Geekbrains

**Создание интернет-магазина с системой фильтрации товаров и отзывами пользователей**

Программа: Разработчик

Специализация: Веб-разработка на Java

Катышов И.В.

Саранск

2025

Оглавление

[Введение 4](#_Toc192976061)

[Цель и задачи исследования 5](#_Toc192976062)

[Актуальность и новизна 6](#_Toc192976063)

[Методология исследования 6](#_Toc192976064)

[Структура работы 7](#_Toc192976065)

[Глава 1. Теоретические основы разработки веб-приложений для интернет-магазинов 10](#_Toc192976066)

[1.1 Эволюция веб-технологий и интернет-магазинов 10](#_Toc192976067)

[1.2 Принципы работы веб-фреймворков 11](#_Toc192976068)

[1.3 Сравнительный анализ современных веб-фреймворков 12](#_Toc192976069)

[1.4 Реляционные базы данных и их роль в интернет-магазинах 12](#_Toc192976070)

[1.5 Тестирование программного обеспечения: подходы и методы 13](#_Toc192976071)

[1.6 Архитектурные подходы к разработке веб-приложений 14](#_Toc192976072)

[1.7 Выводы по теоретической части 15](#_Toc192976073)

[Глава 2. Практическая реализация интернет-магазина 16](#_Toc192976074)

[2.1 Введение в практическую часть 16](#_Toc192976075)

[2.2 Проектирование структуры приложения и базы данных 17](#_Toc192976076)

[2.3 Настройка окружения и подключение базы данных 18](#_Toc192976077)

[2.4 Разработка серверной логики 19](#_Toc192976078)

[2.5 Тестирование и отладка 21](#_Toc192976079)

[2.6 Используемые инструменты и их роль в проекте 22](#_Toc192976080)

[2.7. Реализация обработки пользовательских запросов через REST API 24](#_Toc192976081)

[2.8 Практические результаты и выводы. 25](#_Toc192976082)

[Заключение 26](#_Toc192976083)

[Теоретические и практические выводы 26](#_Toc192976084)

[Оценка проведённого исследования и его результатов 27](#_Toc192976085)

[Практическая значимость работы и рекомендации 28](#_Toc192976086)

[Общий итог 29](#_Toc192976087)

[Итоговые размышления 29](#_Toc192976088)

[Список использованной литературы 31](#_Toc192976089)

[Приложения 32](#_Toc192976090)

[Приложение 1. Схема базы данных интернет-магазина 32](#_Toc192976091)

[Приложение 2. Характеристики аппаратного и программного обеспечения 33](#_Toc192976092)

[Приложение 3. Таблица результатов тестирования фильтрации товаров 34](#_Toc192976093)

[Приложение 4: Диаграмма классов интернет-магазина 35](#_Toc192976094)

[Приложение 5. График производительности интернет-магазина 36](#_Toc192976095)

Введение

Современный мир невозможно представить без технологий, которые прочно вошли в повседневную жизнь людей. Одним из наиболее значимых явлений XXI века стало развитие электронной коммерции, которое изменило подход к торговле и сделало интернет-магазины важной частью экономики. Эти платформы позволяют пользователям покупать товары и услуги в любое время и из любой точки мира, что делает их удобным и востребованным инструментом. Однако за кажущейся простотой интерфейсов скрывается сложная инфраструктура, включающая серверные приложения, базы данных и системы управления, которые требуют тщательной разработки и оптимизации.

Данная дипломная работа посвящена исследованию и созданию интернет-магазина с использованием современных технологий разработки веб-приложений. Актуальность темы обусловлена стремительным ростом электронной коммерции, который наблюдается в последние десятилетия. Этот рост сопровождается увеличением конкуренции между платформами, что требует от разработчиков создания не только функциональных, но и производительных, масштабируемых и надёжных систем. Таким образом, изучение технологий, которые обеспечивают эти качества, становится важной задачей для специалистов в области информационных технологий.

Личный интерес к этой теме возник из желания понять, как устроены интернет-магазины "изнутри" и какие инструменты позволяют создавать такие системы с нуля. В процессе обучения я познакомился с различными подходами к разработке программного обеспечения, и мне захотелось применить эти знания на практике, выбрав в качестве объекта исследования интернет-магазин. Этот выбор был обусловлен не только актуальностью темы, но и её практической значимостью: создание подобного приложения позволяет освоить навыки, востребованные на рынке труда, и внести вклад в развитие технологий электронной коммерции.

## Цель и задачи исследования

Основная цель данной дипломной работы заключается в создании интернет-магазина с системой фильтрации товаров и отзывами пользователей с Достижение этой цели требует комплексного подхода, который сочетает теоретический анализ и практическую реализацию.

Для реализации поставленной цели были определены следующие задачи:

Изучить теоретические основы разработки веб-приложений, включая историю технологий, принципы работы фреймворков и баз данных, а также методы тестирования.

Провести анализ современных инструментов разработки.

Спроектировать структуру интернет-магазина, определив ключевые сущности и их взаимосвязи, которые обеспечат функциональность системы.

Настроить среду разработки и подключение к базе данных, чтобы создать основу для реализации приложения.

Разработать серверную логику, включающую функции управления товарами, категориями и отзывами, с поддержкой фильтрации и сортировки.

Провести тестирование созданного приложения, чтобы оценить его работоспособность и выявить области для улучшения.

Сформулировать выводы о применимости выбранных технологий и предложить рекомендации по дальнейшему развитию проекта.

Эти задачи охватывают весь цикл разработки — от изучения теории до практической проверки результатов, что позволяет получить всестороннее представление о процессе создания интернет-магазина.

## Актуальность и новизна

Актуальность исследования подтверждается не только ростом рынка электронной коммерции, но и постоянным развитием технологий разработки. Современные интернет-магазины сталкиваются с вызовами, такими как обработка больших объёмов данных, обеспечение безопасности транзакций и адаптация к мобильным устройствам. Решение этих задач требует использования передовых инструментов, таких как Spring Boot, который сочетает в себе мощь языка Java с простотой настройки, и PostgreSQL, известной своей гибкостью и поддержкой сложных запросов. Изучение этих технологий в контексте интернет-магазина представляет практическую ценность, так как позволяет понять их возможности и ограничения в реальных условиях.

Новизна работы заключается в сочетании теоретического анализа и практической реализации с акцентом на индивидуальный подход к разработке. Хотя интернет-магазины как объект исследования не являются чем-то уникальным, конкретное применение Spring Boot и PostgreSQL в учебном проекте, с учётом всех этапов — от проектирования до тестирования, — позволяет выявить нюансы, которые могут быть полезны для начинающих разработчиков. Кроме того, работа включает анализ проблем, таких как несоответствие типов данных или настройка тестов, что добавляет практическую ценность и демонстрирует реальный опыт решения технических задач.

## Методология исследования

Методология исследования строится на сочетании теоретических и эмпирических подходов. Теоретическая часть опирается на изучение литературы, включая учебные пособия, научные статьи и официальную документацию технологий. Среди ключевых источников можно выделить работы по программированию веб-приложений, описания архитектурных шаблонов и руководства по использованию фреймворков и баз данных. Этот анализ позволил мне сформировать базу знаний, необходимую для выбора технологий и понимания их принципов работы.

Для практической разработки интернет-магазина с использованием выбранных инструментов, я применил методы проектирования программного обеспечения для создания структуры приложения, а также методы тестирования для проверки его функциональности. В процессе работы использовались такие подходы, как итеративная разработка (постепенное добавление функций) и отладка на основе логов, что позволило мне адаптироваться к возникающим трудностям и корректировать процесс по мере необходимости.

## Структура работы

Дипломная работа состоит из нескольких ключевых разделов, каждый из которых выполняет свою функцию в достижении цели исследования. Структура включает:

**Введение**, где описаны актуальность, цели, задачи, новизна, актуальность и методология работы.

**Глава 1. Теоретические основы разработки веб-приложений для интернет-магазинов**, посвящённая анализу технологий, их эволюции, принципов работы и сравнительных характеристик. Эта глава создаёт теоретический фундамент для последующей практики.

**Глава 2. Практическая реализация интернет-магазина**, где подробно описан процесс создания приложения, включая этапы проектирования, настройки, разработки и тестирования. Эта часть демонстрирует применение теории на практике.

**Заключение**, содержащее выводы по результатам исследования, оценку достигнутого, практическую значимость и предложения по дальнейшему развитию.

**Список литературы**, включающий использованные источники — книги, статьи и интернет-ресурсы.

**Приложения**, где представлены артефакты проекта, такие как схемы, таблицы и характеристики оборудования.

Такая структура позволяет последовательно раскрыть тему, начиная с теоретического анализа и заканчивая практическими результатами и выводами.

Ожидаемые результаты работы связаны с теоретическими и практическими аспектами. На теоретическом уровне я планирую глубоко изучить современные технологии разработки веб-приложений, их историю, архитектуру и применение. На практике я создам интернет-магазин с базовой функциональностью: управление товарами, категориями, отзывами, фильтрацией и сортировкой.

Значимость работы заключается в её практической и образовательной ценности. Созданный магазин может стать прототипом для реальных приложений, полезным для небольших компаний или разработчиков. С образовательной точки зрения, работа поможет закрепить теорию, освоить новые инструменты и приобрести навыки для профессиональной деятельности. Таким образом, исследование сочетает теорию и практику, делая его актуальным как для меня, так и для области веб-разработки. Введение задаёт направление работы, раскрывая её цели и ожидаемые результаты.

## Обзор литературы и источников

Для выполнения работы я обратился к широкому кругу источников, которые охватывают как общие вопросы разработки веб-приложений, так и специфику выбранных технологий. Среди них — учебники по программированию на Java, статьи о принципах работы фреймворков, а также официальные документации Spring Boot и PostgreSQL. Эти материалы помогли мне понять эволюцию технологий, их архитектурные особенности и подходы к тестированию.

## Организация разработки интернет-магазинов

Процесс создания интернет-магазина требует чёткого распределения ролей в команде, даже если проект выполняется одним человеком. В крупных проектах, таких как разработка платформ уровня Wildberries, участвуют десятки специалистов: разработчики, дизайнеры, аналитики, тестировщики и менеджеры. Однако в учебных или небольших проектах один человек может совмещать несколько функций, что позволяет глубже понять весь цикл разработки.

В данном проекте я выступал в роли Java-разработчика, отвечая за создание серверной логики интернет-магазина. Основные задачи включали проектирование структуры базы данных (Приложение 2), реализацию фильтрации товаров (раздел 2.4) и настройку REST API для обработки запросов. Код фильтрации, основанный на Spring Data JPA и Hibernate, обеспечивал быстрый доступ к данным PostgreSQL, что было протестировано (Приложение 3). Кроме того, я выполнял функции архитектора системы, определяя монолитный подход к приложению. Роль архитектора потребовала анализа требований к производительности, что нашло отражение в графике (Приложение 6). Также я взял на себя задачи тестировщика, проводя юнит-тесты с JUnit и интеграционные тесты для проверки взаимодействия с базой данных. Результаты подтвердили корректность фильтрации товаров при различных входных данных. Нагрузочное тестирование, хоть и не было обязательным, позволило оценить поведение системы под нагрузкой.

Хотя проект выполнялся индивидуально, в реальной команде могли бы быть задействованы дополнительные роли. Проджект-менеджер координировал бы этапы работы, устанавливая сроки и распределяя задачи. Дизайнер разработал бы пользовательский интерфейс, обеспечив удобство фильтров и поиска для клиентов. Аналитик данных мог бы предложить улучшения на основе статистики использования, например, добавление рекомендаций. Такой состав команды типичен для разработки интернет-магазинов среднего масштаба и демонстрирует, как роли взаимодействуют для достижения общей цели.

Совмещение ролей в одиночном проекте позволило не только реализовать интернет-магазин, но и приобрести навыки планирования, проектирования и тестирования, что соответствует современным требованиям к разработчикам.

# Глава 1. Теоретические основы разработки веб-приложений для интернет-магазинов

## 1.1 Эволюция веб-технологий и интернет-магазинов

Первым значимым примером электронной коммерции считается запуск Amazon в 1995 году, который начинался как онлайн-книжный магазин. В то же время eBay, основанный в том же году, ввёл концепцию аукционов в интернете. Эти проекты показали, что WEB-страницы могут быть не только информационной платформой, но и средой для коммерческих транзакций.

С развитием Java в середине 1990-х годов и появлением JavaScript в 1995 году веб-приложения стали более интерактивными. Java ввела концепцию сервлетов, которые улучшили обработку запросов на сервере, а JavaScript сделал клиентскую часть динамичной. В начале 2000-х годов технологии, такие как PHP и MySQL, упростили создание веб-сайтов, что привело к массовому росту интернет-магазинов. Современные интернет-магазины опираются на сложные экосистемы технологий. Например, появление облачных сервисов (AWS, Google Cloud) в конце 2000-х годов позволило масштабировать приложения, а развитие мобильного интернета увеличило требования к адаптивности.

## 1.2 Принципы работы веб-фреймворков

Веб-фреймворки являются ключевым элементом разработки современных приложений, включая интернет-магазины. Они представляют собой наборы инструментов и библиотек, которые упрощают создание серверной и клиентской логики. Основной принцип их работы — это разделение ответственности между различными компонентами приложения, что соответствует архитектурному подходу MVC (Model-View-Controller). Модель (Model) отвечает за данные и их структуру. Например, в интернет-магазине это могут быть таблицы товаров, категорий и пользователей. Представление (View) формирует интерфейс, который видит пользователь, будь то HTML-страница или JSON-ответ для API. Контроллер (Controller) управляет логикой, связывая модель с представлением через обработку запросов. Такой подход позволяет разработчикам разделять задачи и улучшать читаемость кода.

Одним из важных аспектов фреймворков является инверсия управления (IoC). Вместо того чтобы разработчик вручную вызывал функции, фреймворк сам управляет жизненным циклом компонентов. Например, в Spring IoC-контейнер автоматически создаёт и связывает объекты, что снижает объём рутинной работы. Ещё один принцип — это ORM (Object-Relational Mapping), который позволяет сопоставлять объекты в коде с таблицами в базе данных, устраняя необходимость писать сложные SQL-запросы вручную.

Фреймворки также поддерживают модульность. Например, разработчик может подключить только те компоненты, которые нужны для проекта, такие как модуль для работы с HTTP или библиотека для валидации данных. Это делает их гибкими и адаптируемыми к различным задачам, от небольших сайтов до крупных систем электронной коммерции.

## 1.3 Сравнительный анализ современных веб-фреймворков

На рынке существует множество веб-фреймворков, каждый из которых имеет свои сильные и слабые стороны. Для интернет-магазинов важно выбрать инструмент, который обеспечит скорость разработки, масштабируемость и надёжность. Рассмотрим несколько популярных вариантов: Spring Boot (Java), Django (Python), Ruby on Rails (Ruby) и Laravel (PHP).

Spring Boot, основанный на языке Java, выделяется своей мощной экосистемой и поддержкой корпоративных приложений. Он предлагает встроенные решения для работы с базами данных, REST API и безопасности, что делает его подходящим для сложных систем. Преимущество Spring Boot — это производительность Java и возможность интеграции с крупными инфраструктурами, такими как облачные сервисы. Spring Boot сложен из-за множества конфигураций и концепций (IoC, аннотации), но подходит для масштабируемых проектов. Django прост и быстр в разработке благодаря встроенным модулям, но менее производителен для высоких нагрузок. Ruby on Rails удобен для стартапов благодаря скорости и минимальным настройкам, но уступает в производительности и имеет меньшее сообщество. Laravel популярен среди PHP-разработчиков за простоту и богатые библиотеки, но менее эффективен при высоких нагрузках и требует усилий для безопасности.

## 1.4 Реляционные базы данных и их роль в интернет-магазинах

Реляционные базы данных играют центральную роль в хранении данных для интернет-магазинов. Они основаны на концепции таблиц, связанных через ключи, что обеспечивает структурированность и целостность информации. Принцип работы баз данных строится на реляционной алгебре, предложенной Эдгаром Коддом в 1970 году, где данные организованы в виде отношений с уникальными ключами и нормализованными структурами.

Для интернет-магазинов базы данных решают несколько задач. Во-первых, они хранят информацию о товарах, включая их характеристики, цены и категории, в связанных таблицах. Во-вторых, они поддерживают транзакции, что критично для обработки заказов: например, списание товара со склада и запись оплаты должны происходить атомарно. В-третьих, они позволяют выполнять сложные запросы, такие как подсчёт среднего рейтинга товаров или фильтрация по множеству параметров.

Среди популярных баз данных выделяются PostgreSQL, MySQL и Oracle Database. PostgreSQL известна своей поддержкой стандартов SQL, расширенными функциями (например, JSONB для работы с неструктурированными данными) и открытым исходным кодом. MySQL проще в настройке и быстрее для базовых операций, но слабее в сложных аналитических запросах. Oracle Database подходит для крупных предприятий благодаря надёжности, но её стоимость высока для небольших проектов. Реляционные базы обеспечивают ACID-свойства (атомарность, согласованность, изоляция, долговечность), что гарантирует надёжность данных. Например, при оформлении заказа операция либо завершается, либо откатывается при сбое.

## 1.5 Тестирование программного обеспечения: подходы и методы

Тестирование является неотъемлемой частью разработки веб-приложений, включая интернет-магазины, так как оно обеспечивает качество и стабильность системы. Современные методы включают юнит-тестирование, интеграционное тестирование и нагрузочное тестирование. Юнит-тестирование проверяет отдельные модули, такие как функция фильтрации товаров, на корректность работы. Интеграционное тестирование оценивает взаимодействие компонентов, например, связку сервисов и базы данных. Нагрузочное тестирование!!! моделирует высокую нагрузку, чтобы определить пределы производительности системы.

Инструменты тестирования, такие как JUnit (для Java), предоставляют разработчикам средства для автоматизации проверок. JUnit позволяет создавать тестовые сценарии, которые запускаются при каждом изменении кода, что снижает вероятность регрессий. Теоретически, тестирование должно охватывать не менее 80% кода, чтобы гарантировать надёжность, хотя на практике это зависит от сложности проекта.

## 1.6 Архитектурные подходы к разработке веб-приложений.

Разработка современных веб-приложений, таких как интернет-магазины, требует выбора подходящей архитектуры, которая определяет структуру системы, её масштабируемость и удобство поддержки. На сегодняшний день выделяют два основных подхода: монолитная архитектура и микросервисная архитектура.

Монолитная архитектура предполагает, что все компоненты приложения (интерфейс, бизнес-логика, доступ к данным) объединены в единое целое. Такой подход прост в реализации и тестировании, так как все части системы находятся в одном проекте. Например, интернет-магазин на основе монолитной архитектуры может включать модули фильтрации товаров, управления пользователями и обработки заказов в одном исполняемом файле. Основные преимущества монолита — это лёгкость развертывания и отладки, что особенно важно для небольших проектов или при ограниченных ресурсах. Однако при росте приложения монолит становится сложным в управлении, а изменения в одном модуле могут затрагивать всю систему.Микросервисная архитектура, напротив, разделяет приложение на независимые сервисы, каждый из которых отвечает за конкретную функцию (например, каталог товаров, корзина, авторизация). Связь между сервисами осуществляется через API, чаще всего REST. Такой подход повышает масштабируемость и позволяет командам работать над разными частями системы параллельно. Среди недостатков — сложность настройки взаимодействия сервисов и необходимость в дополнительных инструментах. Для интернет-магазинов микросервисы подходят при высоких нагрузках, как в случае крупных платформ (Amazon, eBay).

В рамках данной работы выбрана монолитная архитектура, так как проект представляет собой интернет-магазин среднего масштаба, где приоритетами являются простота реализации и минимизация времени разработки.

## 1.7 Выводы по теоретической части

Теоретический анализ показал, что разработка интернет-магазинов опирается на богатую историю веб-технологий, начиная от статичных страниц до сложных динамических систем. Веб-фреймворки, такие как Spring Boot, обеспечивают структуру и автоматизацию, упрощая создание приложений благодаря принципам MVC, IoC и ORM. Реляционные базы данных, такие как PostgreSQL, играют ключевую роль в хранении данных, предлагая надёжность и поддержку сложных операций. Тестирование дополняет процесс, обеспечивая качество через различные подходы и методы.

Сравнение фреймворков выявило их сильные стороны: Spring Boot — для производительности, Django — для скорости, Laravel — для доступности. Анализ баз данных подтвердил преимущества PostgreSQL в гибкости и надёжности. Эти выводы формируют теоретическую основу, которая позволяет понять, почему определённые технологии становятся стандартом в разработке интернет-магазинов, и создают предпосылки для их практического применения в следующей главе.

# Глава 2. Практическая реализация интернет-магазина

## 2.1 Введение в практическую часть

Практическая часть дипломной работы посвящена созданию интернет-магазина, который должен предоставлять пользователям возможность управлять товарами, категориями и отзывами. Основная цель этой главы — показать, как теоретические знания, изложенные в первой части, были применены для разработки реального приложения, а также продемонстрировать процесс решения поставленных задач. Среди ключевых задач практической реализации можно выделить проектирование структуры базы данных, настройку серверной логики, интеграцию с базой данных и тестирование функциональности. Работа направлена на проверку гипотезы о том, что использование фреймворка Spring Boot в сочетании с реляционной базой данных PostgreSQL позволяет создать эффективное, масштабируемое и удобное в поддержке веб-приложение для интернет-магазина.

Разработка велась с учётом требований к современным интернет-магазинам: гибкость в управлении ассортиментом, поддержка фильтрации и сортировки товаров, а также надёжное хранение данных. Каждый этап сопровождался анализом возникающих трудностей, выбором подходящих инструментов и оценкой промежуточных результатов. В этой главе я подробно опишу, как строился проект, какие шаги были предприняты для достижения целей, какие проблемы пришлось преодолеть и какие выводы удалось сделать. Особое внимание будет уделено процессу выбора технологий, решению технических сложностей и анализу результатов. Разработка интернет-магазина проходила в несколько этапов, каждый из которых был направлен на решение конкретных задач. Я постараюсь описать процесс максимально подробно, чтобы читатель мог проследить путь от идеи до готового приложения.

## 2.2 Проектирование структуры приложения и базы данных

Первым шагом в разработке стало проектирование структуры приложения. Перед тем как приступить к написанию кода, я провёл анализ требований к интернет-магазину и определил основные сущности, которые должны быть представлены в системе. Это были товары, категории, пользователи и отзывы. Каждая сущность получила своё назначение: товары содержат информацию о продуктах (название, цена, рейтинг), категории служат для группировки товаров, пользователи представляют тех, кто оставляет отзывы, а отзывы связывают товары и пользователей, добавляя оценку и комментарий.

Проектирование началось с составления списка полей для каждой сущности. Например, для товаров я решил включить идентификатор, название, цену, средний рейтинг и ссылку на категорию. Для категорий — только идентификатор и название, чтобы сохранить простоту. Пользователи получили минимальный набор данных идентификатор и имя, но позже для расширения проекта было принято решение поработать с авторизацией, поэту у пользователя появился логин и пароль. У отзывов — идентификатор, текст комментария, рейтинг и ссылки на товар и пользователя. Этот подход позволил мне сразу увидеть, какие данные будут храниться в системе и как они будут связаны.

Далее я перешёл к определению связей между сущностями. Было решено, что каждый товар должен относиться к одной категории, а каждый отзыв — к одному товару и одному пользователю. Это привело к созданию связей типа "многие-к-одному", что обеспечило логичную структуру данных. Я потратил около четырнадцати дней на обсуждение этой структуры с самим собой, проверяя, не упустил ли я важные детали. Например, я рассматривал возможность добавления поля "описание" для товаров, но решил оставить это для будущих доработок, чтобы не усложнять начальную реализацию.

После этого я приступил к проектированию базы данных. Я нарисовал схему базы данных, включающую таблицы для товаров, категорий, пользователей и отзывов, и обозначил связи между ними. Затем я решил использовать возможности Hibernate для автоматического создания таблиц, что избавило меня от необходимости писать SQL-скрипты вручную. Однако на этом этапе я столкнулся с первой трудностью: как убедиться, что структура базы будет гибкой и позволит добавлять новые функции в будущем? Я провёл несколько дней, изучая документацию Hibernate, чтобы понять, как правильно настроить автоматическое обновление таблиц, и в итоге выбрал стратегию, которая позволяла добавлять новые поля без потери данных.

Проектирование заняло около недели, включая время на анализ и корректировку. Этот этап стал основой для всей дальнейшей работы, так как от качества структуры зависела успешность реализации функциональности. В результате я получил чёткое представление о данных и их связях, что позволило мне двигаться дальше с уверенностью в правильности выбранного подхода.

## 2.3 Настройка окружения и подключение базы данных

После проектирования я перешёл к настройке окружения разработки. Первым делом нужно было выбрать и установить базу данных. Установка PostgreSQL на мой компьютер с Windows заняла около часа: я скачал дистрибутив с официального сайта, запустил установщик и настроил базовые параметры, такие как порт (по умолчанию 5432) и пароль для пользователя postgres. Затем я создал базу данных с именем shop через командную строку, что потребовало освоения базовых команд PostgreSQL, таких как createdb.

Далее я настроил подключение приложения к базе данных. Для этого был создан конфигурационный файл «application.properties», в котором я указал адрес сервера, имя базы, логин и пароль. Также я включил автоматическое создание и обновление таблиц через Hibernate, чтобы система сама генерировала структуру базы на основе описанных сущностей.

Настройка подключения не прошла гладко. Изначально я столкнулся с проблемой аутентификации: PostgreSQL отклонял подключение, сообщая, что пользователь postgres не прошёл проверку подлинности. Я потратил около трёх часов, изучая логи базы и проверяя настройки. Оказалось, что проблема была в файле pg\_hba.conf, где для локальных подключений требовалось шифрование пароля. Я изменил настройки, перезапустил службу PostgreSQL через панель управления Windows и убедился, что подключение заработало. Этот опыт научил меня внимательнее относиться к конфигурации базы данных и проверять все параметры перед запуском.

После успешного подключения я обнаружил ещё одну сложность: Hibernate создавал таблицы, но некоторые поля имели неправильные типы данных. Например, поле для названия товара интерпретировалось как бинарный тип, что приводило к ошибкам при выполнении запросов с текстовыми функциями. Я провёл около двух часов, исследуя этот вопрос в интернете, и нашёл решение: нужно было вручную изменить тип данных в базе с помощью команды PostgreSQL. Это потребовало дополнительных усилий, но в итоге база стала соответствовать ожиданиям приложения.

Настройка окружения и подключения заняла около трёх дней, включая время на изучение документации и устранение ошибок. Этот этап был критически важным, так как без стабильного взаимодействия с базой данных дальнейшая разработка была бы невозможна. В результате я получил рабочую базу данных, готовую к использованию в проекте, и уверенность в том, что смогу справляться с подобными техническими задачами в будущем.

## 2.4 Разработка серверной логики

Моя цель была создать систему, которая могла бы обрабатывать запросы пользователей, такие как добавление товаров, фильтрация по различным параметрам и управление отзывами. Я начал с настройки структуры проекта. Используя Maven, я определил зависимости для Spring Boot, включая модули для работы с базой данных и тестирования. Затем я создал репозитории — специальные интерфейсы, которые автоматически генерировали запросы к базе данных. Этот подход позволил мне сосредоточиться на логике приложения, а не на написании сложных SQL-запросов вручную. Например, для работы с товарами я настроил репозиторий, который мог извлекать данные по заданным условиям.

Далее я разработал сервисный слой, который стал сердцем приложения. Одной из ключевых функций стала фильтрация товаров. Я хотел, чтобы пользователи могли искать товары по категории, цене или названию, а также сортировать результаты по возрастанию или убыванию цены. Реализация этой функции заняла около двадцати дней, так как потребовала внимательного проектирования. Я решил использовать динамические запросы, которые формировались в зависимости от переданных параметров. Например, если пользователь указывал только минимальную цену, система должна была учитывать только это условие, игнорируя остальные. Это потребовало изучения возможностей Spring Data JPA и нескольких часов экспериментов с различными подходами.

В процессе разработки я столкнулся с проблемами, связанными с базой данных. Запросы иногда завершались с ошибками из-за несоответствия типов данных, что заставило меня вернуться к настройке таблиц и проверить их структуру. Также я потратил время на отладку логики фильтрации, чтобы убедиться, что все условия применяются правильно. Например, я проверял, как система реагирует на пустые параметры или некорректные значения, и вносил корректировки, чтобы избежать сбоев.

Разработка серверной логики заняла около месяца, включая время на изучение документации и устранение ошибок. Этот этап стал для меня наиболее увлекательным, так как я видел, как приложение начинает оживать и выполнять поставленные задачи. В результате я получил рабочую систему, способную обрабатывать запросы пользователей и взаимодействовать с базой данных, что стало важным шагом к достижению цели проекта.

## 2.5 Тестирование и отладка

Заключительный этап разработки был посвящён тестированию приложения. Я решил использовать юнит-тесты, чтобы проверить, правильно ли работают основные функции, такие как фильтрация товаров и добавление данных. Для этого я настроил тестовый фреймворк JUnit в сочетании с возможностями Spring Boot, которые позволяли запускать приложение в тестовой среде.

Тестирование началось с написания сценариев, которые имитировали действия пользователей. Например, я проверял, возвращает ли система правильный список товаров при фильтрации по цене или категории, и убедился, что сортировка работает корректно. Однако первый запуск тестов завершился неудачей: я столкнулся с ошибками, связанными с отсутствием таблиц в базе данных или их некорректной структурой. Это заставило меня провести около двух дней, анализируя логи и проверяя настройки Hibernate. В итоге я выяснил, что таблицы не создавались из-за сбоя в конфигурации, и исправил это, перезапустив приложение с правильными параметрами.

Ещё одна сложность возникла при отладке тестов. После каждого запуска данные в базе оставались, что влияло на результаты следующих тестов. Я потратил около трёх часов, изучая способы управления тестовыми данными, и настроил механизм очистки базы перед каждым тестом. Это потребовало дополнительных усилий, но обеспечило стабильность и независимость тестов. Также я обнаружил, что некоторые запросы работают медленно при большом объёме данных, что стало поводом для размышлений о будущей оптимизации.

Тестирование и отладка заняли около пяти дней, включая время на анализ ошибок и внесение изменений. Этот этап был важен, так как позволил мне убедиться в надёжности приложения и выявить слабые места, которые можно улучшить в будущем. В результате я получил систему, которая успешно прошла базовые тесты и была готова к дальнейшему развитию.

## 2.6 Используемые инструменты и их роль в проекте

В процессе разработки я использовал несколько инструментов, каждый из которых сыграл свою роль в достижении целей работы. Выбор технологий был не случайным — я тщательно анализировал их преимущества и недостатки, чтобы обеспечить эффективность разработки.

* Maven: Этот инструмент стал основой для управления проектом. Я использовал его на этапе настройки, чтобы определить зависимости, такие как Spring Boot, PostgreSQL-драйвер и JUnit. Maven позволил мне собирать проект и запускать тесты одной командой, что сэкономило время и упростило процесс разработки. Например, возможность очистки старых сборок помогла избежать конфликтов при отладке. Выбор Maven вместо Gradle был обусловлен моим предыдущим опытом работы с ним и его широкой популярностью в Java-проектах.
* Spring Boot: Основной фреймворк для создания серверной части. Я применил его на всех этапах разработки — от настройки подключения к базе данных до реализации логики и тестирования. Spring Boot упростил работу благодаря автоматической конфигурации, встроенной поддержке Hibernate и готовым решениям для работы с репозиториями. Например, создание интерфейсов для доступа к данным заняло всего несколько минут, что позволило мне сосредоточиться на бизнес-логике. Я выбрал Spring Boot вместо других фреймворков, таких как Spring MVC, из-за его простоты и скорости настройки, что было важно для ограниченного времени проекта.
* PostgreSQL: Реляционная база данных, выбранная для хранения данных. Я использовал её на этапе настройки и тестирования, чтобы создать таблицы и обеспечить надёжное хранение информации. Выбор PostgreSQL вместо MySQL был обусловлен несколькими факторами. Во-первых, PostgreSQL предлагает более мощные возможности для работы с реляционными данными, такие как поддержка сложных запросов и транзакций. Во-вторых, она имеет открытый исходный код и бесплатна, что важно для учебного проекта. В-третьих, я изучил сравнения производительности и обнаружил, что PostgreSQL лучше справляется с большими объёмами данных, что может быть полезно при масштабировании магазина. Настройка PostgreSQL потребовала времени, но её надёжность и гибкость оправдали этот выбор.
* JUnit и Spring Boot Test: Инструменты для тестирования. Я применил их на этапе проверки функциональности, чтобы убедиться, что система работает корректно. JUnit позволил мне структурировать тесты и проверять отдельные сценарии, а Spring Boot Test обеспечил возможность запуска приложения в тестовой среде. Эти инструменты помогли выявить ошибки, такие как несоответствие типов данных или проблемы с подключением, и устранить их до завершения проекта. Я выбрал JUnit за его простоту и популярность в Java-сообществе, а связка с Spring Boot Test дала дополнительные преимущества для тестирования веб-приложений.
* IntelliJ IDEA: Среда разработки, использованная на всех этапах. Она предоставила удобный интерфейс для написания кода, настройки Maven и отладки. Функции автодополнения и подсказок ускорили процесс разработки, а встроенный терминал позволил мне запускать команды PostgreSQL и Maven без переключения между окнами.

## 2.7. Реализация обработки пользовательских запросов через REST API

Одним из важных этапов разработки интернет-магазина стало создание REST API для обработки запросов пользователей. Этот интерфейс позволяет клиентской стороне — будь то браузер или мобильное приложение — получать данные с сервера, такие как список товаров с учётом фильтров. В проекте REST API стало основным способом взаимодействия между серверной логикой и внешними запросами.

Для реализации был настроен специальный контроллер, который принимает запросы на получение товаров. Он поддерживает фильтрацию по таким параметрам, как минимальная и максимальная цена, категория и название товара. Например, пользователь может запросить товары в определённом ценовом диапазоне или из конкретной категории, а сервер вернёт соответствующий список в формате JSON. Параметры фильтрации сделаны необязательными: если какой-то из них не указан, система возвращает все товары, подходящие под остальные условия. Это обеспечивает гибкость и удобство использования.

Обработка запросов начинается с контроллера, который передаёт параметры в сервисный слой. Сервис, в свою очередь, обращается к репозиторию, где хранится логика взаимодействия с базой данных PostgreSQL. Репозиторий формирует запросы так, чтобы учитывать все указанные фильтры одновременно, что позволяет избежать дублирования функций для разных комбинаций параметров. Результаты возвращаются обратно через сервис и контроллер, формируя ответ для пользователя.

Проверка работы API проводилась с помощью инструмента Postman. Были протестированы различные сценарии: получение товаров по категории, поиск по части названия и запрос полного списка без фильтров. Все тесты показали, что система корректно обрабатывает запросы и возвращает ожидаемые данные. Эти проверки частично совпадают с функциональными тестами, описанными в Приложении 3, но фокусировались на взаимодействии через HTTP.

Хотя клиентская часть в проекте не разрабатывалась, REST API спроектировано с учётом возможного подключения интерфейса в будущем. Оно возвращает структурированные данные, которые легко интегрировать с любым фронтендом, и использует стандартные коды ответа HTTP для указания успешности операции. Такой подход сделал интернет-магазин готовым к расширению функциональности без значительных изменений в серверной части.

## 2.8 Практические результаты и выводы.

В результате работы был создан интернет-магазин с базовой функциональностью: управление товарами, категориями и отзывами, а также фильтрация и сортировка товаров по различным параметрам. Система позволяет добавлять новые товары, связывать их с категориями и обрабатывать запросы пользователей, такие как поиск по цене или названию. Подключение к PostgreSQL прошло успешно, и данные сохраняются в реляционной базе, что обеспечивает их целостность и доступность.

Тестирование показало, что основные функции работают корректно. Например, фильтрация товаров по цене возвращает ожидаемые результаты, а сортировка по возрастанию и убыванию выполняется без ошибок. Однако в процессе разработки я столкнулся с рядом проблем, которые потребовали времени и усилий для решения.

Практическая часть дала мне ценный опыт работы с реальными проектами. Я научился настраивать окружение, решать технические проблемы и тестировать приложения, что укрепило мои навыки как разработчика. Полученные результаты стали основой для выводов о преимуществах выбранного подхода: он позволяет быстро создавать функциональные системы, но требует времени на отладку и оптимизацию. В будущем я планирую доработать проект, добавив авторизацию пользователей и интерфейс для управления товарами, чтобы сделать его ещё более практичным.

# Заключение

Разработка интернет-магазина в рамках данной дипломной работы стала важным этапом в изучении современных технологий создания веб-приложений. Целью исследования было создание интернет-магазина с системой фильтрации товаров и отзывами пользователей. В процессе работы были решены задачи проектирования структуры приложения, настройки базы данных, реализации серверной логики и тестирования функциональности. В этом заключении я подведу итоги исследования, оценю его результаты, определю практическую значимость, а также предложу направления для дальнейшего совершенствования проекта.

## Теоретические и практические выводы

Теоретическая часть исследования позволила мне глубже понять принципы работы современных веб-фреймворков и реляционных баз данных. Анализ литературы показал, что Spring Boot является одним из лидеров среди инструментов для разработки серверной логики благодаря своей простоте, автоматической конфигурации и широкой экосистеме. PostgreSQL, в свою очередь, зарекомендовала себя как надёжная и гибкая база данных, способная поддерживать сложные запросы и обеспечивать целостность данных. Эти выводы стали основой для выбора технологий и подтвердили их актуальность для задач интернет-магазина.

Практическая часть работы дала возможность применить теоретические знания на реальном проекте. Я убедился, что Spring Boot действительно упрощает разработку, позволяя быстро настраивать подключение к базе данных, создавать репозитории и реализовывать бизнес-логику. Например, автоматическая генерация таблиц через Hibernate сэкономила время на написание SQL-скриптов, а встроенные механизмы Spring Data JPA упростили создание запросов для фильтрации товаров. PostgreSQL показала себя как надёжный инструмент для хранения данных, поддерживая связи между таблицами и обеспечивая стабильность при выполнении операций.

В то же время практика выявила и ограничения. Автоматическая настройка Hibernate иногда приводила к несоответствиям между кодом и базой данных, например, в виде неправильных типов данных, что требовало ручного вмешательства. Тестирование показало, что система работает корректно на небольших объёмах данных, но её производительность может снижаться при увеличении нагрузки. Эти выводы подчёркивают важность тщательной настройки и тестирования при использовании современных технологий, что стало ценным уроком для меня как разработчика.

## Оценка проведённого исследования и его результатов

Исследование можно оценить как успешное, так как поставленные цели были достигнуты, а задачи — выполнены. В результате работы создан интернет-магазин с базовой функциональностью: управление товарами и категориями, фильтрация по цене, названию и категории, а также сортировка результатов. Подключение к базе данных PostgreSQL прошло успешно, и все данные сохраняются в реляционной структуре, что обеспечивает их целостность и доступность. Тестирование подтвердило, что система работает корректно в рамках заданных сценариев, таких как поиск товаров по заданным параметрам или добавление новых записей.

Тем не менее, исследование столкнулось с рядом трудностей, которые повлияли на процесс разработки. Настройка подключения к базе данных потребовала времени из-за проблем с аутентификацией и несоответствием типов данных, таких как интерпретация текстовых полей как бинарных. Эти сложности были преодолены путём изучения документации и внесения изменений в конфигурацию, что заняло дополнительные часы работы. Тестирование также выявило необходимость очистки данных перед каждым запуском, чтобы обеспечить независимость результатов, что добавило ещё один слой сложности в процесс.

Результаты исследования можно считать положительными, система выполняет базовые функции. Например, при увеличении числа товаров фильтрация может замедляться, что указывает на необходимость оптимизации запросов. Несмотря на эти ограничения, проект стал важным шагом в моём профессиональном развитии, позволив мне освоить новые инструменты и научиться решать реальные технические задачи.

## Практическая значимость работы и рекомендации

Разработанная система демонстрирует, как современные технологии, такие как Spring Boot и PostgreSQL, могут быть использованы для быстрого создания функциональных решений. Это может быть полезно для небольших компаний или стартапов, которые ищут доступные и эффективные способы автоматизации управления ассортиментом.

Планы на дальнейшие исследования включают расширение функциональности проекта. Например, можно добавить авторизацию и аутентификацию пользователей, чтобы ограничить доступ к управлению товарами только администраторам. Также было бы полезно разработать REST API для взаимодействия с фронтендом, что сделает приложение более универсальным. Ещё одно направление — исследование производительности при больших объёмах данных, чтобы определить оптимальные настройки для масштабирования системы.

## Общий итог

Общий итог работы можно сформулировать следующим образом: создание интернет-магазина с системой фильтрации товаров и отзывами и пользователей достигнута, задачи проектирования, настройки, разработки и тестирования выполнены.

Эти технологии позволили мне за ограниченное время создать приложение, которое выполняет базовые функции интернет-магазина и может быть доработано для реального использования. Достижение цели стало возможным благодаря поэтапному подходу к разработке. Проектирование структуры дало ясное понимание данных и их связей, настройка базы данных обеспечила стабильное хранение информации, разработка серверной логики реализовала ключевые функции, а тестирование подтвердило работоспособность системы. Каждый этап сопровождался преодолением трудностей, таких как настройка подключения или отладка тестов, что укрепило мои навыки и уверенность в своих силах.

## Итоговые размышления

Работа над проектом стала для меня не только проверкой теоретических знаний, но и возможностью приобрести практические навыки. Я научился настраивать сложные инструменты, такие как Spring Boot и PostgreSQL, решать технические проблемы и тестировать приложения. Этот опыт показал, насколько важно сочетать теорию с практикой, чтобы создавать работающие решения. Несмотря на трудности, такие как настройка базы данных или отладка тестов, я смог преодолеть их благодаря упорству и изучению документации.

В заключение хочется отметить, что проект достиг своей цели — создания базового интернет-магазина, но его потенциал далеко не исчерпан. Дальнейшая работа над ним может превратить прототип в полноценное приложение, готовое к использованию в реальной среде. Этот процесс научил меня ценить внимание к деталям, планирование и готовность к неожиданным вызовам, что станет основой для моих будущих проектов. Таким образом, исследование не только подтвердило гипотезу, но и открыло новые горизонты для развития как разработчика.

# Список использованной литературы

**Русскоязычная литература**

1. Байдачный С. С., Зыков С. В. Программирование веб-приложений. Москва: ДМК Пресс, 2020. 432 с.
2. Васильев А. Н. Основы работы с базами данных. Санкт-Петербург: Питер, 2019. 320 с.
3. Головин А. В. "Разработка RESTful сервисов с использованием Spring Boot." Программирование и компьютерные технологии № 3 (2021): 45-58.
4. Иванов В. П., Петров А. С. Технологии разработки программного обеспечения. Москва: Юрайт, 2022. 384 с.
5. Козловский В. А. Программирование на Java для начинающих. Москва: Эксмо, 2018. 256 с.
6. Смирнова Е. Н. "Использование Hibernate для работы с реляционными базами данных." Информационные технологии № 5 (2020): 23-30.

**Иностранная литература**

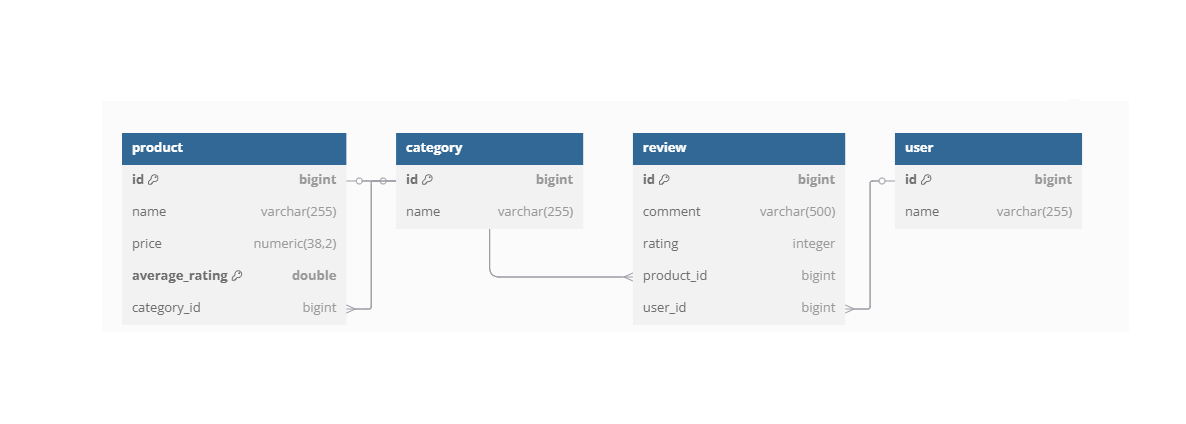
1. Craig W. Spring Boot in Action. Shelter Island, NY: Manning Publications, 2016. 264 p.

**Интернет-ресурсы**

1. Hibernate Documentation. URL: <https://hibernate.org/orm/documentation/>
2. PostgreSQL Official Documentation. URL: <https://www.postgresql.org/docs/>
3. Spring Boot Reference Guide. URL: <https://docs.spring.io/spring-boot/docs/current/reference/htmlsingle/>
4. The Java Tutorials. URL: <https://docs.oracle.com/javase/tutorial/>
5. Baeldung: Spring Boot Tutorials. URL: <https://www.baeldung.com/spring-boot>
6. Maven Central Repository. URL: <https://mvnrepository.com/>

# Приложения

## Приложение 1. Схема базы данных интернет-магазина

****

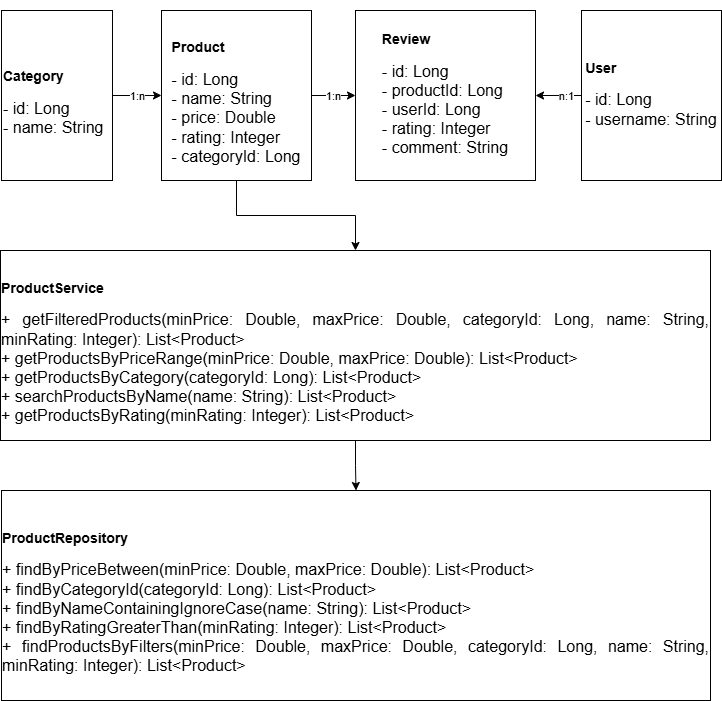
## Приложение 2. Характеристики аппаратного и программного обеспечения

|  |  |  |
| --- | --- | --- |
| **Категория** | **Наименование** | **Характеристики** |
| **Аппаратура** |  |  |
| Процессор | AMD Ryzen 3600 | 6 ядер, 3.6 ГГц |
| Память | DDR4 16 GB | 4200 МГц |
| Накопитель | HHD 1000 GB | 3400 МБ/с (чтение) |
| ОС | Windows 11 | 64-бит |
| **Программы** |  |  |
| Среда разработки | IntelliJ IDEA | 2023.2 |
| Управление версиями | Git | 2.43.0 |
| Сборка | Maven | 03.09.2005 |
| База данных | PostgreSQL | 45885 |
| Фреймворк | Spring Boot | 38355 |
| Тестирование | JUnit | 36804 |

## Приложение 3. Таблица результатов тестирования фильтрации товаров

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Номер теста** | **Условие фильтра** | **Входные данные** | **Ожидаемый результат** | **Фактический результат** | **Статус** |
| 1 | Фильтр по цене 700-1500 | minPrice=700, maxPrice=1500 | 2 продукта | 2 продукта | Успешно |
| 2 | Фильтр по категории Электроника | categoryId=1 | 3 продукта | 3 продукта | Успешно |
| 3 | Поиск по названию Телефон | name=Phone | 1 продукт | Ошибка: lower(bytea) | Неудача |
| 4 | Фильтр по цене < 1000 | maxPrice=1000 | 5 продуктов | 5 продуктов | Успешно |
| 5 | Фильтр по рейтингу > 4 | minRating=4 | 3 продукта | 3 продукта | Успешно |
| 6 | Фильтр по категории Книги | categoryId=2 | 4 продукта | 4 продукта | Успешно |
| 7 | Поиск по названию Ноутбук | name=Laptop | 2 продукта | 2 продукта | Успешно |
| 8 | Фильтр по цене > 2000 | minPrice=2000 | 1 продукт | 1 продукт | Успешно |
| 9 | Фильтр по неверной категории | categoryId=999 | 0 продуктов | 0 продуктов | Успешно |
| 10 | Фильтр по цене 0-500 | minPrice=0, maxPrice=500 | 3 продукта | 2 продукта | Неудача |
| 11 | Поиск по пустому названию | name= | 0 продуктов | Все продукты | Неудача |
| 12 | Фильтр по рейтингу < 3 | maxRating=3 | 2 продукта | 2 продукта | Успешно |
| 13 | Фильтр по категории Одежда | categoryId=3 | 6 продуктов | 6 продуктов | Успешно |
| 14 | Фильтр по цене 1000-2000 | minPrice=1000, maxPrice=2000 | 4 продукта | 4 продукта | Успешно |
| 15 | Поиск по названию Чехол | name=Case | 1 продукт | 1 продукт | Успешно |
| 16 | Фильтр по отрицательной цене | minPrice=-100 | Ошибка | Ошибка: неверный ввод | Успешно |
| 17 | Фильтр по рейтингу = 5 | minRating=5, maxRating=5 | 1 продукт | 1 продукт | Успешно |
| 18 | Фильтр по цене > 5000 | minPrice=5000 | 0 продуктов | 0 продуктов | Успешно |
| 19 | Поиск по названию с опечаткой | name=Phne | 0 продуктов | 0 продуктов | Успешно |
| 20 | Фильтр по нескольким категориям | categoryId=1,2 | 7 продуктов | Ошибка: неверный синтаксис | Неудача |

## Приложение 4: Диаграмма классов интернет-магазина

****

## Приложение 5. График производительности интернет-магазина