Министерство образования Республики Беларусь

Учреждение образования

БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ

Инженерно-экономический факультет

Кафедра экономической информатики

Дисциплина Распределенные системы обработки информации

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

к курсовому проекту на тему:

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

Выполнил: студент гр. 074002

Кофанова Мария Сергеевна

Проверил: ассистент кафедры ЭИ

Лыщик Артем Петрович

Минск 2023

**СОДЕРЖАНИЕ**

[Введение 5](#_Toc133957000)

[1 Описание деятельности интернет-магазина обуви 6](#_Toc133957001)

[2 Постановка задачи разработки программного средства для осуществления продаж в интернет-магазине обуви 8](#_Toc133957002)

[3 Модели представления интернет-магазина обуви и их написания 11](#_Toc133957003)

[3.1 Диаграмма вариантов использования системы 15](#_Toc133957004)

[3.2 Диаграмма последовательностей приложения 16](#_Toc133957005)

[3.3 Диаграмма состояний процесса поиска обуви 17](#_Toc133957006)

[3.4 Диаграмма компонентов 18](#_Toc133957007)

[3.5 Диаграмма развертывания 18](#_Toc133957008)

[3.6 Диаграмма классов 19](#_Toc133957009)

[4 Разработка информационной модели интернет-магазина обуви 20](#_Toc133957010)

[5 Архитектурные решения для разработки интернет-магазина по продаже обуви 22](#_Toc133957011)

[6 Описание алгоритмов, реализующих бизнес-логику веб-приложения 25](#_Toc133957012)

[7 Руководство по использованию интернет-магазина обуви 26](#_Toc133957013)

[7.1 Работа в режиме пользователя 26](#_Toc133957014)

[7.1 Работа в режиме администратора интернет-магазина обуви 29](#_Toc133957015)

[Заключение 34](#_Toc133957016)

[Список использованных источников 35](#_Toc133957017)

[Приложение А (обязательное) Диаграмма классов 36](#_Toc133957018)

[Приложение Б (обязательное) Схема алгоритма работы программы 37](#_Toc133957021)

[Приложение В (обязательное) Листинг кода алгоритмов, реализующих основную бизнес-логику 38](#_Toc133957024)

[Приложение Г (обязательное) Листинг скрипта генерации базы данных 44](#_Toc133957027)

# **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 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.

# **2 ПОСТАНОВКА ЗАДАЧИ РАЗРАБОТКИ ПРОГРАММНОГО СРЕДСТВА ДЛЯ ОСУЩЕСТВЛЕНИЯ ПРОДАЖ В ИНТЕРНЕТ-МАГАЗИНЕ ОБУВИ**

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:

* просмотр информации о пользователях;
* блокировка пользователей;
* удаление пользователей;
* добавление/удаление товаров;
* редактирование товаров.

Пользователю доступны следующие функции:

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

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.

# **3 МОДЕЛИ ПРЕДСТАВЛЕНИЯ ИНТЕРНЕТ-МАГАЗИНА ОБУВИ И ИХ ОПИСАНИЕ**

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 3.1 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:

* админ;
* клиент;
* доставщик.



Рисунок 3.1 – Контекстная диаграмма верхнего уровня

Диаграмма верхнего уровня декомпозируется на 4 блока: «Добавление товара в корзину», «Оформление заказа», «Контроль выполнения заказа» и «Доставка товара». Декомпозиция представлена на рисунке 3.2.



Рисунок 3.2 – Декомпозиция контекстной диаграммы верхнего уровня

Произведём декомпозицию блока «Проконтролировать выполнение заказа», изображенную на рисунке 3.3, который делится на следующие подпроцессы: «Проверка наличия товаров», «Отправка данных о заказе на склад» и «Подготовка заказа к доставке».



Рисунок 3.3 – Декомпозиция блока «Контроль выполнения заказа»

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.

## **3.1 Диаграмма вариантов использования системы**

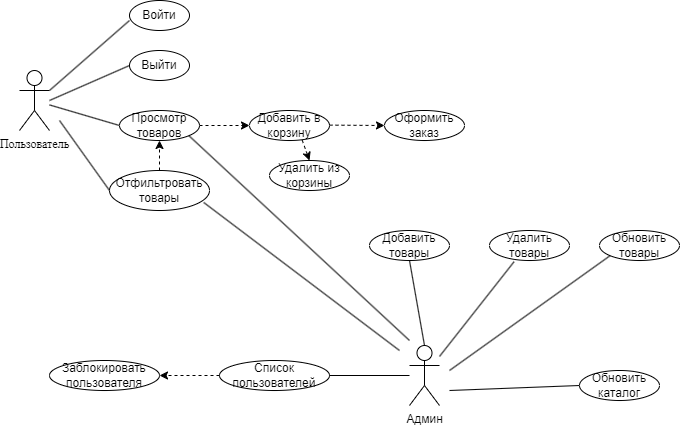
****

Рисунок 3.4 – 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/обновление товаров;
* просмотр и фильтрация товаров;
* блокировка пользователя;
* просмотр списка пользователей;
* обновление каталога.

Клиенту доступны следующие функции:

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

Все пользователи могут авторизоваться.

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

## **3.2 Диаграмма последовательностей приложения**

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.

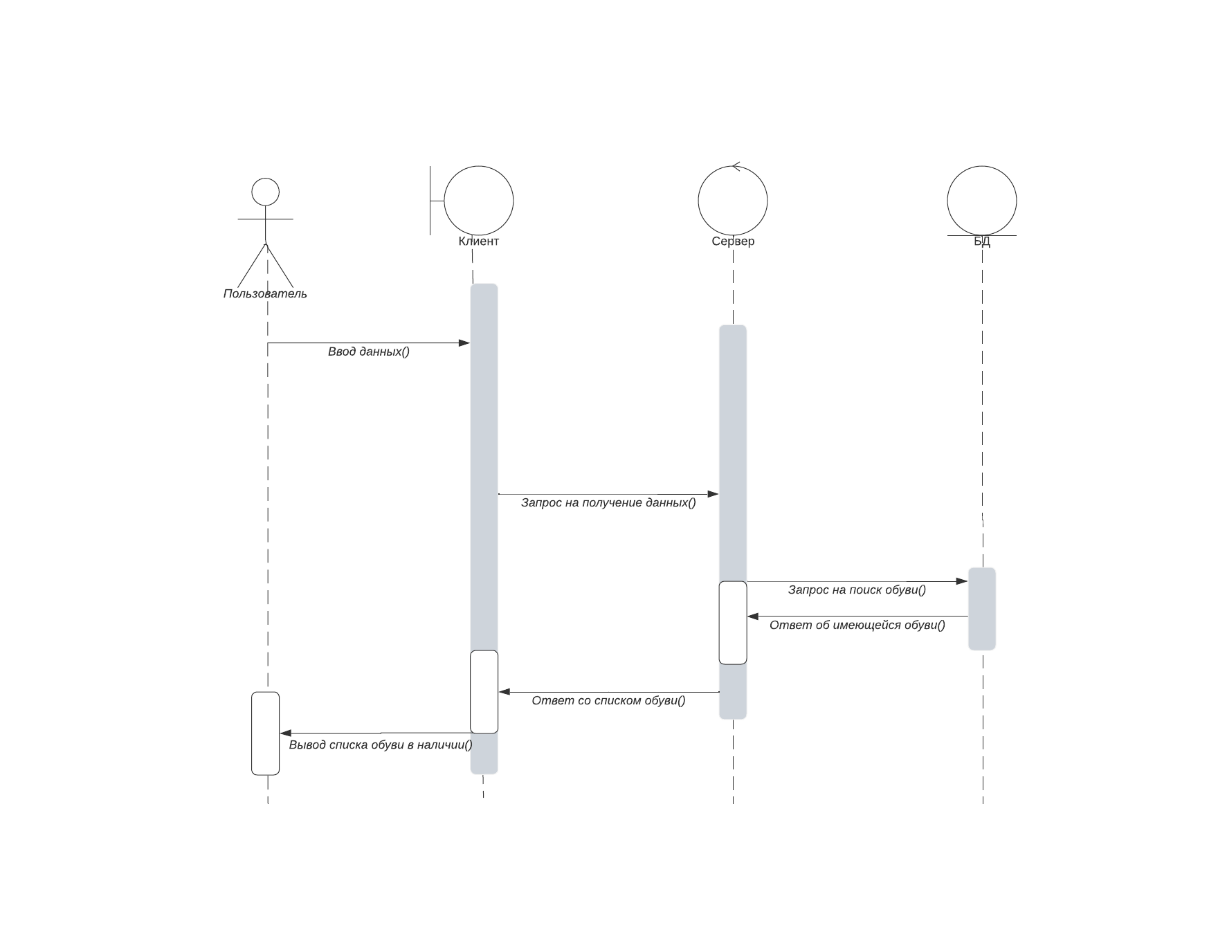


Рисунок 3.5 – Диаграмма последовательности

## **3.3 Диаграмма состояний процесса поиска обуви**

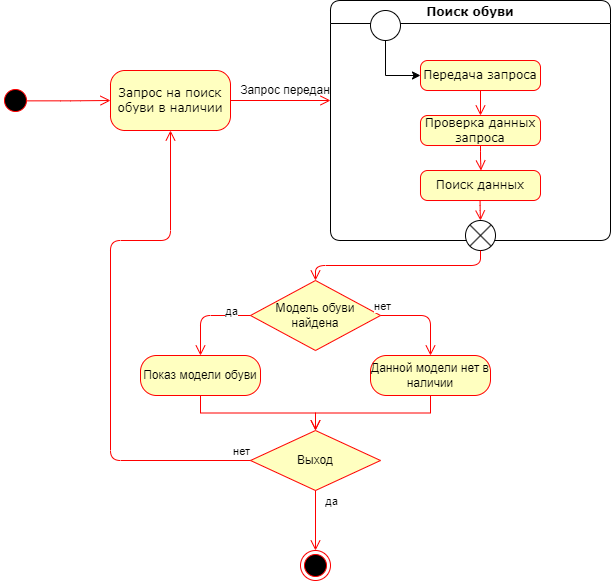
Диаграмма состояний показывает, как объект переходит из одного состояния в другое. Диаграммы состояний служат для моделирования динамических аспектов системы. Данная диаграмма полезна при моделировании жизненного цикла объекта.

Рисунок 3.6 – Диаграмма состояний

Изначально сервер находится в режиме ожидания запросов клиента, затем он принимает введенные пользователем данные. Далее запрос пользователя передается на проверку, происходит поиск данных. В случае нахождения модели обуви необходимой клиенту происходит ее показ. В обратном случае – данной модели нет в наличии. Далее происходит выход из системы, если же пользователь выбирает обратное действие, то процесс начинается заново, с ввода пользователем необходимой информации для поиска новой модели обуви.

## **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:

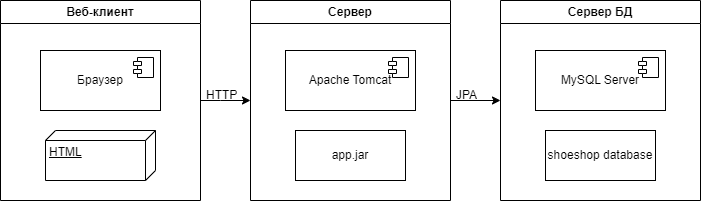
* используются в компонентно-ориентированных разработках для описания систем с сервис-ориентированной архитектурой;
* показывают структуру самого кода;
* могут использоваться для фокусировки на отношениях между компонентами, скрывая при этом детализацию спецификации;
* помогают в информировании и разъяснении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

Рисунок 3.7 – Диаграмма компонентов

## **3.5 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.

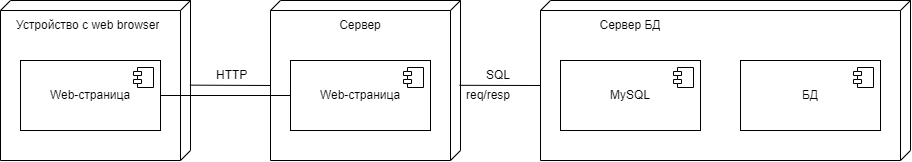


Рисунок 3.8 – Диаграмма развертывания

## **3.6 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. Диаграмма классов представлена в приложении А.

# **4 РАЗРАБОТКА ИНФОРМАЦИОННОЙ МОДЕЛИ ИНТЕРНЕТ-МАГАЗИНА ОБУВИ**

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.

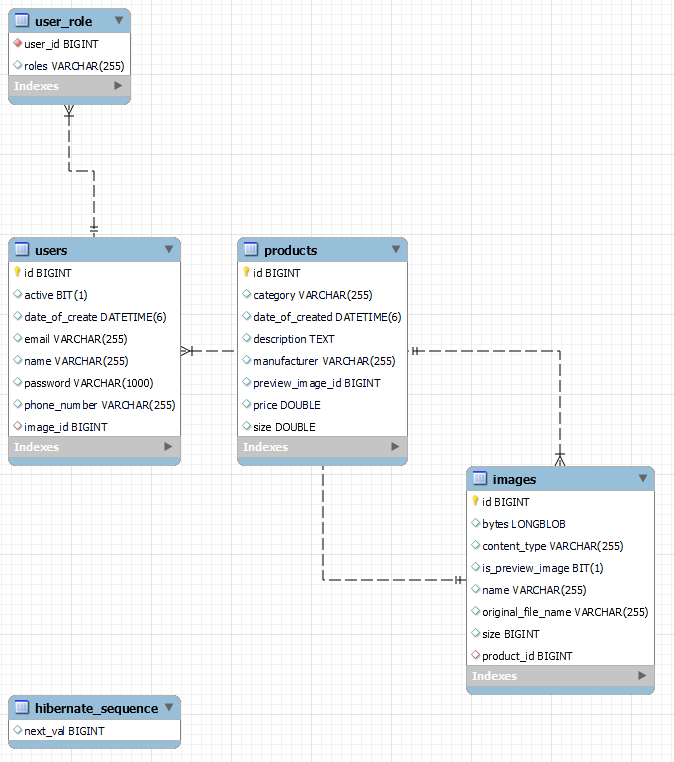
Информационная модель баз данных представлена на рисунке 4.1.

Рисунок 4.1 – Информационная модель базы данных

На данной информационной модели представлено 6 сущностей: user\_role, users, products, images, 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 hibernate\_sequence содержит один атрибут: next\_val.

# **5 АРХИТЕКТУРНЫЕ РЕШЕНИЯ ДЛЯ РАЗРАБОТКИ ИНТЕРНЕТ-МАГАЗИНА ПО ПРОДАЖЕ ОБУВИ**

При разработке веб-приложения был использован архитектурный паттерн MVC.

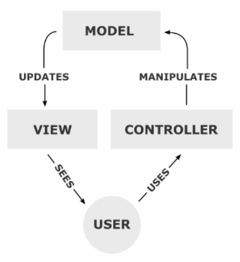


Рисунок 5.1 – Структура архитектурного паттерна MVC

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.

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 5.2 – Принцип работы шаблонов FreeMarker

# 

# **6 ОПИСАНИЕ АЛГОРИТМОВ, РЕАЛИЗУЮЩИХ БИЗНЕС-ЛОГИКУ ВЕБ-ПРИЛОЖЕНИЯ**

Схема алгоритма работы приложения при входе с ролью администратора представлена в Приложении А.

Из нее видно, что сначала администратор направляется на панель администратора, и оттуда он может работать с товарами, либо с пользователями.

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

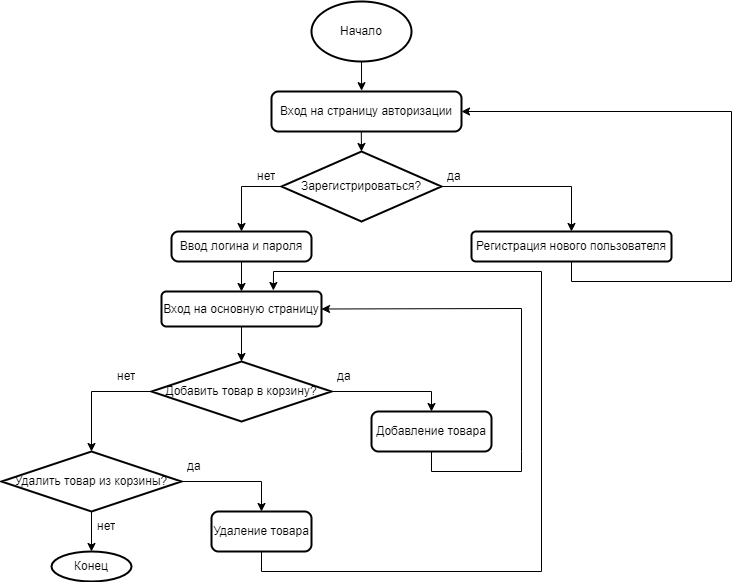
Далее рассмотри схему алгоритма процессов входа и добавления товаров в корзину, представленную на рисунке 6.1.

Рисунок 6.1 – 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.

# **7 РУКОВОДСТВО ПО ИСПОЛЬЗОВАНИЮ ИНТЕРНЕТ-МАГАЗИНА ОБУВИ**

Приложение работает в двух режимах: пользователь и администратор.

## **7.1 Работа в режиме пользователя**

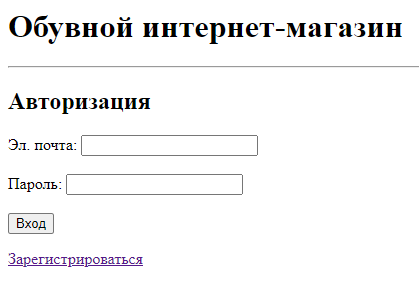
Форма авторизации представлена на рисунке 7.1.

Рисунок 7.1 – Форма авторизации

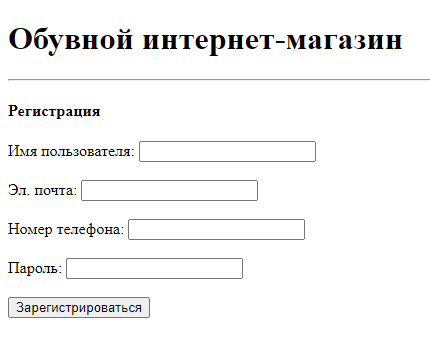
Если же пользователь еще не авторизован, то он должен нажать на кнопку «Зарегистрироваться», после чего будет перенаправлен на форму регистрации, которая представлена на рисунке 7.2.

Рисунок 7.2 – Форма регистрации

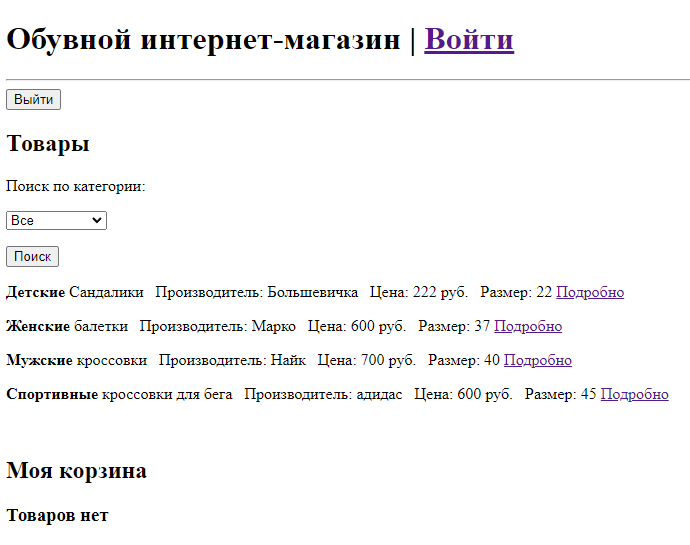
После входа появляется главная страница, где представлены кнопка выйти, чтобы выйти из личного кабинета; поиск обуви по категории; список доступных моделей обуви и корзина. Главная страница представлена на рисунке 7.3.

Рисунок 7.3 – Главная страница интернет-магазина

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

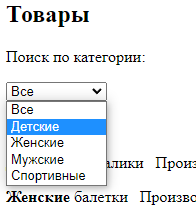
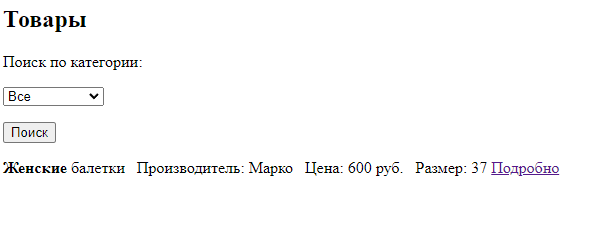
Поиск по категории представлен в виде выпадающего списка. Поиск представлен на рисунке 7.4. Результат поиска по категории «Женские» на рисунке 7.5.

Рисунок 7.4 – Выпадающий список поиска по категории

Рисунок 7.5 – Результат поиска по категории «Женские»

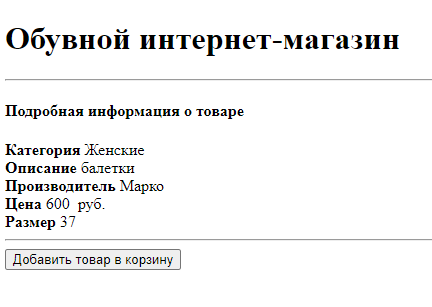
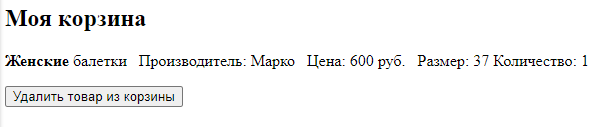
Также, если пользователь нажмет на кнопку «Подробнее», то увидит подробную информацию о модели обуви, такие как: категория, описание, производитель, цена и размер. Страница с подробной информацией о товаре представлена на рисунке 7.6.

Рисунок 7.6 – Страница с подробной информацией 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. Корзина представлена на рисунке 7.7.

Рисунок 7.7 – Корзина пользователя

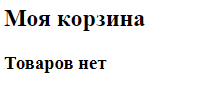
Если пользователь нажимает кнопку «Удалить товар из корзины», то корзина очищается. Пустая корзина представлена на рисунке 7.8.

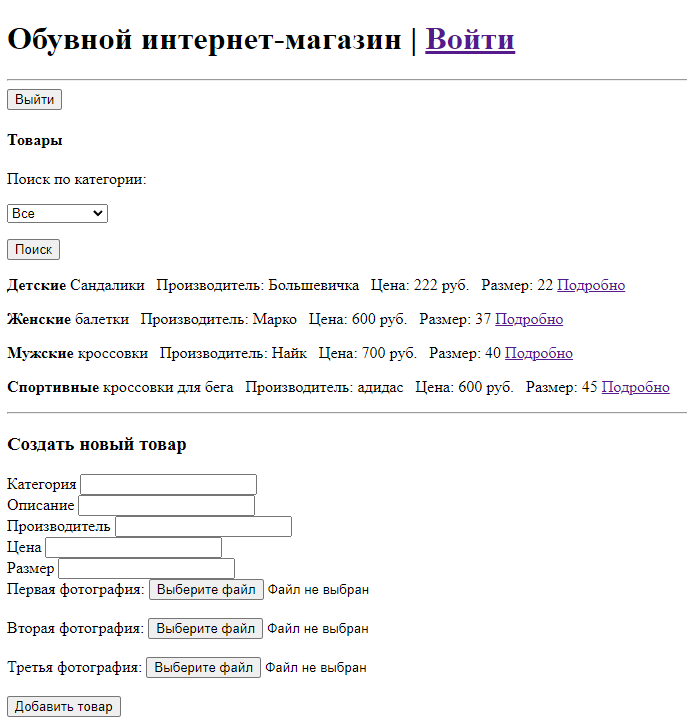
Рисунок 7.8 – Пустая корзина

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.

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

## **7.1 Работа в режиме администратора интернет-магазина обуви**

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 7.9.

Рисунок 7.9 – Главная страница для администратора

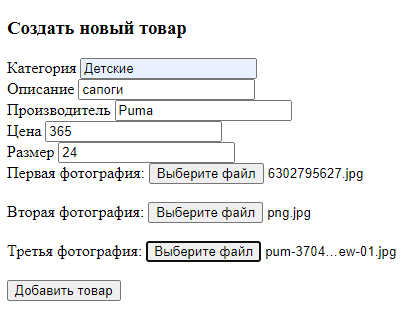
 Администратор может добавлять новые товары в каталог. Процесс создания нового товара представлен на рисунке 7.10.

Рисунок 7.10 – Процесс создания нового товара

После создания товара он появляется в каталоге. Каталог товаров с добавленным новым товаром представлен на рисунке 7.11.

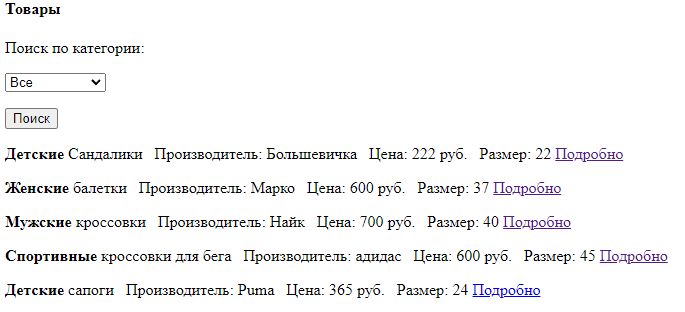


Рисунок 7.11 – Новый добавленный товар в каталоге

Подробная информация о новом добавленном товаре представлена на рисунке 7.12.

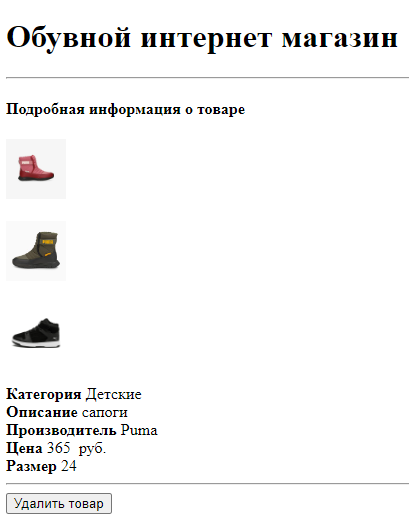


Рисунок 7.12 – Подробная информация о новом товаре

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 7.13.

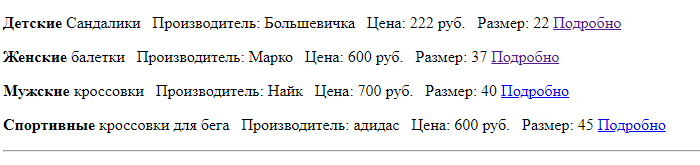


Рисунок 7.13 – Каталог после удаления товара

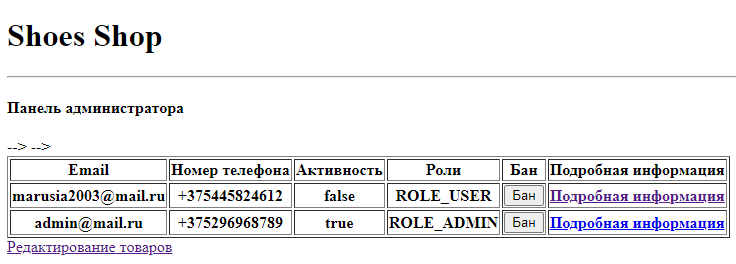
Панель администратора, в которой администратор может заблокировать пользователя, увидеть его электронный адрес, номер телефона, активность, роль, посмотреть подробную информацию, представлена на рисунке 7.14.

Рисунок 7.14 – Панель администратора

Когда администратор нажимает кнопку «Бан», в поле «Активность» активность меняется с true на false. На рисунке 7.15 изображена измененная активность заблокированного пользователя.

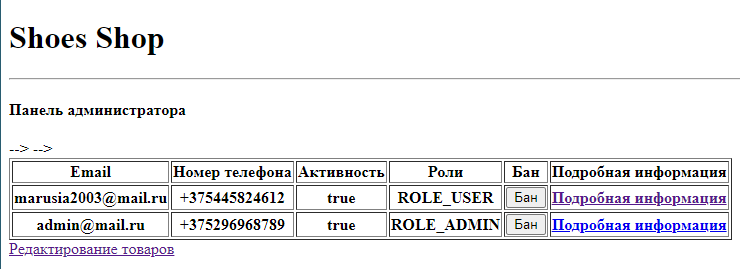


Рисунок 7.15 – Измененная активность заблокированного пользователя

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 7.16.

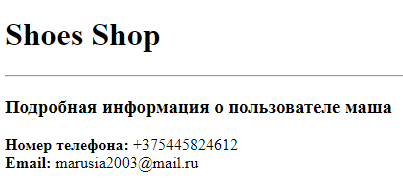


Рисунок 7.16 – Подробная информация о клиенте

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)**

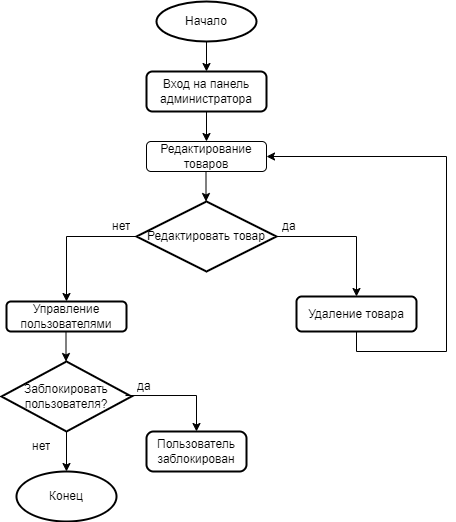
# **Диаграмма классов**



# **ПРИЛОЖЕНИЕ Б**

# **(обязательное)**

# **Схема алгоритма работы программы**



# **ПРИЛОЖЕНИЕ В**

# **(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));

return "admin-products";

}

Продолжение приложения В

@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);

return ResponseEntity.ok()

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

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

Продолжение приложения В

.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

@RequiredArgsConstructor

public class ShoeController {

private final ShoesService shoesService;

private final UserService userService;

private final BucketService bucketService;

private final UserRepository userRepository;

Продолжение приложения В

@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,

@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);

Продолжение приложения В

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(){

return "login";

}

@GetMapping("/registration")

public String registration() {

return "registration";

}

@PostMapping("/registration")

public String createUser(User user, Model model){

if(!userService.createUser(user)){

Продолжение приложения В

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 Г**

# **(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 Г

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,

`user\_id` bigint DEFAULT NULL,

Продолжение приложения Г

PRIMARY KEY (`id`)

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