление товаров из корзины;
* оформление заказа;
* просмотр блога;
* добавление отзывов.

Все пользователи могут авторизоваться.

Таким образом, с помощью диаграммы вариантов использования были описаны функции приложения.

## **1.4 Разработка информационной модели предметной области**

Информационная модель представляет собой формализованное описание на языке информатики некоторой части реального мира, которая может быть изучена с целью управления и автоматизации социально-экономических процессов. Она позволяет представить информацию об объектах и явлениях мира в виде структурированных данных, которые могут быть обработаны компьютерной системой. Информационная модель может быть использована для проектирования и создания информационных систем, в том числе для автоматизации бизнес-процессов, управления проектами, анализа данных и других задач.

Информационная модель баз данных представлена на рисунке 1.6.

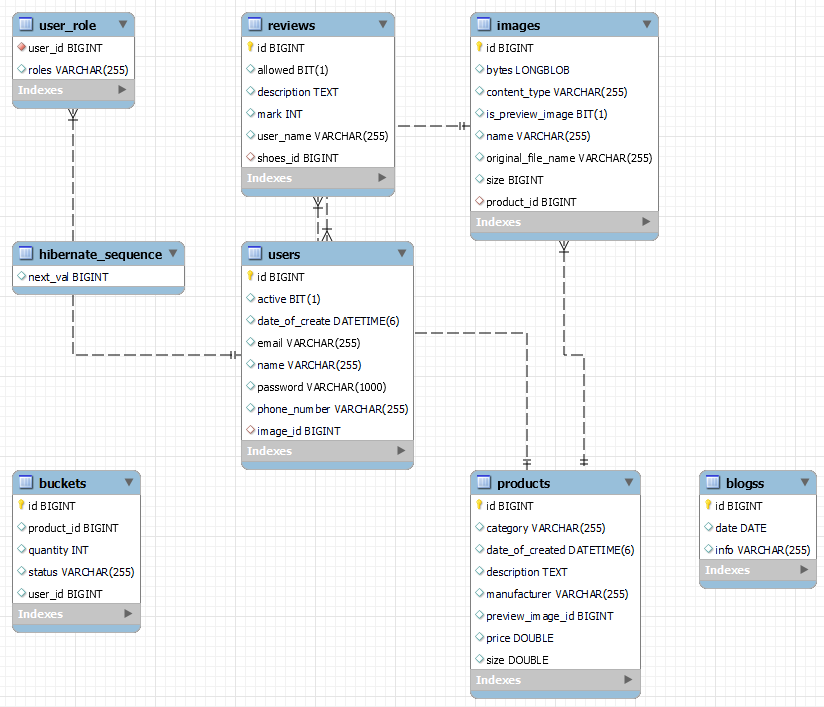


Рисунок 1.6 – Информационная модель базы данных

На данной информационной модели представлено 8 сущностей: user\_role, users, products, images, buckets, blogs и hibernate\_sequence.

Сущность user\_role предназначена для хранения информации о пользователях и администраторов. Содержит в себе 2 атрибута: user\_id – номер пользователя, roles – пользователь/администратор.

Сущность Users хранит информацию о пользователях. Содержит в себе следующие атрибуты:

* id – номер пользователя;
* date\_of\_create – дата создания пользователя;
* email – электронная почта пользователя;
* name – имя пользователя;
* password – пароль;
* phone\_number – номер пользователя;
* image\_id – номер изображения.

Сущность products хранит информацию о товаре (обуви). Содержит в себе следующие атрибуты:

* id – номер товара (обуви);
* category – категория товара (обуви);
* date\_of\_create – дата создания;
* description – описание товара (обуви);
* manufacturer – производитель товара;
* preview \_image\_id – количество фотографий;
* price – цена товара;
* size – размер обуви.

Сущность images хранит информацию о изображениях. Содержит в себе следующие атрибуты:

* id – номер изображения;
* bytes – байты изображения;
* content\_type – тип контента;
* is\_preview\_image – главное изображение;
* name – имя изображения;
* original\_file\_name – оригинальное имя файла;
* size – размер изображения;
* product\_id – номер товара.

Сущность Buskets хранит информацию о корзине. Содержит в себе следующие атрибуты:

* id – номер корзины;
* product\_id – номер товара;
* quantity – количество товаров;
* status – статус корзины;
* user\_id – номер пользователя;
* phone\_number – номер пользователя;

Сущность Blogs хранит информацию о статьях, размещенных в блоге. Содержит в себе следующие атрибуты:

* id – номер статьи в блоге;
* date – дата создания статьи;
* info – информация о статье.

## **1.5 UML-модели представления программного средства и их описание**

Для описания системы используется язык UML, который представляет собой графический язык для моделирования объектов в области разработки ПО, бизнес-процессов, системного проектирования и организационных структур. UML является открытым стандартом и использует графические обозначения для создания абстрактной модели системы, называемой UML-моделью. Он был разработан для определения, визуализации, проектирования и документирования программных систем. UML не является языком программирования, однако на основе UML-моделей возможна генерация кода.

## **1.5.1 Диаграмма последовательности**

Диаграмма последовательности (Sequence diagram) – это графическое представление взаимодействия между объектами в рамках конкретного сценария или процесса, показывающее последовательность вызовов методов и передачу сообщений между объектами в определенном порядке.

Диаграммы последовательности часто используются в процессе проектирования программных систем для отображения взаимодействия между компонентами или модулями. Они позволяют разработчикам и аналитикам лучше понимать поведение системы и выявлять возможные проблемы в процессе ее работы.

Для построения диаграммы последовательности используются объекты, которые представляют собой экземпляры классов или компонентов системы. Взаимодействие между объектами отображается с помощью стрелок, которые представляют вызовы методов или передачу сообщений. Порядок взаимодействия определяется расположением объектов на диаграмме, а также направлением стрелок.

Диаграмма представлена на рисунке 1.7.

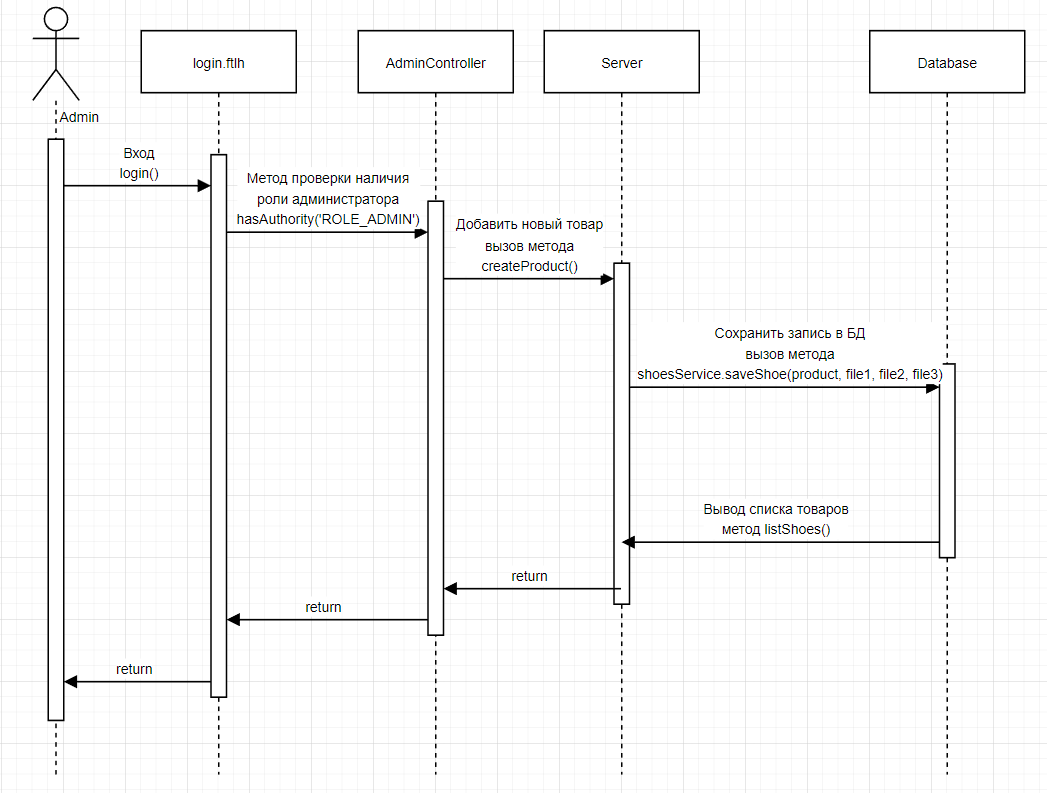


Рисунок 1.7 – Диаграмма последовательности

Сначала администратор входит в систему, затем идет проверка наличия роли администратора с помощью метода has Authority(‘ROLE\_ADMIN’). После проверки роли администратору доступна панель администратора, где он может добавить новый товар с помощью метода createProduct(). После чего запись о новом товаре сохраняется в базе данных с помощью вызова метода shoesService.saveShoe(product, file1, file2, file3). Затем выводится список товаров с уже добавленным с помощью метода listShoes().

## **1.5.2 Диаграмма состояний**

Диаграмма состояний отображает переход объекта из одного состояния в другое, что является ключевым для моделирования динамических процессов в системе. Она полезна при отображении жизненного цикла объекта, поскольку описывает процесс изменения состояний только одного экземпляра класса, который реагирует на внешние события. Это отличает диаграмму состояний от других диаграмм, которые могут моделировать несколько объектов. Жизненный цикл объекта наиболее значим для реактивных объектов, чье текущее состояние и поведение зависят от прошлых состояний. Однако диаграммы состояний важны не только для моделирования динамики отдельного объекта, они также могут использоваться для создания исполняемых систем через прямое и обратное проектирование.

Диаграмма состояния представлена на рисунке 1.8.

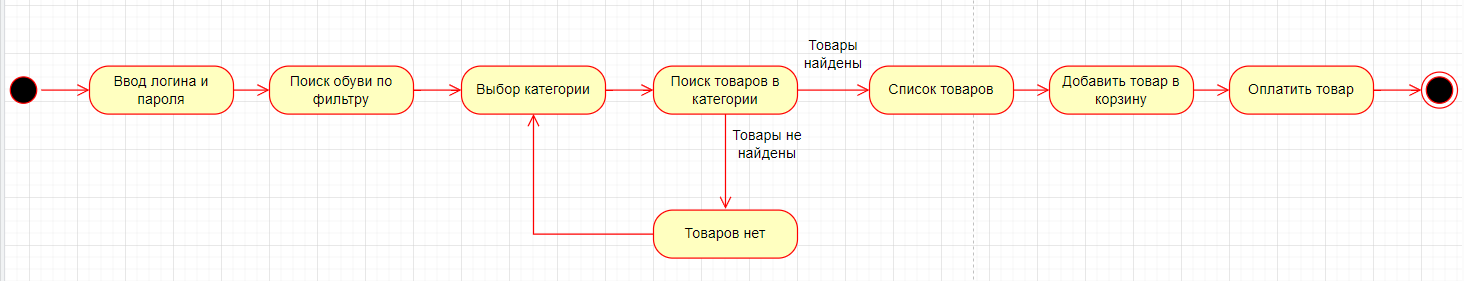


Рисунок 1.8 – Диаграмма состояний

## **Диаграмма компонентов**

Диаграммы компонентов используются для визуализации организации компонентов системы и зависимостей между ними. Они позволяют получить высокоуровневое представление о компонентах системы.

Компонентами могут быть программные компоненты, такие как база данных или пользовательский интерфейс; или аппаратные компоненты, такие как схема, микросхема или устройство; или бизнес-подразделение, такое как поставщик, платежная ведомость или доставка.

Компонентные диаграммы:

* используются в компонентно-ориентированных разработках для описания систем с сервис-ориентированной архитектурой;
* показывают структуру самого кода;
* могут использоваться для фокусировки на отношениях между компонентами, скрывая при этом детализацию спецификации;
* помогают в информировании и разъяснении функций создаваемой системы заинтересованным сторонам.

Диаграмма компонентов представлена на рисунке 1.9.

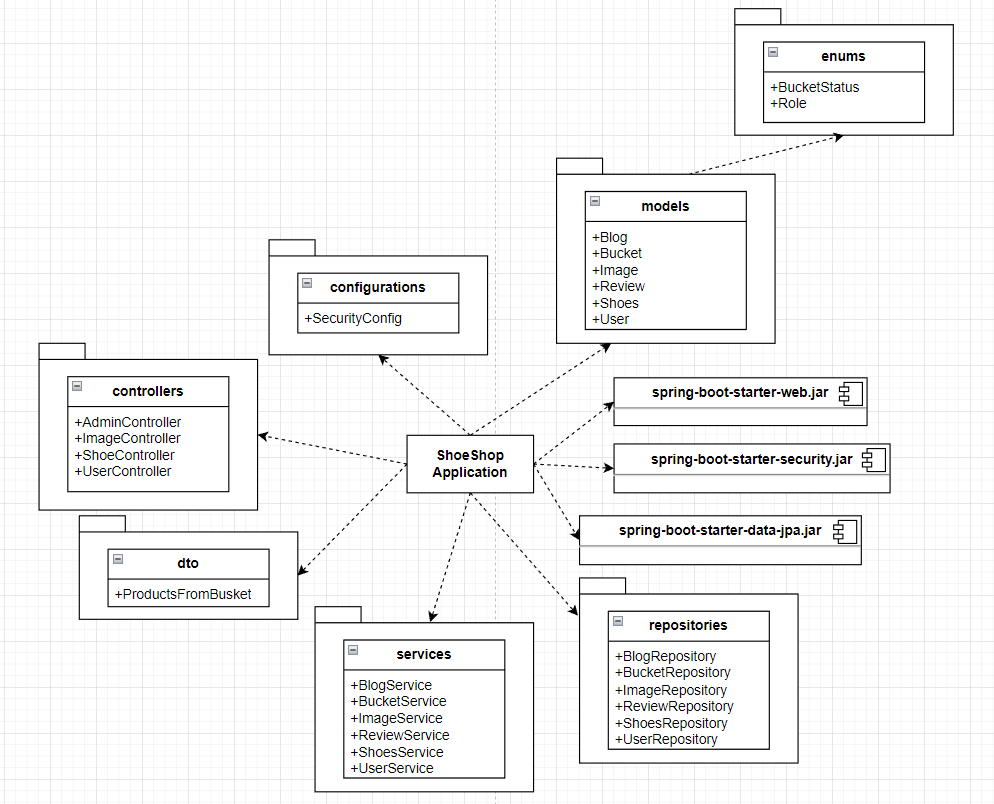


Рисунок 1.9 – Диаграмма компонентов

## **1.5.4 Диаграмма развертывания**

Диаграмма развертывания – это тип UML-диаграммы, который используется для отображения архитектуры выполнения системы, включая аппаратные и программные среды исполнения, а также промежуточное программное обеспечение, которое соединяет их.

Основное назначение диаграмм развертывания – визуализация физической архитектуры системы, показывая, как она будет развернута на аппаратном обеспечении.

В отличие от других типов UML-диаграмм, которые описывают логические компоненты системы, диаграммы развертывания используются для моделирования аппаратной топологии системы.

Диаграмма развертывания представлена на рисунке 1.10

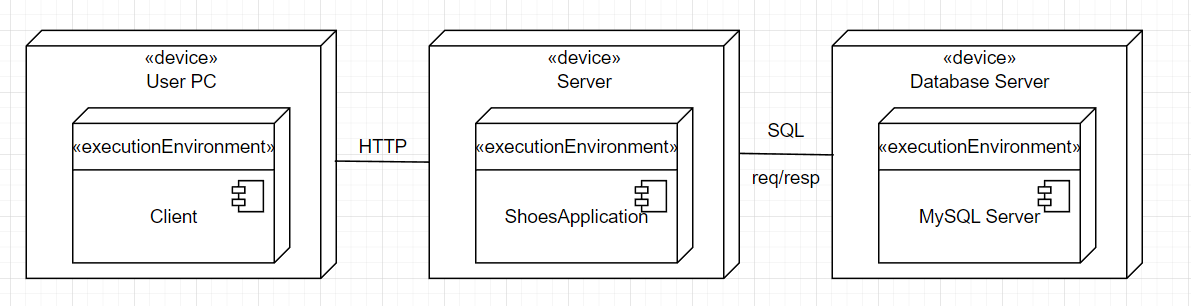


Рисунок 1.10 – Диаграмма развертывания

На диаграмме изображены три узла: User PC, Server и Database Server, так же присутствуют устройства, использованные данными узлами.

## **1.5.5 Диаграмма классов**

Диаграмма классов является средством визуализации статических связей между классами системы и типов классов, которые используются в ней. На диаграмме классов также отображаются атрибуты и операции классов, а также ограничения, которые определяют связи между ними. Интерпретация диаграммы классов может значительно варьироваться в зависимости от точки зрения, так как классы могут представлять как элементы программной системы, так и сущности предметной области в процессе анализа.

В пакете controller присутствует 4 класса: ShoeController, AdminController, UserController и ImageController. Классы пакета controller представлены на рисунке 1.11.

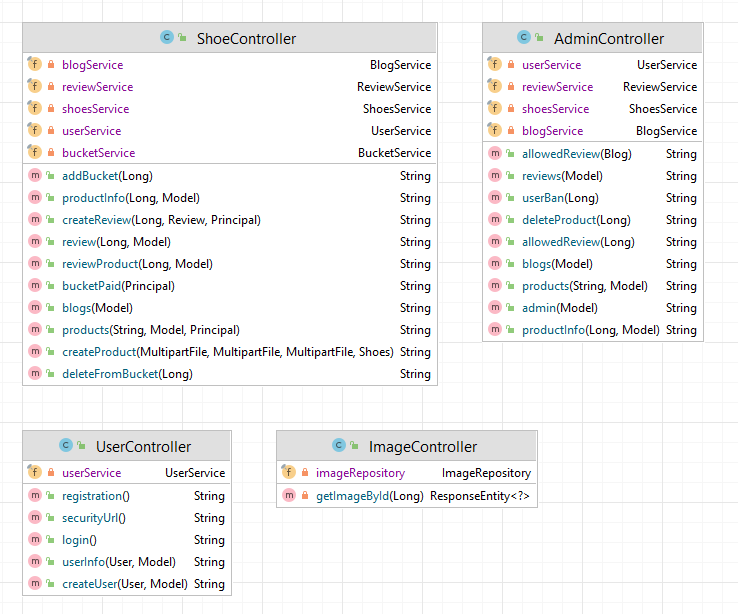


Рисунок 1.11 – Классы пакета controller

В пакете models 5 классов: Image, Shoes, Review, Blog, Bucket, User. Эти классы содержат методы, которые обрабатывают запросы от клиентов и возвращают данные или представления. В нем находятся все сущности. Информация об этих сущностях хранится и обрабатывается в базе данных. Также в данном пакете присутствует пакет enums, в котором есть 2 сущности: BucketShoes и Role. Классы пакета models представлены на рисунке 1.12.

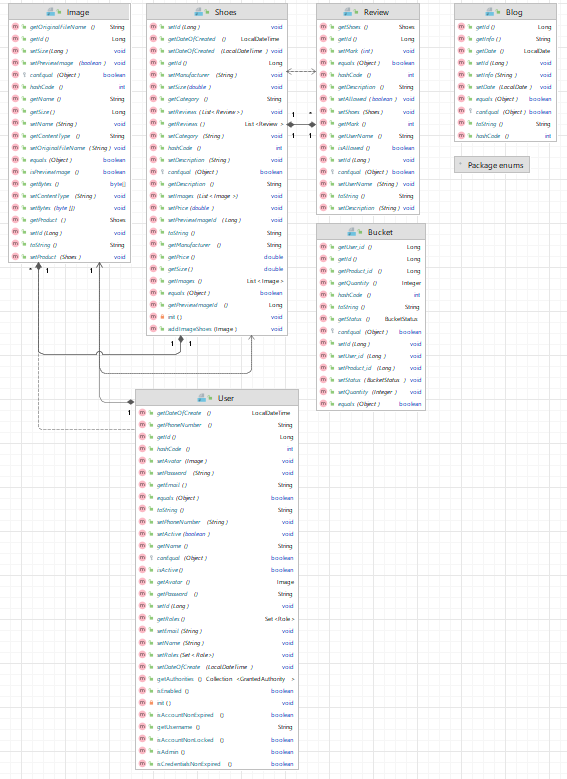
****

Рисунок 1.12 – Классы пакета models

В пакете services присутствует 6 классов: ShoesService, BucketService, UserService, ReviewService, BlogService, CustomUserDetailsService.

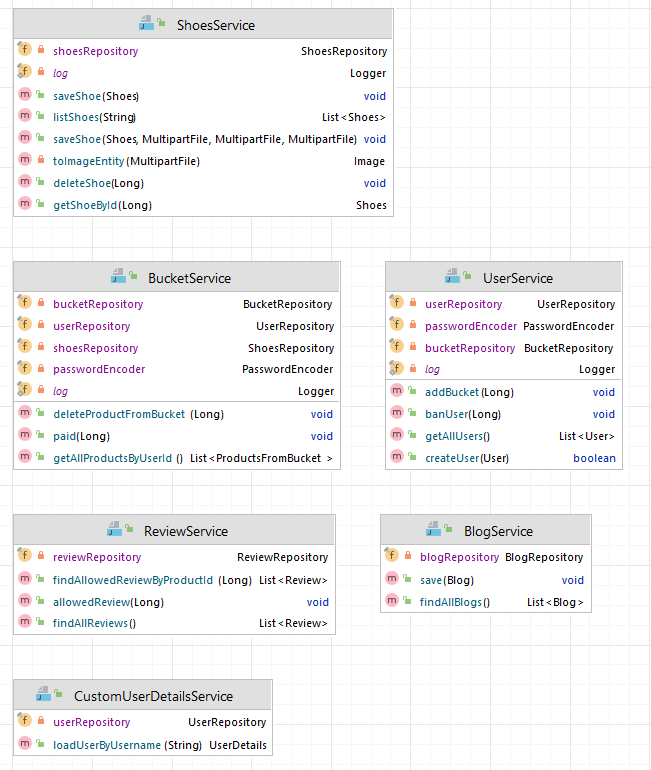


Рисунок 1.13 – Классы пакета services

# **2 ПРОЕКТИРОВАНИЕ И КОНСТРУИРОВАНИЕ АВТОМАТИЗИРОВАННОЙ СИСТЕМЫ УПРАВЛЕНИЯ ЗАКАЗАМИ В ИНТЕРНЕТ-МАГАЗИНЕ ОБУВИ**

## **2.1 Постановка задачи**

Для удобства осуществления процесса покупки обуви и повышения удобства пользователя необходимо разработать веб-приложение, в котором будет предусмотрено создание ролей пользователя (покупателя) и администратора.

При реализации данного программного средства было необходимо выполнить следующие действия:

* провести анализ предметной области;
* разработать базу данных на языке Structured query language (SQL);
* разработать back-end и front-end приложения;
* выполнить связывание back-части приложения с базой данных;
* создать пользовательский интерфейс в HTML-форм;
* выполнить связывание back-end и front-end -частей приложения.

Администратору доступны следующие функции:

* регистрация;
* авторизация;
* добавление/удаление товара;
* просмотр подробной информации о товаре;
* одобрение на размещение отзывов пользователей;
* добавление информации в блог;
* блокировка/разблокировка пользователя.

Пользователю доступны следующие функции:

* регистрация;
* авторизация;
* просмотр товаров и подробной информации о них;
* добавление/ удаление товаров из корзины;
* фильтрация товаров;
* оплата заказа;
* добавление отзывов;
* просмотр блога.

## **2.2 Обоснование выбора компонентов и технологий для реализации программного средства**

Для написания back-части приложения был использован язык Java и фреймворк SpringBoot.

Java – широко используемый язык программирования для написания интернет-приложений. Язык Java широко использовался на протяжении более двух десятилетий. Миллионы приложений Java используются и сегодня. Java – это многоплатформенный, объектно-ориентированный и сетецентрический язык, который сам по себе может использоваться как платформа. Это быстрый, безопасный и надежный язык программирования для всего: от мобильных приложений и корпоративного ПО до приложений для работы с большими данными и серверных технологий [1].

Самым популярным языком программирования на GitHub является JavaScript. Далее идут Python, Java, Typescript и C#. На шестой позиции располагается C++, сместивший на седьмую строку язык PHP. Кроме того, в первую десятку входят Shell, C и Ruby [2].

Основные преимущества языка Java [3]:

1. Язык Java прост для изучения.

При разработке Java было уделено большое внимание простоте языка, поэтому программы на Java, по сравнению с программами на других языках, проще писать, компилировать, отлаживать и изучать.

1. Java – это объектно-ориентированный язык.

Это позволяет создавать модульные программы, исходный код которых может использоваться многократно.

1. Язык Java не зависит от платформы.

Одним из основных преимуществ языка Java является возможность переноса программ из одной системы в другую. Поскольку программы на Java не зависят от платформы как на уровне исходного кода, так и на двоичном уровне, их можно запускать в различных системах, что особенно важно для программ, предназначенных для World Wide Web.

Spring Boot – это полезный проект, целью которого является упрощение создания приложений на основе Spring. Он позволяет наиболее простым способом создать web-приложение, требуя от разработчиков минимум усилий по его настройке и написанию кода.

Spring Boot обладает большим функционалом, но его наиболее значимыми особенностями являются: управление зависимостями, автоматическая конфигурация и встроенные контейнеры сервлетов [4].

В качестве СУБД была взята одна из популярных – MySQL.

MySQL – одна из наиболее используемых систем управления базами данных. MySQL управляет реляционными базами данных, то есть такими, в которых таблицы связаны между собой.

MySQL работает по принципу клиент-сервер. Компьютер пользователя (клиент) отправляет запрос. Сервер баз данных его обрабатывает и предоставляет ответ. Именно поэтому часто можно услышать понятие MySQL-сервер. Это сервер, на котором хранится база данных [5].

У MySQL есть ряд преимуществ [6]:

* высокая скорость работы;
* поддержка практически всех CMS;
* бесплатная лицензия;
* надежная и простая система безопасности;
* поддержка нескольких типов таблиц: MyISAM, InnoDB.
* плагины, позволяющие упростить и настроить работу под себя;
* в одной таблице может содержаться несколько миллионов записей.

MySQL универсальна и применяется при разработке веб-сайтов, веб-приложений и корпоративных баз данных начального уровня.

Таким образом была поставлена задача разработки веб-приложения для управления заказами интернет-магазина обуви, описаны основные функциональные требования к системе, а также приведены основные технологии, используемые для реализации бизнес-логики.

## **2.3 Архитектурные решения**

При разработке веб-приложения был использован архитектурный паттерн MVC.

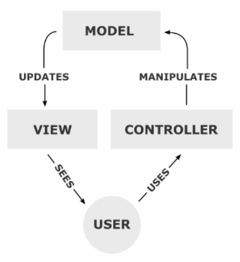


Рисунок 2.1 – Структура архитектурного паттерна MVC

MVC расшифровывается как «модель-представление-контроллер» (от англ. model-view-controller). Это способ организации кода, который предполагает выделение блоков, отвечающих за решение разных задач. Один блок отвечает за данные приложения, другой отвечает за внешний вид, а третий контролирует работу приложения [7].

Компоненты MVC:

1. Модель – этот компонент отвечает за данные, а также определяет структуру приложения. Например, если создается To-Do приложение, код компонента model будет определять список задач и отдельные задачи.
2. Представление – этот компонент отвечает за взаимодействие с пользователем. То есть код компонента view определяет внешний вид приложения и способы его использования.
3. Контроллер – этот компонент отвечает за связь между model и view. Код компонента controller определяет, как сайт реагирует на действия пользователя. По сути, это мозг MVC-приложения.

Для реализации разработки и развертывания приложения был выбран Java-фреймворк Spring Boot [8]:

1. Он предоставляет возможность быстрой и безопасной настройки среды для разработки программного обеспечения. Это означает, что разработчики могут сразу приступить к созданию приложений, не тратя время на настройку инструментов и фреймворков.
2. Уменьшает объем и упрощает процесс разработки за счет использования аннотаций для более простого понимания кода и конфигураций boilerplate, которые автоматически копируют/вставляют для повторяющихся функций.
3. Облегчает создание и тестирование приложений на базе Java, предоставляя стандартную настройку для модульных и интеграционных тестов.
4. Включает в себя встроенные HTTP-серверы Jetty и Tomcat для тестирования веб-приложений. Позволяет легко подключаться к базам данных и сервисам очередей, таким как Oracle, PostgreSQL, MySQL, MongoDB, Redis, Solr, ElasticSearch, Rabbit MQ, ActiveMQ и многим другим.
5. Облегчает управление зависимостями с помощью стартовых пакетов. Каждый пакет охватывает набор зависимостей под одним именем и предоставляет доступ к различным технологиям Spring и аналогичным ему. Например, для создания веб-приложения Spring разработчикам понадобится зависимость spring-boot-starter-web, а для быстрого доступа к библиотекам тестирования – spring-boot-starter-test и так далее.
6. Spring Boot оснащен встроенным контейнером сервлетов. Это позволяет избавиться от потребности в его отдельной настройке и развертывании на нем приложения. Благодаря встроенной в Spring Boot поддержке контейнеров сервлетов, разработанные решения могут легко запускаться на встроенном сервере, таком как Tomcat. В то же время разработчики программного обеспечения могут просто переключиться на другие контейнеры, включая Jetty, Undertow, Resin или Wildfly.
7. Обеспечивает легкий доступ к интерфейсу командной строки, что делает разработку и тестирование приложений Spring Boot, созданных на Java или Groovy, более быстрым.
8. Упрощает интеграцию с Spring и предоставляет доступ к другим инструментам фреймворка, таким как Spring Data, Spring Security, Spring ORM и Spring JDBC.

## **2.4 Описание алгоритмов, реализующих ключевую бизнес-логику разрабатываемого программного средства**

Блок-схема – это графическое представление последовательности действий и принимаемых решений в процессе решения конкретной задачи или выполнения определенной операции. Она состоит из различных геометрических фигур, таких как прямоугольники, ромбы, круги и стрелки, которые соединяются друг с другом, образуя цепочки или ветвления. Блок-схема используется в программировании, инженерии и других областях для упрощения понимания сложных процессов и их анализа, а также для разработки программ и алгоритмов.

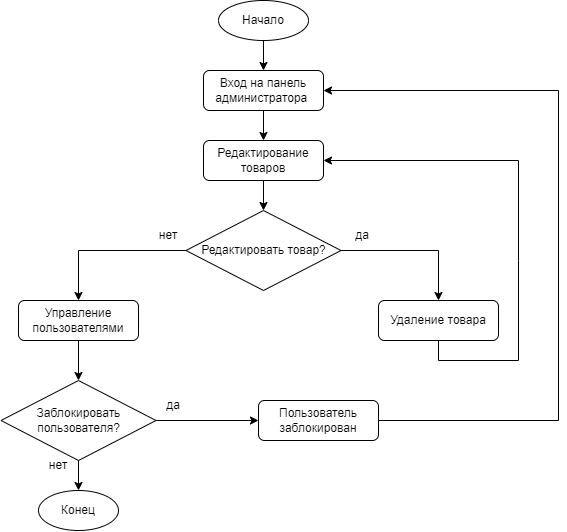
Схема алгоритма работы приложения при входе с ролью администратора представлена на рисунке 2.2.

Рисунок 2.2 – Схема алгоритма работы приложения в роли администратора

Из нее видно, что сначала администратор направляется на панель администратора, где он может редактировать товары или же управлять пользователями.

При работе с товарами доступно создание, редактирование и удаления обуви, в итоге все изменения проводятся в базе данных.

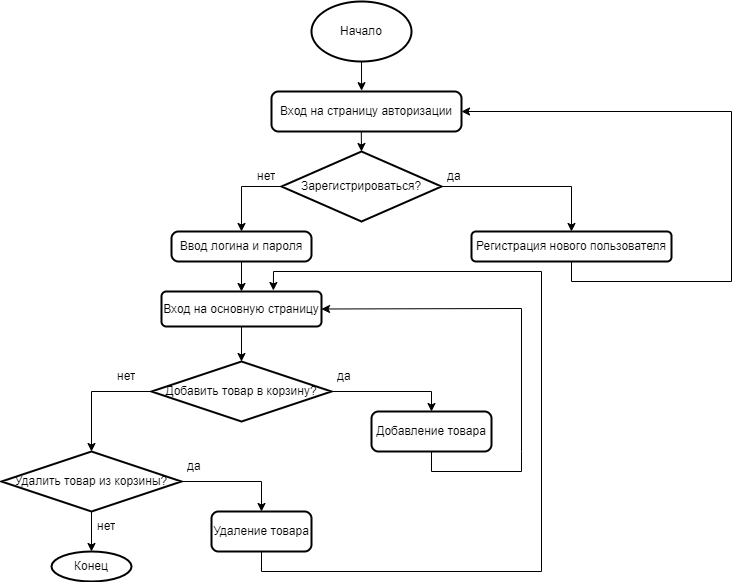
Далее рассмотри схему алгоритма процессов входа и добавления товаров в корзину, представленную на рисунке 2.3.

Рисунок 2.3 – Схема алгоритма действий для пользователя

Из схемы видно, что сначала пользователь заходит на страницу авторизации. Если пользователь уже авторизован, то он вводит логин и пароль, если нет, то происходит регистрация нового пользователя. Далее пользователь входит на основную страницу. Если он хочет добавить товар в корзину, то происходит его добавление, а затем переход на основную страницу. В обратном случае пользователь может удалить товар из корзины, и затем снова происходит переход на основную страницу. В противном случае процесс заканчивается.

## **2.5 Проектирование пользовательского интерфейса**

Пользовательский интерфейс разработан с использованием механизма шаблонов Apache FreeMarker – это: библиотеки Java для генерации текстового вывода (HTML-страницы, xml, файлы конфигурации, исходный код и.т.д. На вход подается шаблон, например html в котором есть специальные выражения, подготавливаются данные, соответствующие этим выражениям, а Freemarker динамически вставляет эти данные и получается динамически заполненный документ.

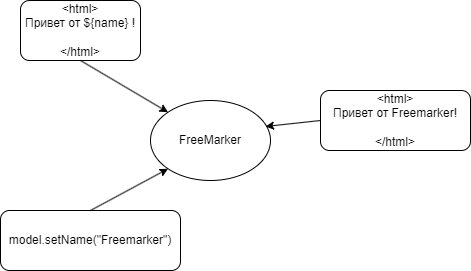


Рисунок 2.4 – Принцип работы шаблонов FreeMarker

## **2.6 Методы и средства, используемые для обеспечения безопасности данных**

Для обеспечения безопасности данных в веб-приложении были использованы следующие методы и средства.

1. При регистрации нового пользователя, пароль хэшируется с использованием функции хэширования BCryptPasswordEncoder. Это позволяет сохранять пароли в безопасном виде, так как хэш нельзя преобразовать обратно в исходный пароль.
2. В приложении был настроен протокол HTTPS, который обеспечивает шифрование данных между сервером и клиентом. Это предотвращает возможность перехвата данных и защищает их от несанкционированного доступа.
3. Была реализована защита от CSRF-атак (межсайтовая подделка запроса). Для этого в коде был использован токен CSRF, который генерируется на сервере и передается вместе с формой. Когда форма отправляется на сервер, токен CSRF проверяется для обеспечения подлинности запроса.
4. В приложении был использован Hibernate, который обеспечивает безопасность от SQL-инъекций. Hibernate автоматически создает параметризованные запросы, которые позволяют избежать SQL-инъекций.
5. Входные данные пользователя проверяются на соответствие определенным правилам. Для этого в приложении используется Spring Validation API.
6. Была реализована система ролей и разграничения прав доступа. Различные пользователи имеют разные права доступа к функционалу приложения. Например, администратор имеет право на удаление и редактирование данных, а пользователь – только на просмотр.

# **3 ТЕСТИРОВАНИЕ И ПРОВЕРКА РАБОТОСПОСОБНОСТИ ПРОГРАММНОГО СРЕДСТВА**

Для тестирования использовалась библиотека Mockito – это Java библиотека для создания и использования мок-объектов в тестировании.

Мок-объекты – это объекты, которые имитируют поведение реальных объектов, но не содержат реальных реализаций методов. Использование мок-объектов в тестировании позволяет изолировать тестируемый код от зависимостей и фокусироваться на конкретных сценариях использования.

Mockito предоставляет удобный API для создания мок-объектов и установки их поведения. Она также обладает мощными возможностями для верификации вызовов методов и аргументов в тестах.

Mockito является популярным инструментом в Java сообществе для модульного тестирования и интеграционного тестирования Java приложений.

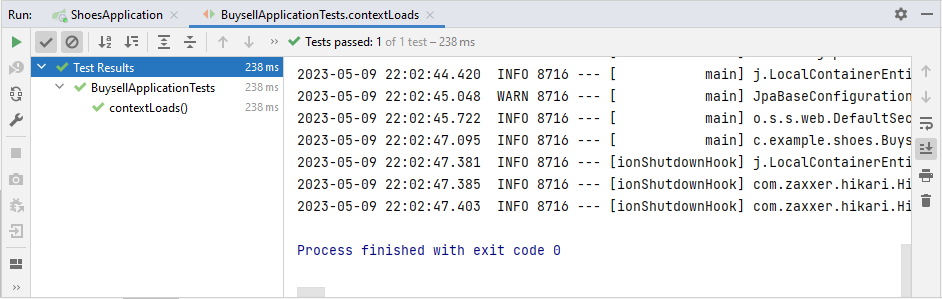
Для тестирования работы данных методов было проведено функциональное тестирование и разработан тестовый класс BuysellApplicationTests. Результат тестирования приведен на рисунке 3.1.

Рисунок 3.1 – Результат тестирования

Программа также подвергалась ручному тестированию, в ходе которого было выявлено, что присутствуют проверки на полях с введением электронного адреса при регистрации и авторизации, что представлено на рисунках 3.2 и 3.3 соответственно. А также при добавлении отзыва, если указывается оценка больше 5 баллов, также выдает сообщение о том, что правильно будет ввести в данную ячейку, что представлено на рисунке 3.4.

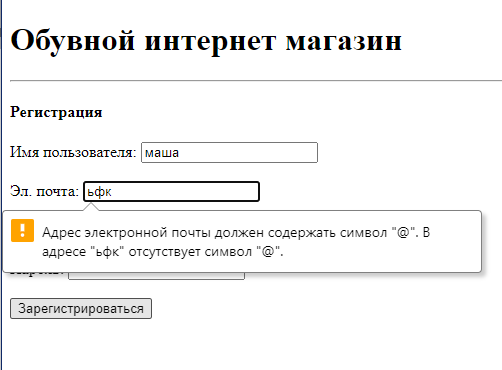
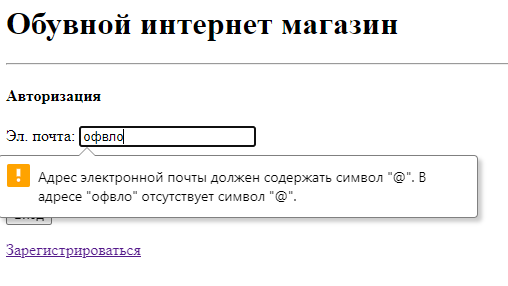


Рисунок 3.2 – Ошибка ввода электронного адреса при регистрации



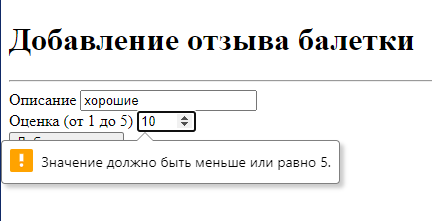
Рисунок 3.3 – Ошибка ввода электронного адреса при авторизации

Рисунок 3.4 – Ошибка ввода оценки при добавлении отзыва

# **4 РУКОВОДСТВО ПО РАЗВЕРТЫВАНИЮ И ИСПОЛЬЗОВАНИЮ ПРОГРАММНОГО СРЕДСТВА**

Приложение работает в двух режимах: пользователя и администратора. Рассмотри сначала работу приложения в режиме пользователя.

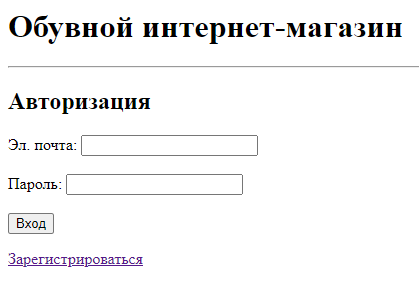
Когда пользователь заходит в приложение, сразу он видит форму авторизации. Форма авторизации представлена на рисунке 4.1.

Рисунок 4.1 – Форма авторизации

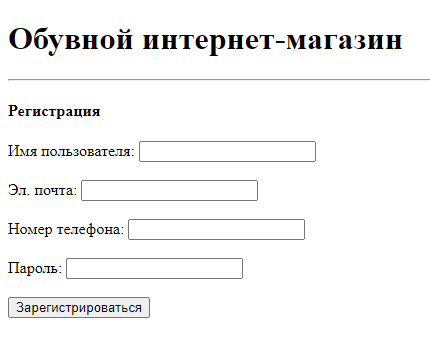
Если же пользователь еще не авторизован, то он должен нажать на кнопку «Зарегистрироваться», после чего будет перенаправлен на форму регистрации, которая представлена на рисунке 4.2.

Рисунок 4.2 – Форма регистрации

После входа появляется главная страница, где представлены кнопка выйти, чтобы выйти из личного кабинета; поиск обуви по категории; список доступных моделей обуви, отзывы к ним, корзина и блог. Главная страница представлена на рисунке 4.3.

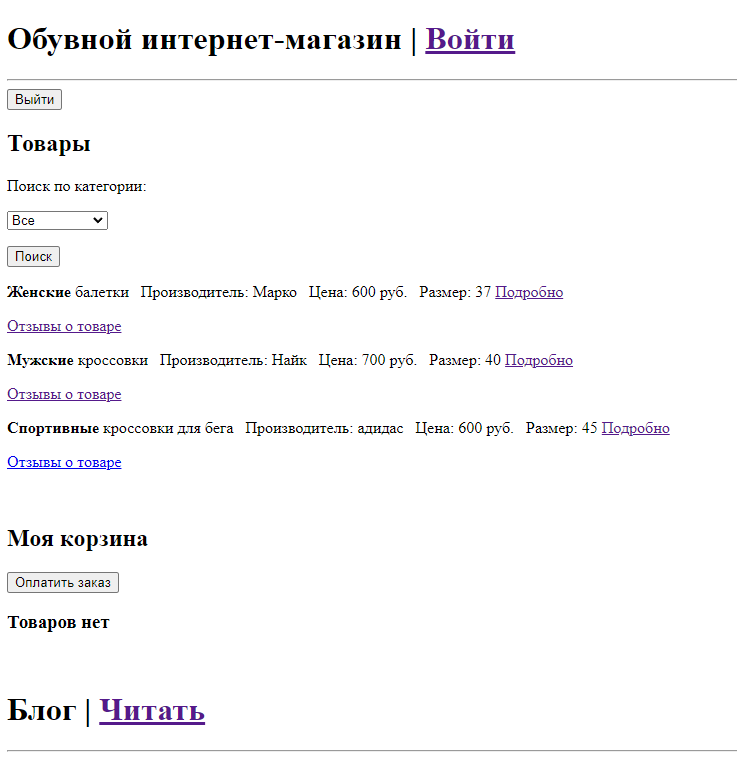


Рисунок 4.3 – Главная страница интернет-магазина

Если пользователь нажимает кнопку выйти, то он снова оказывается на странице авторизации.

Поиск по категории представлен в виде выпадающего списка. Поиск представлен на рисунке 4.4. Результат поиска по категории «Женские» на рисунке 4.5.

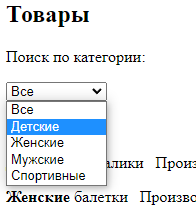


Рисунок 4.4 – Выпадающий список поиска по категории

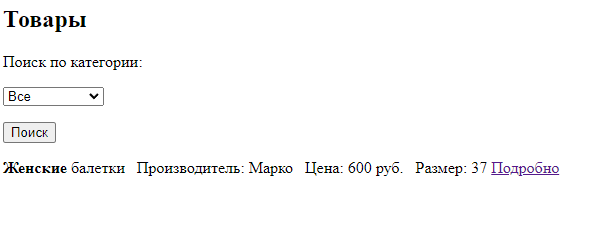


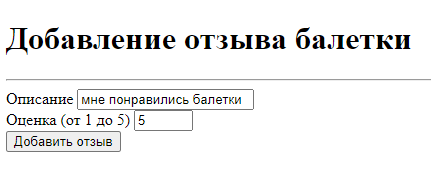
Рисунок 4.5 – Результат поиска по категории «Женские»

Также, если пользователь нажмет на кнопку «Подробнее», то увидит подробную информацию о модели обуви, такие как: категория, описание, производитель, цена и размер. Страница с подробной информацией о товаре представлена на рисунке 4.6.



Рисунок 4.6 – Страница с подробной информацией о товаре

Если пользователь захочет добавить отзыв о товаре, то он может нажать кнопку «Добавить отзыв о товаре». Появляется форма, в которой можно ввести сам текст отзыва и оценку от 1 до 5. Посмотреть отзывы можно нажав кнопку «Отзывы о товаре». Пример введения отзыва представлен на рисунке 4.7.

Рисунок 4.7 – Добавление отзыва о товаре

Если пользователь нажимает кнопку «Добавить товар в корзину», то его перенаправляет на главную страницу, где можно будет увидеть, что товар добавлен в корзину. Также присутствует функция оплатить товар. Корзина представлена на рисунке 4.8.

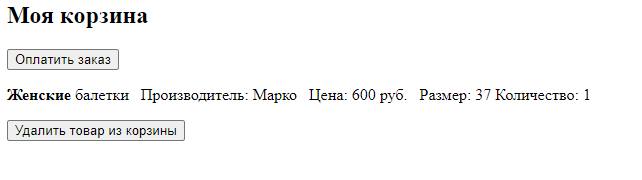


Рисунок 4.8 – Корзина пользователя

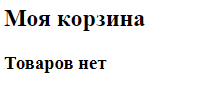
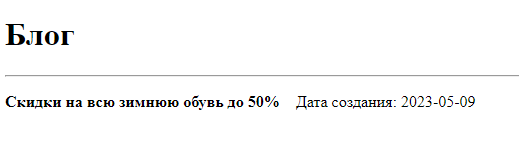
Если пользователь нажимает кнопку «Удалить товар из корзины», то корзина очищается. Пустая корзина представлена на рисунке 4.9.

Рисунок 4.9 – Пустая корзина

Если пользователь зашел в систему, затем добавил необходимые модели в корзину и вышел из системы, то при повторном входе в систему список его товаров корзине вновь отобразится.

Пользователь может просмотреть информацию в блоге. Страница блога представлена на рисунке 4.10.

Рисунок 4.10 – Страница блога

Таким образом, были рассмотрены функции пользователя, далее рассмотрим функции администратора.

Теперь рассмотрим работу приложения в режиме администратора.

Сначала администратор так же, как и пользователь должен авторизоваться. Затем он направляется на главную страницу, отличие которой от главной страницы для пользователя в том, что для администратора на ней есть функция «Создать новый товар», которая представлена на рисунке 4.11.

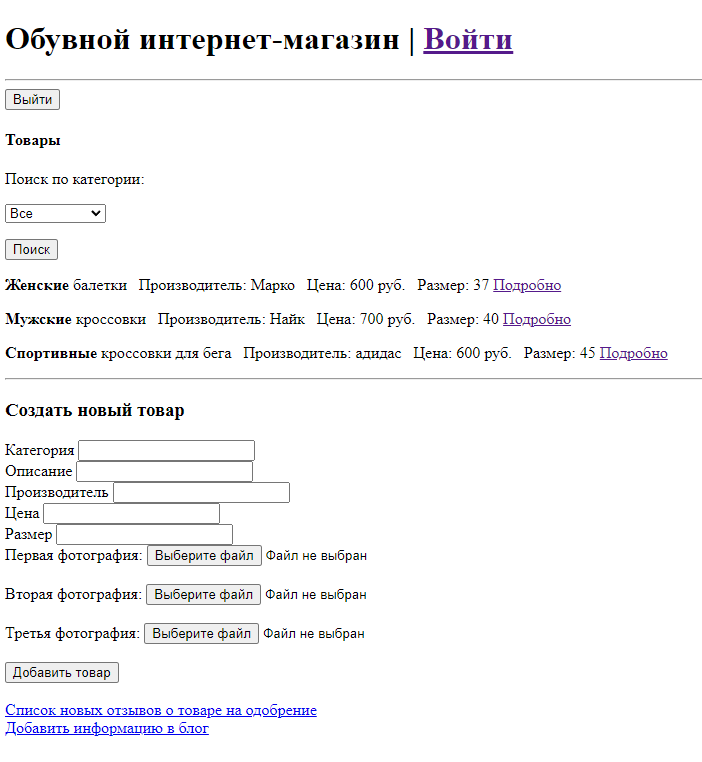


Рисунок 4.11 – Главная страница для администратора

Администратор может добавлять новые товары в каталог. Процесс создания нового товара представлен на рисунке 4.12.

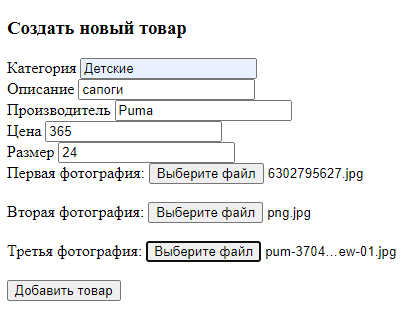


Рисунок 4.12 – Процесс создания нового товара

После создания товара он появляется в каталоге. Каталог товаров с добавленным новым товаром представлен на рисунке 4.13.

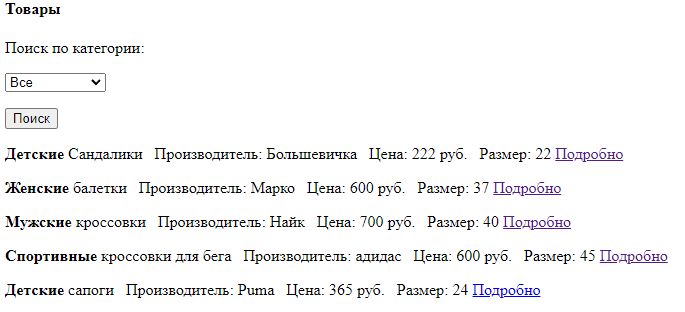


Рисунок 4.13 – Новый добавленный товар в каталоге

Подробная информация о новом добавленном товаре представлена на рисунке 4.14.

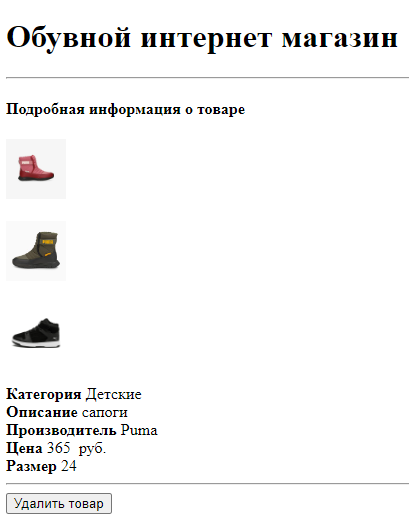


Рисунок 4.14 – Подробная информация о новом товаре

Если администратор нажимает на кнопку «Удалить товар», то его вернет на главную страницу и в каталоге этого товара уже не будет. Каталог после удаления товара представлен на рисунке 4.15.

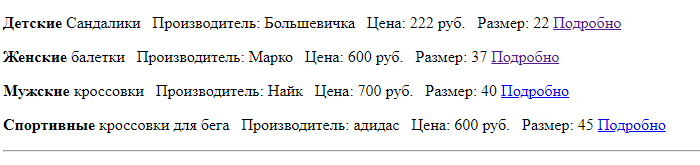


Рисунок 4.15 – Каталог после удаления товара

Присутствует функция одобрения отзывов, оставленных пользователями. После проверки отзыва администратор нажимает кнопку «Одобрить размещение и отзыв добавляется для просмотра другими пользователями. Страница отзывов, которые ждут одобрения администратором представлена на рисунке 4.16.

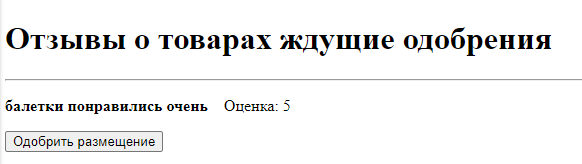


Рисунок 4.16 – Страница отзывов, которые находятся в ожидании одобрения администратора

Присутствует возможность добавления новой информации, такой как новости или статьи в блог. Добавление информации в блог представлена на рисунке 4.17.

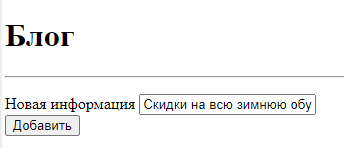


Рисунок 4.17 – Добавление информации в блог

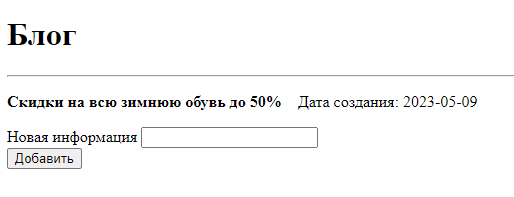
Результат добавления новой информации в блог представлен на рисунке 4.18.

Рисунок 4.18 – Результат добавления информации в блог

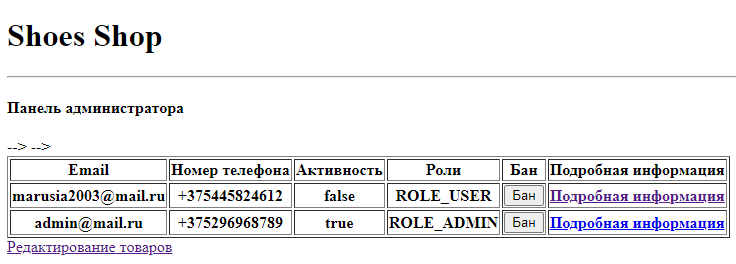
Панель администратора, в которой администратор может заблокировать пользователя, увидеть его электронный адрес, номер телефона, активность, роль, посмотреть подробную информацию, представлена на рисунке 4.19.

Рисунок 4.19 – Панель администратора

Когда администратор нажимает кнопку «Бан», в поле «Активность» активность меняется с true на false. На рисунке 4.20 изображена измененная активность заблокированного пользователя.

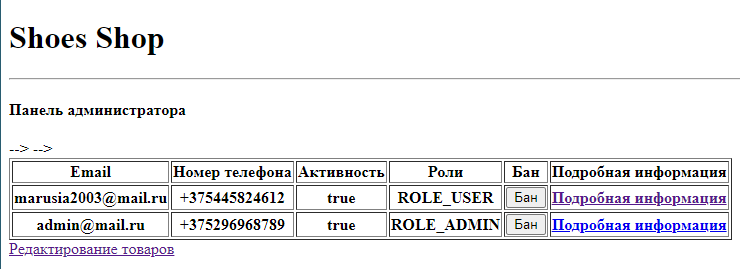


Рисунок 4.20 – Измененная активность заблокированного пользователя

Чтобы разблокировать пользователя, администратору нужно снова нажать кнопку «Бан» и активность изменится снова на true.

Просмотреть подробную информацию о пользователе администратор может нажав кнопку «Подробная информация», где указаны имя пользователя, его номер телефона и электронный адрес. Подробная информация о пользователе представлена на рисунке 4.21.

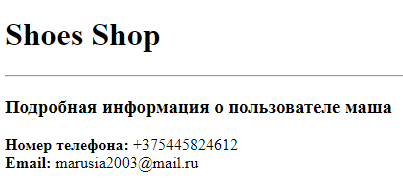


Рисунок 4.21 – Подробная информация о клиенте

В итоге была подробно рассмотрена работа программы в двух режимах и дана исчерпывающая инструкция по пользованию интернет-магазином обуви.

# **ЗАКЛЮЧЕНИЕ**

При выполнении курсовой работы были исследованы процессы продажи обуви посредством интернет-магазина и доказана актуальность автоматизации некоторых процессов, принадлежащих данной сфере человеческой деятельности. Далее была поставлена цель о выделении задачи разработки системы. Затем предметная область детализировалась и разрабатывались диаграммы UML для последующей разработки программы. Были разработаны схемы работы важнейших функций программы.

В результате была разработана программа, позволяющая безопасно, быстро и удобно производить продажу обуви через Интернет. Были разработаны различные классы и методы работы с информацией, представленной в них.

В приложении реализованы режимы администратора и пользователя. Также были использованы современные технологии авторизации и аутентификации, а также ограничения доступа к ресурсам, право на доступ к которым имеют конкретные группы пользователей.

Приложение имеет дружественный пользовательский интерфейс, описанный в руководстве пользователя. В приложении реализована работа с базой данных SQL при помощи фреймворка Spring.

Программу можно улучшить путем создания дочерних классов и расширения функционала всех видов пользователей.

Таким образом, программа устойчива к различным непредвиденным ситуациям и хорошо масштабируема за счет использования объектно-ориентированного программирования и использования современных технологий разработки.

# **СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ**

[1] Что такое Java? [Электронный ресурс]. – Режим доступа : <https://aws.amazon.com/ru/what-is/java/>.

[2] Рейтинг востребованности языков программирования [Электронный ресурс]. – Режим доступа : <https://www.tadviser.ru/index.php>.

[3] Преимущества Java [Электронный ресурс]. – Режим доступа : <https://www.ibm.com/docs/ru/aix/7.2?topic=monitoring-advantages-java>.

[4] Знакомство со Spring Boot [Электронный ресурс]. – Режим доступа : <https://topjava.ru/blog/introducing-spring-boot>.

[5] Что такое MySQL [Электронный ресурс]. – Режим доступа : <https://help.reg.ru/support/hosting/bazy-dannykh/chto-takoye-mysql>.

[6] Что такое MySQL [Электронный ресурс]. – Режим доступа : <https://www.nic.ru/help/chto-takoe-mysql_8510.html>.

[7] Что такое MVC: рассказываем простыми словами [Электронный ресурс]. – Режим доступа : <https://ru.hexlet.io/blog/posts/chto-takoe-mvc-rasskazyvaem-prostymi-slovami>.

[8] Плюсы и минусы использования Spring Boot [Электронный ресурс]. – Режим доступа : <https://scand.com/ru/company/blog/plyusy-i-minusy-ispolzovaniya-spring-boot/>.

# **ПРИЛОЖЕНИЕ А**

# **(обязательное)**

# **Отчет о проверке на заимствования в системе «Антиплагиат»**

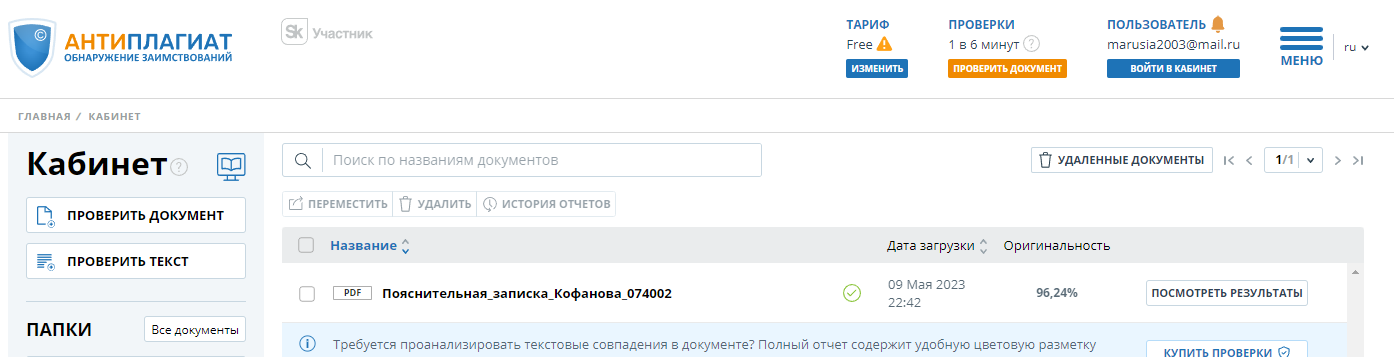


Рисунок А.1 – Отчет о проверке на заимствования в системе «Антиплагиат»

# **ПРИЛОЖЕНИЕ Б**

# **(обязательное)**

# **Листинг кода программы**

AdminController.java

package com.example.shoes.controllers;

import com.example.shoes.models.Shoes;

import com.example.shoes.services.ShoesService;

import com.example.shoes.services.UserService;

import lombok.RequiredArgsConstructor;

import org.springframework.security.access.prepost.PreAuthorize;

import org.springframework.stereotype.Controller;

import org.springframework.ui.Model;

import org.springframework.web.bind.annotation.GetMapping;

import org.springframework.web.bind.annotation.PathVariable;

import org.springframework.web.bind.annotation.PostMapping;

import org.springframework.web.bind.annotation.RequestParam;

@Controller

@RequiredArgsConstructor

@PreAuthorize("hasAuthority('ROLE\_ADMIN')")

public class AdminController {

private final UserService userService;

private final ShoesService shoesService;

@GetMapping("/admin")

public String admin(Model model){

model.addAttribute("users",userService.getAllUsers());

return "admin";

}

@PostMapping("/admin/user/ban/{id}")

public String userBan(@PathVariable("id") Long id) {

userService.banUser(id);

return "redirect:/admin";

}

@GetMapping("/admin/products")

public String products(@RequestParam(name = "category", required = false) String category, Model model) {

model.addAttribute("products", shoesService.listShoes(category));

Продолжение приложения Б

return "admin-products";

}

@PostMapping("/product/delete/{id}")

public String deleteProduct(@PathVariable Long id) {

shoesService.deleteShoe(id);

return "redirect:/admin/products";

}

@GetMapping("admin/product/{id}")

public String productInfo(@PathVariable Long id, Model model) {

Shoes shoe = shoesService.getShoeById(id);

model.addAttribute("shoe", shoe);

model.addAttribute("images", shoe.getImages());

return "admin-shoe-info";

}

}

ImageController.java

package com.example.shoes.controllers;

import com.example.shoes.models.Image;

import com.example.shoes.repositories.ImageRepository;

import lombok.RequiredArgsConstructor;

import org.springframework.core.io.InputStreamResource;

import org.springframework.http.MediaType;

import org.springframework.http.ResponseEntity;

import org.springframework.web.bind.annotation.GetMapping;

import org.springframework.web.bind.annotation.PathVariable;

import org.springframework.web.bind.annotation.RestController;

import java.io.ByteArrayInputStream;

@RestController

@RequiredArgsConstructor

public class ImageController {

private final ImageRepository imageRepository;

@GetMapping("/images/{id}")

private ResponseEntity<?> getImageById(@PathVariable Long id) {

Image image = imageRepository.findById(id).orElse(null);

Продолжение приложения Б

return ResponseEntity.ok()

.header("fileName", image.getOriginalFileName())

.contentType(MediaType.valueOf(image.getContentType()))

.contentLength(image.getSize())

.body(new InputStreamResource(new ByteArrayInputStream(image.getBytes())));

}

}

ShoeController.java

package com.example.shoes.controllers;

import com.example.shoes.models.Shoes;

import com.example.shoes.models.User;

import com.example.shoes.models.enums.Role;

import com.example.shoes.repositories.UserRepository;

import com.example.shoes.services.BucketService;

import com.example.shoes.services.CustomUserDetailsService;

import com.example.shoes.services.ShoesService;

import com.example.shoes.services.UserService;

import lombok.RequiredArgsConstructor;

import org.springframework.security.authentication.UsernamePasswordAuthenticationToken;

import org.springframework.security.core.Authentication;

import org.springframework.security.core.context.SecurityContextHolder;

import org.springframework.stereotype.Controller;

import org.springframework.ui.Model;

import org.springframework.web.bind.annotation.GetMapping;

import org.springframework.web.bind.annotation.PathVariable;

import org.springframework.web.bind.annotation.PostMapping;

import org.springframework.web.bind.annotation.RequestParam;

import org.springframework.web.multipart.MultipartFile;

import java.io.IOException;

import java.security.Principal;

@Controller

Продолжение приложения Б

@RequiredArgsConstructor

public class ShoeController {

private final ShoesService shoesService;

private final UserService userService;

private final BucketService bucketService;

private final UserRepository userRepository;

@GetMapping("/")

public String products(@RequestParam(name = "category", required = false) String category, Model model, Principal principal) {

if(principal==null){

return "redirect:/login";

}

model.addAttribute("products", shoesService.listShoes(category));

model.addAttribute("buckets", bucketService.getAllProductsByUserId());

if(((UsernamePasswordAuthenticationToken) principal).getAuthorities().contains(Role.ROLE\_ADMIN)){

return "admin/products";

}

if(((UsernamePasswordAuthenticationToken) principal).getAuthorities().contains(Role.ROLE\_USER)){

return "products";

}

return "redirect:/login";

}

@GetMapping("/product/{id}")

public String productInfo(@PathVariable Long id, Model model) {

Shoes shoe = shoesService.getShoeById(id);

model.addAttribute("shoe", shoe);

model.addAttribute("images", shoe.getImages());

return "shoe-info";

}

@PostMapping("/product/create")

public String createProduct(@RequestParam("file1") MultipartFile file1, @RequestParam("file2") MultipartFile file2,

Продолжение приложения Б

@RequestParam("file3") MultipartFile file3, Shoes product) throws IOException {

shoesService.saveShoe(product, file1, file2, file3);

return "redirect:/";

}

@PostMapping("/product/addBucket/{id}")

public String addBucket(@PathVariable Long id) {

userService.addBucket(id);

return "redirect:/";

}

@PostMapping("/bucket/delete/{id}")

public String deleteFromBucket(@PathVariable Long id){

bucketService.deleteProductFromBucket(id);

System.out.println();

return "redirect:/";

}

}

UserController.java

package com.example.shoes.controllers;

import com.example.shoes.models.User;

import com.example.shoes.services.UserService;

import lombok.RequiredArgsConstructor;

import org.springframework.stereotype.Controller;

import org.springframework.ui.Model;

import org.springframework.web.bind.annotation.GetMapping;

import org.springframework.web.bind.annotation.PathVariable;

import org.springframework.web.bind.annotation.PostMapping;

@Controller

@RequiredArgsConstructor

public class UserController {

private final UserService userService;

@GetMapping("/login")

public String login(){

Продолжение приложения Б

return "login";

}

@GetMapping("/registration")

public String registration() {

return "registration";

}

@PostMapping("/registration")

public String createUser(User user, Model model){

if(!userService.createUser(user)){

model.addAttribute("errorMessage", "Пользователь с email: "+ user.getEmail()+" уже существует");

return "/registration";

}

return "/redirect:/login";

}

@GetMapping("/user/{user}")

public String userInfo(@PathVariable("user") User user, Model model) {

model.addAttribute("user", user);

// model.addAttribute("products", user.getProducts());

return "user-info";

}

@GetMapping("/hello")

public String securityUrl(){

return "hello";

}

}

# **ПРИЛОЖЕНИЕ В**

# **(обязательное)**

# **Листинг скрипта генерации базы данных**

CREATE TABLE `users` (

`id` bigint NOT NULL AUTO\_INCREMENT,

`active` bit(1) DEFAULT NULL,

`date\_of\_create` datetime(6) DEFAULT NULL,

`email` varchar(255) DEFAULT NULL,

`name` varchar(255) DEFAULT NULL,

`password` varchar(1000) DEFAULT NULL,

`phone\_number` varchar(255) DEFAULT NULL,

`image\_id` bigint DEFAULT NULL,

PRIMARY KEY (`id`),

UNIQUE KEY `UK\_6dotkott2kjsp8vw4d0m25fb7` (`email`),

KEY `FK17herqt2to4hyl5q5r5ogbxk9` (`image\_id`),

CONSTRAINT `FK17herqt2to4hyl5q5r5ogbxk9` FOREIGN KEY (`image\_id`) REFERENCES `images` (`id`)

) ENGINE=InnoDB AUTO\_INCREMENT=4 DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_0900\_ai\_ci;

CREATE TABLE `images` (

`id` bigint NOT NULL,

`bytes` longblob,

`content\_type` varchar(255) DEFAULT NULL,

`name` varchar(255) DEFAULT NULL,

`original\_file\_name` varchar(255) DEFAULT NULL,

`size` bigint DEFAULT NULL,

`product\_id` bigint DEFAULT NULL,

`is\_preview\_image` bit(1) DEFAULT NULL,

PRIMARY KEY (`id`),

Продолжение приложения В

KEY `FKghwsjbjo7mg3iufxruvq6iu3q` (`product\_id`),

CONSTRAINT `FKghwsjbjo7mg3iufxruvq6iu3q` FOREIGN KEY (`product\_id`) REFERENCES `products` (`id`)

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_0900\_ai\_ci;

CREATE TABLE `user\_role` (

`user\_id` bigint NOT NULL,

`roles` varchar(255) DEFAULT NULL,

KEY `FKj345gk1bovqvfame88rcx7yyx` (`user\_id`),

CONSTRAINT `FKj345gk1bovqvfame88rcx7yyx` FOREIGN KEY (`user\_id`) REFERENCES `users` (`id`)

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_0900\_ai\_ci;

CREATE TABLE `products` (

`id` bigint NOT NULL,

`category` varchar(255) DEFAULT NULL,

`description` text,

`manufacturer` varchar(255) DEFAULT NULL,

`price` double DEFAULT NULL,

`size` double DEFAULT NULL,

`date\_of\_created` datetime(6) DEFAULT NULL,

`preview\_image\_id` bigint DEFAULT NULL,

PRIMARY KEY (`id`)

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_0900\_ai\_ci;

CREATE TABLE `buckets` (

`id` bigint NOT NULL,

`product\_id` bigint DEFAULT NULL,

`quantity` int DEFAULT NULL,

`status` varchar(255) DEFAULT NULL,

Продолжение приложения В

`user\_id` bigint DEFAULT NULL,

PRIMARY KEY (`id`)

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_0900\_ai\_ci;