Аннотация

Выпускная квалификационная работа (бакалаврская работа) на тему «Разработка маркетплейса для продажи товаров на Java Spring» содержит 69 страницу пояснительной записки и 13 листов графического материала.

МАРКЕТПЛЕЙС, ВЕБ-ПРИЛОЖЕНИЕ, ИНТЕРНЕТ-МАГАЗИН, БАЗА ДАННЫХ, JAVA, SPRING FRAMEWORK, BOOTSTRAP.

Объект работы - веб-приложение, онлайн-магазин, для компании «UBuySell».

Целью работы является переход компании «UBuySell» на собственную платформу по продаже товаров для увеличения прибыли с продаж и облегчения управления товарами внутри сервиса.

Теоретическое исследование проводилось методом анализа литературы и нормативных справочников. Были изучены технические документы используемых в разработке инструментов.

В первой главе был выполнен анализ предметной области и сформулированы требований к разрабатываемой АИС.

Во второй главе была представлена функциональная модель, описаны использованные алгоритмы, спроектирована АИС.

В третьей главе приведен результат работы, описан пользовательский интерфейс разработанной системы. Проведен обзор функциональных возможностей, предоставленных сервисом.

В выпускной квалификационной работе решается задача автоматизации покупки и продажи товаров.

Содержание

[Введение 6](#bookmark4)

1. Анализ предметной области 8
   1. [Бизнес требования 8](#bookmark6)
      1. [Исходные данные 8](#bookmark8)
      2. [Возможности бизнеса 8](#bookmark10)
      3. [Бизнес цели 8](#bookmark12)
      4. [. Критерии успеха 9](#bookmark14)
      5. [Видение решения 9](#bookmark16)
      6. [. Бизнес риски 9](#bookmark18)
      7. [Предположение зависимости 10](#bookmark20)
      8. [Основные функции приложения 10](#bookmark22)
      9. [MVP 11](#bookmark24)
      10. [Стейкхолдеры 11](#bookmark26)
      11. [Приоритеты проекта 12](#bookmark28)
      12. [Варианты использования 12](#bookmark30)
   2. [Разработка модели данных 13](#bookmark37)
      1. Выделение существительных глаголов 13
      2. [Разработка концептуальной карты 14](#bookmark35)
      3. Разработка ER-модели 14
   3. [Моделирование бизнес-процессов 16](#bookmark39)
   4. [Разработка пользовательских требований 19](#bookmark41)
      1. [Диаграмма вариантов использования 19](#bookmark43)
      2. [Спецификации варианта использования 20](#bookmark45)
   5. [Разработка функциональных требований 23](#bookmark47)
   6. [Разработка нефункциональных требований 25](#bookmark49)
      1. [Требования к пользовательским интерфейсам 25](#bookmark51)
      2. [Требования к удобству использования 25](#bookmark53)
      3. [Требования к производительности 26](#bookmark55)
      4. [Требования к безопасности 26](#bookmark57)
      5. [Требования к надежности 26](#bookmark59)
      6. [Бизнес-правила 27](#bookmark61)
   7. [Выводы по главе 27](#bookmark63)
2. Проектирование информационной системы 28
   1. Функциональная модель системы 28
   2. [Разработка алгоритмов системы 29](#bookmark68)
      1. [Алгоритм авторизации 29](#bookmark70)
      2. [Алгоритм регистрации 31](#bookmark72)
      3. [Алгоритм создания товара 32](#bookmark74)
      4. [Алгоритм покупки товара 34](#bookmark76)
      5. [Алгоритм создания отчета 35](#bookmark78)
   3. База данных 37
      1. [Выбор системы управления базой данных 37](#bookmark80)
      2. [Разработка моделей для базы данных 37](#bookmark82)
         1. [Модель User 38](#bookmark84)
         2. [Модель Seller 38](#bookmark86)
         3. [Модель Product 39](#bookmark88)
         4. [Модель Review 40](#bookmark90)
         5. [Модель StatusRequest 40](#bookmark92)
         6. [Модель Cart 41](#bookmark94)
         7. [Модель Purchase 41](#bookmark96)
      3. [UML диаграмма таблиц базы данных 42](#bookmark98)
   4. [Выводы по главе проектирование 43](#bookmark100)
3. [Реализация информационной системы 44](#bookmark102)
   1. Выбор инструментария для разработки 44
      1. Серверная часть 44
      2. [Клиентская часть 46](#bookmark107)
      3. [Проектирования интерфейса пользователя 47](#bookmark109)
   2. [Описание интерфейса 48](#bookmark111)
   3. [Выводы по главе 58](#bookmark115)

[Заключение 59](#bookmark117)

[Список использованных источников 61](#bookmark119)

Приложение А 63

Приложение Б 64

Введение

Бизнес компания «UBuySell» основан на перепродаже товаров. Свой ассортимент она закупает в Китае, являясь посредником между производителем и покупателем. Деятельность компании распространяется на территории Российской Федерации.

Купленный за рубежом товар реализуется на различных онлайн сервисах по продаже товаров.

За предоставление места размещения товара сервисы взымают комиссию за каждый проданный товар, 20% от суммы покупки. Компания не довольна тем, что ее клиенты находятся на разных сервисах. Также для менеджеров компании затруднительно контролировать продажи внутри нескольких сервисов, следовать правилом каждой отдельной платформы. Поэтому руководство желает перейти на свою площадку. Так в целях увеличения прибыли, и повышения узнаваемости компании, она приняла решение, создать свой онлайн-магазин, на котором она сможет реализовывать продукцию и не терять потенциальную прибыль, которая сейчас уходит на выплату процентов с продаж.

Объектом исследования в работе является маркетплейс для продажи и покупки товаров.

Предметом исследования являются процессы регистрации новых пользователей, продавцов внутри сервиса, процесс купли-продажи товаров.

Актуальность работы обусловлена необходимостью перехода на свою онлайн-площадку для продажи товаров

Целью работы является повышение прибыли за счет продажи товаров на собственной площадке, и за счет привлечения сторонних продавцов.

Для достижения поставленной цели требуется решить следующие задачи:

* разработать пользовательские требования в системе;
* разработать функциональные требования к системе;
* разработать нефункциональные требования к системе;
* разработать функциональную модель системы;
* разработать UML диаграммы;
* разработать информационную систему;
* провести анализ реализации;
* спроектировать информационную систему;
* реализовать информационную систему на стороне сервера;
* реализовать информационную систему на стороне клиента.

**1 Анализ предметной области**

* 1. Бизнес требования

Поступил запрос на создание веб-приложения для компании «UBuySell» на котором она смогла бы реализовывать свои товары. Компания желает перейти на собственную площадку и дать возможность другим продавцам также продавать свои товары. В данной главе будут сформированы требования к сервису, описаны основные функции, сценарии использования.

* + 1. Исходные данные

Компания «UBuySell» реализует свои товары на сторонних маркетплейсах. Организация выплачивает процент с каждого проданного товара. Вследствие чего руководство приняло решение разработать свою площадку ведения бизнеса.

* + 1. Возможности бизнеса

Руководство компании желает быть независимой от других подобных платформ для продажи товаров, не платить комиссию за присутствие на этих платформах. Также сама компания сможет впоследствии зарабатывать на комиссии от продажи товаров от других клиентов приложения, в связи с этим было принято решение разработать данное приложение.

* + 1. Бизнес цели

В качестве бизнес-целей были выделены следующие пункты:

* снизить расходы от продажи товаров, перестав платить комиссию сторонним маркетплейсам;
* получение дополнительной прибыли от продажи товаров другими продавцами.
  + 1. . Критерии успеха

Критерия успеха в данной работе будет являться следующее:

* 100% реализация базовой функциональности к 01.07.2023;
* система внедрена в эксплуатацию;
* проведение первых десяти покупок зарегистрированными пользователями.
  + 1. Видение решения

Решение представляет собой веб-приложение на Java Spring на котором пользователи смогут продавать товары. Также решение будет содержать панель администратора, внутри самого приложения. Приложение будет содержать минимальный функционал, позволяющий регистрироваться как покупатель и продавец, выставление товаров, их покупка.

* + 1. . Бизнес риски

На данный момент можно выделить следующие риски, касающиеся процесса разработки и может повлиять на конечный продукт:

* компании может не хватить ресурсов для выполнения бизнес- целей в срок;
* может возникнуть нехватка времени, в случае расширения требований к сервису;
* смена команды разработчиков проекта может увеличить время разработки.
  + 1. Предположение зависимости

Следующие зависимости будут актуальны на протяжении всего процесса разработки:

* разработчики не будут привлекаться для работы на других проектах;
* не первом этапе не будет разработка мобильного приложения;
* руководство компании заинтересовано в разработке программного продукта;
* в рамках первой итерации не планируется реализация оплаты.
  + 1. Основные функции приложения

В качестве основных функций приложения было выделено следующее:

* регистрация/авторизация пользователей;
* регистрация продавца;
* управление витриной товаров;
* создание товаров;
* управление каталогом товаров;
* управление корзиной;
* покупка товаров;
* управление пользователями;
* оценка товаров, формирование рейтинга;
* оповещения по электронной почте;
* получение отчета о продажах для продавца.
  + 1. MVP

Был сформирован список минимального набора функций, без которого нет смысла разрабатывать программный продукт:

* регистрация/авторизация пользователей;
* регистрация продавца;
* создание товаров;
* управление пользователями;
* управление корзиной;
* покупка товаров.
  + 1. Стейкхолдеры

В данной главе будут представлены ключевые участники проекта, ценность программного продукта для них, их отношение к нему, интересы и ограничения.

Пользователи-покупатели

* ценность: возможность покупки товаров;
* отношение: средняя заинтересованность;
* интересы: покупка требующихся товаров по наиболее низкой цене;
* ограничения: отсутствуют.

Пользователи-продавцы

* ценность: возможность продажи товаров;
* отношение: высокая заинтересованность;
* интересы: продажа наибольшего количества товаров;
* ограничения: цены или рейтинг конкурентов могут быть более привлекательными для покупателей.

Руководство компании

* ценность: выставление собственных товаров внутри свое сервиса;
* отношение: высокая заинтересованность;
* интересы: продажа товаров без наценки, также заработок за счет комиссии от продажи у других продавцов;
* ограничения: отсутствуют.
  + 1. Приоритеты проекта

Функции:

* все функции MVP должны быть выполнены.

Качество:

* критическая функциональность должна работать без ошибок.

Срок:

* версия MVP должны быть запущена в тестовую эксплуатацию в границах ожидаемого срока 1 года (12 месяцев).

Персонал:

* предполагаемый персонал: 1 фронт-энд разработчик, 1 бек-энд разработчик, менеджер проекта.
  + 1. Варианты использования

Пользователи-покупатели

* регистрация;
* авторизация;
* управление витриной;
* покупка товаров;
* оценка товаров;
* возможность подать заявку на то, чтобы стать продавцом.

Пользователи-продавцы

* регистрация;
* авторизация;
* выставление товаров на продажу;
* оповещение о изменении состояния товара;
* изменение информации о ранее созданных товаров;
* просмотр отчета о продажах.

Администратор

* авторизация;
* просмотр информации о пользователях;
* блокировка пользователей;
* одобрение/отклонение заявок на получение статуса продавца;
  1. Разработка модели данных
     1. Выделение существительных глаголов

В ходе анализа бизнес-требований был сформирован список из существительных объектов, глаголов и существительных-ролей (Таблица 1.1).

Таблица 1.1- Существительные и глаголы

|  |  |  |
| --- | --- | --- |
| Существительное | Глагол | Роли |
| Приложение | Регистрируется | Продавец |
| Пользователь | Создает | Покупатель |
| Товар | Продает | Администратор |
| Описание | Управляет |  |
| Рейтинг | Блокирует |  |
| Отзыв | Одобряет |  |
| Отчет | Отклоняет |  |
| Заявка | Удаляет |  |
|  | Формирует |  |

* + 1. Разработка концептуальной карты

В ходе разработки концептуальной картой основной сущностью было выявлен товар. Эту сущность еще называют ресурсом. Данный ресурс имеет различные атрибуты, такие как, описание, данные о продавце, отзывы, рейтинг.



Рисунок 1.1 - Концептуальная карта

Когда пользователь регистрируется он может создать запрос на получение статуса продавца, который принимает или отклоняет администратор. Продавец может создать, редактировать товар, а также запросить отчет по продажам данного товара. Отчеты формируются на основе статистики, которая ведется по продажам товара.

* + 1. Разработка ER-модели

В ходе анализа было выявлено, что одной из основных сущностей является «Товар». На рисунке 1.2 представлена модель данных «сущность связь» без атрибутов.

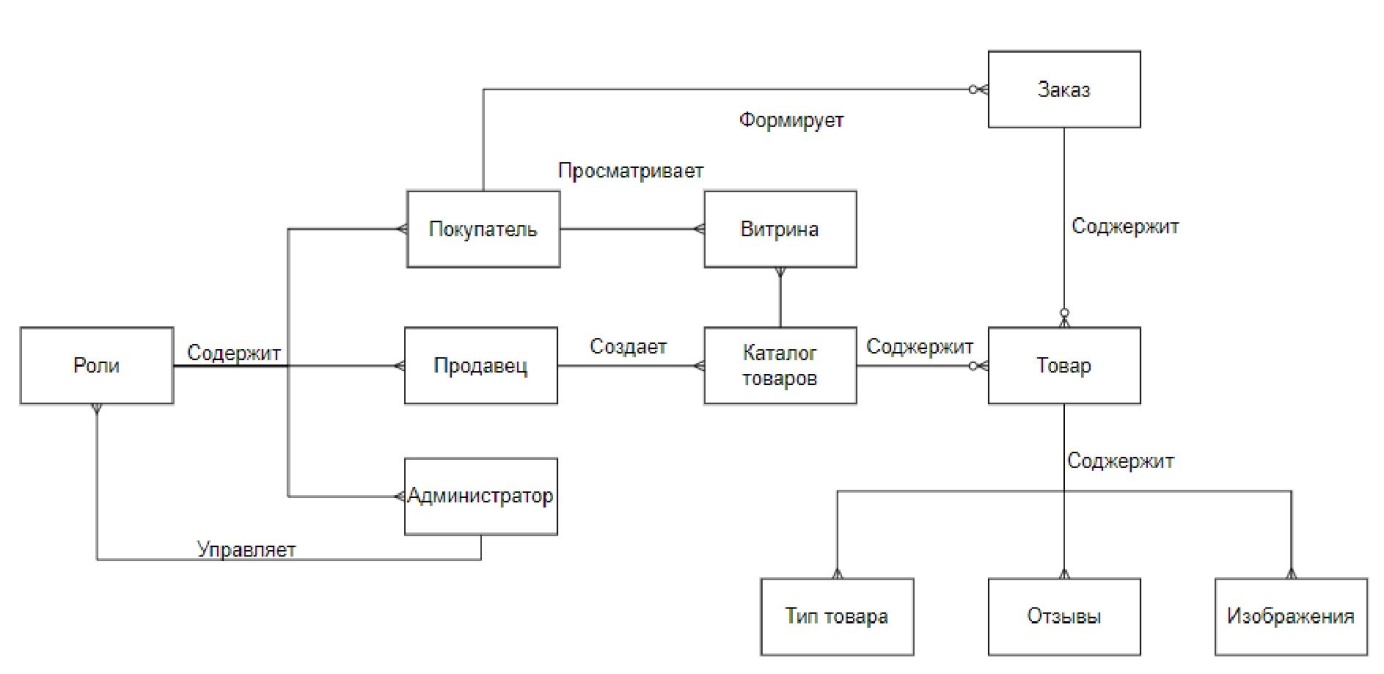


Рисунок 1.2 - Модель данных “сущность-связь” без атрибутов

На модели сущность-связь с атрибутами изображены основные поля каждой сущности. Основная сущность - товар обязательно содержит название, описание, изображения, информацию о продавце, отзывы. Модель

изображена на рисунке 1.3.

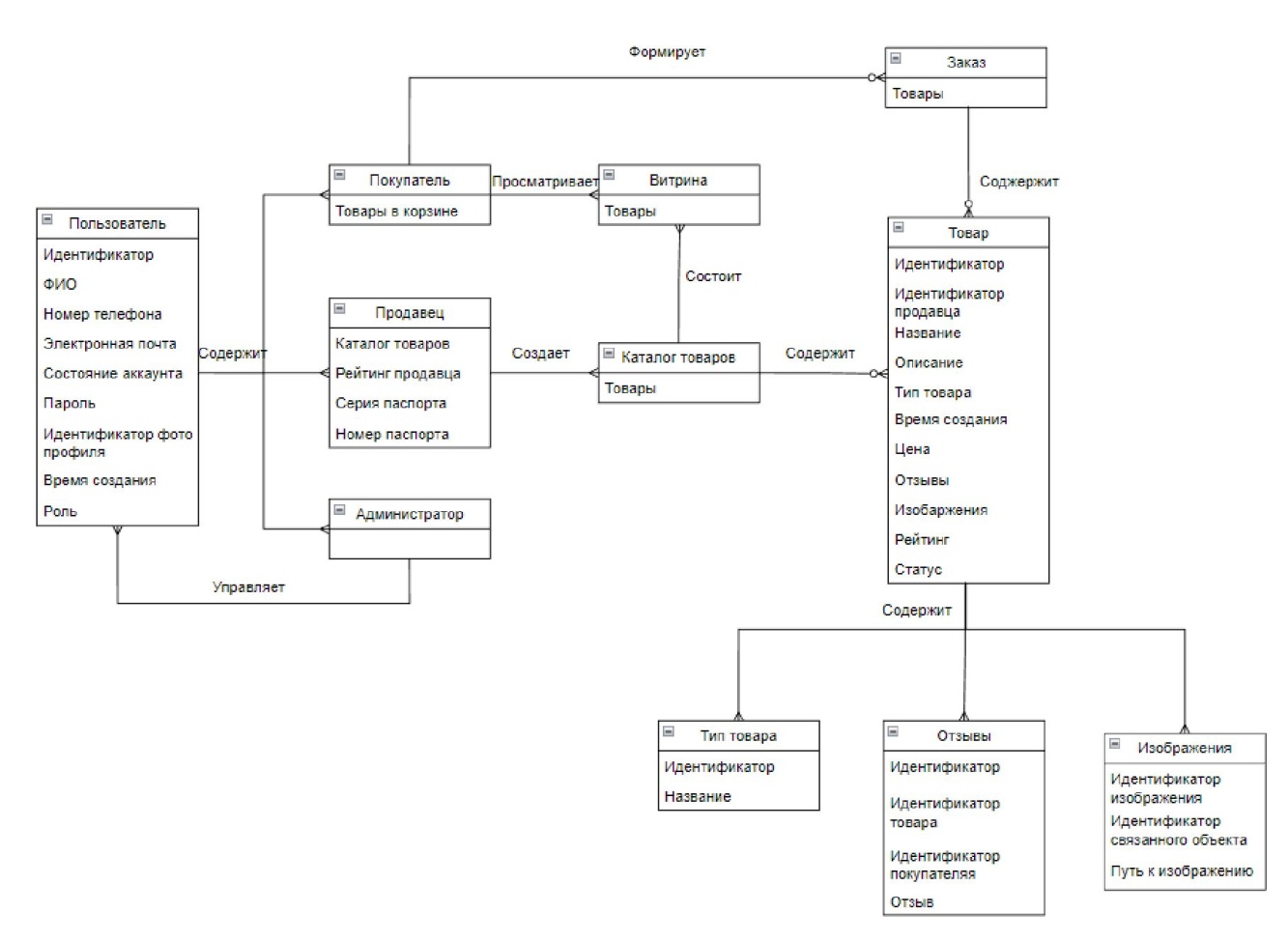


Рисунок 1.3 - Модель данных “сущность-связь” с атрибутами

* 1. Моделирование бизнес-процессов

Внутри сервиса планируется использовать систему ролей, таких как покупатель, продавец и администратор. Так как каждый пользователь регистрируется в качестве покупателя, и, если у него есть желание стать продавцом, то он может подать такую заявку.

На рисунке 1.4 представлена BMPN диаграмма процесса подачи заявки на получение статуса продавца.

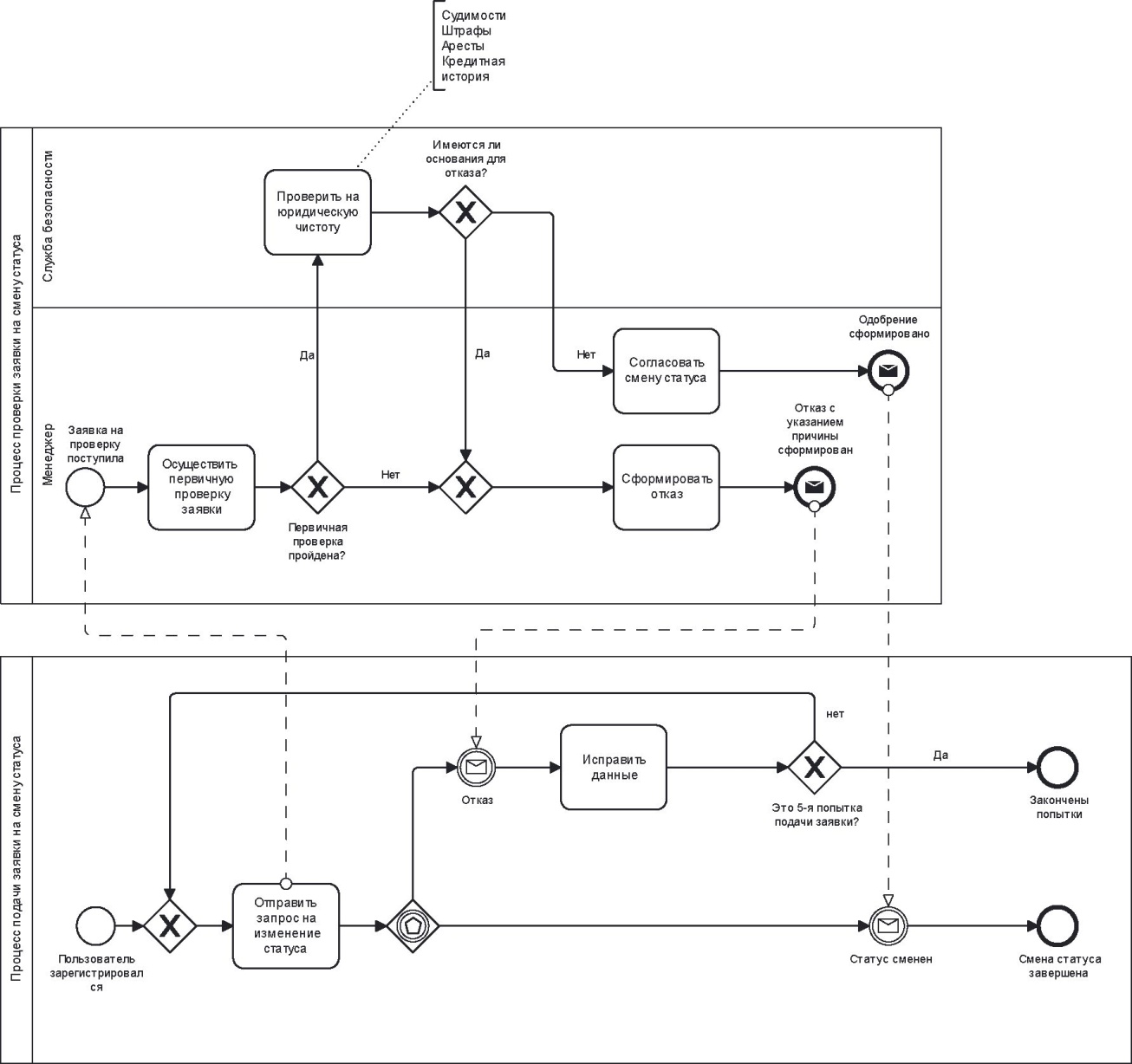


Рисунок 1.4 - BMPN диаграмма процесса подачи заявки на получение статуса продавца

Для получения статуса продавца, пользователь сначала должен зарегистрироваться затем заполнить заявку на смену статуса. Заявка проверяется в два этапа: сначала идет первичная проверка на корректность введенных данных, далее идет проверка пользователя на его юридическую чистоту. Чтобы заявка была одобрена нужно чтобы она прошла два этапа проверки. В случае отказа на смену статуса ее можно отправить повторно до пяти раз, если количество заявок превышает данное число, то заявку подать будет невозможно.

При создании товара было решено модерировать все созданные товары для того, чтобы избежать большого числа ошибочных объявлений, и для того, чтобы конечный пользователь был доволен продуктом и не был обманут продавцом.

На рисунке 1.5 представлена BMPN диаграмма процесса создания и выкладки товара в сервисе.

загрузил ли пользователь непристойные фотографии

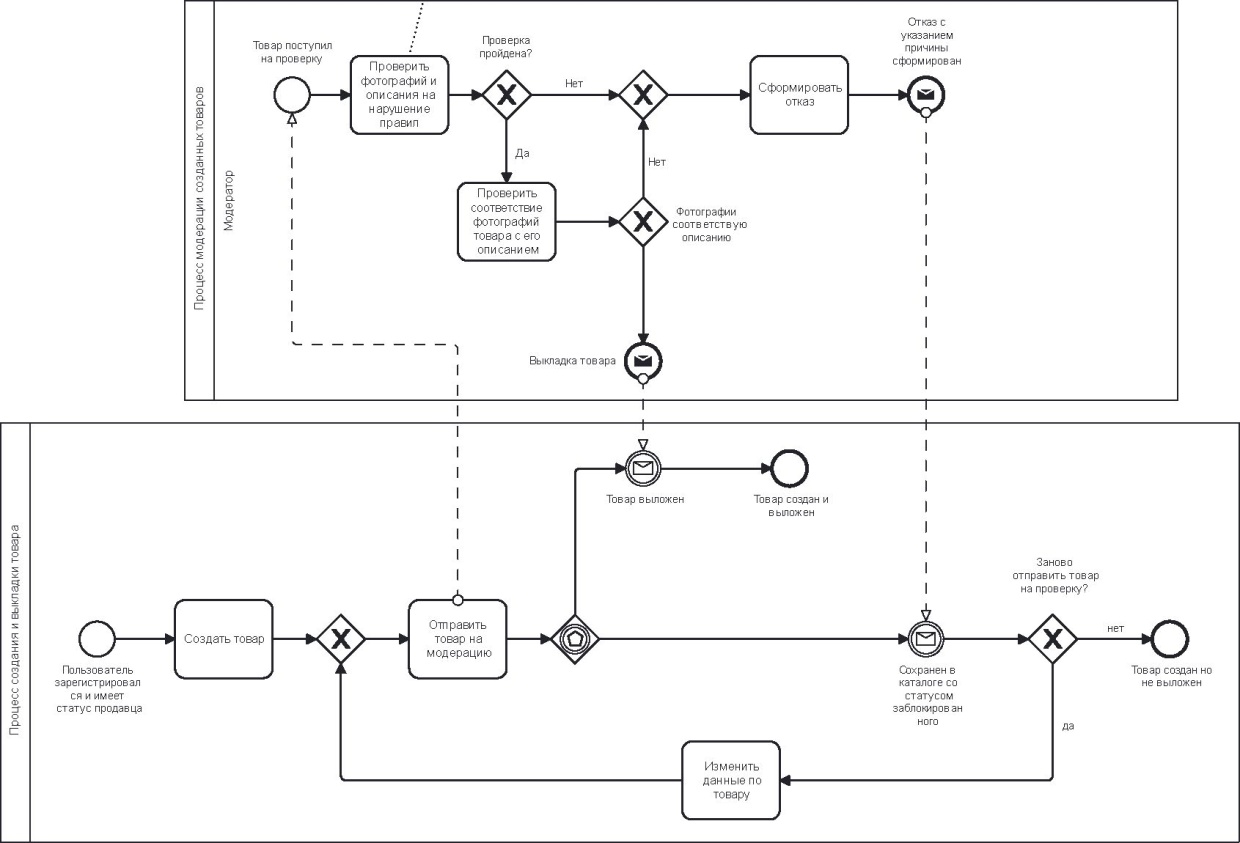


Рисунок 1.5 - BMPN диаграмма процесса создания и выкладки товара в

сервисе

Для того чтобы создать товар пользователь должен зарегистрироваться в системе. Далее нужно зайти в меню создания товара, внести нужны данные о товаре, добавить фото товара. Далее товар проходит проверку на то, нет ли в описании нецензурных слов и не загрузил ли пользователь непристойные фотографии, то после этого, идет проверка на соответствие товара и его описания, после прохождения всех проверок товар будет доступен для покупки либо, в случае отказа будет сохранен в каталоге продавца, но недоступен для покупателя.

Процесс покупки является одним из самых важных поэтому было принято решение вводить проверки на наличие товара, чтобы не произошло ситуаций, когда товар считается купленным, в то время как он уже закончился на складе.

На рисунке 1.6 представлена BMPN диаграмма процесса покупки товара.

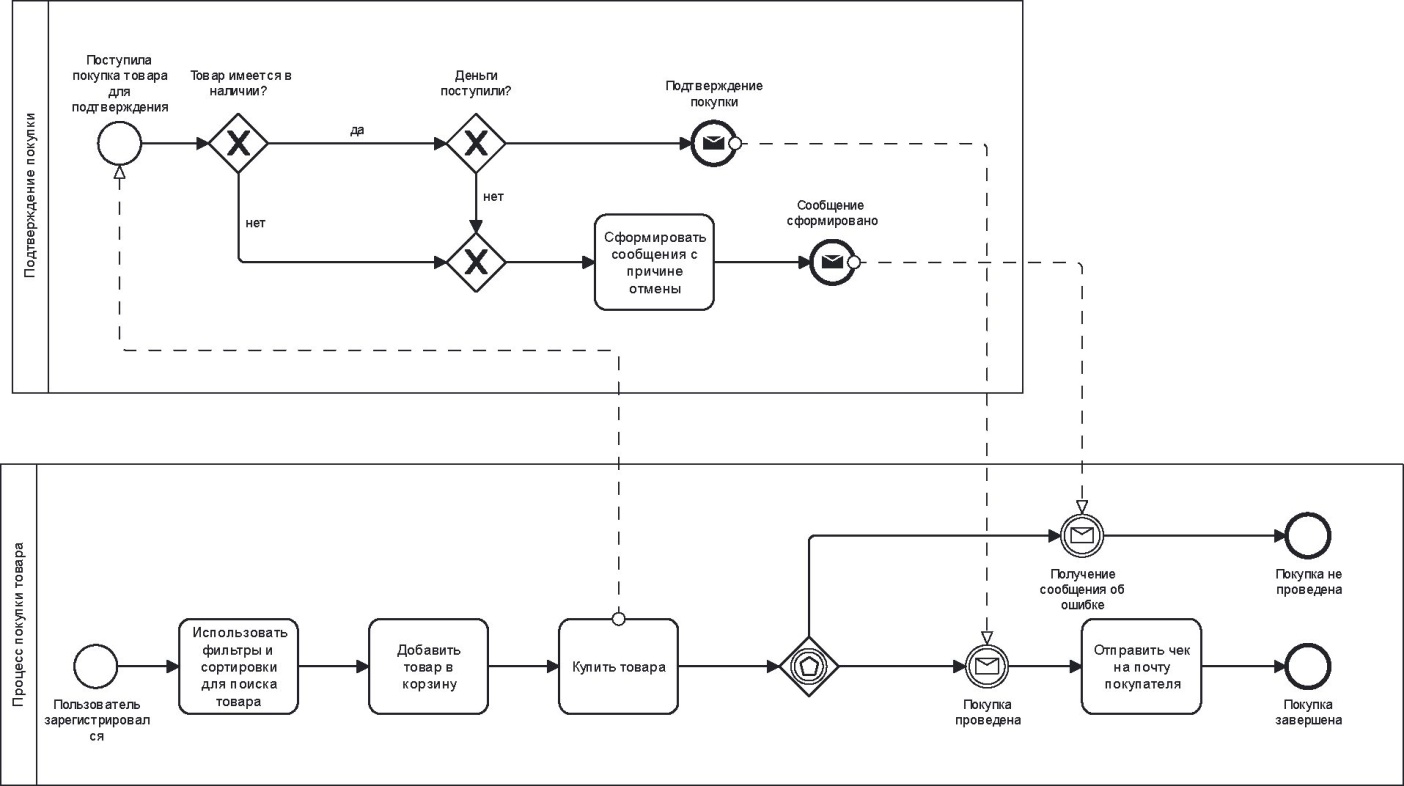


Рисунок 1.6 - BMPN диаграмма процесса покупки товара.

Для того чтобы купить товар пользователь сначала должен зарегистрироваться в системе. Используя фильтрацию и сортировку, 18

пользователь должен выбрать нужный ему товар и добавить его в корзину. В корзине пользователь покупает товар. После покупки идет проверка на то есть ли данный товар в наличии у продавца, если есть, то далее идет проверка на то поступили ли деньги. Только после этих двух проверок происходит регистрация покупки и на почту пользователя отправляется информация о его покупке.

* 1. Разработка пользовательских требований

Следующим этапом после описания основных бизнес-процессов будет описание сценариев использования. В данной главе будет рассмотрено то, как каждый из типов пользователей будет взаимодействовать с системой. Будут описаны шаги каждого сценария, основной и альтернативный поток прецедента.

* + 1. Диаграмма вариантов использования

Участники в диаграмме вариантов использования является:

* администратор;
* покупатель;
* продавец.

Варианты использования:

* авторизоваться;
* создание товара;
* покупка товара;
* получение статуса продавца для пользователя;
* управление пользователями, обработка запросов;

Диаграмма вариантов использования представлена на Рисунок 1.7

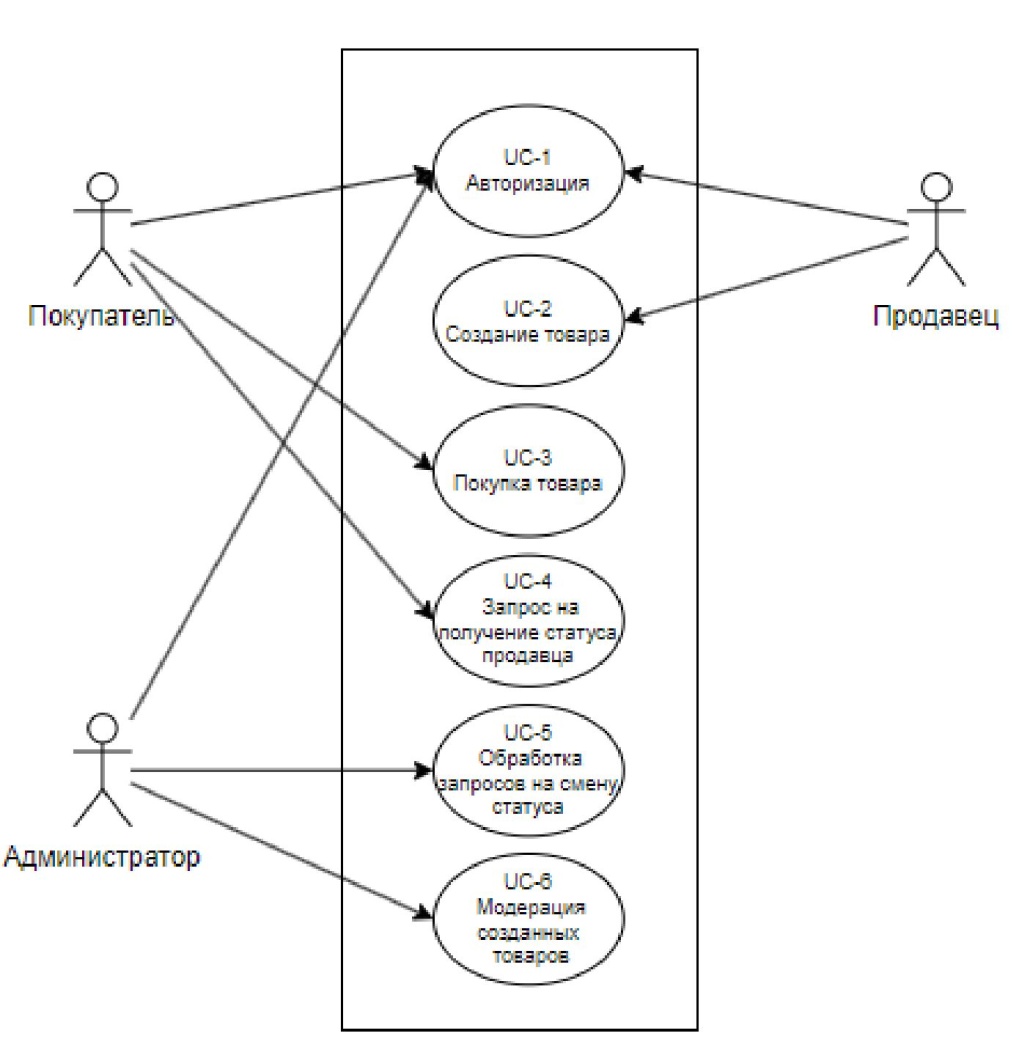


Рисунок 1.7 - Сценарии использования

* + 1. Спецификации варианта использования

В соответствии с методологией UML каждый прецедент должен быть описан в виде спецификации, содержащей основные свойства прецедента и сценарий.

В качестве шаблона спецификации для описания прецедента был выбран шаблон со следующей информацией:

* имя прецедента;
* краткое описание;
* актеры, задействованные в прецеденте;
* основной поток прецедента;
* расширение потока (опционально).

Вариант использования - Авторизация представлен в таблице 1.2.10

Таблица 1.2 - Авторизация

|  |  |
| --- | --- |
| **UC-1 Авторизация** | |
| Определение/Ценность | Операция для авторизации пользователя в приложении |
| Пользователь | Продавец/Покупатель/Администратор |
| Предусловие | Пользователь не авторизован в системе |
| Основной сценарий | 1. Пользователь открывает приложение 2. Система отображает страницу авторизации 3. Пользователь вводит логин и пароль 4. Система проверяет введенные данные 5. Система разрешает вход |
| Альтернативные пути | 5.а.1 Ведены неверные имя пользователя или пароль  5.а.2 Приложение предлагает отправить данные для входа на почту 5.b Нет соединения с сервером |
| Результат | Пользователь авторизовался |

Вариант использования - Создание товара представлен в таблице 1.3

Таблица 1.3 - Создание товара

|  |  |
| --- | --- |
| **UC-2 Создание товара** | |
| Определение/Ценность | Операция предназначена для создания товара в приложении |
| Пользователь | Продавец |
| Предусловие | Пользователь авторизован в системе |
| Основной сценарий | 1. .Пользователь открывает свой каталог товаров 2. .Система отображает возможность создания товара 3. .Пользователь инициирует создание нового товара 4. .Система отображает конструктор 5. .Пользователь вносит информацию о товаре и загружает фотографии 6. .Система отображает информацию об успешном создании товара и об отправке его на проверку. |
| Результат | Товар создан |

Вариант использования - Покупка товара представлен на таблице 1.4

Таблица 1.4 - Покупка товара

|  |  |
| --- | --- |
| **UC-3 Покупка товара** | |
| Определение/Ценность | Операция предназначена покупки товара |
| Пользователь | Покупатель |
| Предусловие | Пользователь авторизован в системе |
| Основной сценарий | 1. .Пользователь открывает список товаров 2. .Система отображает все товары 3. .Пользователь использует фильтры и сортировку чтобы найти нужный товар. |

|  |  |
| --- | --- |
|  | 1. .Система отображает товары с учетом введенных пользователем данных. 2. .Пользователь добавляет товар в корзину 3. .Пользователь открывает корзину. 4. .Система отображает товары, добавленные в корзину. 5. .Пользователь инициирует покупку товара. 6. . Система проверяет наличие товара у продавца, проверяет поступление средств покупателя. |
| Альтернативные пути | 9.а.1 У продавца закончился товар.  9.а.2 Покупка отменена  9.b.1 Средства не поступили  9.b.2 Покупка отменена  9.с.1 Нет соединения с сервером |
| Результат | Товар куплен, на почту покупателя отправляется письмо с данными по покупке. |

Вариант использования - Запрос на получение статуса продавца представлен в таблице 1.5

Таблица 1.5 - Запрос на получение статуса продавца

|  |  |
| --- | --- |
| **UC-4 Запрос на получение статуса продавца** | |
| Определение/Ценность | Операция предназначена для получения покупателем статуса продавца |
| Пользователь | Покупатель |
| Предусловие | Пользователь авторизован в системе |
| Основной сценарий | 1. Пользователь открывает профиль. 2. Система отображает возможность смены статуса 3. Пользователь инициирует смену статуса 4. Система выводит форму заявки 5. Пользователь заполняет форму 6. Система отображает информацию об успешном заполнении формы. |
| Результат | Запрос отправлен |

Вариант использования - Обработка запроса на смену статуса представлен в таблице 1.6

Таблица 1.6 - Обработка запроса на смену статуса

|  |  |
| --- | --- |
| **UC-5 Обработка запроса на смену статуса** | |
| Определение/Ценность | Операция предназначена для обработки запросов на получение статуса продавца |
| Пользователь | Администратор |
| Предусловие | Пользователь открыл список запросов на смену статуса |

|  |  |
| --- | --- |
| Основной сценарий | 1. Пользователь открыл запрос пользователя. 2. Система отображает информацию в запросе 3. Пользователь проверят корректность запроса 4. Пользователь одобряет запрос 5. На почту автора запроса отправляется сообщение о результате проверки |
| Альтернативные пути | 1. .а.1 Пользователь отклоняет запросов 2. .а.2 На почту автора запроса отправляется   сообщение о результате проверки |
| Результат | Запрос обработан. |

Вариант использования - Модерация созданного товара представлен в таблице 1.7

Таблица 1.7 - Модерация созданного товара

|  |  |
| --- | --- |
| **UC-6 Модерация созданного товара** | |
| Определение/Ценность | Операция предназначена для обработки запросов об выставлении новых товаров |
| Пользователь | Администратор |
| Предусловие | Пользователь открыл список созданных товаров |
| Основной сценарий | 1. Пользователь открыл информацию товара. 2. Система отображает товар, описание и фотографии 3. Пользователь проверят товар на соблюдение правил, корректность введенных данных 4. Пользователь одобряет/отклоняет запрос 5. На почту автора запроса приходит сообщение о результате проверки |
| Результат | Запрос обработан. |

* 1. Разработка функциональных требований

На основании пользовательских требований и бизнес-процесса были разработаны функциональные требования, представленные в таблице 1.8

Таблица 1.8 - Функциональные требования

|  |  |
| --- | --- |
| **Идентификатор** | **Функции/требования** |
| **FR-1** | **Авторизация** |
| **FR-1.1** | Система должна отображать страницу авторизации |
| **FR-1.2** | Система должна оповещать пользователя об ошибках при неправильно введенных данных |

|  |  |
| --- | --- |
| **FR-2** | **Создание товара** |
| **FR-2.1** | Система должна предоставлять продавцу возможность создать товар |
| **FR-2.2** | Система должна предоставлять возможность администратору показывать список созданных товаров |
| **FR-2.3** | Система должна предоставлять возможность администратору отклонять или принимать созданные товары |
| **FR-2.4** | Система должна предоставлять возможность администратору описывать причину отказа на размещение товара |
| **FR-3** | **Покупка товара** |
| **FR-3.1** | Система должна предоставлять возможность покупателю открывать витрину товаров |
| **FR-3.2** | Система должна предоставлять возможность покупателю просматривать товары используя фильтры и сортировки |
| **FR-3.3** | Система должна предоставлять возможность добавлять товар в корзину |
| **FR-3.4** | Система должна предоставлять возможность покупателю просматривать список товаров в корзине |
| **FR-3.5** | Система должна предоставлять возможность покупателю купить товар |
| **FR-3.6** | Система должна предоставлять возможность отправки чека о покупке на почту покупателя |
| **FR-4** | **Запрос на получение статуса на продавца** |
| **FR-4.1** | Система должна отображать профиль пользователя |
| **FR-4.2** | Система должна отображать возможность смены статуса |
| **FR-4.3** | Система должна отображать список заявок на смену статуса |
| **FR-4.4** | Система должна предоставлять возможность смены статуса |
| **FR-4.5** | Система должна предоставлять возможность администратору принимать или отклонять запрос на смену статуса |
| **FR-4.6** | Система должна предоставлять возможность отправки на электронную почту причины отклонения запроса на смену статуса |
| **FR-5** | **Управление пользователями** |
| **FR-5.1** | Система должна отображать список всех пользователей |
| **FR-5.2** | Система должна предоставлять возможность блокировать пользователей |
| **FR-5.3** | Система должна сделать невозможным вход для  заблокированных пользователей |
| **FR-6** | **Управление товарами** |
| **FR-6.1** | Система должна отображать каталог товаров продавца |
| **FR-6.2** | Система должна предоставлять возможность продавцу удалять товар |
| **FR-6.3** | Система должна предоставлять возможность продавцу изменять информацию о товаре |
| **FR-7** | **Отчеты** |
| **FR-7.1** | Система должна предоставлять возможность продавцу формировать отчет о продажах и отзывах за определенный период времени |

* 1. Разработка нефункциональных требований

На основании пользовательских требований и бизнес-процесса были разработаны функциональные требования, представленные в таблицах 1.9­1.14.

* + 1. Требования к пользовательским интерфейсам

Требования к пользовательским интерфейсам представляют собой требования к тому, что должен видеть пользователь сервиса. Требования к стилю и прочие требования к визуальной части. Требования к пользовательским интерфейсам представлены на таблице 1.9.

Таблица 1.9— Требования к пользовательским интерфейсам

|  |  |
| --- | --- |
| UI-1 | Витрина товаров должна иметь сортировку и фильтрацию по различным по категории товаров и по конкретному типу в категории |
| UI-2 | Весь пользовательский интерфейс должен быть выполнен в едином стиле |
| UI-3 | Страница просмотра списка товаров должна представлять собой список из двенадцати товаров и переключение между страницами |
| UI-4 | Система должна имеет адаптированную верстку под различные экраны |

* + 1. Требования к удобству использования

Требования к удобству использования подразумевают требования к простоте использования, эффективности взаимодействия, специальным возможностям. Требования представлены на таблице 1.10.

Таблица 1.10— Требования к удобству использования

|  |  |
| --- | --- |
| USE-1 | Система должна позволить пользователю создать товар минимальным количеством действий: указывать только  критические поля товара:   1. Название. 2. Категория товара 3. Цена 4. Количество |

|  |  |
| --- | --- |
| USE-2 | При просмотре витрины товаров должны отображаться только те товары, которые имеются в наличии |
| USE-3 | При просмотре своего каталога товаров продавец должен видеть все свои товары, доступные к покупке и недоступные |

* + 1. Требования к производительности

Требования к производительности - это требования для различных операций внутри сервиса. Требования к производительности представлены на таблице 1.11.

Таблица 1.11— Требования к производительности

|  |  |
| --- | --- |
| PER-1 | Время отклика системы на запрос пользователя не должно превышать 1 секунды |

* + 1. Требования к безопасности

Требования к безопасности, которые ограничивают доступ к данным или к возможности использования сервиса представлены в таблице 1.12. Таблица 1.12— Требования к безопасности

|  |  |
| --- | --- |
| SEC-1 | Не авторизованные пользователь может лишь просматривать товары, для дальнейших действий требуется авторизация |
| SEC-2 | Все запросы к API, кроме авторизации, должны быть подписаны токеном. |
| SEC-3 | Пароли пользователей, и паспортные данные продавца должны хранится в зашифрованном виде |

* + 1. Требования к надежности

Требования к надежности представлены на таблице 1.13.

Таблица 1.13— Требования к надежности

|  |  |
| --- | --- |
| REL-1 | При возникновении ошибки при проведении покупки пользователь должен быть уведомлен об ошибке |
| REL-2 | Система должна иметь возможность создания резервной копии базы данных |

1.6.6 Бизнес-правила

Бизнес правила представлены в таблице 1.14.

Таблица 1.14— Бизнес-правила

|  |  |
| --- | --- |
| BR-1 | После каждой проведенной покупки проверяется наличие товара у текущего продавца и если он закончился, то объявление становится невидимым для дальнейших  покупателей. |
| BR-2 | Каждый созданный товар должен проходить проверку администратором |

* 1. Выводы по главе

В ходе разработки требований на разработку система по продаже товаров для компании «UBuySell» была выявлена и описана проблема компании - повышение прибыли с продаж товаров. Компания желает увеличить прибыль с каждого проданного товара и хочет получать дополнительный процент с продаж других продавцов. Руководство компании поручило решать данную проблему.

Решение представляет собой маркетплейс который позволяет выставлять и покупать товары. В ходе анализа на разработку маркетплейса были разработаны модели бизнес-процессов деятельности маркетплейса, выявлены стейкхолдеры и разработаны требования:

* бизнес-требования к системе;
* 5 пользовательских требований в виде модели вариантов

использования;

* 26 функциональных требований;
* нефункциональные требования.

Полученных результатов достаточно для выполнения следующей работы - проектирования информационной системы.

2 Проектирование информационной системы

* 1. Функциональная модель системы

В соответствии с функциональными требованиями, описанными в предыдущей главе сформирована функциональная модель, показывающая входы, выходы и функции. Схема изображена на рисунке 2.1.

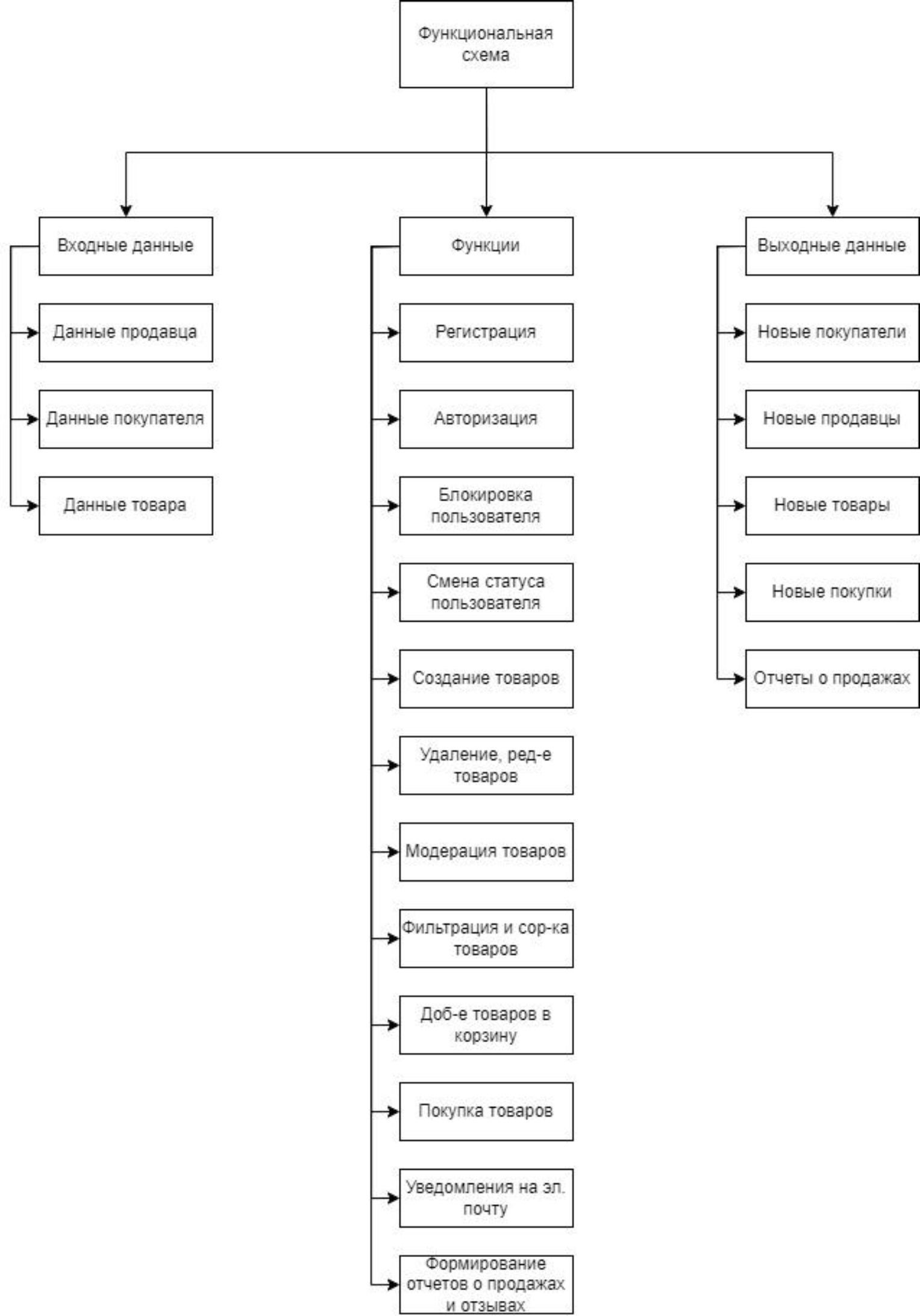


Рисунок 2.1 - Функциональная модель системы

Основные из представленных в схеме функций описаны в следующем разделе.

* 1. Разработка алгоритмов системы

Чтобы подробно рассмотреть основные функции, показанные в функциональной модели, были применены блок-схемы. Блок-схемы являются графическими инструментами, которые используются для описания алгоритмов и процессов. Блок-схемы удобны в использовании, так как они позволяют визуализировать алгоритм и легко понять его структуру.

* + 1. Алгоритм авторизации

На основе анализа существующих сервисов был разработан алгоритм авторизации. Для реализации авторизации в классе User, описывающего пользователя, был реализован интерфейс UserDetails и библиотеки Spring Security. Так мы отмечаем для Spring что данный класс является пользователем. Также был создан класс SecurityConfig [Приложение А] который наследует абстрактный класс WebSecurityConfigurerAdapter. В данном классе переопределены некоторые методы для установки настроек безопасности. В этом классе были перечислены области сайта, к которым имеет доступ неавторизованный пользователь, была обозначена страница авторизации, также была установлен метод шифрования паролей пользователей.

Сам процесс авторизации происходит следующим образом: пользователь вводит свой логин и пароль в соответствующие поля на странице авторизации, после чего идет поиск пользователя с таким же логином, если пользователь найден, то идет сравнение с зашифрованным паролей. Для шифрования паролей был использован метод шифрования

библиотеки Spring Security BCryptPasswordEncoder,

сложностью 8. Если пользователь найден и введенный пользователь авторизован, если нет, то отбрасывается странице авторизации появляется сообщение об ошибке.

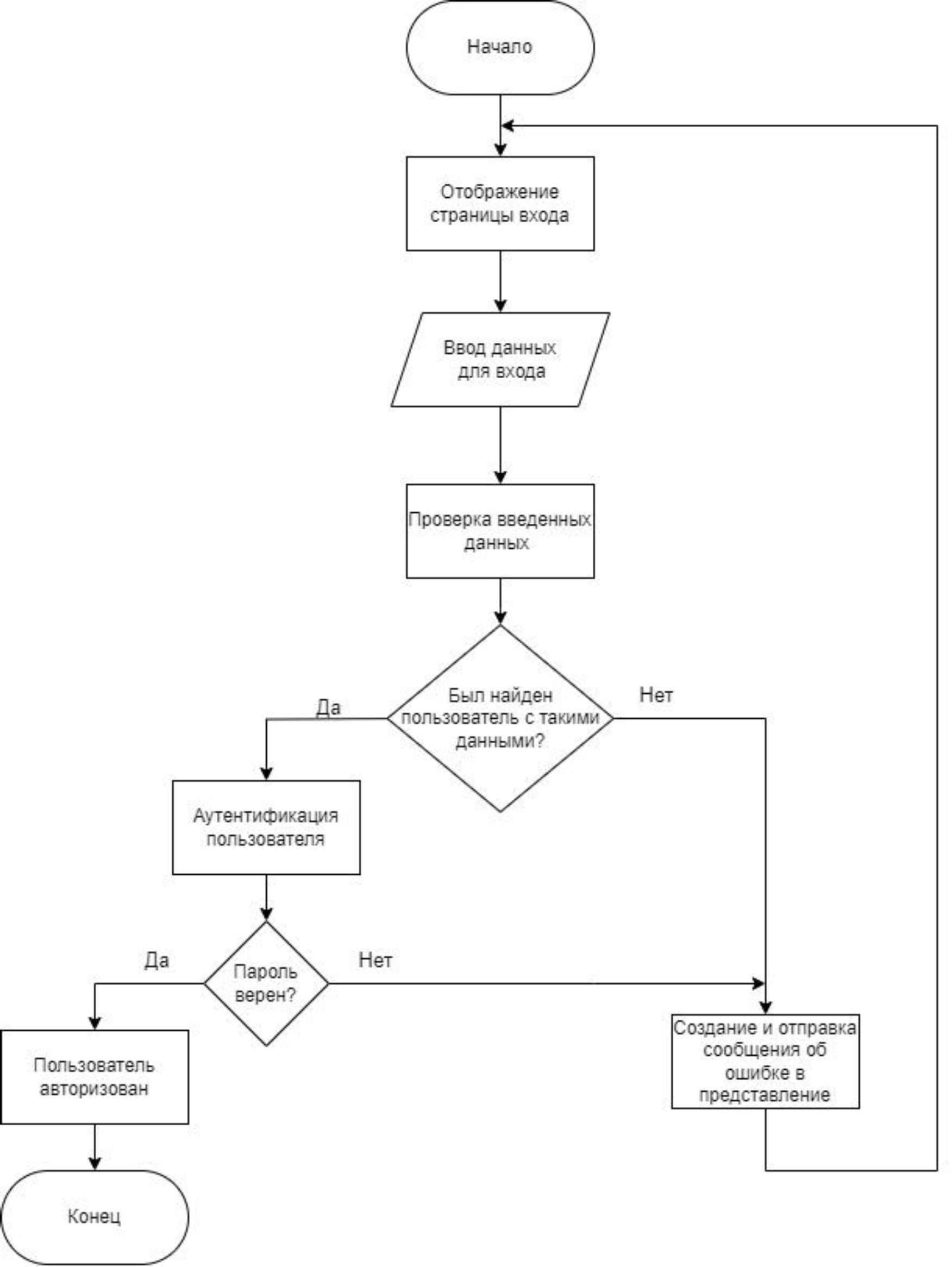
Блок-схема которого изображена на рисунке 2.2.

с установленной

пароль верен, то

исключение и на

Рисунок 2.2 - Блок-схема реализации алгоритма авторизация



* + 1. Алгоритм регистрации

Если пользователь впервые зашел в приложение ему нужно зарегистрироваться. В ходе проектирования было принято решение что каждый пользователь при регистрации будет являться покупателем, чтобы стать продавцом ему, в будущем, потребуется создать заявку для этого.

Блок-схема алгоритма регистрации представлена на рисунке 2.3.

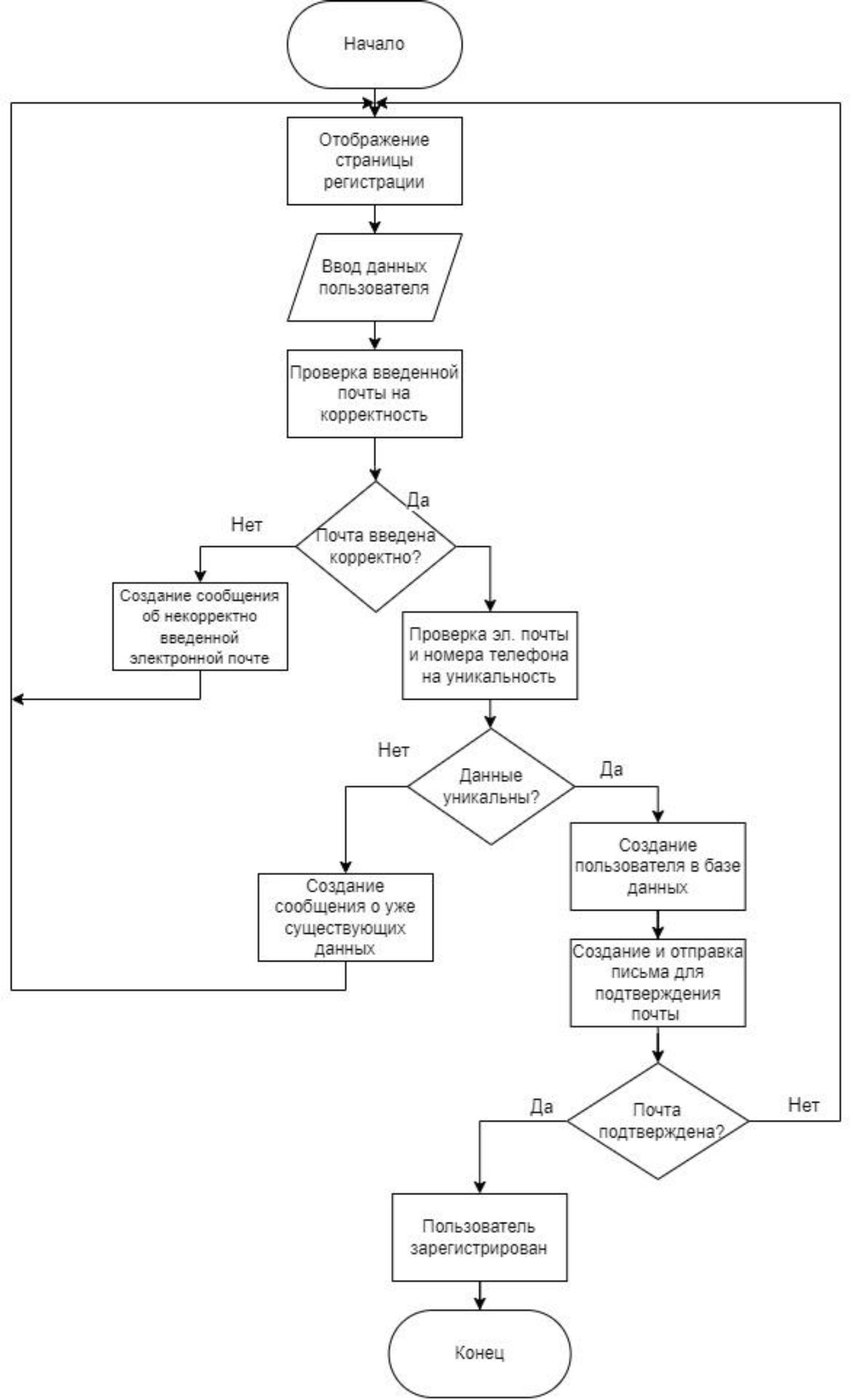


Рисунок 2.3 - Блок-схема реализации алгоритма регистрации

Вводятся такие данные как: имя, фамилия, номер телефона, электронная почта и пароль. После того как пользователь нажал кнопку регистрации, ему нужно подтвердить адрес электронной почты, который он ввел. На его почту приходит письмо с подтверждением. Только после подтверждения почты пользователь может авторизоваться и начать полноценно пользоваться сайтом.

* + 1. Алгоритм создания товара.

Так как товар, внутри сервиса, является основным объектом, было принято решение усложнить сам процесс создания товара, что отражено в блок-схема алгоритма создания товара.

После того как пользователь вводит все данные о товаре и нажимает кнопку создания, на сервер отправляется POST запрос о создании нового товара. Данные запроса приходят в контроллер, в котором вызывается метод класса ProductService [Приложение Б]. В методе создания обрабатываются введенные пользователем данные. После этого товару присваивается статус валидации, он сохраняется в базу данных и ожидает проверки.

При создании товаров продавец может допустит множество ошибок, которые система может не предвидеть и не среагировать, поэтому было принято решение, чтобы каждый созданный товар проверял человек. Это усложняет процесс, но большим плюсом служит, то, что минимизируется количество товаров с некорректно указанными данными. После того как товар пройдет пройдёт процедуру проверки, он станет доступен для покупки. Если же товар не пройдет проверку, то он остается невидим для покупателей, но его может видеть продавец в своём списке товаров. О результатах проверки продавец уведомляется письмом на электронную почту.

Блок-схема алгоритма создания товара изображена на рисунке 2.4.

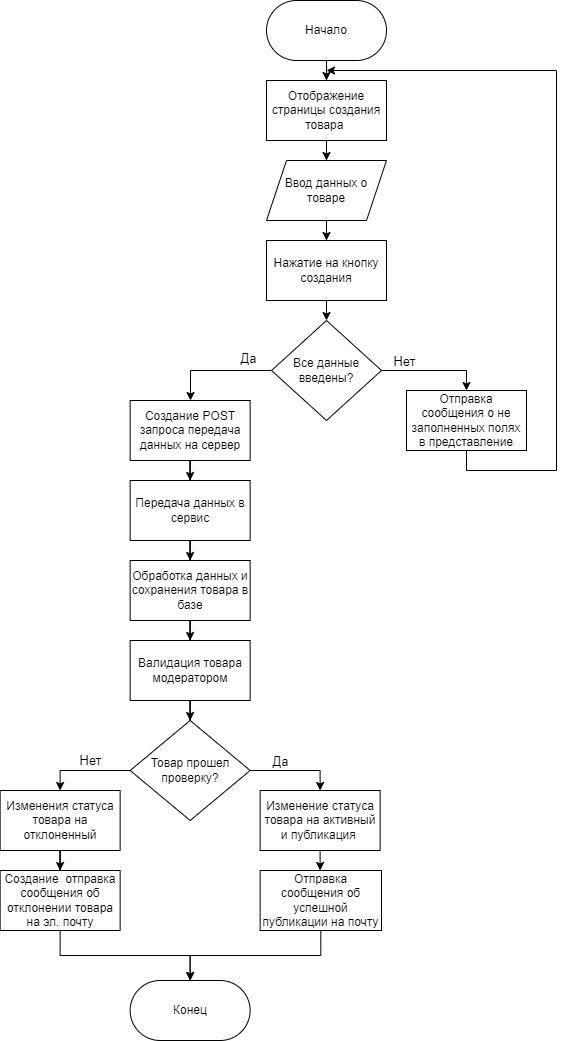


Рисунок 2.4 – Блок-схема реализации алгоритма создания товара

* + 1. Алгоритм покупки товара

Покупка товара является не менее важной функцией приложения, поэтому, как и при создании товара было учтено немало параметров для того, чтобы этот процесс проходил без ошибок и непредвиденных ситуаций. Блок- схема алгоритма покупки товара представлена на рисунке 2.5.

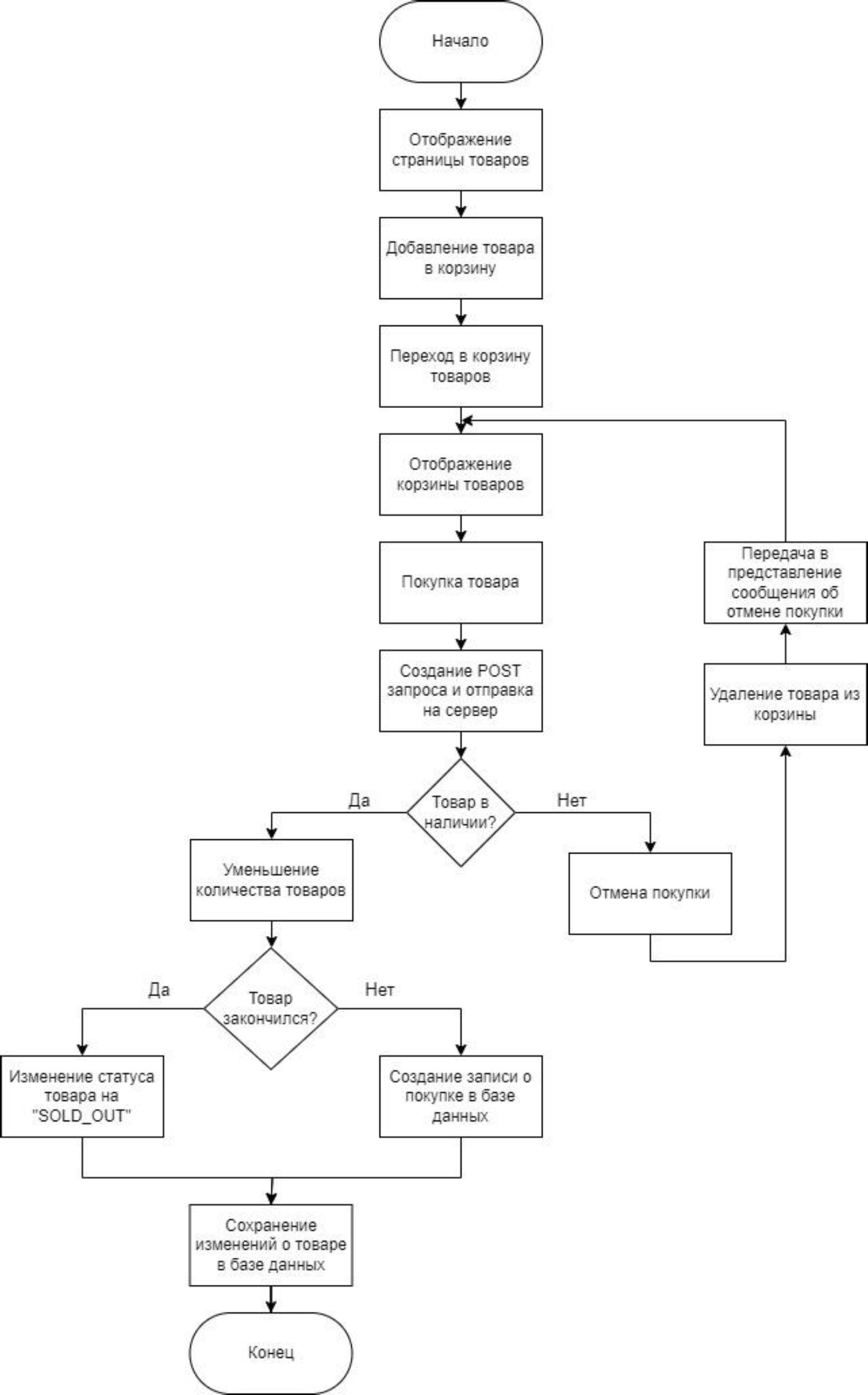


Рисунок 2.5 - Блок-схема реализации алгоритма покупки товара

Покупку пользователь может совершить двумя способами:

* сразу на странице товара;
* добавить товары в корзину и совершить покупку всех товаров на странице корзины.

После того как пользователь нажимает на кнопку покупки товара. На сервер отправляется POST запрос, который поступает в контроллер и передается в ProductService. Сначала проверяется наличие товара. Если товара нет, то операция отменится, товар удаляется из корзины. Далее создается сообщение о товаре, который закончился, и пользователь перенаправляется обратно на страницу корзины и на ней отображается сформированное сообщение. Если же товар есть в наличии, то покупка регистрируется и количество товара уменьшается в базе данных. Далее проверяется, не закончился ли товар после покупки. Если товар закончился, то его статус меняется на закончившийся. После этого товар становится недоступен для покупки перестает показываться покупателям и продавцу приходит письмо на электронную почту о том, что товар закончился, но он доступен для просмотра продавцу.

2.2.5 Алгоритм создания отчета

Для того чтобы понимать, насколько хорошо, или плохо идут продажи товаров, было принято решение реализовать получение отчета по продажам и отзывам для продавца. Для того, чтобы пользователю загрузить Excel файл, ему нужно выбрать нужный промежуток дат для отчетов и выбрать путь сохранения. Блок-схема создания отчета изображен на рисунке 2.6.

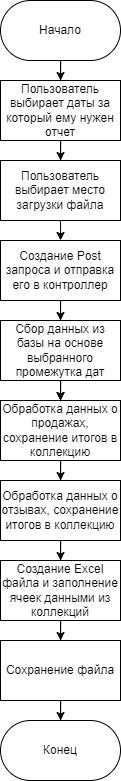


Рисунок 2.6 - Блок-схема реализации алгоритма создания отчетов

Созданный Excel файл будет содержать два листа:

* отчет о продажах, где будут показаны все проданные товары, общая выручка от их продажи;
* отчет об отзывах покупателей, оценку товара на текущий момент количество оценок, средний рейтинг за все время.

**2.3 База данных**

* + 1. Выбор системы управления базой данных

Для работы сайта необходима база данных. Для взаимодействия с базами данных нужно использовать систему управления базами данных (СУБД).

Для проекта была будет использоваться реляционная СУБД. Структура таких баз данных позволяет связывать информацию из различных таблиц с помощью внешних ключей (или индексов), которые используются для уникальной идентификации любого фрагмента данных в этой таблице. Нереляционные базы не подходят так как в основе они используют хеш- таблицы, деревья, хранящие только записи, не имея возможности создать внешние ключи.

Для данной работы будет использоваться MySQL - бесплатный продукт с открытым исходным кодом от Oracle. Отличается стабильностью и хорошим тестированием обновлений до их внедрения. MySQL можно доработать под свои нужды или поискать готовые исправления в обширной библиотеке профильного сообщества.

* + 1. Разработка моделей для базы данных.

Модели представляют собой отдельные таблицы в базе данных и такие же классы в Java. Они используются для хранения бизнес-логики, необходимой для каждой модели. Для сервиса маркетплейса выделю следующие основные модели:

* + - 1. Модель User

Модель User - это модель пользователя в сервисе. В классе присутствует поле роли, которое определяет возможности человека в сервисе. В зависимости от роли, зарегистрированный пользователю могут быть доступны определенные функции, которые недоступны людям с другой ролью. Содержит следующие поля:

* id - идентификатор;
* email - электронная почту;
* password - пароль;
* activationCode - код для подтверждения электронной почты;
* phoneNumber - номер телефона;
* firstName - имя пользователя;
* surname - фамилия;
* avatarId - идентификатор фотографии в таблице фотографий;
* role - роль пользователя;
* isActive - поле определяющее активен ли пользователь, если значение false то пользователь считается заблокированным и он не может авторизоваться;
* creationDate - дата и время создания пользователя.
  + - 1. Модель Seller

Модель Seller - это модель, определяющая дополнительные поля для класса User. Так как пользователь заключает некое соглашение, в момент становления продавцом, он предоставляет дополнительные данные о себе, которые хранятся в этой таблице, также хранится поле рейтинга, которое окажется полезным для покупателя при выборе товара.Содержит следующие поля:

* id - идентификатор;
* userid - идентификатор пользователя;
* fullName - ФИО;
* passposrtSeries - серия паспорта;
* passportNumber - номер паспорта;
* rating - рейтинг продавца на основе оценок его товаров.
  + - 1. Модель Product

Модель Product - это модель товара, который создает продавец. Каждый товар имеет статус, который показывает его текущее состояние и этот статус определяет доступен ли данный товар для покупки. Содержит следующие поля:

* id - идентификатор;
* title - название товара;
* description - описание товара;
* price - цена;
* parentCategory - общая категория товара;
* childCategory - подкатегория товара из общей категории, необязательно поле;
* productStatus - статус товара, показывающий его текущее состояние, он может иметь следующие значения:
* в процессе проверки;
* активен;
* отклонен;
* закончился;
* удален.
* amount - количество товара, которое указывается при создании товара, данное количество уменьшается в процессе покупок и, когда товар заканчивается, у него меняется статус и он становится недоступен к покупке;
* sales - счетчик продаж товара, используется при сортировке;
* isInfinite - булевое поле, означающее что товар не закончится, если поле равно true, то для товара не будет считаться количество и он всегда будет активен;
* previewImageId - идентификатор первой фотографии товара, которое будет отображаться при просмотре всех товаров;
* userId - идентификатор продавца;
* rating - рейтинг товара на основе оценок пользователей;
* creationDate - дата и время создания товара;
* images - идентификаторы фотографий из таблицы фотографий.
  + - 1. Модель Review

Модель Review - это модель отзыва на товар. Каждый зарегистрированный пользователь может оставить отзыв и поставить оценку товару. На основе этих оценок будет формироваться его рейтинг. Содержит следующие поля:

* id - идентификатор;
* productId - идентификатор товара;
* rating - оценка товара;
* reviewText - текст отзыва;
* userId - идентификатор покупателя, оставившего отзыв;
* creationDate - дата и время написания отзыва.
  + - 1. Модель StatusRequest

Модель StatusRequest - это модель запроса пользователя на смену статуса на продавца. Содержит следующие поля:

* id - идентификатор;
* userid - идентификатор пользователя, создавшего запрос;
* fullName - ФИО;
* passposrtSeries - серия паспорта;
* passportNumber - номер паспорта;
* description - описание товара;
* state - статус заявки, может принимать следующие значения:
* создан;
* одобрен;
* отклонен.
* reasonOfDeviation - причина отклонения. При обработке заявки, если заявка отклоняется по той или иной причине, всегда указывается причина отклонения;
* images - идентификаторы фотографий из таблицы фотографий.
  + - 1. Модель Cart

Модель Cart - это модель которая определяет корзину, каждый покупатель добавляет в корзину товар, который будет сохранятся в соответствующей таблице. Содержит следующие поля:

* id - идентификатор;
* productId - идентификатор товара;
* userId - идентификатор покупателя;
* amount - количество товара.
  + - 1. Модель Purchase

Модель Purchase - этот модель отвечает за регистрацию покупки товара. После того как пользователь покупает товар/товары, создается один или несколько новых записей в таблице. В дальнейшем на основе этой 41

таблицы будет строится отчет о продажах для продавца за определенный период времени. Этот отчет будет формироваться в Excel файл и продавец сможет его скачать. Содержит следующие поля:

* id - идентификатор;
* productId - идентификатор товара;
* clientId - идентификатор покупателя;
* sellerId - идентификатор продавца;
* amount - количество товара;
* price - цена покупки;
* purchaseDate - дата и время покупки.

2.3.3 UML диаграмма таблиц базы данных

UML Диаграмма взаимодействия таблиц в базе данных изображена на

рисунке 2.7

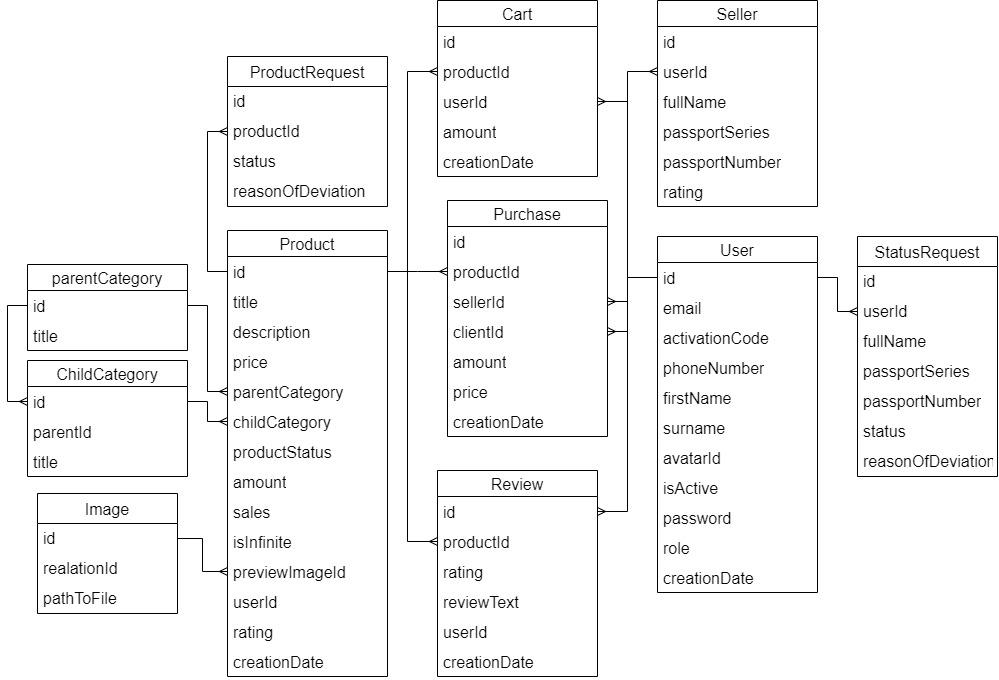


Рисунок 2.7 - UML диаграмма таблиц базы данных

2.4 Выводы по главе проектирование

В данной главе была разработана функциональная схема проектируемой информационной системы. Были показаны блок схемы следующих алгоритмов:

* авторизация;
* регистрация;
* создание товара;
* покупка товара;
* создание отчета.

Для работы с базой данных была выбрана СУБД MySQL. Также была представлена UML диаграмма взаимодействия таблиц в базе данных, были описаны основные модели, использующиеся в системе:

* User;
* Seller;
* Product;
* Review;
* StatusRequest;
* Cart;
* Purchase.

Полученных в ходе работы данных достаточно для дальнейшей разработки системы.

3 Реализация информационной системы

В данной главе будут рассмотрены технологии, использованные в разработке. Будут продемонстрированы страницы сайта и описан основной функционал для разных пользователей.

* 1. Выбор инструментария для разработки
     1. Серверная часть

Выбор языка программирования - один из самых важных шагов в разработке программного обеспечения. Каждый язык имеет свои преимущества и недостатки, и выбор зависит от конкретных задач и потребностей проекта.

Одним из наиболее популярных языков программирования является Java [1]. Java является объектно-ориентированным языком

программирования, который широко используется для создания веб­приложений, мобильных приложений, игр и других программных продуктов.

Одной из самых популярных библиотек для создания веб-приложений на Java является Spring [2]. Spring - это фреймворк для разработки Java- приложений, который предоставляет множество инструментов и функций для упрощения процесса разработки.

Одной из главных возможностей Spring является создание веб­приложений. Spring предоставляет множество инструментов для создания веб-приложений, включая Spring MVC [3], Spring Boot [4] и Spring Cloud. Spring MVC - это фреймворк для создания веб-приложений, которая предоставляет множество инструментов для управления запросами и

ответами. Spring Boot - это фреймворк для создания автономных приложений на Java, который позволяет создавать приложения с

минимальными настройками. Spring Cloud - это фреймворк для создания 44

микро-сервисов на Java, который предоставляет множество инструментов для управления микро-сервисами.

Spring также предоставляет множество инструментов для работы с базами данных, включая Spring Data и Spring JDBC.

Для работы с базой данных используется JPA (Java Persistence API) [5] Repository - это механизм, который позволяет упростить работу с базами данных в приложениях на Java. Spring предоставляет удобный механизм внедрения зависимостей, который позволяет автоматически создавать экземпляры репозиториев и использовать их в коде приложения. Благодаря этому, разработчик может сосредоточиться на бизнес-логике приложения, не тратя время на написание рутинного кода для работы с базами данных. JPA Repository также обеспечивает удобный интерфейс для выполнения CRUD- операций (Create, Read, Update, Delete) с сущностями базы данных, что позволяет сократить количество кода и упростить его поддержку. Кроме того, использование JPA Repository позволяет легко переключаться между различными базами данных, так как Spring поддерживает множество провайдеров JPA.

Также в процессе разработки использовалась библиотека Lombok [6], которая использовалась для сокращения объема шаблонного кода.

Веб-приложение Java Spring - это система, состоящая из трех основных компонентов: контроллера, сервиса и репозитория. Контроллер отвечает за обработку запросов от клиента и передачу данных сервису. Сервис, в свою очередь, обрабатывает запросы, используя данные из репозитория, и возвращает результаты обратно контроллеру. Репозиторий отвечает за хранение и управление данными.

Взаимодействие между компонентами происходит следующим образом: контроллер получает HTTP запрос [7] от клиента и передает его в сервис. Сервис использует репозиторий для получения необходимых данных и обрабатывает запрос. Затем сервис возвращает результаты обратно в контроллер, который формирует ответ и отправляет его клиенту.

Контроллер, сервис и репозиторий являются независимыми компонентами, что позволяет легко изменять их функциональность без влияния на другие компоненты системы. Кроме того, использование Java Spring позволяет легко интегрировать различные библиотеки и фреймворки, что упрощает разработку и поддержку веб-приложения.

* + 1. Клиентская часть

Для разработки клиентской части приложения используются HTML [8], CSS [9] и JavaScript [10]. HTML (HyperText Markup Language) - это язык разметки, который используется для создания структуры веб-страницы. CSS (Cascading Style Sheets) это язык описания внешнего вида веб-страницы, который позволяет разработчикам создавать красивые и современные дизайны. JavaScript - это язык программирования, который используется для создания интерактивных элементов на веб-страницах.

HTML и CSS используются для создания структуры и внешнего вида веб-страницы, а JavaScript используется для создания интерактивных элементов, таких как кнопки, формы и другие элементы, которые пользователь может взаимодействовать с ними.

Одним из главных преимуществ использования HTML, CSS и JavaScript является то, что они являются стандартными технологиями для веб-разработки. Это означает, что они поддерживаются всеми современными браузерами и могут быть использованы на любой платформе.

Помимо перечисленных инструментов будет использоваться Bootstrap [11]. Bootstrap - это бесплатный фреймворк для разработки веб-сайтов и приложений. Он предоставляет набор готовых компонентов и стилей, которые позволяют быстро и легко создавать красивые и адаптивные интерфейсы. Bootstrap использует HTML, CSS и JavaScript, и поддерживает множество браузеров и устройств, включая мобильные устройства. Он также имеет множество расширений и плагинов, которые позволяют расширять его функциональность и улучшать производительность.

* + 1. Проектирования интерфейса пользователя

Перед тем как приступать к разработке клиентской части, нужно примерное понимание как должна выглядеть система, какие в ней будут страницы, стиль приложения.

Веб-интерфейс приложения - это одна или несколько веб-страниц, это пользовательский интерфейс, для взаимодействия с сервисом.

Схема интерфейса пользователя-покупателя показана на рисунке 3.1:



Рисунок 3.1- Схема интерфейса пользователя

Перейдя на главную, страницу пользователь может посмотреть список всех товаров, воспользовавшись фильтром и сортировками. Далее может посмотреть интересующий товар и посмотреть профиль продавца.

Также пользователь может перейти к себе в профиль. Если он не авторизован, то он переведется на страницу авторизации. Авторизованный пользователь может посмотреть свою корзину, а также перейти на страницу создания запроса на то, чтобы стать продавцом.

Если пользователь является продавцом, то он может посмотреть его товары, причем он видит все свои товары, даже те, которые недоступны к покупке по той или иной причине. Также он может вызвать меню создания нового товара.

Если пользователь является администратором, то ему доступны 3 страницы, недоступные трем другим пользователям:

* страница всех пользователей, где он может просматривать информацию о всех пользователях, менять их роли, блокировать их;
* страница заявок на смену статуса;
* страница заявок на создание нового товара.

3.2 Описание интерфейса

На рисунке 3.2 изображена главная страница.

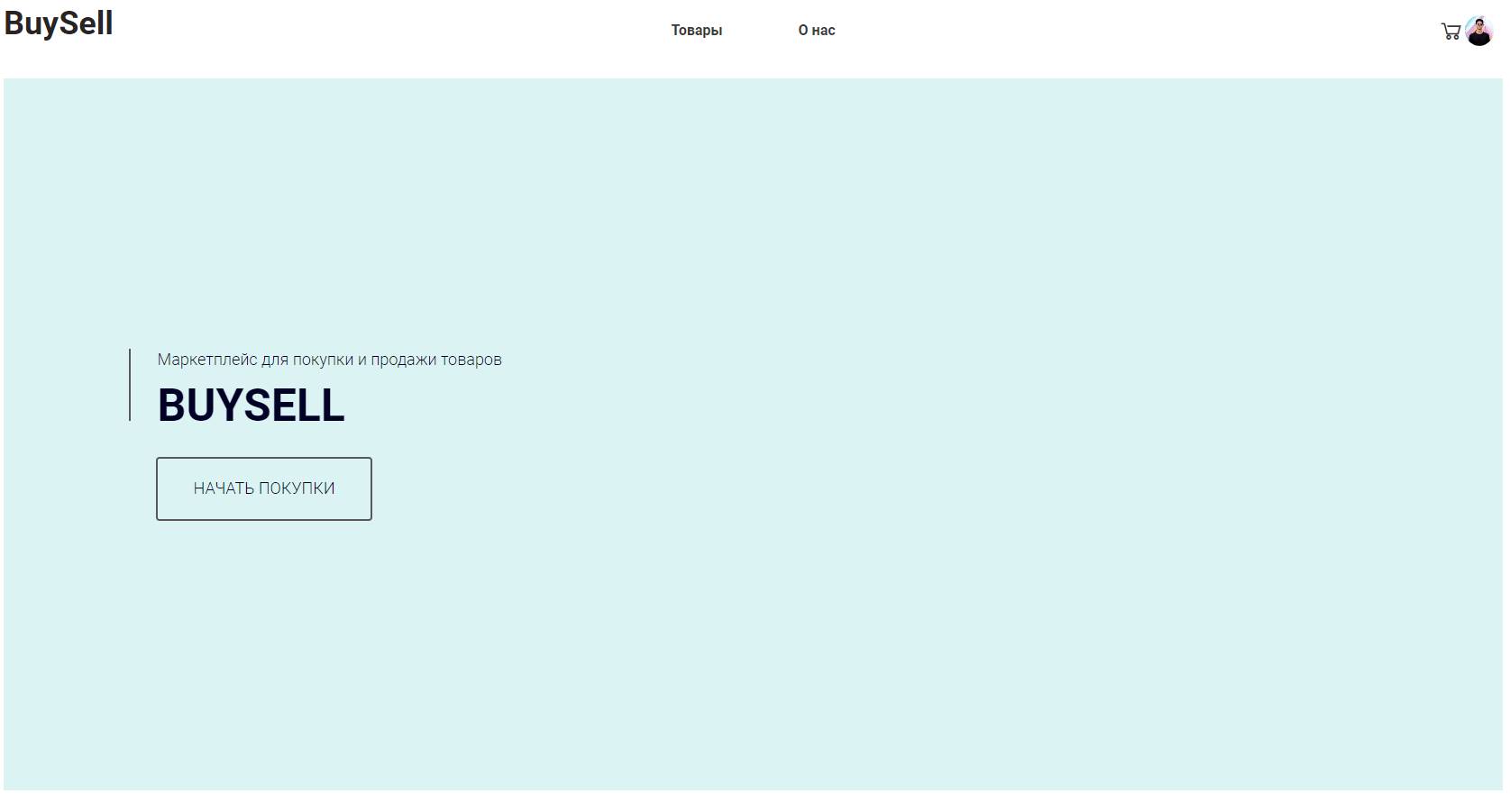
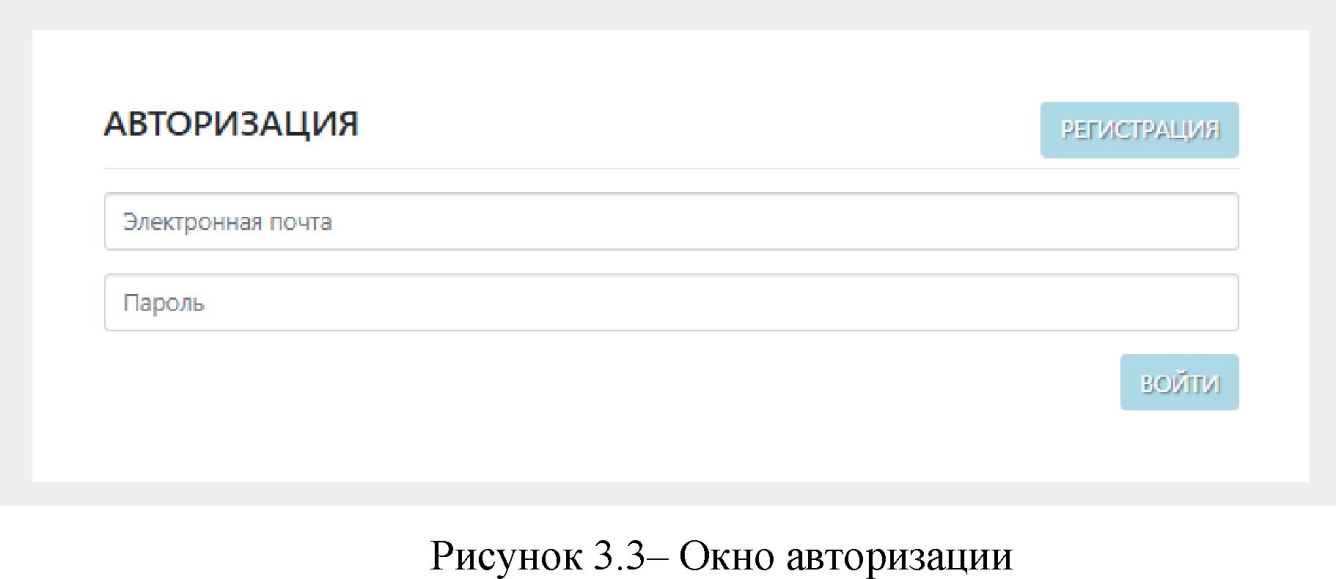


Рисунок 3.2- Главная страница сайта

Навигационная панель представляет собой ссылки для перехода между страницами. Панель закреплена сверху и отображается на всех страницах. На панели присутствует логотип, который переводит пользователя на главную страницу. Также есть отдельный пункт товары, переводящий на страницу с товарами. В правой части есть аватар пользователя, нажав на который пользователя переместится в свой профиль. Также рядом с аватаром есть иконка корзины, нажав на которую пользователь перенесется в свою корзину.

Одной из главных функций веб приложений, является авторизация. В данном приложении неавторизованный пользователь обладает только ограниченным функционалом. Он может просматривать витрину товаров, просматривать страницу отдельного товара, заходить в профили продавцов. Такой пользователь не может ни выкладывать товары, ни покупать, ни пользоваться корзиной. Для доступа к этим функция нужна авторизация.

На рисунке 3.3 представлено окно авторизации.



На случай, если пользователь не зарегистрирован в системе, в правом верхнем углу окна есть кнопка регистрации, открывающая поля для регистрации.

На рисунке 3.4 представлена окно регистрации.

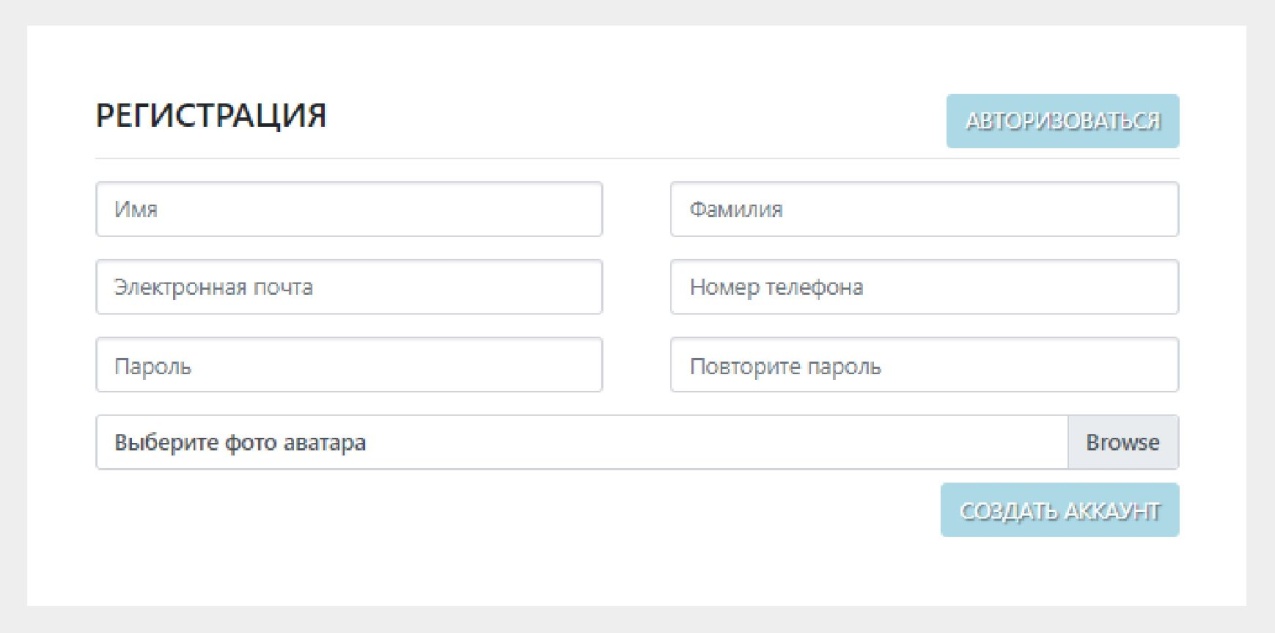


Рисунок 3.4 - Окно регистрации

После того как пользователь вводит свои данные, нажимает кнопку регистрации, ему на электронную почту приходит письмо с подтверждением адреса его электронной почты. Если пользователь зарегистрировался и не подтвердил свой электронный ящик, то он не сможет авторизоваться.

После того как пользователь зарегистрировался и авторизовался он может приступить к покупкам. Для этого он может перейти на страницу товаров нажав на соответствующую ссылки на навигационной панели.

На рисунке 3.5 изображена страница товаров

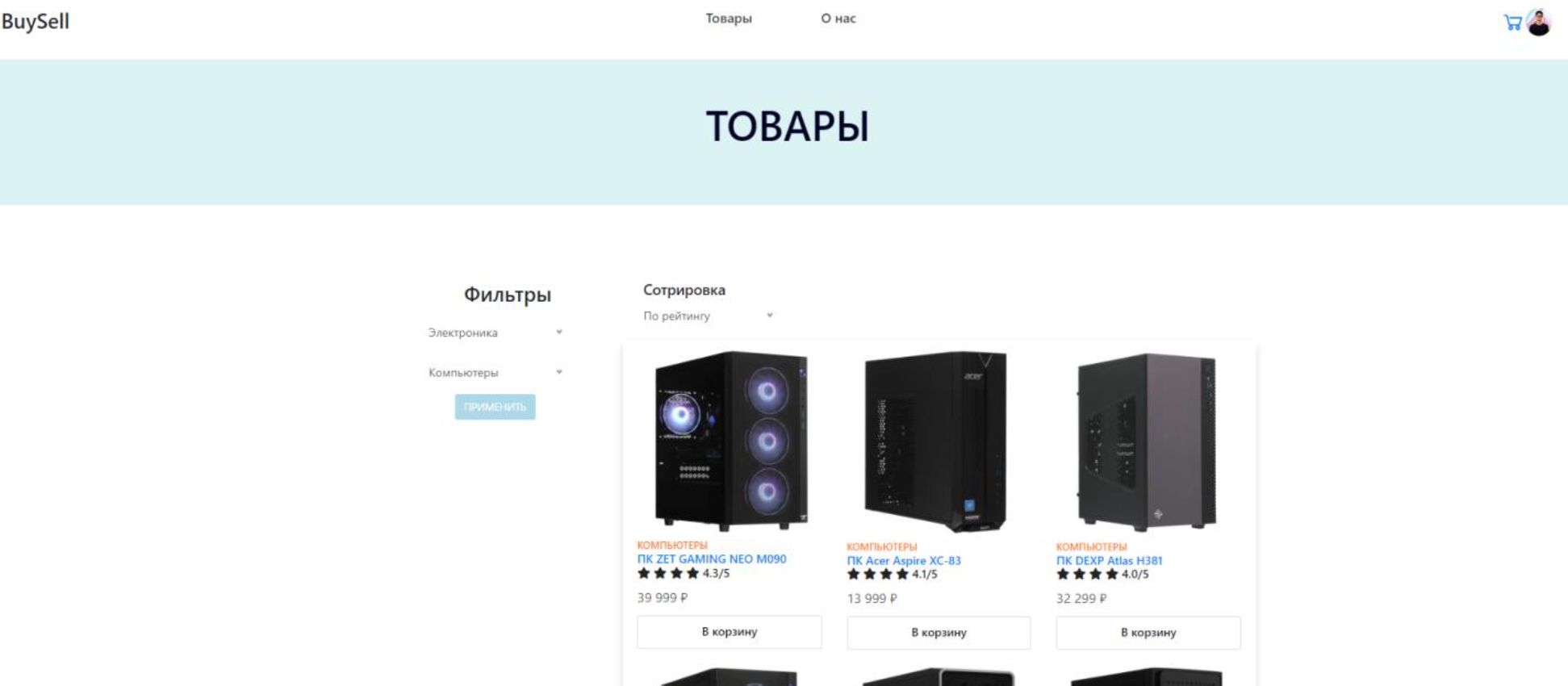


Рисунок 3.5 - Страница товаров

Перед пользователем отображаются все доступные для покупки товары. Он может воспользоваться сортировкой или фильтрами чтобы ему было легче выбрать нужный ему продукт.

На сайте присутствуют следующие сортировки:

* по цене по возрастанию;
* по цене по убыванию;
* по рейтингу;
* по заказам.

По умолчанию товары сортируются по рейтингу.

Фильтрация в приложении выполнена так, что у товара есть основной тип, например электроника или одежда, и подтип, дочерний от выбранного основного типа, для электроники, например, это смартфоны и компьютеры. При создании товара продавец должен указать хотя бы один, главный тип, чтобы товар можно было отнести в отдельную группу. Выбор подкатегории необязателен при создании товара, но продавец, как правило, заинтересован в указании подкатегории, так как, если у товара не будет данного поля, то при фильтрации он будет отображаться только при выборе основного типа.

Для функции фильтрации была применена Ajax технология [12]. Ajax (Asynchronous JavaScript and XML) - это технология, которая позволяет обновлять содержимое веб-страницы без перезагрузки всей страницы. Это достигается путем отправки запросов на сервер с помощью JavaScript и получения ответов в формате XML [13] или JSON [14].

С точки зрения системы, Ajax запросы позволяют уменьшить нагрузку на сервер, так как только часть страницы обновляется, а не вся страница. Это также позволяет ускорить загрузку страницы, так как только необходимые данные загружаются.

С точки зрения пользователя, Ajax запросы позволяют создавать более интерактивные и быстрые веб-приложения. Например, при отправке формы, пользователь может увидеть сообщение об успешной отправке без перезагрузки всей страницы.

Когда пользователь выбирает главный тип в фильтрах, срабатывает событие которые формирует Ajax запрос. Этот запрос поступает на Rest- контроллер, который передаёт данные в сервис, где идет обращение к репозиторию. В качестве ответа контроллер возвращает JSON объект подтипов выбранной категории. Эти данные обрабатываются на стороне клиента. Поступившая информация обрабатывается и на ее основе заполняется список подтипов фильтра. После этого пользователь уже может выбрать конкретный тип товара.

В приложении используется пагинация - постраничное отображение информации. Пагинация позволяет пользователям легко перемещаться по страницам и быстро находить нужную информацию.

Реализацию пагинации можно реализовать двумя способами - на стороне клиента, и на стороне сервера. При реализации на стороне клиента - все объекты отправляются в представление и с использованием JavaScript реализовать постранично отображение. То есть каждый раз будет показываться часть товаров, по очереди. Этот метод бы позволил отображать товары во время фильтрации, чтобы пользователь мог выбрать нужный фильтр и сегмент с товарами сразу обновлялся, используя Ajax запросы, это было бы очень быстро для пользователя. Но при увеличении количества продавцов и, соответственно, товаров на сайте, пришлось бы при каждой загрузке товаров обрабатывать тысячи объектов, и все на стороне клиента, это было бы уже не так производительно и поэтому было принято решение о реализации пагинации на стороне сервера.

Постраничное отображение было реализовано на стороне сервера при помощи библиотеки Spring. При первой загрузке для пользователя загружается только 12 товаров и используются сортировка (по идентификатору товара) и фильтры (по рейтингу) по умолчанию. При переключении между страницами, происходит перезагрузка страницы и загрузятся следующие 12 объектов для отображения.

Пользователь выбрал интересующий его товар и перешел на его страницу для детального изучения. На рисунке 3.6 изображена страница товара.

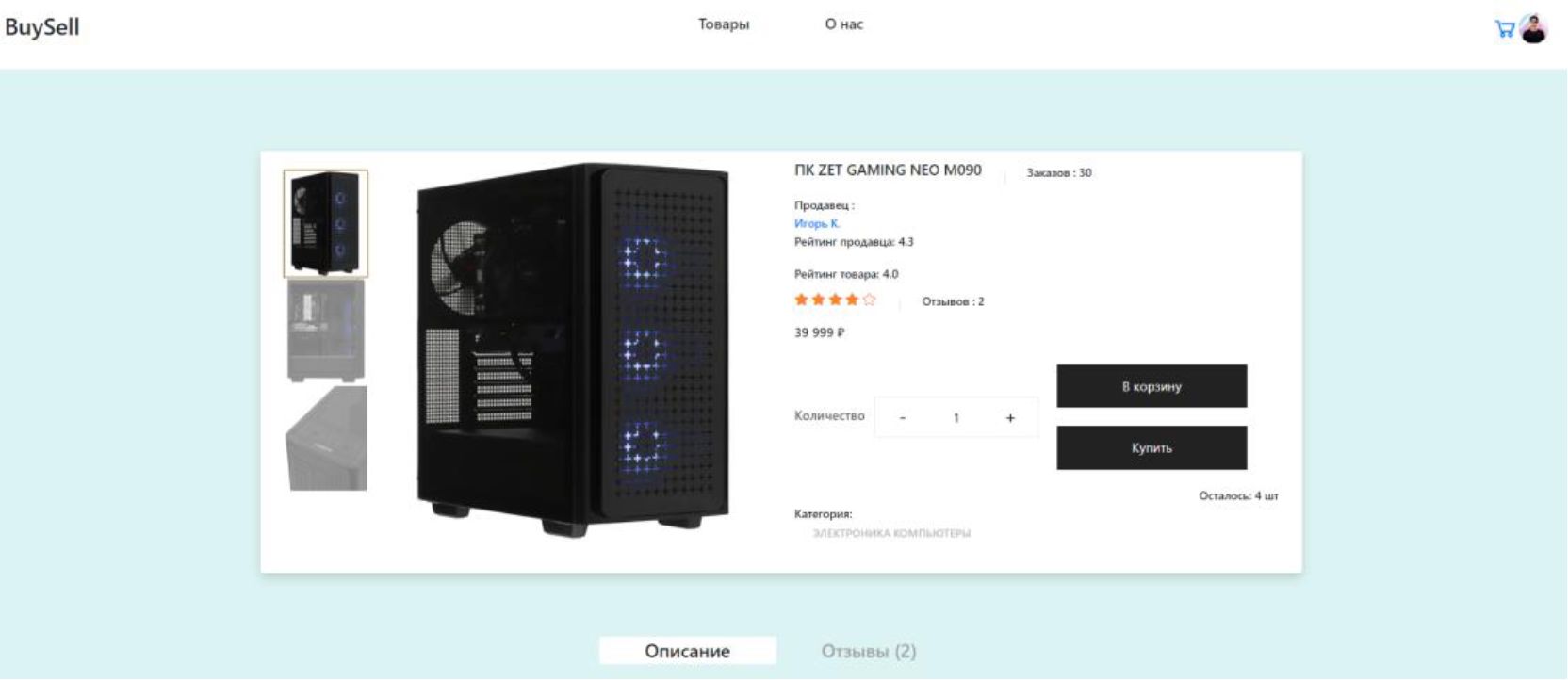


Рисунок 3.6 - Страница товара

На странице товара можно найти его описание, количество заказов, рейтинг, количество отзывов. Также на странице товара можно добавить товар в корзину и выбрать желаемое количество товаров. Помимо перечисленного со страницы товара можно попасть в профиль продавца, где можно посмотреть какие он продает товары.

Каждый зарегистрированный пользователь может оставить отзыв о товаре, перейдя в соответствующий раздел. На рисунке 3.7 изображена раздел страница товара с отзывами

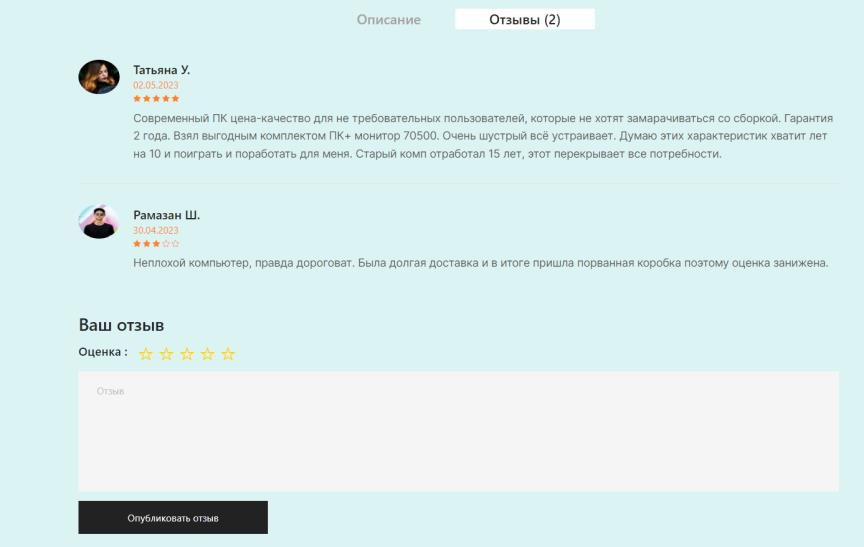


Рисунок 3.7 - Отзывы пользователей о товаре  
53

В разделе отзывов видны все отзывы покупателей, их оценки, на их основе можно сформировать решение о покупке товара. Если товар покупателю понравился или, наоборот, не понравился, то он может поставить ему оценку, написать отзыв. Отзыв будет отображаться данном разделе, вместе с оценкой. Оценка товара влияет не только на рейтинг товара, но и на рейтинг самого продавца. Рейтинг товара - это среднее значение всех поставленных оценок. Рейтинг продавца - это среднее значение рейтингов его товаров.

После того как пользователь выбрал нужные товары и добавил их в корзину, он может их купить, перейдя на страницу корзины. Интерфейс корзины пользователя изображен на рисунке 3.8.

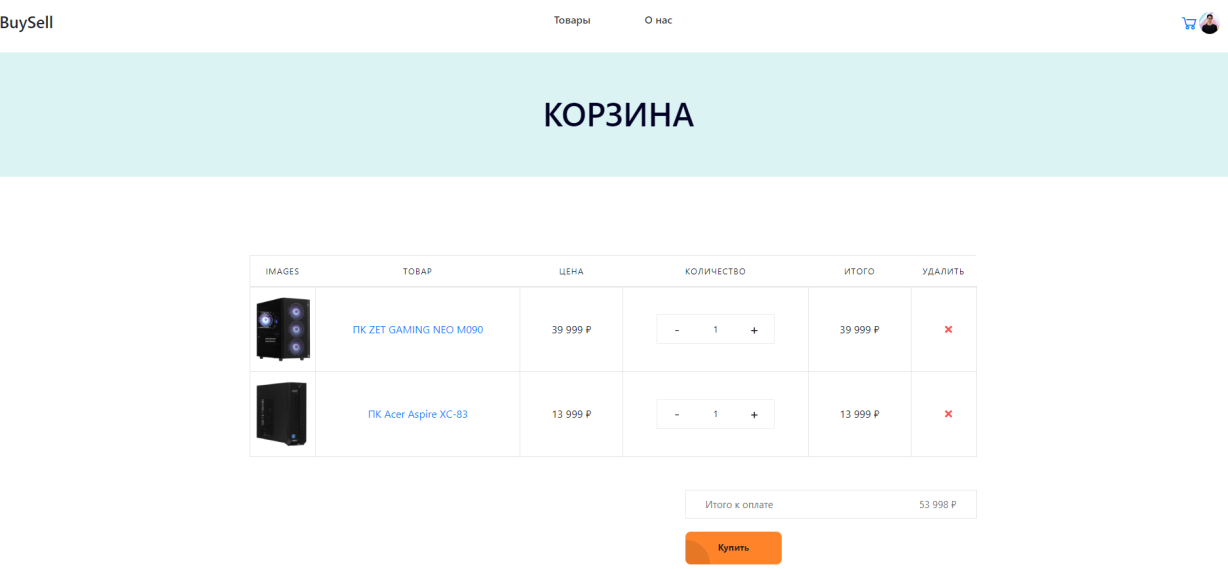


Рисунок 3.8 - Страница корзины пользователя

На странице корзины можно управлять количество товаров, удалять их. После того как пользователь нажмет кнопку купить, и операция пройдет успешно пользователю на электронную почту придет письмо, с чеком по его заказу.

Были рассмотрены возможности покупателя внутри сервиса. Теперь стоит рассмотреть, что может делать продавец в приложении.

Каждый пользователь после того, как зарегистрируется, по умолчанию является покупателем. Чтобы стать продавцом он должен перейти в свой 54

профиль и нажать на соответствующую кнопку. Профиль пользователя изображен на рисунке 3.9.

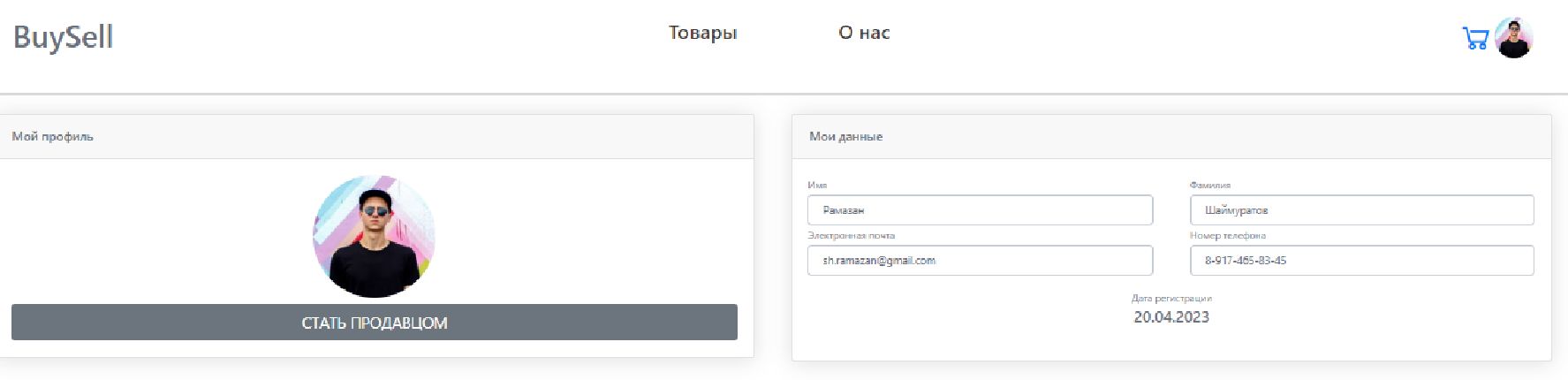


Рисунок 3.9 - Страница профиля пользователя

Далее будет отображена небольшая форма, где пользователь будет указывать дополнительные данные о себе. Далее данную заявку будет рассматривать администратор и после того, как будет принято решение, о результате пользователь будет уведомлен по электронной почте. Если пользователю будет отказано, то в сообщении будет причина отказа. Пользователь, в случае отказа, может подать заявку до 5 раз.

Если пользователь станет продавцом, то ему станет доступны новые функции. Например, он сможет создавать свои товары с помощью конструктора. Товары будут отображаться в его профиле. На рисунке 3.10 представлен профиль продавца.

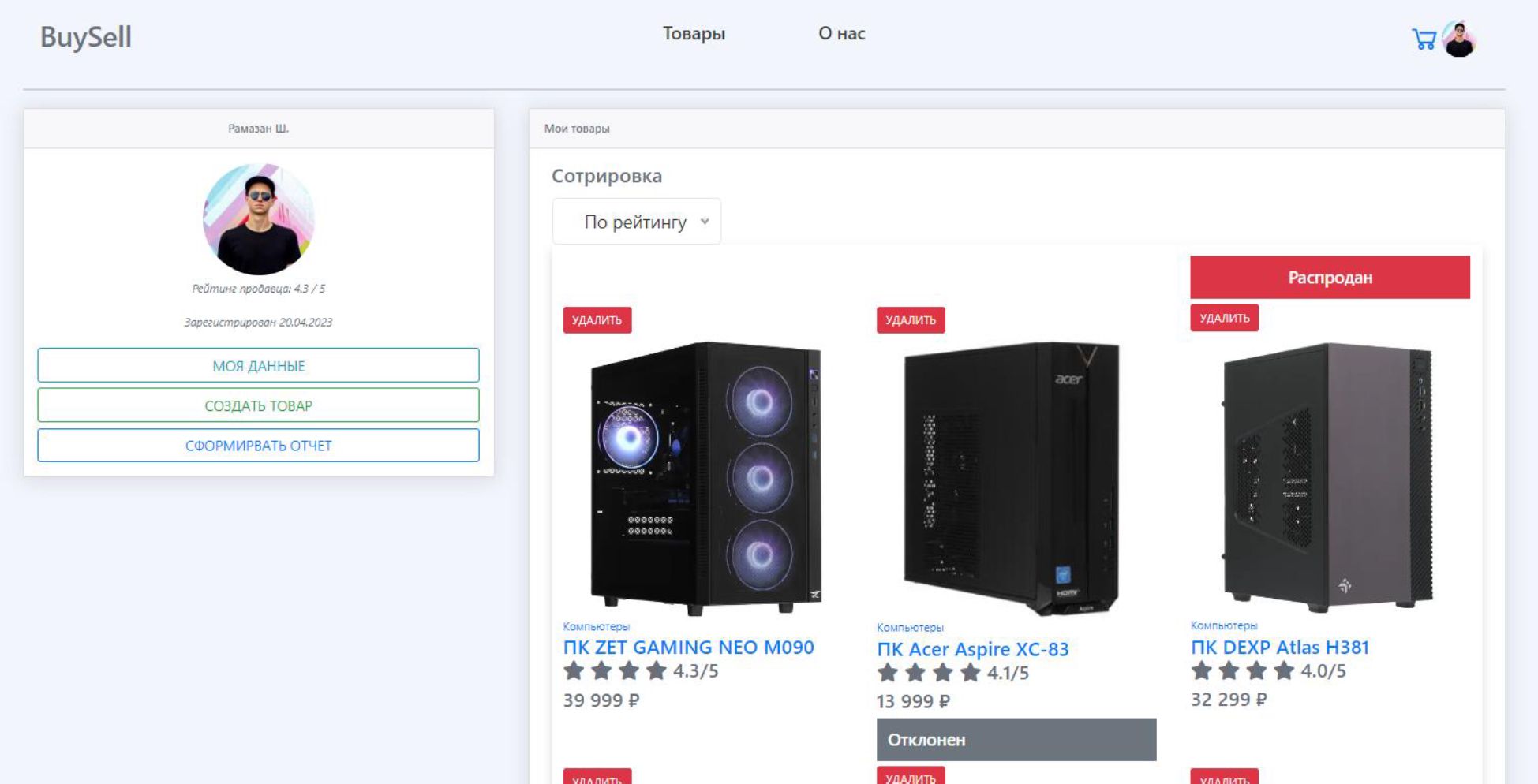


Рисунок 3.10 - Страница товаров продавца

Когда продавец просматривает свои товары он видит их все, и активные, и неактивные, неактивные товары отмечены соответствующим надписью. Существующие товары он может редактировать и удалять. При изменении статуса товара, на почту продавца придет соответствующее сообщение.

Покупатель при переходе на профиль продавца видит только те товары, которые можно купить, активные.

Также для продавца доступны отчеты по продажам и отзывам, а excel. Работа с таблицами была реализована с помощью библиотеки Apache POI [15]. Apache POI представляет собой API, который позволяет использовать файлы MS Office в Java приложениях. Библиотека включает в себя классы и методы для чтения и записи информации в документы MS Office. Он может выбрать период, за который ему нужен отчет. Далее сформируются отчет по продажам на основе таблицы Purchase, которая хранит информацию о покупках, и отчет по отзывам покупателей на основе таблицы Review. Сформируется Excel файл, содержащий две книги: отчет по продажам и отчет по отзывам. После того как файл будет готов его можно скачать. На рисунках 3.11, 3.12 изображены отчеты по продажами и отзывам.

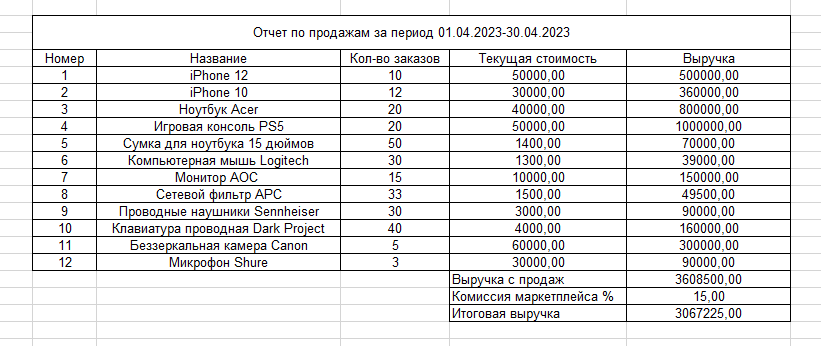


Рисунок 3.11 - Скриншот таблицы отчета по продажам в Excel

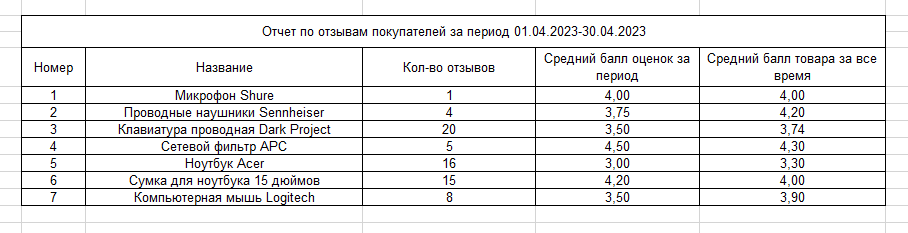


Рисунок 3.12 - Скриншот таблицы отчета по отзывам в Excel

Отчет по продажам позволяет выявить самые продаваемые товары и товары, пользующиеся низкой популярностью. Таблица отзывов служит для выявления проблем в самой продукции, то есть понять какие товары, по мнению покупателей, являются плохими, и, возможно, принять меры по устранению проблем с этими товарами, чтобы конечный покупатель остался доволен.

Возможности покупателе и пользователей разобраны, осталось описать функции администратора. В задачи администратора входит:

* обработка заявок на смену статуса продавца;
* обработка заявок на создание товара;

- управление пользователями, если, например, пользователь

нарушил правила площадки то администратор может его заблокировать.

Если пользователь блокируется, то в его аккаунт становится неактивным и он не может авторизоваться под своими данными и получает ошибку входа. На рисунке 3.13 представлена таблица пользователей, которую видит администратор если зайдет на данную страницу.

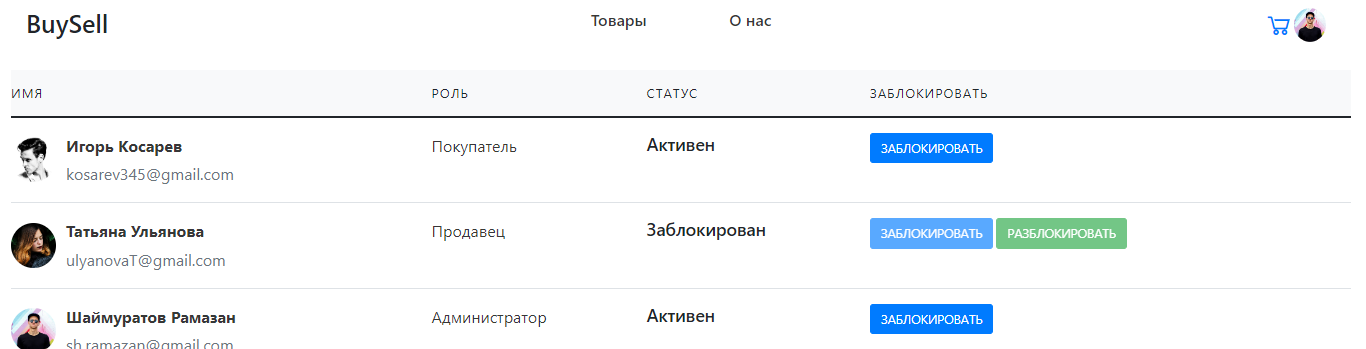


Рисунок 3.13 - Таблица пользователей

В таблице видны имена пользователей, их почты, роли и статус. Также нажав на профиль, можно детально изучить профили людей.

3.3 Выводы по главе

В ходе работы была разработана архитектура системы, которая представляет собой веб-приложение, построенное с использованием фреймворка Spring.

Было произведено проектирование интерфейса пользователя. Клиентская часть была построена с использованием HTML+CSS, а также языка программирования JavaScript. В ходе работы был описан функционал покупателя, продавца и администратора внутри сервиса, были показаны следующие страницы сайта:

* главная страница;
* страница входа;
* страница товаров;
* страница конкретного товара;
* страница корзины пользователя;
* страница профиля пользователя;
* панель администратора.

Также был показан пример отчетов, которые может сформировать продавец на основе его продаж.

Заключение

В результате выполнения выпускной работы было разработано веб­приложение маркетплейса реализованного на языке Java Spring. Приложение обладает функциями регистрации, авторизации, создания/покупки товаров, добавление товаров в корзину. Также система позволяет пользователям становится продавцами. Сервис предоставляет возможность пользователям оставлять отзывы на товары и формировать рейтинг товара. Формируя рейтинг товаров, формируется рейтинг продавца. Для продавцов доступна возможность получения отчетов за желаемый период времени, который содержит в себе информацию о продажах и информацию об оценках покупателей.

В первой главе был проведен анализ бизнес-процессов. Был проведен анализ бизнес-требований. Была выявлена и описана проблема компании - повышение прибыли с продаж товаров. В ходе анализа на разработку маркетплейса, выявлены стейкхолдеры и выявлены сценарии использования системы. На основе сценариев использования были сформированы функциональные и нефункциональные требования к системе.

Во второй главе была представлена функциональная модель системы, описывающая основные функции системы, входные данные и результат. Были детально описаны алгоритмы регистрации, авторизации, создания товара, покупки товара. В главе также была описаны основные модели в системе, описаны их функции и их взаимодействие между собой. В конце главы была представлена диаграмма все таблиц в базе данных и их связи.

В третье главе были описана архитектура информационной системы, перечислены использующие инструменты разработки. Для серверной части использовался язык Java, фреймворк Spring. Клиентская часть была разработана с использованием HTML+CSS и языка JavaScript. В главе также продемонстрированы основные окна интерфейса приложения, описаны примененные при разработке технологии, изложены функции, доступные пользователям с разными ролями.

Таким образом при выполнении работы были решены все поставленные задачи. Решение представляет собой маркетплейс, где пользователи могут продавать и покупать товары. Внедрение системы позволить компании «UBuySell» увеличить прибыль с продажи собственных товаров внутри сервиса и получать дополнительную прибыль за счет комиссии с продаж других продавцов сайта.

Список использованных источников

1. Г. Шилдт Java. Полное руководство, издатель: Диалектика, 12-е издание - 2022г.
2. Официальная документация по Spring Framework [Электронный ресурс] // Documentation: [сайт] URL:

<https://docs.spring.io/spring-framework/reference/>(дата обращения: 10.02.2023)

1. Хабр - сообщество IT-специалистов [Электронный ресурс] //Spring MVC - основные принципы: [сайт] URL:

<https://habr.com/ru/articles/336816/>(дата обращения: 10.02.2023)

1. Документация по Spring Boot [Электронный ресурс] // Documentation: [сайт] URL:<http://spring-projects.ru/projects/spring-boot/>(дата обращения: 10.02.2023)
2. Официальная документация по Java [Электронный ресурс] // Documentation: [сайт] URL:

<https://docs.oracle.com/javaee/6/tutorial/doc/bnbpz.html/>(дата обращения:

10.02.2023)

1. Хабр - сообщество IT-специалистов [Электронный ресурс] //Lombok - полное руководство: [сайт] URL:

<https://habr.com/ru/companies/piter/articles/676394/>(дата обращения:

10.02.2023)

1. Википедия - свободная энциклопедия [Электронный ресурс] // HTTP: [сайт] URL: [https: //ru. wikipedia.org/wiki/HTTP/](https://ru.wikipedia.org/wiki/HTTP) (дата обращения: 20.10.2022)
2. Документация по HTML [Электронный ресурс] // Documentation: [сайт] URL: [https: //html .spec. whatwg.org/multipage/](https://html.spec.whatwg.org/multipage/) (дата обращения:

10.02.2023)

1. Документация по CSS [Электронный ресурс] // Documentation: [сайт] URL:<https://www.w3schools.com/cssref/>(дата обращения: 10.02.2023)
2. Сайт о программировании Metanit [Электронный ресурс] // JavaScript: [сайт] URL: [https://metanit.eom/web/j avascript/ 1.2.php/](https://metanit.com/web/javascript/1.2.php/) (дата

обращения: 10.02.2023)

1. Официальная документация по Bootstrap [Электронный ресурс] // Documentation: [сайт] URL: [https://getbootstrap.com/docs/4.1/getting-](https://getbootstrap.com/docs/4.1/getting-started/introduction/)

[started/introduction/](https://getbootstrap.com/docs/4.1/getting-started/introduction/) (дата обращения: 10.02.2023)

1. Документация о AJAX технологии [Электронный ресурс] // Documentation: [сайт] URL:

<https://developer.mozilla.org/ru/docs/Web/Guide/AJAX/>(дата обращения:

10.02.2023)

1. Википедия - свободная энциклопедия [Электронный ресурс] // XML: [сайт] URL: [https: //ru.wikipedia.org/wiki/XML/](https://ru.wikipedia.org/wiki/XML/) (дата обращения:

20.10.2022)

1. Википедия - свободная энциклопедия [Электронный ресурс] // JSON: [сайт] URL: [https: //ru.wikipedia.org/wiki/JSON/](https://ru.wikipedia.org/wiki/JSON/) (дата обращения:

20.10.2022)

1. Официальная документация по библиотеке Apache POI [Электронный ресурс] // Documentation: [сайт] URL:<https://poi.apache.org/> (дата обращения: 10.02.2023)

Приложение А

Листинг класса конфигурации SecurityConfig.

@EnableWebSecurity

@RequiredArgsConstructor

@EnableGlobalMethodSecurity(prePostEnabled = true)

public class SecurityConfig extends WebSecurityConfigurerAdapter {

private final CustomUserDetailsService userDetailsService;

@Override

protected void configure(HttpSecurity http) throws Exception {

http

.authorizeRequests()

.antMatchers("/", "/activate/\*" ,"/product/\*\*", "/productImages/\*\*",

"/requestImages/\*\*", "/avatarImages/\*\*", "/registration", "/user/\*\*", "/static/\*\*", "/filter/\*\*", "/load/\*\*")

.permitAll()

.anyRequest().authenticated()

.and()

.formLogin()

.loginPage("/login")

.permitAll()

.and()

.logout()

.permitAll();

}

@Override

protected void configure(AuthenticationManagerBuilder auth) throws Exception {

auth.userDetailsService(userDetailsService)

.passwordEncoder(passwordEncoder());

}

@Bean

public PasswordEncoder passwordEncoder() {

return new BCryptPasswordEncoder(8);

}

}

Приложение Б

Листинг класса ProductService

@Service

@Slf4j

@RequiredArgsConstructor

public class ProductService {

Pageable sortByPriceDec = PageRequest.of(0, 3, Sort.by("price").descending());

Pageable sortByPriceInc = PageRequest.of(0, 3, Sort.by("price").ascending());

Pageable sortBySalesDec = PageRequest.of(0, 5, Sort.by("sales").descending());

Pageable sortByRatingDec = PageRequest.of(0, 5, Sort.by("rating").descending());

private final ProductRepository productRepository;

private final UserRepository userRepository;

private final PurchaseRepository purchaseRepository;

private final CartRepository cartRepository;

private final ProductImageRepository productImageRepository;

private final ProductRequestRepository productRequestRepository;

private final ReviewRepository reviewRepository;

public Page<Product> getFilteredProducts(String title, Long parentId, Long childId, Pageable pageable, String sortValue) {

//if (title != null) return productRepository.findByTitleAndProductStatus(title,

ProductStatus.ACTIVE);\

if (sortValue != null) {

pageable = processPageable(pageable, sortValue);

}

if (childId == null) {

if (parentId == null || parentId == -1) {

return productRepository.findByProductStatus(ProductStatus.ACTIVE, pageable);

} else {

return productRepository.findByParentCategoryAndProductStatus(parentId,

ProductStatus.ACTIVE, pageable);

}

} else {

return

productRepository.findByParentCategoryAndChildCategoryAndProductStatus(parentId, childId,

ProductStatus.ACTIVE, pageable);

}

}

private Pageable processPageable(Pageable pageable, String sortValue) {

Pageable newPageable;

switch (sortValue){

case PRICE\_DECREASING : {

newPageable = PageRequest.of(pageable.getPageNumber(), pageable.getPageSize(), sortByPriceDec.getSort());

}

break;

case PRICE\_INCREASING : {

newPageable = PageRequest.of(pageable.getPageNumber(), pageable.getPageSize(), sortByPriceInc.getSort());

}

break;

case SALES\_DECREASING : {

newPageable = PageRequest.of(pageable.getPageNumber(), pageable.getPageSize(), sortBySalesDec.getSort());

}

break;

default: {

newPageable = pageable;

}

break;

}

return newPageable;

}

public void saveProduct(Product product) { productRepository.save(product);

}

public void saveProduct(Principal principal, Product product, boolean isInfinite, MultipartFile file1, MultipartFile file2, MultipartFile file3, Integer childCategory) throws IOException {

User user = getUserByPrincipal(principal);

product.setUserId(user.getId());

ProductImage image1 = null;

ProductImage image2 = null;

ProductImage image3 = null;

if (file1.getSize() != 0) {

image1 = toImageEntity(file1);

image1.setPreviewImage(true);

}

if (file2.getSize() != 0) {

image2 = toImageEntity(file2);

}

if (file3.getSize() != 0) {

image3 = toImageEntity(file3);

}

product.setInfinite(!isInfinite);

product.setProductStatus(ProductStatus.ACTIVE);

if (childCategory != null) {

product.setChildCategory(childCategory);

}

ProductRequest productRequest = new ProductRequest();

log.info("Saving new Product. Title: {}; Author email: {}", product.getTitle(), user.getEmail());

Product productFromDb = productRepository.save(product);

image1.setProductId(productFromDb.getId());

ProductImage previewImage = productImageRepository.save(image1);

if (image2 != null) {

image2.setProductId(productFromDb.getId());

productImageRepository.save(image2);

}

if (image3 != null) {

image3.setProductId(productFromDb.getId());

productImageRepository.save(image3);

}

productRequest.setProductId(productFromDb.getId());

productRequest.setStatus(RequestState.ACCEPTED);

productRequest.setReason(ProductRequest.NO\_REASON);

productRequestRepository.save(productRequest);

productFromDb.setPreviewImageId(previewImage.getId());

productRepository.save(productFromDb);

}

public User getUserByPrincipal(Principal principal) {

if (principal == null) return new User();

return userRepository.findByEmail(principal.getName());

}

public ProductImage toImageEntity(MultipartFile file) throws IOException {

ProductImage image = new ProductImage();

image.setName(file.getName());

image.setOriginalFileName(file.getOriginalFilename());

image.setContentType(file.getContentType());

image.setSize(file.getSize());

image.setBytes(file.getBytes());

return image;

}

public void deleteProduct(User user, Long id) {

Product product = productRepository.findById(id).orElse(null);

if (product != null) {

long userid = userRepository.findById(product.getUserId()).getId();

if (userid == user.getId()) {

product.setProductStatus(ProductStatus.DELETED);

productRepository.save(product);

log.info("Product with id = {} was deleted", id);

} else {

log.error("User: {} haven't this product with id = {}", user.getEmail(), id);

}

} else {

log.error("Product with id = {} is not found", id);

}

}

public Product getProductById(Long id) {

return productRepository.findById(id).orElse(null);

}

public void registerPurchase(Principal principal, long productId) throws IOException { Product product = productRepository.getById(productId);

Purchase purchase = new Purchase();

purchase.setProductId(productId);

purchase.setPrice(product.getPrice());

purchase.setAmount(123);

purchase.setClientId(getUserByPrincipal(principal).getId());

purchase.setSellerId(product.getUserId());

product.addSalesValue(123);

if (!product.isInfinite()) {

product.calculateNewAmount(purchase.getAmount());

}

Calendar start = Calendar.getInstance();

start.set(Calendar.DAY\_OF\_MONTH, 1);

start.set(Calendar.MONTH, 3);

Calendar end = Calendar.getInstance();

end.set(Calendar.MONTH, 4);

end.set(Calendar.DAY\_OF\_MONTH, end.getActualMaximum(Calendar.DAY\_OF\_MONTH));

List<Purchase> purchases =

purchaseRepository.findAllBySellerIdAndDateOfPurchaseBetween(product.getUserId(), start, end);

List<Product> products = productRepository.findByUserId(product.getUserId());

SalesReportController.createSalesReport(purchases, products);

productRepository.save(product);

purchase.setReturned(false);

log.info("User with id: {} bought product with id: {}, {} pcs.", purchase.getClientId(), purchase.getProductId(), purchase.getAmount());

purchaseRepository.save(purchase);

}

public void addProductToCart(Principal principal, long id) {

Cart cart = new Cart();

cart.setProductId(id);

cart.setAmount(1);

cart.setUserId(getUserByPrincipal(principal).getId());

cartRepository.save(cart);

}

public Page<Product> getCreatedProducts(Pageable pageable) {

return productRepository.findByProductStatus(ProductStatus.VALIDATING, pageable);

}

public void saveChanges(Product product) { productRepository.save(product);

}

public Page<Product> findExistingProducts(Long id, Pageable pageable) {

return productRepository.findByUserIdAndProductStatusNot(id, ProductStatus.DELETED, pageable);

}

public List<Product> findByIdInAndProductStatus(List<Cart> cartList, ProductStatus productStatus) {

return

productRepository.findByIdInAndProductStatus(cartList.stream().map(Cart::getProductId).collect(Collec tors.toList()), productStatus);

}

public Map<Review, String> getReviewToAuthorNameMap(Long productId) {

return reviewRepository.findByProductId(productId).stream()

.collect(Collectors.toMap(review -> review, review ->

userRepository.findById(review.getUserId()).getShortName(), (a, b) -> b));

}

public List<Review> getAllProductReviews(Long productId) { return reviewRepository.findByProductId(productId);

}

public void saveReview(Review review) { reviewRepository.save(review);

}

public long getIdFromUserOrDefault(Principal principal) {

if (principal == null) { return -1;

} else {

return getUserByPrincipal(principal).getId();

}

}