**Реферат**

ФИО

Подпись

Дата

Лист

1

ДП 00.00.ПЗ

Разраб.

*Корень М.А.*

Пров.

Комкова А.В.

Н. контр.

Нистюк О.А.

Утв.

Блинова Е.А.

Реферат

Лит.

Листов

1

БГТУ 1-40 05 01, 2025

Пояснительная записка дипломного проекта содержит 79 страниц пояснительной записки, 13 таблицы, 0 формул, 55 иллюстрации, 8 источников литературы, 6 приложений.

Веб-приложение, python, django, javascript, REACT, postgresql, docker

Целью дипломного проекта является создание веб-приложения для проведения онлайн-аукционов, которое объединяет удобство торговой площадки и функциональность для взаимодействия пользователей.

В первой главе проводится обзор литературы и постановка задачи по тематике дипломного проекта.

Вторая глава посвящена обзору средств разработки, а также проектирование системы и содержит описание технологий, использованных во время создания проекта, а также необходимые процессы.

В третьей главе описывается процесс разработки, принципы функционирования и назначение созданных компонент проекта.

В четвертой главе описывается контрольный пример, с проведением тестирования, и показывается поведения системы в различных ситуациях.

В пятой главе описана методика установки и руководство пользователя, позволяющее подробно понять интерфейс программного средства.

В шестой главе приводится расчет экономических параметров и себестоимости программного средства.

В заключении подводятся итоги проделанной работы и оцениваются достигнутые результаты в целом.

Объем графической части дипломного проекта составляет 0 листа А3.

У

**Abstract**

ФИО

Подпись

Дата

Лист

1

ДП 00.00.ПЗ

Разраб.

*Корень М.А.*

Пров.

Комкова А.В.

Н. контр.

Нистюк О.А.

Утв.

Блинова Е.А.

Abstract

Лит.

Листов

1

БГТУ 1-40 05 01, 2025

Explanatory note of the diploma project 79 pages of explanatory note, 13 tables, 0 formulas, 55 illustrations, 8 sources of literature, 6 appendices.

WEB APPLICATION, python, django, javascript, REACT, postgresql, docker.

The aim of the diploma project is to create a web application for conducting online auctions, combining the convenience of a trading platform with functionality for user interaction.

The first chapter provides a literature review and defines the project objectives.

The second chapter is dedicated to an overview of development tools and system design, including a description of the technologies used during the project creation, as well as the necessary processes.

The third chapter describes the development process, principles of operation, and the purpose of the created project components.

The fourth chapter presents a test case with system testing and demonstrates the system’s behavior in various situations.

The fifth chapter covers the installation procedure and user manual, allowing for a detailed understanding of the software interface.

The sixth chapter provides detailed calculations of the economic parameters and overall cost of the software product.

The conclusion summarizes the completed work and thoroughly evaluates the overall results achieved.

The volume of the graphic part of the diploma project is 0 sheets A3.

У

Оглавление

ФИО

Подпись

Дата

Лист

1

ДП 00.00.ПЗ

Разраб.

*Корень М.А.*

Пров.

Комкова А.В.

Н. контр.

Нистюк О.А.

Утв.

Блинова Е.А.

Оглавление

Лит.

Листов

2

БГТУ 1-40 05 01, 2025

[Введение 5](#_Toc198276981)

[1 Обзор аналогов и постановка задачи 6](#_Toc198276982)

[1.1 Обзор аналогов 6](#_Toc198276983)

[1.1.1 E-auction 6](#_Toc198276984)

[1.1.2 Elot 8](#_Toc198276985)

[1.1.3 Torgi 9](#_Toc198276986)

[1.2 Постановка задачи 10](#_Toc198276987)

[1.3 Выводы по разделу 10](#_Toc198276988)

[2 Проектирование веб-приложения 11](#_Toc198276989)

[2.1 Описание средств разработки 11](#_Toc198276990)

[2.1.1 Обоснование выбора среды разработки 11](#_Toc198276991)

[2.1.2 Фреймворки и библиотеки 12](#_Toc198276992)

[2.1.3 Инструменты для клиентской части 12](#_Toc198276993)

[2.1.4 Инструменты для серверной части 12](#_Toc198276994)

[2.1.5 Хранилища данных 13](#_Toc198276995)

[2.1.6 Архитектура взаимодействия внутри приложения 13](#_Toc198276996)

[2.2 Проектирование базы данных 14](#_Toc198276997)

[2.2.1 Диаграмма базы данных 14](#_Toc198276998)

[2.2.2 Описание таблиц базы данных 16](#_Toc198276999)

[2.3 Диаграмма вариантов использования 19](#_Toc198277000)

[2.4 Проектирование контейнеризации 20](#_Toc198277001)

[2.5 Проектирование основных алгоритмов 20](#_Toc198277002)

[2.5.1 Алгоритм создания аукциона 21](#_Toc198277003)

[2.5.2 Алгоритм проведения аукциона 22](#_Toc198277004)

[2.6 Выводы по разделу 23](#_Toc198277005)

[3 Разработка веб-приложения 24](#_Toc198277006)

[3.1 Общая структура 24](#_Toc198277007)

[3.2 Разработка серверной части веб-приложения 24](#_Toc198277008)

[3.2.1 Реализация процесса регистрации 29](#_Toc198277009)

[3.2.2 Реализация процесса авторизации 30](#_Toc198277010)

[3.2.3 Реализация процесса поиска аукциона 31](#_Toc198277011)

[3.2.4 Реализация процесса создания аукционов 32](#_Toc198277012)

[3.2.5 Реализация процесса чата 33](#_Toc198277013)

[3.3 Реализация клиентской части веб-приложения 34](#_Toc198277014)

[3.4 Контейнеризация 38](#_Toc198277015)

[3.5 Выводы по разделу 39](#_Toc198277016)

[4 Тестирование веб-приложения 41](#_Toc198277017)

[4.1 Тестирование авторизации и регистрации 41](#_Toc198277018)

[4.2 Тестирование создания аукциона 44](#_Toc198277019)

[4.3 Тестирование создания аукциона 46](#_Toc198277020)

[4.4 Тестирование чата покупателя и продавца 48](#_Toc198277021)

[4.5 Выводы по разделу 48](#_Toc198277022)

[5 Руководство пользователя 49](#_Toc198277023)

[5.1 Руководство для пользователя 49](#_Toc198277024)

[5.1.1 Руководство для покупателя 49](#_Toc198277025)

[5.1.2 Руководство для продавца 54](#_Toc198277026)

[5.1.3 Руководство для администратора 58](#_Toc198277027)

[5.2 Руководство по развертыванию приложения 60](#_Toc198277028)

[5.3 Выводы по разделу 62](#_Toc198277029)

[6 Технико-экономическое обоснование проекта 64](#_Toc198277030)

[6.1 Общая характеристика разрабатываемого программного средства 64](#_Toc198277031)

[6.2 Исходные данные для проведения расчетов и маркетинговый анализ 65](#_Toc198277032)

[6.3 Обоснование цены программного средства 65](#_Toc198277033)

[6.3.1 Расчет затрат рабочего времени на разработку 66](#_Toc198277034)

[6.3.2 Расчет основной заработной платы 67](#_Toc198277035)

[6.3.3 Расчет дополнительной заработной платы 67](#_Toc198277036)

[6.3.4 Отчисления в Фонд социальной защиты населения 68](#_Toc198277037)

[6.3.5 Расчет суммы прочих прямых затрат 68](#_Toc198277038)

[6.3.6 Расчет суммы накладных расходов 69](#_Toc198277039)

[6.3.7 Сумма расходов на разработку 69](#_Toc198277040)

[6.3.8 Определение цены, оценка эффективности 70](#_Toc198277041)

[6.4 Выводы по разделу 71](#_Toc198277042)

[Заключение 83](#_Toc198277043)

[Список использованной литературы 73](#_Toc198277044)

[ПРИЛОЖЕНИЕ А. Трехуровневая архитектура веб-приложения 74](#_Toc198277045)

[ПРИЛОЖЕНИЕ Б. Логическая схема базы данных 75](#_Toc198277046)

[ПРИЛОЖЕНИЕ В. Диаграмма вариантов использования 76](#_Toc198277047)

[ПРИЛОЖЕНИЕ Г. Процесс создания аукциона 77](#_Toc198277048)

[ПРИЛОЖЕНИЕ Д. Процесс проведения аукциона 78](#_Toc198277049)

[ПРИЛОЖЕНИЕ Е. Результаты подсчетов экономического обоснования 79](#_Toc198277050)

Введение

ФИО

Подпись

Дата

Лист

1

ДП 00.00.ПЗ

Разраб.

*Корень М.А.*

Пров.

Комкова А.В.

Н. контр.

Нистюк О.А.

Утв.

Блинова Е.А.

Введение

Лит.

Листов

1

БГТУ 1-40 05 01, 2025

В современном мире цифровых технологий электронная коммерция и онлайн-торги набирают популярность, привлекая миллионы пользователей. Удобство, безопасность и взаимодействие в реальном времени делают такие платформы ключевыми для аукционов и торговли. Однако их успешная реализация требует глубокого анализа, продуманной разработки и использования современных технологий.

Целью дипломного проекта является создание веб-приложения для проведения онлайн-аукционов, которое объединяет удобство торговой площадки и функциональность для взаимодействия пользователей. Приложение позволит пользователям участвовать в торгах, делать ставки, отслеживать результаты и взаимодействовать с другими участниками.

Для достижения цели необходимо решить следующие задачи:

* провести анализ существующих аналогов, определив их сильные и слабые стороны для дальнейшего совершенствования проекта;
* спроектировать архитектуру веб-приложения, при этом учесть все требования, связанные с масштабируемостью и безопасностью системы;
* реализовать веб-приложение, обеспечив его удобство использования как для обычных пользователей, так и для администраторов платформы;
* провести тестирование системы, чтобы убедиться в ее полной функциональности и надежности работы в различных условиях;
* подготовить документацию и рекомендации по эксплуатации веб-приложения для любого пользователя.

В результате будет создана платформа, обеспечивающая удобство для участников торгов и эффективные инструменты для администраторов, поддерживающих управление и безопасность системы.

У

1 Обзор аналогов и постановка задачи

ФИО

Подпись

Дата

Лист

1

ДП 01.00.ПЗ

Разраб.

*Корень М.А.*

Пров.

Комкова А.В.

Н. контр.

Нистюк О.А.

Утв.

Блинова Е.А.

1 Обзор аналогов

и постановка задачи

Лит.

Листов

5

БГТУ 1-40 05 01, 2025

1.1 Обзор аналогов

В условиях стремительного развития электронной коммерции и онлайн-торгов платформы для проведения аукционов приобретают особую значимость, обеспечивая удобство и доступность для пользователей. Создание такого рода систем требует глубокого изучения существующих решений и четкого определения целей разработки.

Анализ существующих на рынке аналогов является важным этапом разработки любого веб-приложения. Это позволяет выявить сильные и слабые стороны текущих решений, определить нужды пользователей, а также найти возможности для улучшения и внедрения уникальных функций.

На рынке уже существуют платформы для аукционов, однако многие из них не предоставляют пользователям достаточного функционала, как, например, возможность удобного взаимодействия между ролями, глубокий анализ статистики или поддержка гибкой настройки категорий товаров. Это создает пространство для разработки нового решения, которое может эффективно устранить эти ограничения.

В данном разделе будут рассмотрены приложения-аналоги для проведения онлайн-аукционов и будут выявлены их достоинства и недостатки.

1.1.1 E-auction

E-auction [1] – это платформа, специализирующаяся на продаже арестованного и конфискованного имущества, включая бытовую технику, транспорт, и мебель. Это официальный аукцион, организованный Республиканским унитарным предприятием «БелЮрОбеспечение».

Официальным организатором аукциона выступает Республиканское унитарное предприятие «БелЮрОбеспечение», которое обеспечивает надежность и законность всех этапов проведения торгов. Каждый участник может быть уверен, что процедура покупки осуществляется в строгом соответствии с законодательством. Благодаря E-auction [1], арестованное имущество находит новых владельцев, а средства от продажи идут на выполнение государственных обязательств или другие предусмотренные законом нужды.

Платформа E-auction [1] помогает арестованному и конфискованному имуществу быстро находить новых владельцев. Средства, вырученные в ходе аукциона, направляются на выполнение различных государственных обязательств или используются для других целей, предусмотренных законодательством. Пример интерфейса данного сервиса представлен на рисунке 1.1.

У

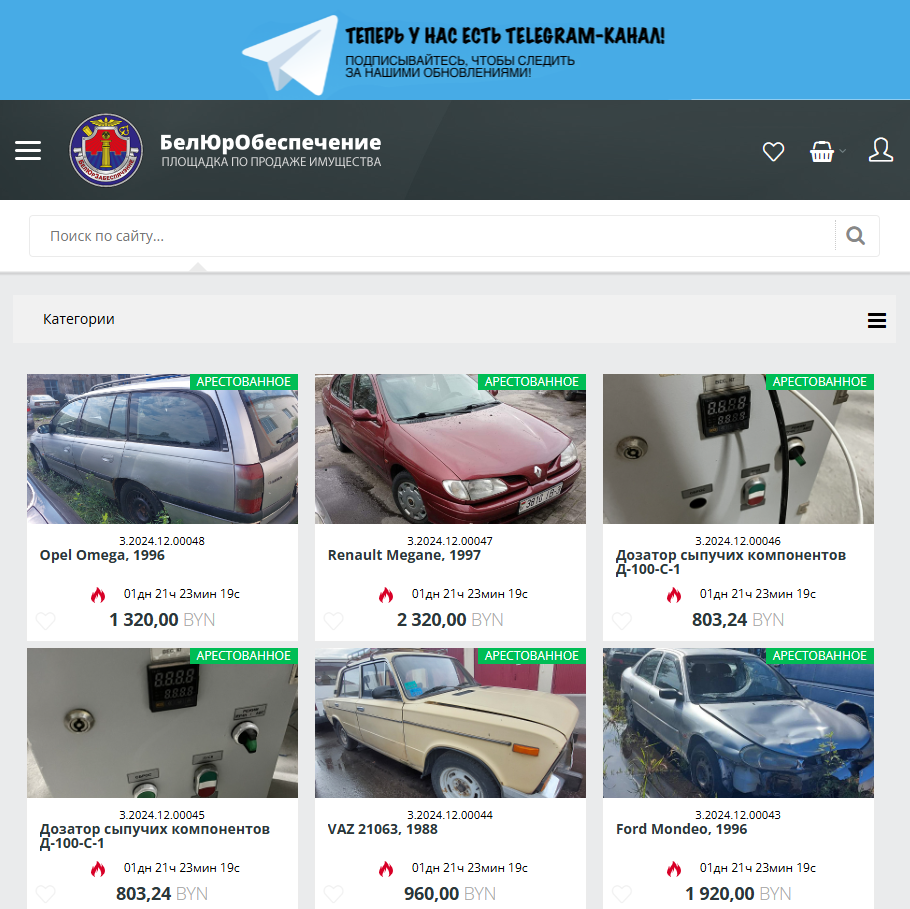


Рисунок 1.1 – Интернет-аукцион E-auction

Основной функционал E-auction заключается в организации электронных торгов, предоставляющих возможность приобретения конфискованного, арестованного или иного имущества. Пользователи получают доступ к большому количеству лотов, среди которых автомобили, недвижимость, бытовая техника и мебель. Это делает сервис подходящей для различных покупателей.

Для удобства поиска доступны фильтры по категориям, цене, состоянию товара и местоположению. Каждое предложение сопровождается детальным описанием, фотографиями и условиями участия, что позволяет участникам более тщательно оценить объект перед подачей ставок.

На данной платформе можно выделить следующие преимущества:

* широкий выбор лотов в различных категориях;
* наличие фильтров по категориям, цене и местоположению;
* детальные описания лотов с фотографиями и истории ставок;
* функции отслеживания текущих торгов и истории ставок;
* интеграция с государственными системами для прозрачности.

Но также присутствуют и такие недостатки как:

* отсутствие мобильного приложения, что снижает удобство доступа;
* сложный процесс регистрации из-за гарантийного взноса.

Платформа предоставляет обширные возможности для проведения торгов и гарантирует их прозрачность, однако ограничения в доступности и сложная процедура регистрации значительно снижают удобство использования.

1.1.2 Elot

Elot [2] – это торговая площадка, предоставляющая возможность покупки и продажи разнообразных товаров, включая предметы искусства, одежду и оборудование. Платформа подходит как для физических, так и для юридических лиц что расширяет потенциальную клиентскую базу.

Пример интерфейса данного сервиса представлен на рисунке 1.2.

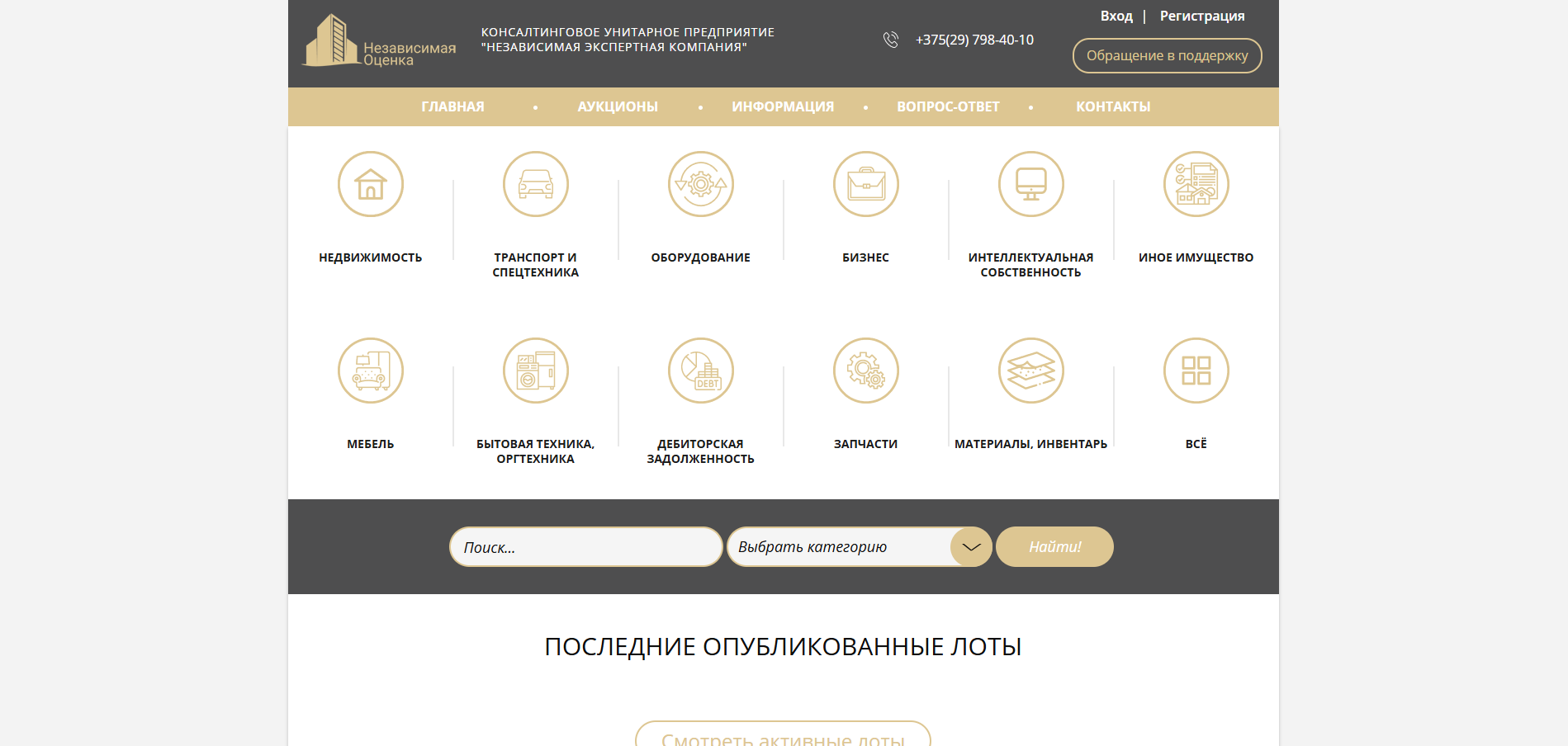


Рисунок 1.2 – Интернет-аукцион Elot

Elot предоставляет удобный инструмент для проведения торгов по продаже и аренде различного имущества, включая объекты недвижимости, транспортные средства и промышленное оборудование. Платформа рассчитана как на частных пользователей, так и на корпоративных клиентов, обеспечивая широкий спектр возможностей для участия в аукционах.

Сервис включает функции поиска и фильтрации объектов по ключевым запросам, таким как тип, стоимость, регион и статус лота. Пользователи могут ознакомиться с детальной информацией по каждому предложению, включая технические характеристики, фотографии и условия участия в торгах.

Для участия в аукционе требуется регистрация и выполнение определенных условий, таких как подача заявки и внесение гарантийного депозита.

Elot также предоставляет пользователям возможность отслеживать активные аукционы, участвовать в торгах в реальном времени и просматривать завершенные сделки. Прозрачность и надежность процессов гарантируются за счет интеграции с государственными системами и строгого соблюдения нормативных требований. Это делает Elot надежным решением для проведения торгов и поиска имущества на ресурсе.

Ключевые достоинства платформы включают:

* удобный дизайн для пользователя;
* возможность отслеживать аукционов в реальном времени.

Однако имеются и определенные ограничения:

* ограниченная аналитика для пользователей;
* отсутствие функций взаимодействия между участниками.

Elot эффективно справляется с организацией торгов, обеспечивая надежность и прозрачность, но для повышения удобства требует расширения аналитических возможностей и инструментов взаимодействия пользователей.

1.1.3 Torgi

Torgi [3] – это электронная площадка для аукционов, где можно приобрести имущество, находящееся в государственной собственности, а также арестованное или конфискованное имущество. Этот ресурс предоставляет доступ к широкому спектру предложений в различных категориях.

Пример интерфейса данного сервиса представлен на рисунке 1.3.

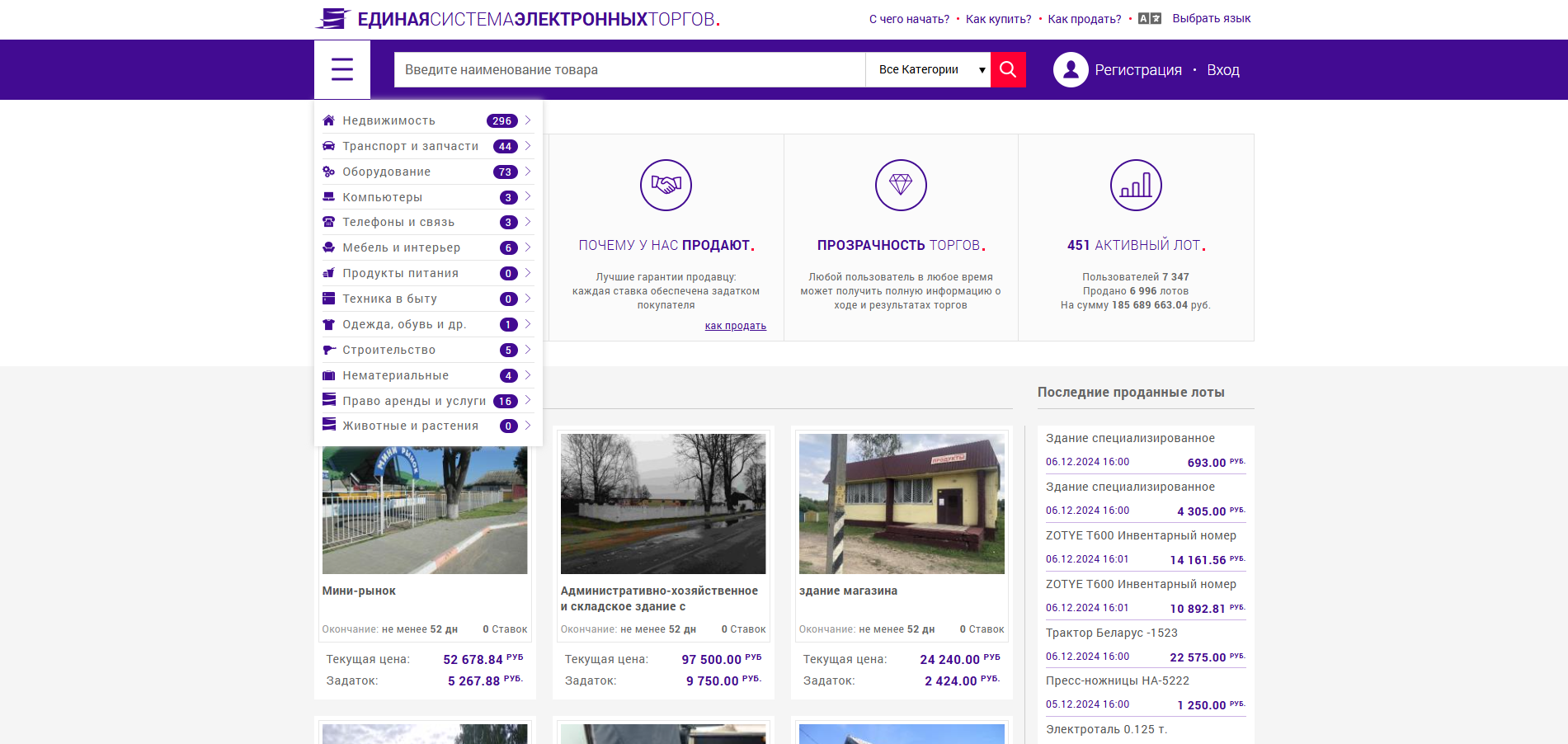


Рисунок 1.3 – Интернет-аукцион Torgi

Сервис предлагает просмотр лотов в категориях недвижимости, транспорта, оборудования и земельных участков с фильтрами по типу имущества, региону и параметрам, а также поиском по ключевым словам. Лоты содержат подробные описания, условия участия и документы.

Для участия необходима регистрация, подача заявки и, при необходимости, внесение депозита. Платформа поддерживает отслеживание торгов в реальном времени, подачу ставок и уведомления о результатах. Torgiпредоставляет аналитику и отчетность по завершенным торгам, а интеграция с государственными реестрами гарантирует безопасность сделок. Интерфейс удобен, стабилен и адаптирован для больших объемов данных.

Основные преимущества платформы, следующие:

* поддержка отчетности для организаторов торгов;
* поиск по ключевым словам.

При этом существуют некоторые недостатки:

* сложный интерфейс для новых пользователей;
* отсутствие персонализированных рекомендаций по лотам.

Torgi эффективно организует торги, обеспечивая надежность и прозрачность, но для улучшения пользовательского опыта требуется упрощение интерфейса и добавление персонализированных функций.

1.2 Постановка задачи

На основе анализа аналогов были определены ключевые функции, которые должны быть реализованы в разрабатываемом веб-приложении для онлайн-аукционов. Важно, чтобы каждая роль в системе имела доступ к соответствующим инструментам для эффективного выполнения своих задач. Рассмотрим функциональные возможности для каждой из ролей в системе.

Для администратора:

* управление категориями (создание, удаление, изменение, назначение);
* блокировка и разблокировка пользователей;
* просмотр статистики сайта (количество пользователей веб-приложения, количество заблокированных и активных пользователей, количество завершенных/незавершенных аукционов, статистика ставок пользователей).

Для продавца:

* управление товарами для аукционов (создание, изменение, удаление);
* формирование крупных аукционов и аукционов одного товара;
* просмотр информации о ставках на свои товары;
* поддержка чата с покупателем.

Для покупателя:

* просмотр крупных аукционов (запланированных, текущих и завершенных) с общей информацией об аукционе и каждом товаре;
* просмотр аукционов одного товара и детальной информации о нем;
* добавление аукционов в избранное;
* участие в торгах, делая свои ставки;
* просмотр истории ставок на товары (выиграл/проиграл);
* поддержка чата с продавцом выигранного товара;
* выставление оценки продавцу;
* статистика выигранных и проигранных аукционов.

Весь этот функционал является необходимым для обеспечения удобства использования и корректной работы веб-приложения, что в свою очередь способствует улучшению опыта пользователей и повышению эффективности системы.

1.3 Выводы по разделу

В результате анализа аналогичных веб-приложений, таких как E-auction, Elot и Torgi, были определены ключевые функциональные возможности для реализации проекта. Основное внимание уделяется регистрации, авторизации, созданию и редактированию профилей пользователей, а также поиску, фильтрации и сортировке аукционов. Проект предусматривает разделение ролей пользователей: администратор, продавец и покупатель.

Особое внимание будет уделено интуитивно понятному интерфейсу, который обеспечит удобство использования для пользователей. Чат для взаимодействия между пользователями будет способствовать улучшению коммуникации.

2 Проектирование веб-приложения

ФИО

Подпись

Дата

Лист

1

ДП 02.00.ПЗ

Разраб.

*Корень М.А.*

Пров.

Комкова А.В.

Н. контр.

Нистюк О.А.

Утв.

Блинова Е.А.

2 Проектирование

веб-приложения

Лит.

Листов

14

БГТУ 1-40 05 01, 2025

2.1 Описание средств разработки

В данном разделе будут рассмотрены различные библиотеки, технологии и подходы, которые будут использованы для создания веб-приложения. Выбор инструментов обосновывается их функциональными возможностями, удобством и соответствием требованиям проекта.

2.1.1 Обоснование выбора среды разработки

Среды разработки (IDE – Integrated Development Environments) – это программные инструменты, которые помогают разработчикам создавать, отлаживать и поддерживать код. Они включают в себя редакторы кода, отладчики, средства для тестирования и другие инструменты, упрощающие процесс разработки. Существует два основных типа сред разработки:

* легковесные редакторы кода, которые подходят для различных языков программирования и дают гибкость за счет расширений;
* интегрированные среды разработки (IDE), такие как PyCharm, которые предоставляют встроенные инструменты для работы с определенными языками.

Visual Studio Code – это легкий, но мощный редактор кода, поддерживающий языки программирования, такие как JavaScript, TypeScript, Python и C++. Его гибкость обеспечивается благодаря обширному набору расширений, которые позволяют интегрироваться с различными фреймворками, например, React и NestJS, а также с базами данных и платформами. Visual Studio Code предоставляет инструменты для написания, тестирования и отладки кода, что делает его универсальным выбором как для фронтенд-, так и для бэкенд-разработки.

PyCharm – это мощная интегрированная среда разработки, специально ориентированная на язык Python. Она поддерживает функции отладки, тестирования, анализа кода и работы с базами данных. PyCharm идеально подходит для работы с фреймворками, такими как Django и Flask, предлагая встроенные инструменты для управления проектами и интеграции с системами контроля версий. Эта IDE становится особенно полезной при разработке серверной части сложных приложений со сложной архитектурой.

Visual Studio Code и PyCharm часто используются совместно для реализации различных задач в процессе разработки.

Visual Studio Code обеспечивает гибкость и широкий спектр возможностей для фронтенд-разработки, позволяя работать с фреймворками, такими как React*.*

У

PyCharm специализируется на Python [4], что делает его идеальным выбором для создания серверной логики и работы с фреймворками вроде Django.

Использование обеих сред разработки позволяет эффективно справляться как с задачами клиентской, так и серверной части веб-приложений, обеспечивая удобство и производительность в рабочем процессе

2.1.2 Фреймворки и библиотеки

Веб-фреймворки и библиотеки являются важными инструментами в разработке веб-приложений. Они помогают упростить и ускорить процесс создания, тестирования и сопровождения кода. Основное различие между фреймворками и библиотеками заключается в степени их контроля над процессом разработки:

* фреймворки предоставляют разработчику структуру и правила, диктуя архитектуру приложения и порядок работы. Они обеспечивают готовые решения для типичных задач, таких как маршрутизация, работа с базами данных и аутентификация, что особенно полезно для серверной разработки;
* библиотеки, напротив, предлагают набор инструментов, которые разработчик может использовать по своему усмотрению. Они не навязывают строгой архитектуры и подходят для выполнения специфических задач, например, создания пользовательского интерфейса.

2.1.3 Инструменты для клиентской части

Для реализации фронтенда запланировано использование библиотеки React [5], которая является одной из самых популярных технологий для разработки современных пользовательских интерфейсов. Она идеально подходит для создания одностраничных приложений, где важны высокая интерактивность, производительность и отзывчивость интерфейса.

Ключевой особенностью является виртуальный DOM (Document ObjectModel*)*, который ускоряет обновление интерфейса, минимизируя прямые изменения в браузере. Благодаря широкому набору инструментов React поддерживает динамичные и удобные интерфейсы, что улучшает пользовательский опыт.

2.1.4 Инструменты для серверной части

Для бэкенда планируется применять фреймворк Django на языке Python. Django содержит мощные инструменты для создания серверной логики, включая управление базами данных, обработку запросов и настройку аутентификации. Django ускоряет разработку, позволяя делать акцент на функционале.

Фреймворк отличается высокой надежностью и поддержкой масштабируемых решений, что делает его подходящим для сложных проектов. Встроенные механизмы безопасности и готовые модули упрощают реализацию типичных задач, таких как управление пользователями и формами.

2.1.5 Хранилища данных

Хранилища данных – это системы, предназначенные для хранения, обработки и управления данными. Они играют ключевую роль в архитектуре веб-приложений, предоставляя надежные механизмы для работы с данными. В современном программировании используются два основных типа хранилищ:

* NoSQL-хранилища, ориентированные на работу с неструктурированными или слабо структурированными данными, например, JSON-документами;
* Реляционная база данных – это мощная объектно-реляционная система управления базами данных (СУБД) с открытым исходным кодом. Она обеспечивает высокую производительность, поддерживает сложные запросы, транзакции, работу с различными типами данных и масштабирование.

Основные преимущества PostgreSQL:

* поддержка географических данных через расширение PostGIS, что делает его отличным выбором для приложений с геолокацией;
* возможность использования хранимых процедур и триггеров для автоматизации сложных операций в базе данных;
* надежная работа с высоконагруженными проектами, где требуется обработка больших объемов данных и внутренним взаимодействием.

PostgreSQL часто используется в паре с фреймворками, такими как Django, которые предоставляют встроенные инструменты для работы с базами данных.

Django использует ORM (Object-Relational Mapping) для упрощения взаимодействия с PostgreSQL, что позволяет разработчикам работать с базой данных через Python-код, не прибегая к написанию сложных SQL-запросов.

На стороне фронтенда React может интегрироваться с серверной частью (например, через API, разработанный на Django), чтобы предоставлять пользователю доступ к данным, хранящимся в PostgreSQL.

Такое сочетание инструментов обеспечивает надежность и производительность веб-приложений, а PostgreSQL выступает центральным элементом для управления данными в этом процессе.

2.1.6 Архитектура взаимодействия внутри приложения

После выбора инструментов разработки необходимо определить архитектурный стиль. Для реализации была выбрана трехуровневая структура, состоящая из клиентской части, логического уровня и слоя данных.

Трехуровневая архитектура широко применяется в современных веб-приложениях благодаря своей гибкости. Такая архитектура позволяет четко разделить ответственность между уровнями, что упрощает разработку, тестирование и масштабирование системы, что в свою очередь удобно для разработчика.

Среди ключевых преимуществ – модульность, которая облегчает обновление отдельных компонентов без влияния на другие, и высокая степень безопасности, так как доступ к данным строго контролируется логическим слоем. Кроме того, такая структура поддерживает независимость клиентской части, что дает возможность использовать разные устройства и интерфейсы без изменения серверной логики и архитектуры.

Данная модель изображена на рисунке 2.1, а также в веб-приложении А.

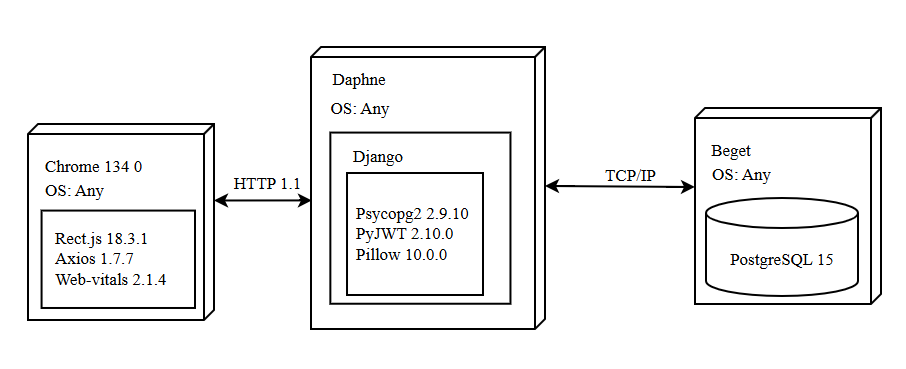


Рисунок 2.1 – Трехуровневая архитектура веб-приложения

Эта модель функционирует по следующему принципу:

* клиентская часть, чаще всего представлена браузером, занимается отображением интерфейса и выполнением действий, описанных функционалом;
* логический слой обрабатывает запросы от пользователя, и на основе различных факторов (например, прав доступа, роли или доступных данных) принимает решение о том, какую информацию предоставить клиенту;
* слой данных хранит информацию в специальном хранилище, доступ к которому осуществляется через сервер, управляющий обработкой запросов.

2.2 Проектирование базы данных

В данном разделе будет рассмотрено проектирование базы данных, включая структуру хранения информации и подходы к ее организации и взаимодействию. Проектирование базы данных – это процесс создания модели для эффективного хранения, управления и обработки данных, обеспечивающий надежность и производительность приложения. Оно необходимо для соответствия функциональным требованиям проекта, упрощения доступа к информации и поддержки масштабируемости системы.

2.2.1 Диаграмма базы данных

Диаграмма таблиц базы данных (Database Table Diagram) представляет собой графическое изображение структуры базы данных, отображающее все таблицы и их связи между собой. Это наглядное средство помогает понять, как данные организованы и как взаимодействуют внутри системы, облегчая анализ структуры данных и выявление потенциальных проблем в проектировании. Диаграмма также позволяет лучше визуализировать отношения таблиц.

На рисунке 2.2 показана диаграмма базы данных, а также в приложении Б.

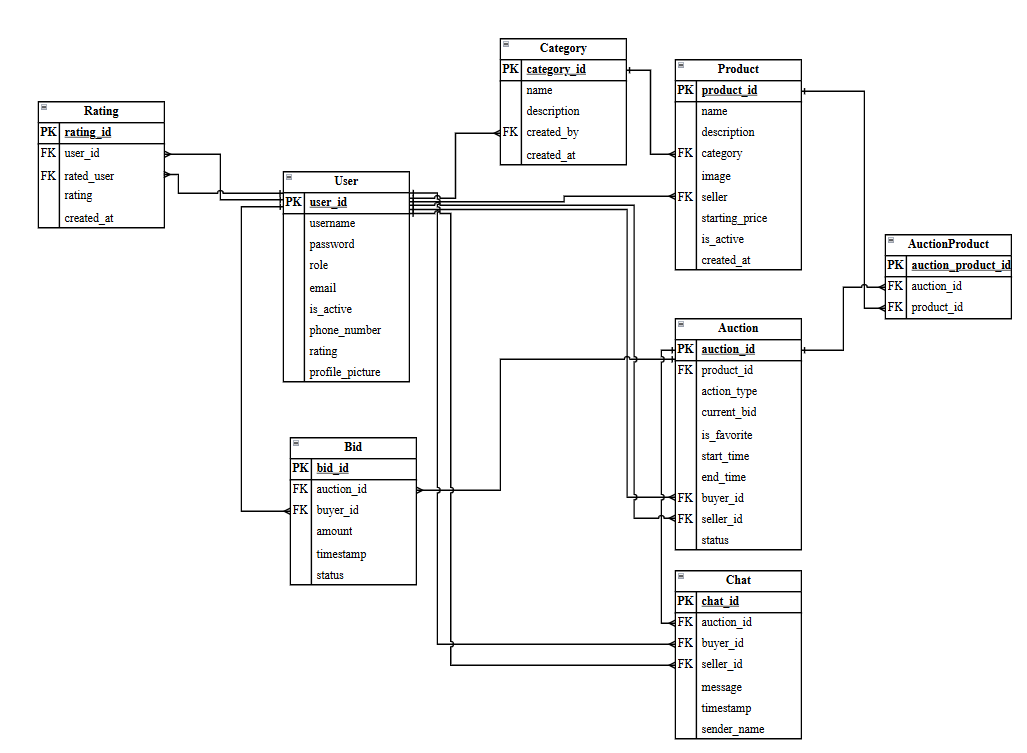


Рисунок 2.2 – Логическая схема базы данных

В базе данных будет 8 таблиц: User, Rating, Category, Product, Auction, AuctionProduct, Bid, Chat. В базе данных будет несколько таблиц для управления пользователями, товарами, аукционами и ставками.

Список таблиц базы данных и их описание можно изучить в таблице 2.1.

Таблица 2.1 – Таблицы базы данных

|  |  |
| --- | --- |
| Таблица | Описание |
| User | Содержит данные об аккаунте |
| Rating | Содержит данные о рейтинге пользователей |
| Category | Содержит данные об категориях |
| Product | Содержит данные об товарах |
| Auction | Содержит данные об аукционах |
| AuctionProduct | Содержит данные об аукционах и товарах |
| Bid | Содержит данные об ставках |
| Chat | Содержит данные о чатах пользователей |

Данный пункт показывает структуру и диаграмму будущей базы данных в которая будет хранить все данные веб-приложения.

2.2.2 Описание таблиц базы данных

Таблица «User» хранит данные пользователей, зарегистрированных в системе. В таблице 2.2 представлено структура «User».

Таблица 2.2 – Структура таблицы «User»

|  |  |  |  |
| --- | --- | --- | --- |
| Название | Тип | Ограничения | Описание |
| user\_id | INTEGER | Primary key | Идентификатор пользователя |
| username | TEXT | Not null | Имя пользователя |
| password | TEXT | Not null | Хэшированный пароль |
| role | TEXT | Not null | Роль пользователя |
| email | TEXT | Not null, Unique | Уникальный адрес электронной почты |
| is\_active | BOOLEAN | Not null | Статус активности пользователя |
| phone\_number | TEXT | Null | Номер телефона пользователя |
| rating | FLOAT | Null | Общий рейтинг пользователя |
| profile\_picture | TEXT | Null | Ссылка на изображение профиля |

Таблица «Rating» хранит информацию о рейтингах, которые пользователи могут выставлять друг другу в процессе взаимодействия на платформе. В таблице 2.3 представлено структура «Rating».

Таблица 2.3 – Структура таблицы «Rating»

|  |  |  |  |
| --- | --- | --- | --- |
| Название | Тип | Ограничения | Описание |
| rating\_id | INTEGER | Primary key | Идентификатор записи рейтинга |
| user\_id | INTEGER | Foreign key | Идентификатор пользователя, поставившего рейтинг |
| rated\_user | INTEGER | Foreign key | Идентификатор пользователя, получившего рейтинг |
| rating | FLOAT | Not null | Числовое значение рейтинга |
| created\_at | DATETIME | Not null | Дата и время создания рейтинга |

Таблица «Category» хранит данные о категориях товаров, включая название, описание и связанные характеристики. В таблице 2.4 представлено структура «Category», что помогает наглядно оценить хранилище и в случае чего переработать или модифицировать хранилище.

Таблица 2.4 – Структура таблицы «Category»

|  |  |  |  |
| --- | --- | --- | --- |
| Название | Тип | Ограничения | Описание |
| category\_id | INTEGER | Primary key | Уникальный идентификатор категории |
| name | TEXT | Not null | Название категории |
| description | TEXT | Null | Описание категории |
| created\_by | INTEGER | Foreign key | Идентификатор пользователя, создавшего категорию |
| created\_at | DATETIME | Not null | Дата и время создания категории |

Таблица «Product» хранит информацию о товарах, размещаемых на аукционах, включая их название, описание, категорию, состояние, фотографии и другие характеристики, что позволяет пользователям и администраторам эффективно управлять лотами и участвовать в торгах. В таблице 2.5 представлено структура «Product».

Таблица 2.5 – Структура таблицы «Product»

|  |  |  |  |
| --- | --- | --- | --- |
| Название | Тип | Ограничения | Описание |
| product\_id | INTEGER | Primary key | Уникальный идентификатор товара |
| name | TEXT | Not null | Название товара |
| description | TEXT | Not null | Описание товара |
| category | INTEGER | Foreign key | Идентификатор категории товара |
| image | TEXT | Not null | Ссылка на изображение товара |
| seller | INTEGER | Foreign key | Идентификатор пользователя, разместившего товар |
| starting\_price | FLOAT | Not null | Начальная цена товара |
| created\_at | DATETIME | Not null | Дата и время создания товара |
| is\_active | BOOLEAN | Not null | Статус товара |

Таблица «Auction» описывает аукционы, на которых продаются товары, включая информацию о времени начала и окончания аукциона, начальной цене, текущем статусе, связанных товарах, а также условиях участия и ставках, что позволяет пользователям отслеживать торги и участвовать в них. В таблице 2.6 представлено структура «Auction»

Таблица 2.6 – Структура таблицы «Auction»

|  |  |  |  |
| --- | --- | --- | --- |
| Название | Тип | Ограничения | Описание |
| auction\_id | INTEGER | Primary key | Идентификатор аукциона |
| product\_id | INTEGER | Foreign key | Идентификатор товара на аукционе |
| action\_type | TEXT | Not null | Тип аукциона (один товар или несколько) |
| current\_bid | FLOAT | Null | Текущая ставка по аукциону |
| is\_favorite | BOOLEAN | Null | Флаг добавления аукциона в избранное |
| start\_time | DATETIME | Not null | Время начала аукциона |
| end\_time | DATETIME | Not null | Время окончания аукциона |
| buyer\_id | INTEGER | Foreign key | Идентификатор покупателя |
| seller\_id | INTEGER | Foreign key | Идентификатор продавца |
| status | TEXT | Not null | Статус аукциона (запланирован, активен, завершен) |

Таблица «AuctionProduct» представляет связь между аукционами и товарами. Связь между двумя табличками представлена как «многие ко многим». Данный подход позволяет манипуливать данными более гибко. На таблице 2.7 представлена структура «AuctionProduct».

Таблица 2.7 – Структура таблицы «AuctionProduct»

|  |  |  |  |
| --- | --- | --- | --- |
| Название | Тип | Ограничения | Описание |
| auction\_product\_id | INTEGER | Primary key | Идентификатор записи |
| auction\_id | INTEGER | Foreign key | Идентификатор аукциона |
| product\_id | INTEGER | Foreign key | Идентификатор товара |

Таблица «Bid» хранит информацию о ставках, сделанных на аукционах. Эта информация весьма важна для сбора статистики всего веб-приложения. Также в после сделанных ставок пользователь создавший аукцион получает возможность просмотреть какие ставки были сделаны на его выставленные товары. Это помогает следить за заинтересованностью к аукционам и торгам в веб-приложении. На таблице 2.8 представлена структура «Bid».

Таблица 2.8 – Структура таблицы «Bid»

|  |  |  |  |
| --- | --- | --- | --- |
| Название | Тип | Ограничения | Описание |
| bid\_id | INTEGER | Primary key | Идентификатор ставки |
| auction\_id | INTEGER | Foreign key | Идентификатор аукциона |
| buyer\_id | INTEGER | Foreign key | Идентификатор пользователя, сделавшего ставку |
| amount | FLOAT | Not null | Сумма ставки |
| timestamp | DATETIME | Not null | Дата и время размещения ставки |
| status | TEXT | Not null | Статус ставки |

Таблица «Chat» хранит сообщения, которые пользователи отправляют в общении на аукционе, включая сообщение, временные метки, информацию об отправителе и получателе. На таблице 2.9 представлена структура «Chat».

Таблица 2.9 – Структура таблицы «Chat»

|  |  |  |  |
| --- | --- | --- | --- |
| Название | Тип | Ограничения | Описание |
| chat\_id | INTEGER | Primary key | Идентификатор сообщения |
| auction\_id | INTEGER | Foreign key | Идентификатор аукциона |
| buyer\_id | INTEGER | Foreign key | Идентификатор пользователя-покупателя |
| seller\_id | INTEGER | Foreign key | Идентификатор пользователя-продавца |
| message | TEXT | Not null | Текст сообщения |
| timestamp | DATETIME | Not null | Дата и время отправки сообщения |
| sender\_name | TEXT | Not null | Имя отправителя сообщения |

Эти таблицы позволяют эффективно управлять пользователями, товарами, аукционами и ставками в веб-приложении, а также обеспечивают возможность общения между пользователями.

2.3 Диаграмма вариантов использования

Веб-приложение должно поддерживать регистрацию и авторизацию пользователей с разделением ролей: администратор, продавец, покупатель.

Администратор сможет управлять категориями товаров для аукционов, блокировать и разблокировать пользователей, видеть статистику как по всему веб-приложению так и по активности пользователей и товарам на аукционах.

Продавец получит возможность управлять своими товарами для аукционов, формировать крупные аукционы и аукционы одного товара, просматривать ставки на свои товары и вести чат с покупателями.

Покупатель сможет просматривать крупные аукционы (запланированные, текущие и завершенные) с информацией о каждом товаре, участвовать в торгах, добавлять аукционы в избранное, просматривать историю ставок на товар, вести чат с продавцом выигранного товара и выставлять оценку продавцу. Диаграмма вариантов использования представлена на рисунке 2.3, а также в приложении В.

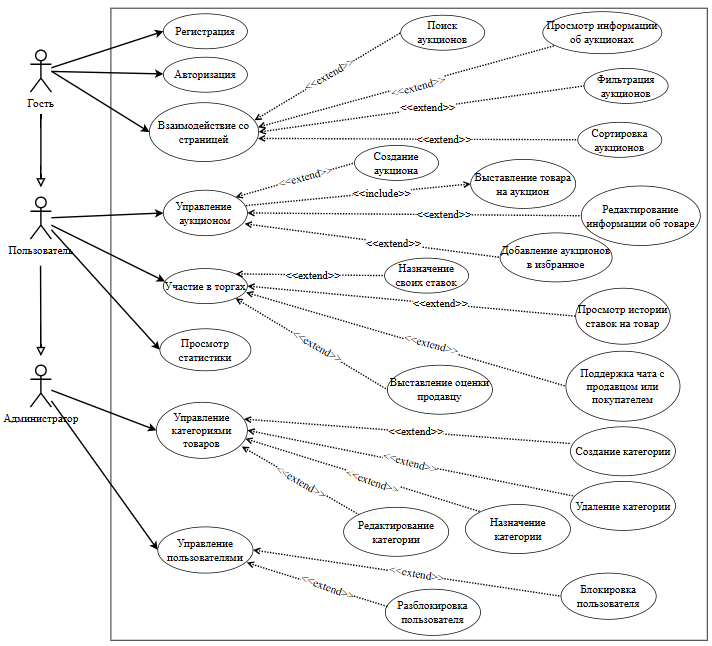


Рисунок 2.3 – Диаграмма вариантов использования

UML диаграмма вариантов использования помогает определить, какие функции доступны для каждой роли в данном веб-приложении. Такой подход очень удобен для оценки возможностей веб-приложения.

2.4 Проектирование контейнеризации

Перед началом разработки приложения важно уделить внимание инфраструктуре, которая будет необходима как в процессе разработки, так и при последующем развертывании. В это входит сервер базы данных, сервер для размещения клиентской и серверной частей приложения. В рамках небольших проектов все эти компоненты можно развернуть на одной физической машине с установленной операционной системой.

Docker – одно из наиболее популярных решений в области контейнеризации, которое предоставляет разработчикам мощный и удобный инструмент для работы с приложениями. Основным преимуществом Docker является его кроссплатформенность. Это значит, что приложение, упакованное в контейнер Docker, можно запускать на любой операционной системе, где установлен Docker, что устраняет проблемы совместимости. Помимо этого, Docker предлагает возможности оркестрации контейнеров, что делает управление многоконтейнерными приложениями гораздо удобнее.

Использование Docker не только упрощает процесс развертывания приложения, но и делает его более надежным и предсказуемым. Это также снижает риск ошибок, связанных с несовместимостью программного обеспечения или неправильной настройкой среды. Кроме того, Docker позволяет разработчикам сосредоточиться на создании функциональности приложения, оставляя вопросы инфраструктуры на уровне автоматизированных инструментов. Такой подход обеспечивает удобство совместной работы, ускоряет процесс разработки и развертывания, а также делает масштабирование приложения более гибким.

2.5 Проектирование основных алгоритмов

В данном разделе будут подробно рассмотрены основные алгоритмы работы приложения, которые играют ключевую роль в достижении целей дипломного проекта. Каждый из этих алгоритмов был разработан с учетом специфики поставленных задач и особенностей архитектуры системы.

Процесс создания алгоритмов был организован поэтапно, начиная с формулировки самых общих, «крупных» команд и заканчивая детализацией каждого шага. На первом этапе формулировались общие принципы работы алгоритма, описывались его ключевые этапы. Далее добавлялись уточнения и более детализированные инструкции. Этот подход, известный как метод последовательного уточнения алгоритма, позволяет шаг за шагом вырабатывать логическую структуру, которая легко воспринимается и понимается. Такой процесс проектирования способствует созданию алгоритмов, которые не только решают поставленные задачи, но и обладают высокой надежностью и качеством. Основное преимущество этого подхода заключается в том, что на каждом этапе можно проверить, насколько алгоритм соответствует ожидаемым результатам, и в случае необходимости внести коррективы.

Каждый алгоритм был представлен в виде блок-схем, что значительно упрощает их восприятие. Блок-схемы позволяют визуализировать процесс выполнения алгоритма, делая его более наглядным и доступным для понимания.

В процессе выполнения дипломного проекта было разработано два основных алгоритма, каждый из которых выполняет важную функцию в рамках общей архитектуры приложения. Все эти алгоритмы были проиллюстрированы блок-схемами, которые подробно отображают логику их работы.

2.5.1 Алгоритм создания аукциона

При попытке создать сам аукцион пользователю необходимо соблюсти ряд определенных условий. Сам процесс создания показан на рисунке 2.4, а также данный алгоритм можно увидеть в приложении Г.

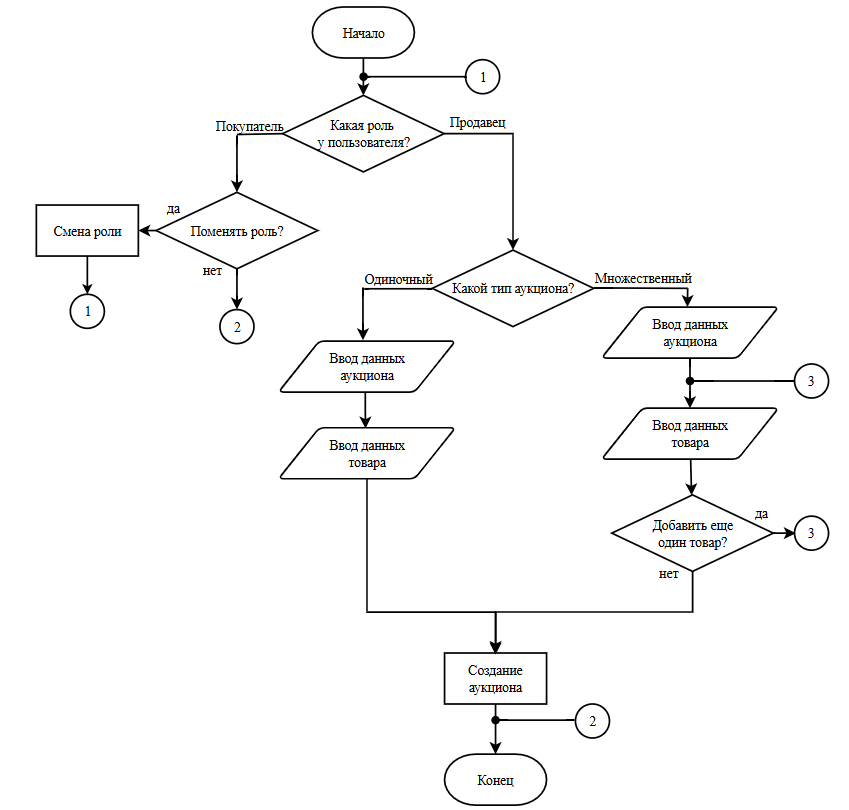


Рисунок 2.4 – Алгоритм создания аукцион

Данная блок-схема описывает процесс покупки товара на аукционе с учетом роли пользователя и типа аукциона.

Процесс начинается с проверки роли пользователя: система определяет, какая роль у пользователя (например, покупатель, продавец или гость). Если роль пользователя не подходит для покупки (например, это продавец или гость), процесс завершается с сообщением «Смена роли». Если роль соответствует покупателю, алгоритм переходит к следующему этапу.

Далее система проверяет, одиночный ли это аукцион или множественный (с двумя и более товарами). Если аукцион одиночный, пользователь вводит данные для покупки одного товара. Если аукцион множественный, пользователь вводит данные для покупки нескольких товаров.

После ввода данных система проверяет, добавлены ли еще товары для покупки в случае множественного аукциона. Если пользователь хочет добавить еще товары, процесс возвращается к этапу ввода данных для множественного аукциона. Если добавление товаров завершено, система создает аукцион.

На финальном этапе аукцион создается и процесс завершен сохранив данные в хранилище данных со всеми введенными параметрами.

2.5.2 Алгоритм проведения аукциона

После того как подойдет подходящее время пользователи могут принять участие и вносить свои ставки. Рисунок 2.5 показывает процесс ставок, а также данный процесс можно увидеть в приложении Д.

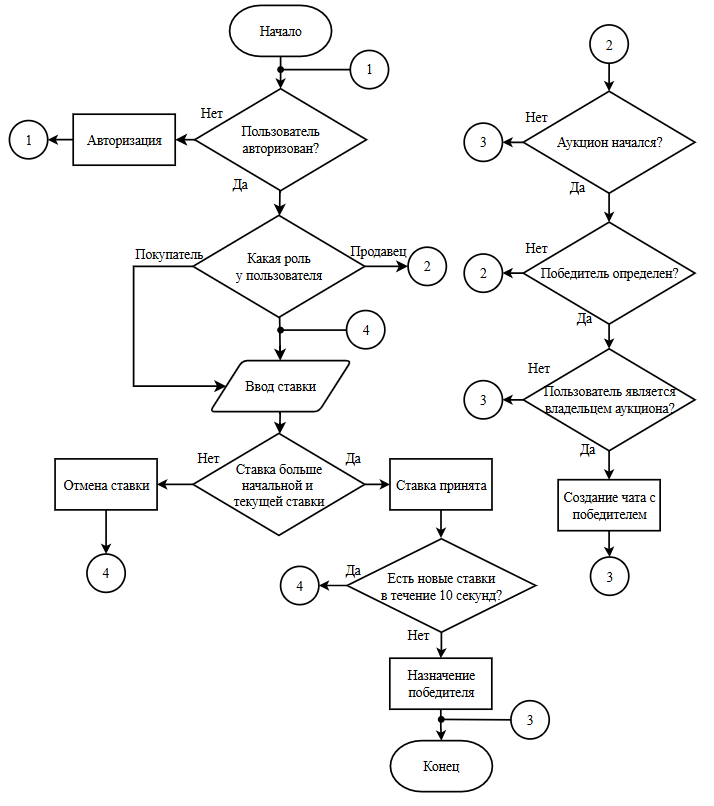


Рисунок 2.5 – Алгоритм проведения аукциона

Данная блок-схема описывает процесс проведения аукциона, начиная с проверки авторизации пользователя и заканчивая определением победителя.

Алгоритм начинается с проверки авторизации пользователя. Если пользователь не авторизован, процесс прекращается. После успешной авторизации выполняется проверка роли пользователя, чтобы убедиться в необходимых правах для участия в аукционе. Если права недостаточны, выполнение завершается.

Далее система проверяет, начался ли аукцион. Если аукцион еще не начался, выполнение приостанавливается до его старта. После начала аукциона вводится начальная ставка. Пользователь имеет возможность установить свою ставку, но только в том случае, если она превышает начальную или текущую ставку. Если ставка не соответствует требованиям, следует отклонение. В случае успешной проверки ставка принимается системой.

Процесс торгов длится до того, пока не появится новый участник со ставкой, превышающей текущую. Если в течение 10 секунд после последней ставки не поступает новая, аукцион завершается, и система назначает победителя.

На этом процесс завершается, и аукцион считается полностью завершенным после чего между продавцом и победителем создается чат.

Таким образом, алгоритм работы аукциона гарантирует соблюдение правил и выполнение действий, необходимых для успешного завершения торгов.

2.6 Выводы по разделу

В этом разделе спроектирован процесс создания веб-приложения, включая этапы определения архитектуры проекта и структуры базы данных.

Для разработки выбраны среды Visual Studio Code и PyCharm. Visual Studio Code используется для работы с фронтенд-компонентами, а PyCharm – для серверной части на Django. Среди выбранных технологий и библиотек – Django для серверной логики, React для клиентской части и PostgreSQL как реляционная база данных. Спроектирована трехуровневая архитектура приложения, разделяющая логику, данные и интерфейс.

Спроектирована структура базы данных, включающая 8 таблиц, каждая из которых подробно описана с учетом функциональности и связей. Это обеспечивает эффективное управление данными и поддержку требований приложения.

Рассмотрена диаграмма вариантов использования, отражающая роли пользователей и их взаимодействие с системой. Это позволило структурировать функциональные требования и задачи для клиентской, и серверной частей.

Для упрощения развертывания будет использован Docker. Контейнеризация обеспечивает независимость от платформы, сокращает время настройки среды и повышает надежность системы.

Спроектированы основные алгоритмы работы приложения, включая создание и проведение аукционов, что в дальнейшем облегчит разработку.

Таким образом, в разделе проектирования заложены основы для реализации веб-приложения: разработана архитектура, выбраны инструменты и технологии, спроектированы база данных и алгоритмы, что дает гарантию надежности системы и ее отказоустойчивости перед ошибками.

3 Разработка веб-приложения

ФИО

Подпись

Дата

Лист

1

ДП 03.00.ПЗ

Разраб.

*Корень М.А.*

Пров.

Комкова А.В.

Н. контр.

Нистюк О.А.

Утв.

Блинова Е.А.

3 Разработка

веб-приложения

Лит.

Листов

13

БГТУ 1-40 05 01, 2025

3.1 Общая структура

Структура проекта разделена на две ключевые части: клиентскую и серверную. Серверная часть управляет логикой веб-приложения и отвечает за обработку запросов от клиента. Клиентская часть фокусируется на пользовательском интерфейсе, взаимодействует с сервером и отображает данные.

3.2 Разработка серверной части веб-приложения

Серверная часть веб-приложения разработана на фреймворке Django. Для реализации различных функций были использованы такие пакеты, как WebSocket для организации реального времени, pyjwt для работы с JSON Web Token, Pillow для обработки изображений и phadne для интеграции с API.

Архитектура проекта в Django основана на принципе «Model-View-Controller» (MVC), где контроллеры представлены представлениями (views), которые обрабатывают запросы и формируют ответы для пользователей. На рисунке 3.1 представлена схема структуры серверной части проекта.

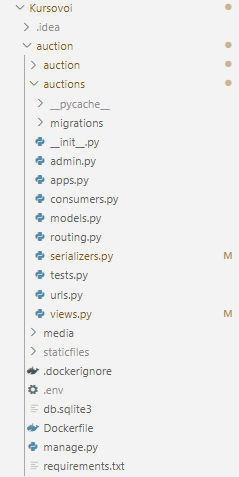


Рисунок 3.1 – Структура серверной части

У

Представления (views) обрабатывают входящие HTTP-запросы, определяют маршруты и выполняют такие действия, как создание, получение, изменение или удаление данных. В представлениях также вызываются бизнес-логика и другие вспомогательные компоненты, что позволяет разделить логику маршрутизации от сложной обработки данных, что значительно повышает читаемость кода и доработку веб-приложения.

Модели (models) реализуют логику работы с данными, описывая таблицы базы данных и взаимодействие с ними. Django использует собственную ORM, которая упрощает создание и выполнение запросов.

Шаблоны (templates) отвечают за отображение информации пользователю, преобразуя данные, полученные от представлений, в HTML-страницы для отображения в браузере.

URL-конфигурация (urls.py) отвечает за сопоставление запросов с соответствующими представлениями и позволяет обмениваться параметрами.

Настройки веб-приложения собираются в файле settings.py, который управляет параметрами конфигурации, такими как базы данных, статические файлы и другие внешние сервисы представлено на листинге 3.1 файл urls.py.

from django.conf import settings  
from django.conf.urls.static import static  
from django.urls import path  
from .views import product\_list, auction\_list, auction\_detail, register, login, profile, update\_profile, rate\_user, \  
 create\_auction, create\_product, get\_categories, refresh\_token, user\_profile\_view, update\_auction, update\_product, \  
 current\_user, product\_detail, add\_to\_favorites, remove\_from\_favorites, is\_favorite, auction\_status, place\_bid, \  
 change\_user\_role, stats\_view, send\_message, get\_messages  
urlpatterns = [  
 path('api/products/', product\_list, name='product\_list'),  
 path('api/create-product/', create\_product, name='create\_product'),  
 path('api/product/<int:pk>/update/', update\_product, name='update\_product'),  
 path('api/product/<int:pk>/', product\_detail, name='product\_detail'),  
 path('api/auctions/', auction\_list, name='auction\_list'),  
 path('api/auction/<int:pk>/', auction\_detail, name='auction\_detail'),]  
if settings.DEBUG:  
 urlpatterns += static(settings.MEDIA\_URL, document\_root=settings.MEDIA\_ROOT)

Листинг 3.1 – Файл urls.py

Точка входа в приложение – файл manage.py, который используется для запуска сервера, выполнения миграций базы данных и других операций.

Этот файл играет ключевую роль в инициализации приложения и его запуске на листинге 3.2 представлен файл manage.py.

const main = async () => {

const app = await NestFactory.create<NestExpressApplication>(AppModule);

app.useGlobalPipes(new ValidationPipe());

const configService = app.get(ConfigService);

app.use(cookieParser());

app.use(compression());

app.use(cors({

origin: configService.get<string>('FRONTEND\_URL'),

credentials: true,

}));

const apiPrefix = 'api';

app.setGlobalPrefix(apiPrefix);

app.useStaticAssets(join(\_\_dirname, '..', "..", 'public'), {

prefix: '/public/',

});

const port = configService.get<number>('PORT\_APP');

await app.listen(port);

main();

Листинг 3.2 – Файл manage.py

Этот файл запускает веб-сервер и выполняет все необходимые операции для старта веб-приложения и его конфигурации для корректной работы.

В рамках данного проекта будут рассмотрены следующие функции:

* регистрация;
* авторизация;
* поиск аукционов;
* создание аукционов;
* чат между продавцом и покупателем.

3.2.1 Реализация процесса регистрации

В любом приложении регистрация и авторизация пользователей играют ключевую роль в обеспечении безопасности и удобства работы с системой. Эти процессы должны быть реализованы таким образом, чтобы гарантировать защиту данных пользователей, а также позволять безопасную интеграцию с другими компонентам и системы. В этом разделе рассмотрим процесс регистрации пользователей и как это происходит.

В данном приложении регистрацию пользователей обрабатывает register endpoint. При отправке запроса на регистрацию с клиентской стороны передаются данные пользователя, такие как email, пароль и имя пользователя. Роль пользователя задается на сервере, когда создается запись в базе данных.

Код логики регистрации представлен в листинге 3.3.

@api\_view(['POST'])  
@permission\_classes([AllowAny])  
@authentication\_classes([])  
def register(request):if request.method == 'POST':

serializer = RegisterSerializer(data=request.data)  
 if serializer.is\_valid():  
 user = serializer.save()  
 refresh = RefreshToken.for\_user(user)  
 access\_token = refresh.access\_token  
 return Response({  
 "message": "User created successfully",  
 "refresh": str(refresh),  
 "access": str(access\_token)  
 }, status=201)  
 return Response(serializer.errors, status=400)

Листинг 3.3 – Код регистрации

После успешной регистрации создаются два типа токенов. Первый – это access токен, который используется для аутентификации запросов к API. Данный токен имеет ограниченный срок действия и отправляется в ответе клиенту для хранения в localStorage. Этот токен прикрепляется к каждому запросу для проверки подлинности пользователя. Второй – это refresh токен, который хранится также в localStorage. Этот токен используется для обновления access токена после его истечения без необходимости повторной авторизации.

Таким образом, два токена – access и refresh – обеспечивают правильный баланс между удобством для пользователя и безопасностью, позволяя поддерживать долгосрочную сессию с минимальными рисками утечек данных.

3.2.2 Реализация процесса авторизации

Процесс авторизации важен и позволяет пользователям получить доступ к защищенным данным после успешной проверки их учетных данных.

Код для логики авторизации будет представлен в листинге 3.4.

@api\_view(['POST'])  
@permission\_classes([AllowAny])   
def login(request):   
 if serializer.is\_valid():  
 user = serializer.validated\_data['user']  
 refresh = RefreshToken.for\_user(user)  
 return Response({'message': 'Login successful',})  
 return Response({'error': 'Invalid credentials'}, status=400)

Листинг 3.4 – Код авторизации

В данном приложении авторизация проходит через несколько этапов. Пользователь передает свои данные, такие как имя и пароль, в запросе. Сервер проверяет эти данные с помощью сериализатора LoginSerializer. Если введенные данные корректны, система находит соответствующего пользователя в базе данных и создает два токена: один для непосредственного доступа к системе (accessтокен) и другой для продления сессии (refresh токен).

Access токен – это краткосрочный токен, который действует ограниченное время (обычно 1 час) и используется для выполнения запросов.

Refresh токен – это более продолжительный токен, действующий, как правило, 1 день. Он сохраняется в localStorage и используется для получения нового access токена после истечения его срока действия.

Если введенные данные неверны, например, пользователь с таким имя не найдено, система возвращает ошибку с кодом 400 и сообщением о недействительных учетных данных. В случае успешного входа клиент получает оба токена. Access токен сохраняется в localStorage для использования в будущих запросах также, как и Refresh токен.

3.2.3 Реализация процесса поиска аукциона

Процесс поиска и фильтрации аукционов в данном приложении является важной частью функциональности, позволяя пользователям находить выгодные предложения по различным критериям. Система фильтрации и сортировки аукционов дает возможность быстро находить нужные товары, подходящие интересам пользователя. В листинге 3.5 будет представлен основной запрос для поиска аукционов.

@api\_view(['GET'])  
@permission\_classes([AllowAny])  
@authentication\_classes([])  
def auction\_list(request):  
 category = request.query\_params.get('category')   
 auction\_type = request.query\_params.get('type')   
 search = request.query\_params.get('search')   
 sort\_by = request.query\_params.get('sort\_by', 'start\_time')   
 auctions = Auction.objects.all()  
 if category:  
 auctions = auctions.filter(related\_products\_\_category\_\_id=category)  
 if auction\_type:  
 auctions = auctions.filter(auction\_type=auction\_type)  
 auctions = auctions.order\_by(sort\_by)  
 serializer = AuctionSerializer(auctions, many=True, context={'request': request})  
 return Response(serializer.data, status=status.HTTP\_200\_OK)

Листинг 3.5 – Код для поиска ауционов

В данном коде процесс поиска аукционов включает несколько ключевых этапов. Фильтрация по категории позволяет системе отбирать аукционы, относящиеся к указанной категории товаров, например, электронике или одежде, если категория указана в запросе, помогая пользователю сузить выбор. Фильтрация по типу аукциона отбирает результаты по заданному формату, например, аукционы с фиксированной ценой или с торгом, что удобно для пользователей, ищущих определенный тип аукциона. Поиск по имени товара, одна из ключевых функций, позволяет находить аукционы запросу, такой как «смартфон», включая товары с совпадением в названии независимо от регистра благодаря использованию icontains, что повышает удобство. Сортировка по дате начала, установленная по умолчанию, отображает аукционы в порядке их старта, что помогает пользователям участвовать в ближайших аукционах, при этом сортировка может быть изменена на другие поля по запросу.

3.2.4 Реализация процесса создания аукционов

Процесс создания аукциона в данном приложении представляет собой сложную операцию, состоящую из нескольких ключевых этапов. Данный процесс позволяет пользователям (продавцам) формировать аукционы для продажи товаров. Процесс весьма важен так как помимо аукциона создает товары.

Авторизация пользователя обязательна для создания аукциона, что обеспечивается декоратором @permission\_classes([IsAuthenticated]), гарантирующим доступ только авторизованным пользователям.

Данные для создания аукциона пользователь отправляет через POST-запрос, а реализация этого процесса представлена в листинге 3.6.

@api\_view(['POST'])

@permission\_classes([IsAuthenticated])

def create\_auction(request):

data = request.data

auction = request.objects.create()

if isinstance(product\_data, dict):

category = Category.objects.get(id=product\_data['category'])

product = Product.objects.create(starting\_price=product\_data['starting\_price'], category=category, seller=request.user, image=product\_data.get('image'))

AuctionProduct.objects.create(auction=auction, product=product)

else:

product = Product.objects.get(id=product\_data)

AuctionProduct.objects.create(auction=auction, product=product)

return Response({'id': auction.id}, status=201)

Листинг 3.6 – Создание аукциона

Проверка данных и обработка ошибок выполняется на сервере. Если обязательные поля, такие как время начала или окончания, отсутствуют, возвращается ошибка. Аукцион создается после успешной проверки с заданием параметров: тип, время, статус и изображение, и сохраняется в базе данных.

Добавление товаров зависит от типа аукциона. В одиночном аукционе при отсутствии товара создается новый товар с заданными характеристиками и связывается с аукционом. В множественном аукционе добавляется несколько товаров, при этом система сообщает о пропущенных элементах или ошибках. Сервер возвращает информацию об аукционе и добавленных товарах либо описания ошибок для их исправления.

Функционал учитывает разные типы аукционов, добавление товаров и обработку ошибок, обеспечивая удобный процесс для пользователей.

3.2.5 Реализация процесса чата

Функция send\_message реализует отправку сообщений в рамках аукциона и формируя чат между участниками (покупателем и продавцом). Функция принимает запрос с данными сообщения. Если один из этих параметров отсутствует, система вернет ошибку.

Также один из важнейших моментов заключается в том, что данный поинт используется в рамках подключения к сокету после победы пользователя.

Логика отправки сообщения представлена в листинге 3.7.

@api\_view(['POST'])  
@permission\_classes([IsAuthenticated])  
def send\_message(request):  
 auction\_id = request.data.get('auction\_id')  
 message = request.data.get('message') required'}, status=status.HTTP\_400\_BAD\_REQUEST)   
 auction = Auction.objects.get(id=auction\_id)  
 buyer = auction.buyer  
 seller = auction.seller  
 sender\_name = buyer.username if request.user == buyer else seller.username  
 async\_to\_sync(get\_channel\_layer().group\_send)(  
 f"auction\_{auction\_id}\_chat",{  
 'type': 'chat\_message',  
 'message': message,  
 'sender\_name': sender\_name})   
 chat = Chat.objects.create(auction=auction,buyer=buyer,seller=seller,message=message,  
 sender\_name=sender\_name)  
return Response({'success': 'Message sent'}, status=status.HTTP\_201\_CREATED)

Листинг 3.7 – View send\_message

Процесс начинается с проверки того, что все обязательные данные присутствуют в запросе. В случае их отсутствия возвращается ошибка с кодом 400. Далее, система пытается найти аукцион по предоставленному ID. Если аукцион не найден, генерируется ошибка с кодом 404.

Затем система проверяет, кто является отправителем сообщения (покупатель или продавец). Если запрос был отправлен покупателем, то его имя будет записано как отправитель, в противном случае – имя продавца.

После этого с помощью WebSockets сообщение передается в соответствующую группу чата, асинхронно с использованием канала WebSocket для передачи данных между участниками аукциона. Канал сообщений получает уникальный идентификатор, который включает ID аукциона, что позволяет обеспечить отправку сообщений только в соответствующие чаты.

Кроме того, сообщение сохраняется в базе данных с указанием аукциона, покупателя, продавца, текста сообщения и имени отправителя. Это позволяет вести историю сообщений для последующего просмотра.

В заключение, если все прошло успешно, система отправляет ответ с кодом 201, подтверждая, что сообщение было отправлено.

3.3 Реализация клиентской части веб-приложения

Клиентская часть веб-приложения создана с акцентом на удобство использования и адаптирована для работы в современных браузерах.

Основной упор сделан на обеспечение интерактивности и высокой производительности интерфейса. Данные загружаются и обновляются в реальном времени без необходимости перезагрузки страницы, что создает ощущение плавного и непрерывного взаимодействия.

Взаимодействие между клиентом и сервером реализовано через асинхронные запросы, что позволяет минимизировать задержки и повышает общую скорость работы приложения. Это особенно важно для пользователей с низкой скоростью подключения, так как интерфейс остается отзывчивым, даже если загрузка данных требует дополнительного времени, так как происходит в фоновом режиме.

Дополнительно была внедрена логика обработки ошибок на стороне клиента, которая информирует пользователя в случае возникновения проблем, таких как недоступность сервера или ошибки при отправке данных. Такой подход делает приложение не только удобным, но и надежным, помогая пользователю лучше понимать, что происходит за кулисами системы.

В целом, клиентская часть веб-приложения сочетает в себе современные технологии и ориентированный на пользователя дизайн, что обеспечивает плавность работы, гибкость и простоту взаимодействия.

На рисунке 3.2 представлена структура проекта.

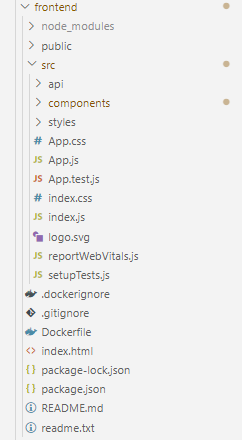


Рисунок 3.2 – Структура клиентской части приложения

Структура клиентской части веб-приложения организована с учетом принципов модульности и удобства разработки.

Папка components содержит все строительные блоки интерфейса, включая как простые элементы (например, кнопки или поля ввода), так и более сложные компоненты (например, формы или карточки). Эти элементы используются повторно во всем приложении.

Папка api предназначена для хранения логики взаимодействия с сервером. Здесь находятся функции и модули, которые отправляют HTTP-запросы и обрабатывают ответы от API. Такой подход позволяет изолировать сетевую логику от остальных частей приложения и повысить читаемость кода.

Папка styles включает файлы, отвечающие за оформление интерфейса. Здесь можно хранить глобальные стили, темы или CSS-модули, которые обеспечивают единый вид приложения и распространяется на все компоненты.

Основные файлы:

* App.js – главный компонент приложения, который выступает в роли корневого узла, объединяющего другие компоненты.
* index.js – точка входа в приложение. Здесь производится подключение корневого компонента к DOM и дальнейшая загрузка.
* App.css и index.css – файлы для локальных и глобальных стилей.

Для управления аутентификацией в проекте используется React Context API, что позволяет централизованно хранить данные о пользователе и давать их любому компоненту приложения. Это упрощает управление состоянием аутентификации и доступ к функциям, таким как вход/выход, смена роли.

Для выполнения запросов к серверу используется библиотека Axios [6], которая обеспечивает удобный интерфейс для работы с API и передачу токенов аутентификации в заголовках запросов. Логика загрузки текущего пользователя реализована в функции loadUser, которая обновляет состояние в зависимости от наличия и валидности токена доступа. Это позволяет сохранять аутентифицированное состояние между сеансами и обеспечивает динамическую подгрузку данных пользователя.

Кроме того, реализована функция смены роли пользователя с отправкой POST-запроса на сервер и обновлением состояния на клиенте в зависимости от роли. Методы входа и выхода управляют аутентификацией, включая удаление токенов из локального хранилища при выходе и загрузку данных пользователя после входа, это удобно как для пользователя, так и для разработчика.

Такой подход дает высокую читаемость и масштабируемость кода, а также упрощает добавление новых функций, связанных с аутентификацией. В листинге 3.8 представлен AuthContext который управляет аутентификацией.

export const AuthProvider = ({ children }) => {

  const [isAuthenticated, setIsAuthenticated] = useState(false);

  const [user, setUser] = useState(null);

  const loadUser = async () => {

    if (isAuthenticated && user) return;

    try {

      const accessToken = localStorage.getItem('access\_token');

      if (accessToken) {

        const response = await axios.get('http://localhost:8000/api/current\_user/', {headers: {

            Authorization: `Bearer ${accessToken}`,},});

        setUser(response.data);

        setIsAuthenticated(true);} else {

        console.error('No access token available');

        setUser(null);

        setIsAuthenticated(false);}

    } catch (error) {

      console.error('Failed to load user:', error);

      setUser(null);

      setIsAuthenticated(false);}};

Листинг 3.8 – AuthContext

Для управления аутентификацией и обновлением токенов используется библиотека Axios с интеграцией перехватчиков запросов и ответов. Это позволяет централизованно обрабатывать авторизационные заголовки и автоматически обновлять токены доступа при их истечении.

Функция refreshToken отвечает за отправку запроса на обновление токена. Данная функция проверяет наличие токена обновления в локальном хранилище и при успешном обновлении сохраняет новый токен доступа. В случае ошибки токены удаляются, чтобы предотвратить использование устаревших данных, и пользователь перенаправляется на страницу входа.

Перехватчики запросов автоматически добавляют токен доступа в заголовок каждого исходящего запроса, обеспечивая удобство и гибкости аутентификации. Перехватчики ответов обрабатывают ошибки 401 (Unauthorized) и инициируют обновление токена, если это необходимо. Если обновление проходит успешно, оригинальный запрос повторяется с новым токеном. В случае неудачи пользователь перенаправляется на страницу входа для обновления.

Такой подход улучшает опыт пользования, обеспечивая плавную аутентификацию и уменьшая нужду вручную управлять токенами в приложении.

В листинге 3.9 представлен процесс обновления токенов.

const refreshToken = async () => {

  try {

    const refreshToken = localStorage.getItem('refresh\_token');

    if (!refreshToken) {

      return null;}

    const response = await axios.post('/api/token/refresh/', {

      refresh: refreshToken,});

    localStorage.setItem('access\_token', response.data.access);

    return response.data.access;

  } catch (error) {

    localStorage.removeItem('access\_token');

    localStorage.removeItem('refresh\_token');

    return null;}};

Листинг 3.9 – Работа с токенами

Приложение организовано с использованием React Router [7] для управления маршрутизацией, что позволяет переключаться между различными компонентами в зависимости от URL. Для обеспечения авторизации и управления состоянием аутентификации приложение оборачивается в AuthProvider, дающий глобальный контекст с доступом к данным о пользователе и токенам.

Пример кода маршрутизации в Routerпредставлен в листинге 3.10.

const App = () => {

  return (<AuthProvider>AuthProvider\*/}

      <Router><Header /><main className="app-content"><Routes>

      <Route path="/" element={<AuctionList />} />

      <Route path="/register" element={<Register />} />

      <Route path="/login" element={<Login />} />

      <Route path="/auctions/:id" element={<AuctionDetail/>}/>

<Route path="/profile/:userId"element={<UserProfile/>}/>

<Routepath="/auction/:id/update"element={<EditAuction/>

<Route path="/product/:id/update" element={<EditProduct/>} />

<Route path="/stats" element={<Stats />}/></Routes></main></Router></AuthProvider>);};

Листинг 3.10 – Настройка маршрутизации

Структура приложения состоит из нескольких маршрутов, каждый из которых соответствует определенному компоненту, отвечающему за конкретный функционал: просмотр списка аукционов, регистрация, вход, профиль пользователя, редактирование профиля, создание аукциона и его деталей, а также просмотр и редактирование продуктов. Главный элемент интерфейса – это Header, который отображается на каждой странице для удобной навигации.

Использование CSS-файла global.css обеспечивает единый стиль для всего приложения, делая его более визуально согласованным. Такой подход к маршрутизации и разделению компонентов обеспечивает модульность, легкость в разработке и дальнейшей поддержке приложения.

3.4 Контейнеризация

Для корректной работы приложения в Docker необходимо заранее подготовить специальные конфигурационные файлы, которые определяют, как приложение будет развертываться. Одним из таких файлов является Dockerfile. Он описывает процесс создания образа, который включает в себя все необходимые зависимости, файлы и настройки для работы приложения. На основе образа создаются контейнеры, которые изолируют приложение от операционной системы и других процессов. Кроме того, для более сложных проектов, где требуется одновременная работа нескольких контейнеров, используется файл docker-compose.yml. Этот файл позволяет описать, как различные контейнеры взаимодействуют между собой, определяя их зависимости, сеть и порядок запуска.

Пример конфигурационного файла Docker, необходимого для создания контейнера клиентской части, представлен в листинге 3.11.

|  |
| --- |
| FROM node:18-alpine  WORKDIR /app  COPY package\*.json ./  RUN npm install  COPY . .  EXPOSE 3000  CMD ["npm", "start"] |

Листинг 3.11 – Пример конфигурационного файла Docker

Для корректной работы приложения, состоящего из нескольких контейнеров, необходимо подготовить специальный файл, который определяет порядок и параметры запуска всех контейнеров, обеспечивающих функционирование приложения. Каждый контейнер содержит отдельную компоненту проекта, и для каждого используется свой Dockerfile. Этот файл позволяет централизованно управлять запуском всех частей серверной части приложения. Пример конфигурации, демонстрирующий запуск серверной части, представлен на листинге 3.12.

|  |
| --- |
| services:  backend:  build:  context: ./auction  dockerfile: Dockerfile  container\_name: django-backend  command: daphne -p 8000 auction.asgi:application  volumes:  - ./auction:/app  - ./auction/media:/app/media  ports:  - "8000:8000"  env\_file:  - ./auction/.env  depends\_on:  - db |

Листинг 3.12 – Пример части конфигурационного файла docker-compose

Этот файл описывает архитектуру, включающей серверную часть, пользовательский интерфейс и базу данных, с инструкциями для взаимодействия.

Серверная часть собирается из каталога с использованием указанного Dockerfile. Она запускается командой для обработки запросов. Каталог с исходным кодом и директория для медиа-файлов монтируются в контейнер для синхронизации данных.

Пользовательский интерфейс собирается из каталога на основе своего Dockerfile. Локальный каталог синхронизируется с контейнерным, что упрощает внесение изменений и тестирование.

База данных запускается с использованием образа. Контейнер использует заранее настроенные параметры: имя пользователя, пароль и название базы данных. Данные хранятся в томе, монтируемом в директорию.

3.5 Выводы по разделу

Данный раздел описывает реализацию приложения, включающую серверную часть на Django и клиентскую часть на React, а также развертывание с использованием Docker и подробным описанием процесса контейнеризации.

Серверная часть реализует функциональности регистрации, авторизации, создания аукционов и чата между продавцом и покупателем, применяя аутентификацию с токенами (access и refresh) для обеспечения безопасности.

Клиентская часть, сделана на React, предоставляет динамичный и удобный пользовательский интерфейс для регистрации, входа, работы с аукционами и общения, используя React Router для быстрой навигации и Context API.

Развертывание приложения осуществляется с помощью Docker, что упрощает настройку, обеспечивает гибкость окружения и поддерживает масштабируемость. Такое сочетание технологий и подходов гарантирует надежность, безопасность и эффективное масштабирование приложения.

4 Тестирование веб-приложения

ФИО

Подпись

Дата

Лист

1

ДП 04.00.ПЗ

Разраб.

*Корень М.А.*

Пров.

Комкова А.В.

Н. контр.

Нистюк О.А.

Утв.

Блинова Е.А.

4 Тестирование

веб-приложения

Лит.

Листов

8

БГТУ 1-40 05 01, 2025

4.1 Тестирование авторизации и регистрации

При тестировании приложения были рассмотрены различные сценарии, которые могли бы привести к ошибкам, включая ввод данных, уже существующих в системе. В этой главе проанализируем ситуации и методы их обработки. Например, на этапе регистрации возможно, что пользователь попытается ввести данные, такие как email или имя пользователя, которые уже зарегистрированы в системе.

Такой сценарий требует валидации и своевременного отображения ошибки, предупреждающей о невозможности повторной регистрации с этими данными такой случай представлен на рисунке 4.1.

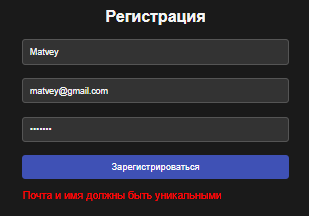


Рисунок 4.1 – Данные существующего пользователя

Для предотвращения отправки неверных данных на сервер реализована валидация на стороне клиента, которая автоматически проверяет введенное значение при изменении поля или попытке отправки формы.

Эта ошибка может возникнуть, если введенный адрес не соответствует стандартному формату email (например, отсутствует символ «@» или доменная часть) такой формат ошибок весьма распространен и требует обработки.

При вводе некорректных данных в поле email система выявляет ошибку и уведомляет пользователя, как показано на рисунке 4.2. Там же видно, что система не только предупредила об ошибке, но и выдало подсказку как правильно следовать формату при вписывании своего email.

У

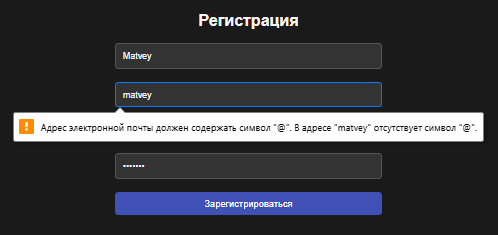


Рисунок 4.2 – Обработка некорректных данных

При отсутствии ввода пароля при регистрации система сразу уведомляет пользователя о необходимости заполнения этого поля. На этапе клиентской валидации отображается сообщение об ошибке, предотвращая отправку формы на сервер. Результат данного теста представлен на рисунке 4.3.

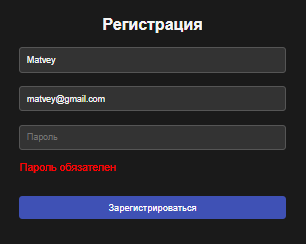


Рисунок 4.3 – Отсутствие пароля

При попытке регистрации с паролем, длина которого менее шести символов, система выполняет проверку на клиентской стороне и сразу информирует пользователя о необходимости указать более сложный пароль. Сообщение об ошибке отображается под полями ввода, предотвращая отправку формы на сервер и дальнейшего создания пользователя.

Такие тесты помогают пользователю понять, что необходимо ввести и предупреждают об ошибка допущенных в написании или пропуске определенных строк в форме. После того как пользователь заметит предупреждение, не произойдет ситуации, когда кто-либо не будет в курсе из-за чего произошла та или иная ошибка так как текст в предупреждении даст необходимую информацию о том, чего нету и что надо ввести дополнительно. Результат данного тестирования представлен на рисунке 4.4.

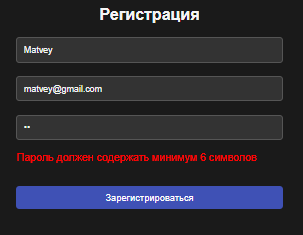


Рисунок 4.4 – Результат c паролем, состоящим из 2 символов

При попытке авторизации без указания пароля система обрабатывает ситуацию, уведомляя пользователя об ошибке. На клиентской стороне это сообщение отображается в удобной для восприятия форме, побуждая пользователя ввести недостающие данные. Таким образом, данный сценарий демонстрирует устойчивость системы к ошибкам ввода и ее способность предотвращать неполные запросы. Результат данного тестирования представлен на рисунке 4.5.

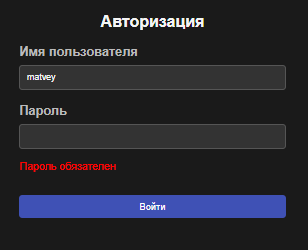


Рисунок 4.5 – Результат отсутствие пароля при авторизации

Таким образом можно сделать вывод что система защиты, авторизации и регистрации реализована верно и предусматривает различные сценарии.

4.2 Тестирование создания аукциона

Тестирование добавления аукциона – ключевая стадия для проверки правильности работы функционала. Первый сценарий – отсутствие дат начала и окончания аукциона, один из важнейших сценариев тестирования веб-приложения. На рисунке 4.6 представлен результат тестирования.

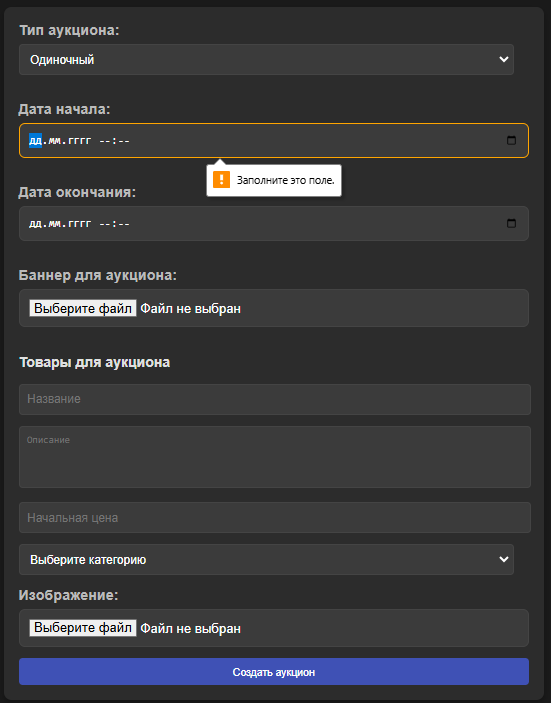


Рисунок 4.6 – Создание аукциона с пустыми полями

При создании аукциона обязательно требуется заполнить все обязательные поля. В случае их отсутствия процесс создания аукциона будет заблокирован. Тестирование подтвердило, что система правильно обрабатывает данный случай, не давая возможность создания аукциона без нужных данных.

Второй сценарий заключается в отсутствии данных в поле для названия товара. Это критическая ошибка, так как название товара является обязательным для корректного создания аукциона. Если это поле не заполнено, система выдает оповещение и предотвращает дальнейшее выполнение процесса создания аукциона. На рисунке 4.7 представлен результат этого тестирования.

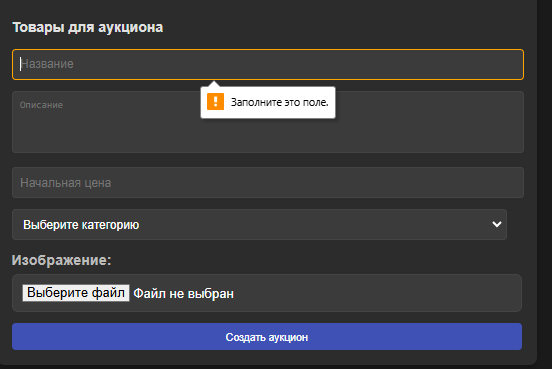


Рисунок 4.7 – Результат попытки создания аукциона без названия товара

Как показали результаты тестирования, создание аукциона невозможно без указания названия товара. Следующий этап тестирования включал попытку создать аукцион без начальной цены.

Тестирование показало, что система не допускает создание аукциона без указания начальной цены, корректно блокируя процесс и предоставляя уведомление об ошибке. Результаты этого теста отображены на рисунке 4.8.

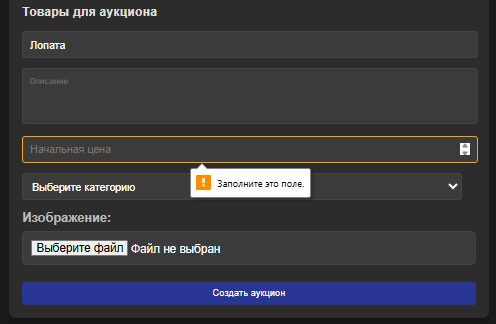


Рисунок 4.8 – Результат попытки создания аукциона без начальной цены

При выполнении всех требований и правильном заполнении обязательных полей процесс создания аукциона проходит успешно. Система корректно обрабатывает введенные данные, и аукцион создается без ошибок, что подтверждает правильность работы функционала в данных условиях.

4.3 Тестирование создания аукциона

В функционале поиска предусмотрены определенные поля для ввода текста и выпадающие меню для выбора параметров. Первый сценарий тестирования предполагает ввод какого-нибудь названия товара.

На рисунке 4.9 представлен результат данного тестирования.

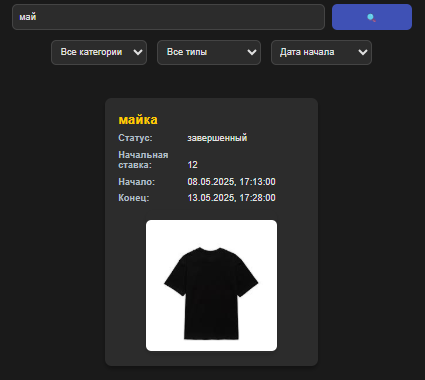


Рисунок 4.9 – Результат поиска товара

Этот тест был успешно выполнен. Следующим этапом было тестирование фильтра по категории «Техника» для поиска неисправностей.

На рисунке 4.10 представлен результат данного тестирования.

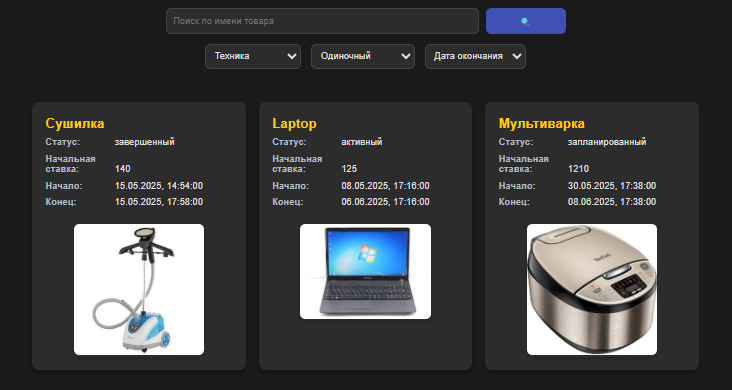


Рисунок 4.10 – Результат использования фильтра по категории

На следующем этапе тестирования тщательно проверялась функциональность фильтрации данных по типу аукциона, гарантирующая работу системы. На рисунке 4.11 представлен результат данного тестирования.

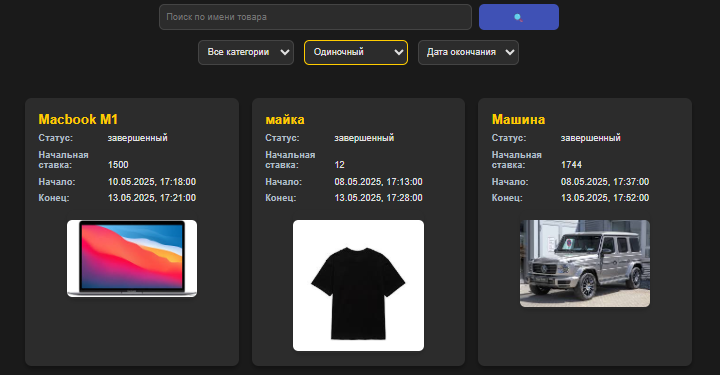


Рисунок 4.11 – Результат фильтрации по типу аукциона

На следующем этапе тестировалась функциональность сортировки по дате окончания аукциона, гарантирующая точное упорядочивание данных в системе. На рисунке 4.12 представлен результат данного тестирования.

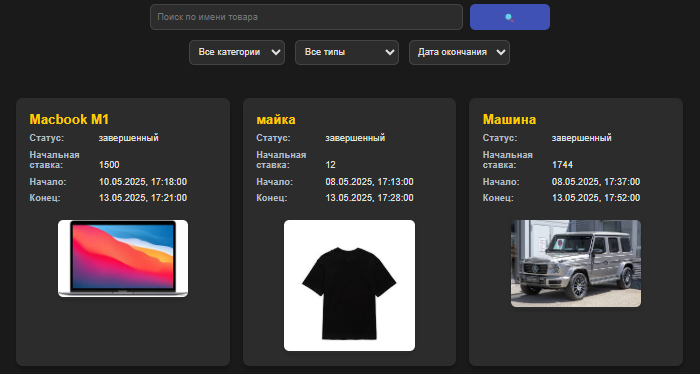


Рисунок 4.12 – Рузультат сортировки аукционов по дате окончания

Тестирование функционала поиска аукционов было проведено успешно, что подтвердило корректную работу фильтров, сортировок и правильное отображение результатов. Все фильтры функционируют согласно ожиданиям, предоставляя пользователю точные и соответствующие запросу результаты.

4.4 Тестирование чата покупателя и продавца

Для тестирования чата нужно выиграть какой-либо аукцион и далее у покупателя и продавца появится доступ к чату. Чат открывается только перед тем покупателем, который является победителем по итогу торгов на аукционе. Данное условие создано для того чтоб после окончания торгов создать максимально удобные условия как для победителя, так и для владельца аукциона. В результате данных действий оба пользователя могут договориться как о способе передачи, так и просто чтоб установить контакт друг с другом.

На рисунке 4.13 представлен результат данного тестирования.

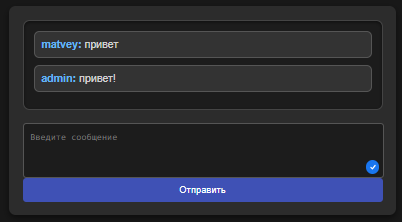


Рисунок 4.13 – Тестирование корректной работы обмена сообщений

Далее была проведена проверка на возможность чтения и ведения чата пользователя который не участвовал в аукционе.

На рисунке 4.14 представлен результат данного тестирования.



Рисунок 4.14 – Результат проверки доступа к чату

Тест был признан успешным, поскольку при попытке просмотра чата он не отобразился. Система корректно отклонила загрузку и предотвратила дальнейшее продолжение процесса, обеспечив правильную обработку ошибки.

4.5 Выводы по разделу

Таким образом, были успешно протестированы функции создания и поиска аукционов, заполнение обязательных полей и отображение чата. Система корректно обрабатывает обязательные поля, фильтрацию данных и ошибки при отсутствии необходимой информации, предотвращая создание аукциона с некорректными данными. При неправильных данных для чата загрузка отклоняется. Все тесты подтвердили стабильность и безопасность приложения.

5 Руководство пользователя

ФИО

Подпись

Дата

Лист

1

ДП 05.00.ПЗ

Разраб.

*Корень М.А.*

Пров.

Комкова А.В.

Н. контр.

Нистюк О.А.

Утв.

Блинова Е.А.

5 Руководство

пользователя

Лит.

Листов

9

БГТУ 1-40 05 01, 2025

5.1 Руководство для пользователя

При открытии веб-приложения, первой страницей является страница со всеми аукционами. В таком режиме можно лишь просматривать общую информацию и нельзя ничего кроме регистрации и авторизации. Если у пользователя еще нет аккаунта, есть возможность перейти на страницу регистрации.

Страница регистрации представлена на рисунке 5.1.

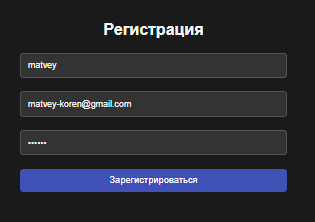


Рисунок 5.1 – Страница регистрации

После нажатия на кнопку «Зарегистрироваться» пользователь автоматически получает роль покупателя. В этой роли пользователь получает доступ к функционалу, позволяющему участвовать в аукционах, делать ставки и взаимодействовать с продавцами посредством чата. Это базовая роль, предназначенная для тех, кто заинтересован в приобретении товаров на платформе.

При желании пользователь может сменить роль на продавца, что откроет доступ к созданию собственных аукционов. Если потребуется, пользователь всегда может вернуться к роли покупателя.

5.1.1 Руководство для покупателя

После регистрации, покупатель получает возможность участия в аукционах посредством ставок, искать сортировать и фильтровать аукционы, выставлять рейтинг продавцам и в случае победы общаться с продавцом и другие важные функции, которые помогают пользователю при работе с ресурсом и накоплении положительного опыта взаимодействия.

На рисунке 5.2 представлена страница домашнего экрана покупателя.

У

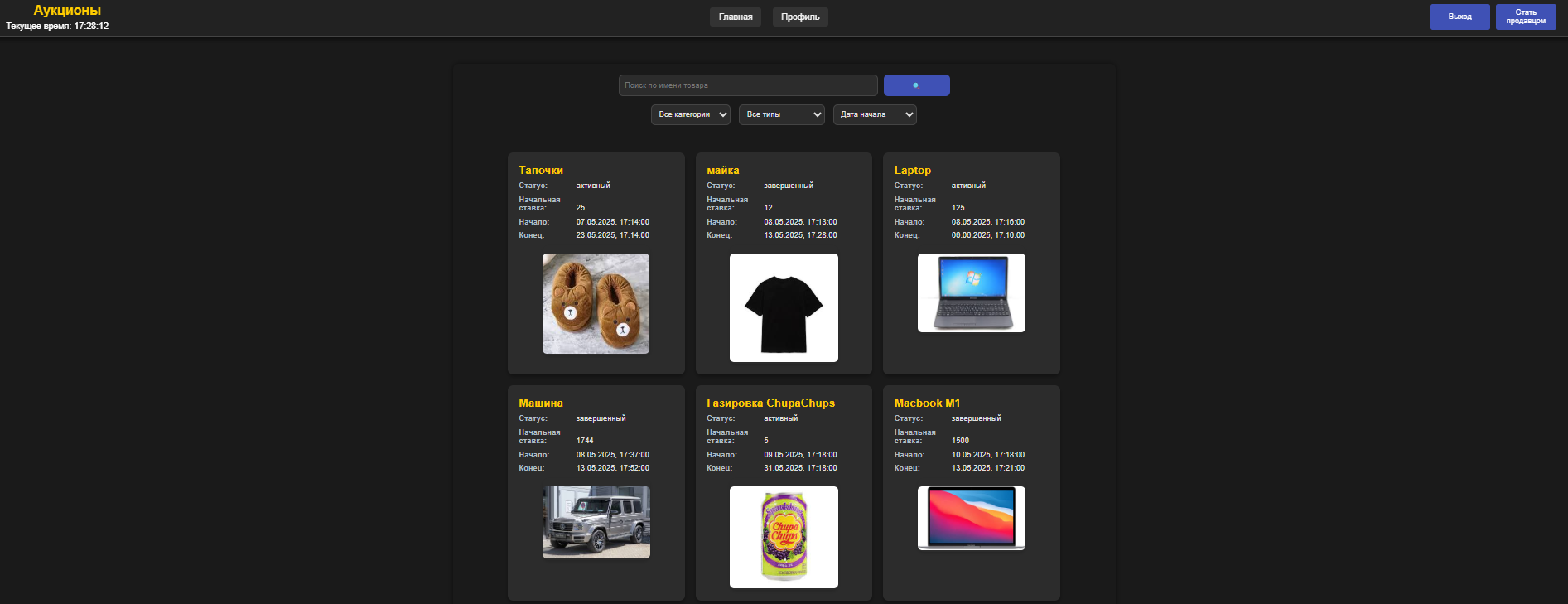


Рисунок 5.2 – Домашняя страница

Далее, если пользователь нажмет на какой-либо аукцион, последует переход. На странице аукциона представлена вся доступная информация, касающаяся товара и самого аукциона, включающая такую информацию как статус и остальное. Нажав на кнопку «Добавить в избранное», пользователь помечает данный аукцион, чтоб вернуться к нему в дальнейшем. На рисунке 5.3 представлена страница с информацией аукциона.

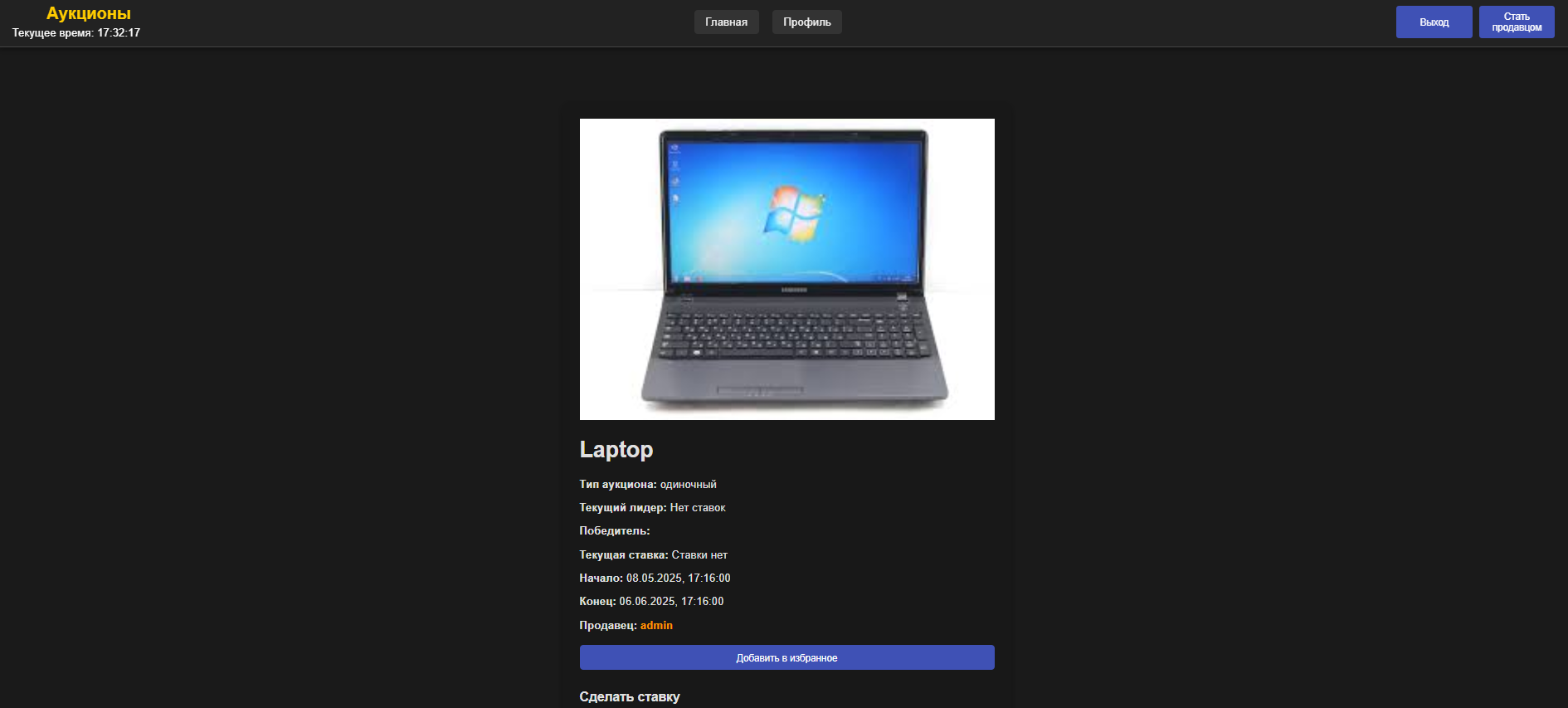


Рисунок 5.3 – Страница аукциона

Следующая страница – это страница профиля, на которую можно попасть, нажав на кнопку «Профиль». Уже на данной странице пользователь может посмотреть и отредактировать свою информацию.

На рисунке 5.4 представлена страница профиля.

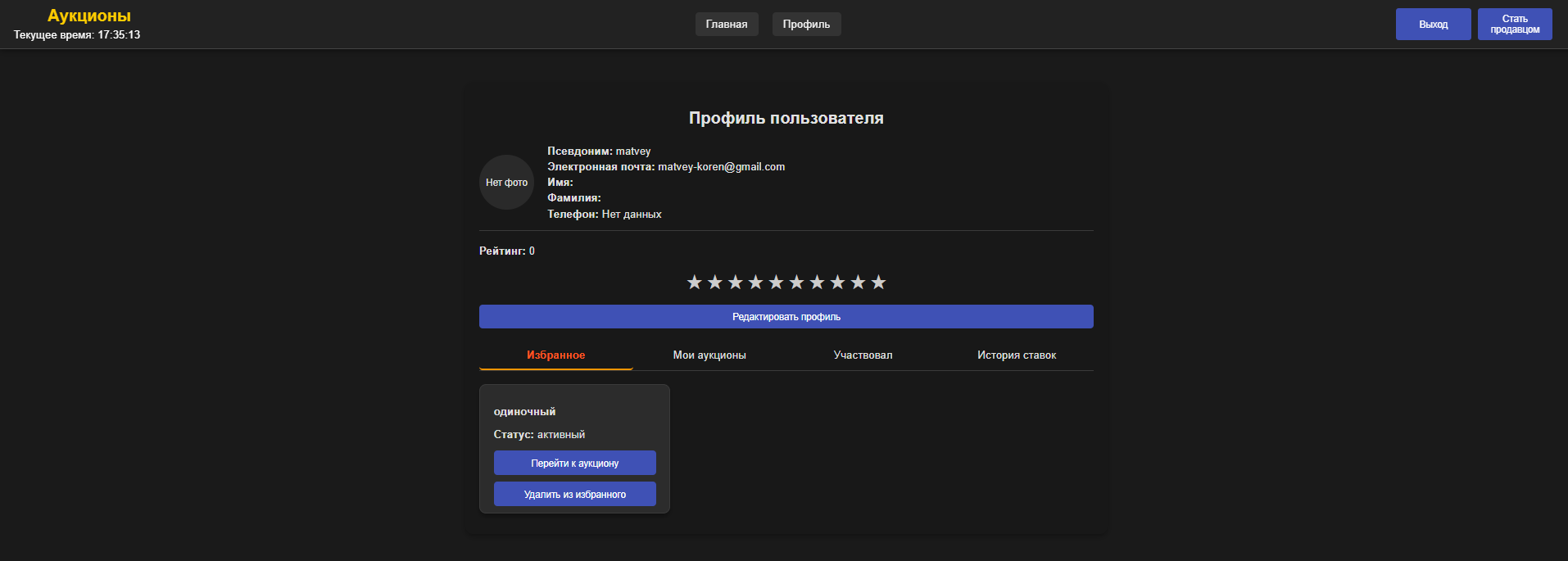


Рисунок 5.4 – Страница профиля

Тут пользователь может отредактировать информацию, которая относится непосредственно к его личности нажав на кнопку «Редактировать профиль». Рисунок 5.5 показывает форму для редактирования.

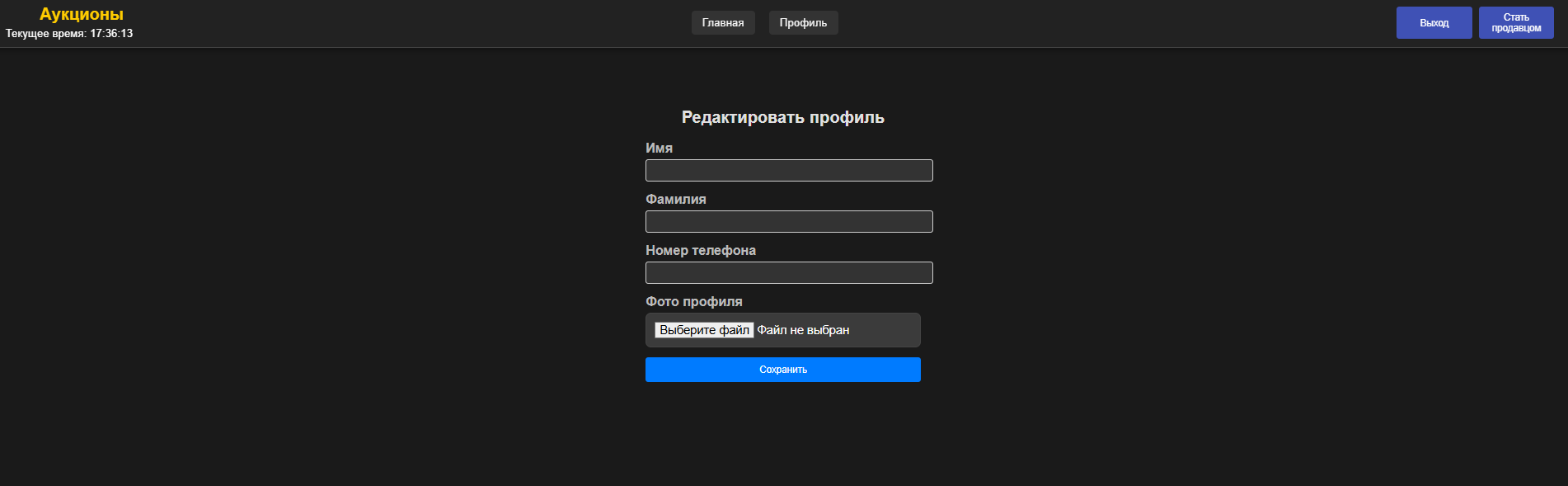


Рисунок 5.5 – Страница редактирования профиля

В профиле пользователь может просматривать аукционы, добавленные в избранное, выбрав вкладку «Избранное». При нажатии на кнопку «Перейти к аукциону» пользователь будет перенаправлен на страницу соответствующих торгов. На этой странице пользователь также может удалить аукцион из избранного, нажав на кнопку «Удалить из избранного», после чего аукцион исчезнет из списка избранных. Рисунок 5.6 показывает вид данной вкладки.

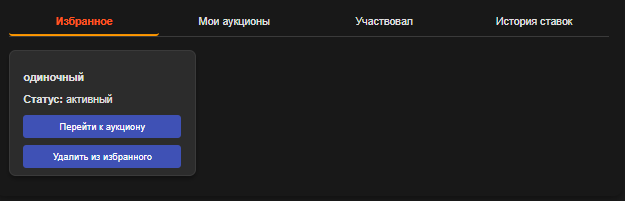


Рисунок 5.6 – Вкладка «Избранное»

Следующая вкладка – это «Мои аукционы», где отображены все аукционы, которые текущий пользователь создал и является владельцем. Рисунок 5.7 показывает результат отображения аукционов.

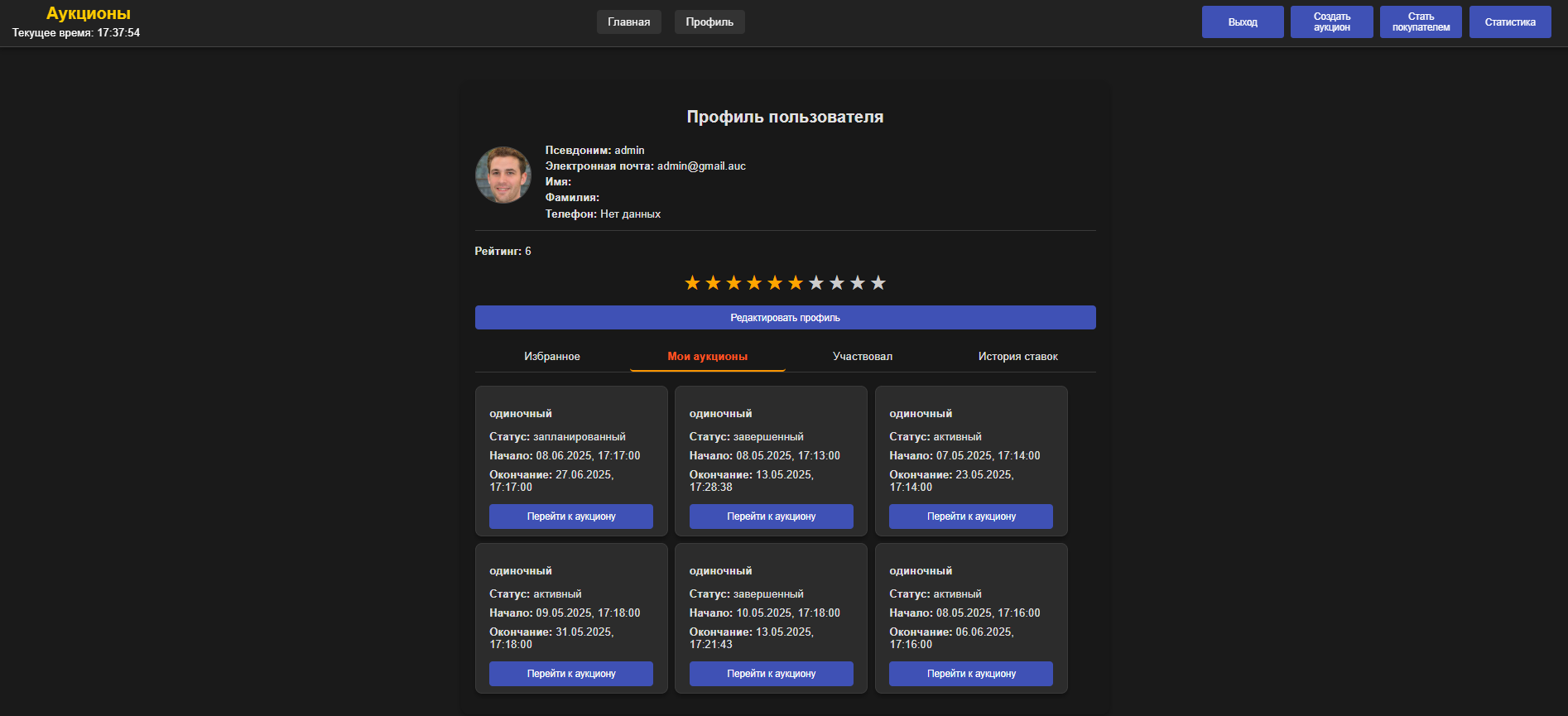


Рисунок 5.7 – Вкладка «Мои аукционы»

Во вкладке «Участвовал» отображается статистика пользователя на ресурсе. Тут же есть фильтры проиграл/выиграл, чтоб пользователь знал, где и что было выиграно/проиграно. Рисунок 5.8 статистика и результаты торгов.



Рисунок 5.8 – Вкладка «Участвовал»

И последняя вкладка «История ставок», где отображены все ставки пользователя, которые когда-либо были сделаны. Рисунок 5.9 показана вкладка с историей ставок пользователя с подробной информацией такой, как типом аукциона ставкой и датой, когда ставка сделана.

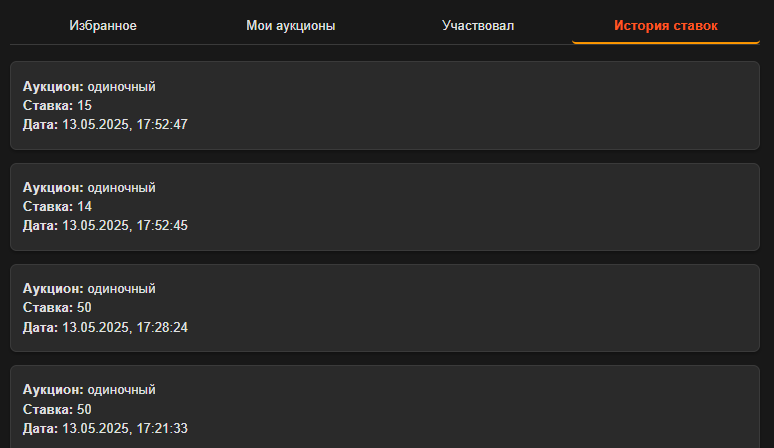


Рисунок 5.9 – Вкладка «История ставок»

При заходе на аукцион со статусом «активный» у пользователя появляется возможность делать ставки нажимая кнопки «Сделать ставку» и «Повысить ставку на 10%». Чтобы сделать ставку, надо ввести число, превышающее текущую лидирующую ставку, а при нажатии соседней кнопки к лидирующей ставке добавляется 10%. Необходимо нажать на «Сделать ставку» и тогда ставка отобразится. Рисунок 5.10 показывает данный функционал.

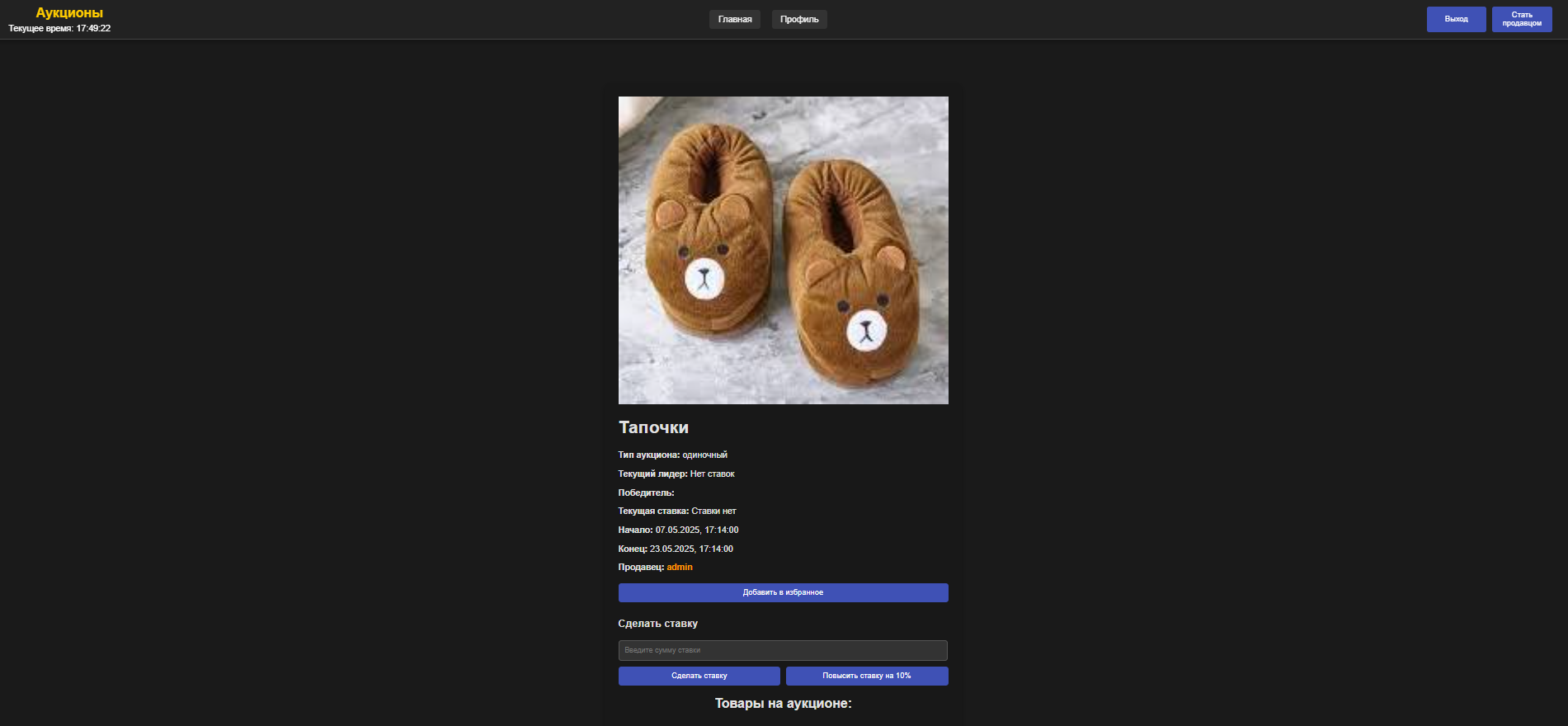


Рисунок 5.10 – Ставки на аукционе

В случае победы пользователя перед ним отобразиться чат, созданный с владельцем данного аукциона. Рисунок 5.11 чат победителя и создателя.

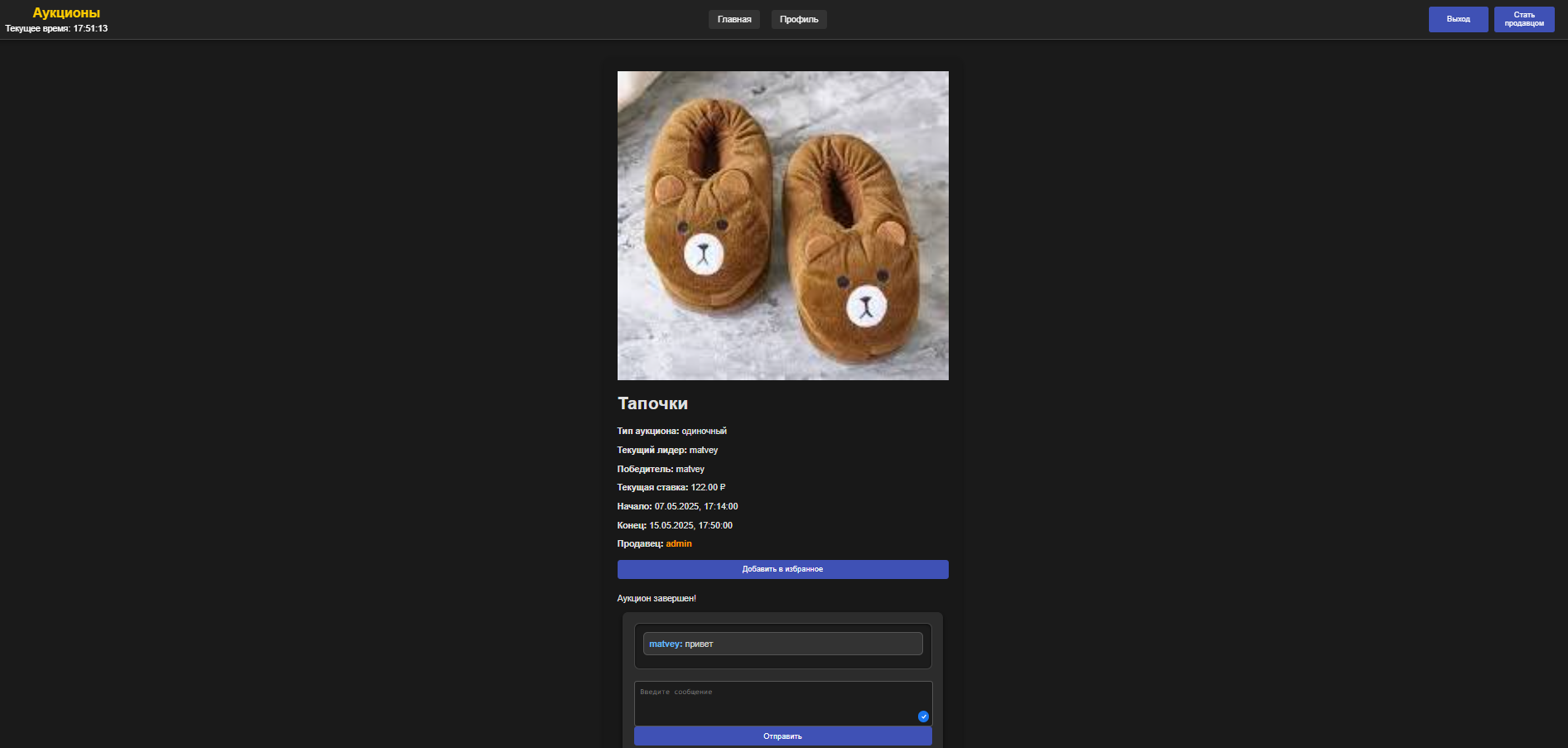


Рисунок 5.11 – Страница с чатом

Следующая страница – это профиль продавца. Если пользователь хочет оставить рейтинг после завершения сделки на аукционе, он может сделать это, перейдя в профиль владельца. Для этого нужно нажать на имя продавца, указанное напротив поля «Продавец». На рисунке 5.6 представлена страница продавца с рейтингом и его информацией.

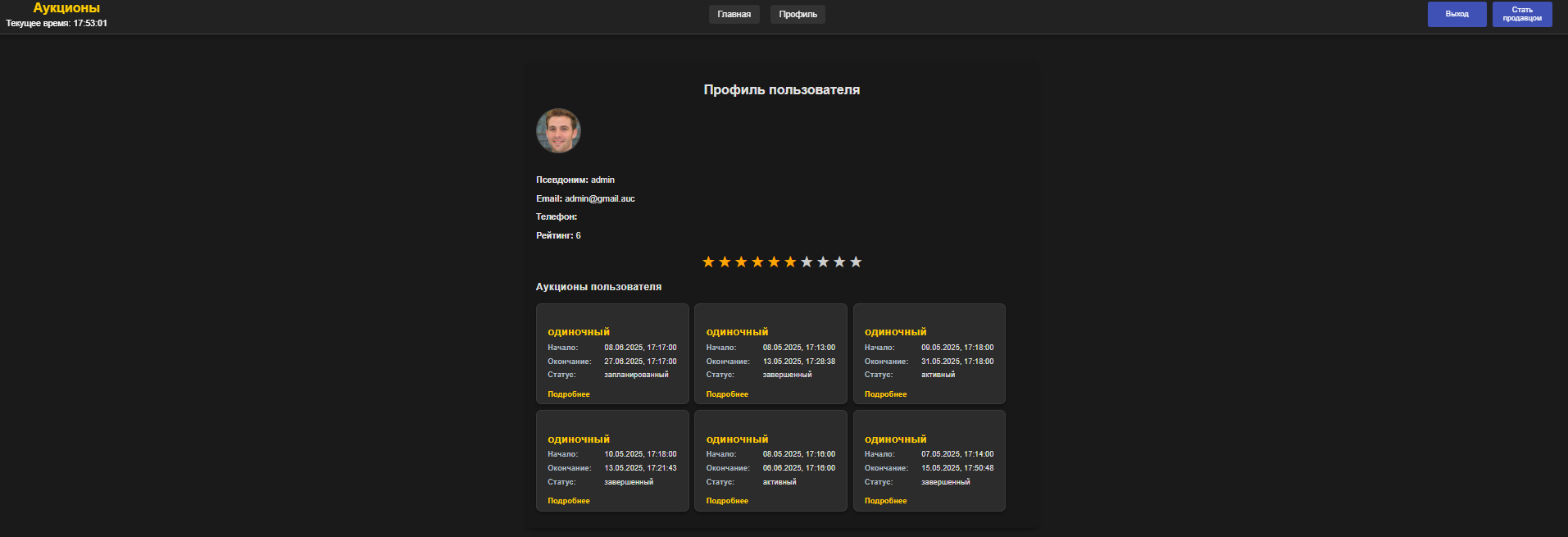


Рисунок 5.12 – Страница продавца

Важно отметить, что пользователь с ролью покупателя не может создавать собственные аукционы, однако он имеет возможность участвовать в аукционах, созданных другими.

5.1.2 Руководство для продавца

Для покупателя функционал меняется после нажатия на кнопку смены роли или же просто «Стать продавцом». На рисунке 5.13 представлена страница с обновленной кнопкой после смены роли.

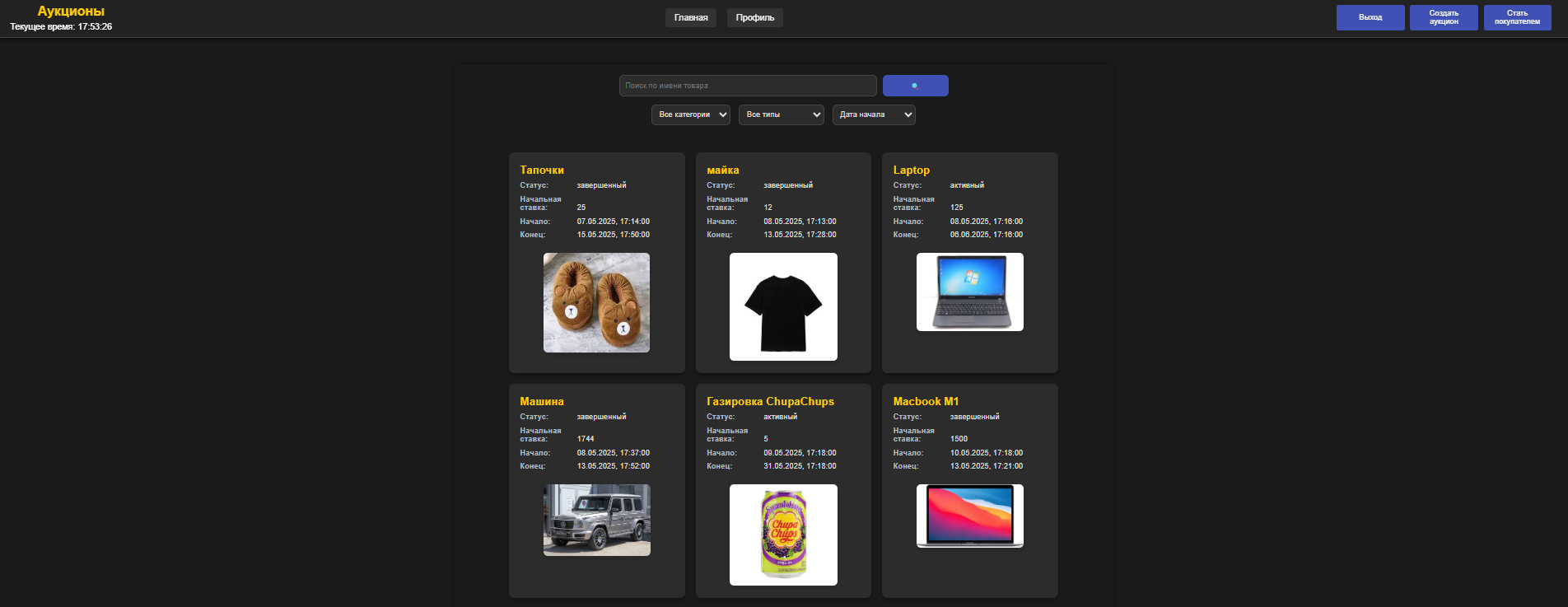


Рисунок 5.13 – Главная страница для продавца

Также можно заметить, что при становлении продавцом теряется возможность делать ставки на аукционе, но помимо этого открывается множество других функций. На рисунке 5.14 представлена страница аукциона.

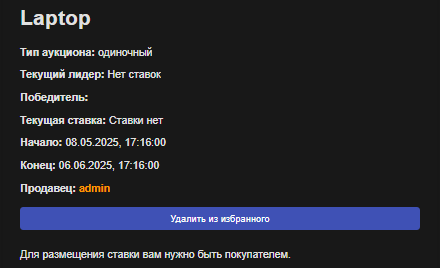


Рисунок 5.14 – Страница аукциона

Нажав на кнопку «Создать аукцион», продавец попадает на страницу с формой для создания аукциона, где нужно ввести все обязательные поля для того, чтоб создать ауцион. После заполнения всех обязательных данных, пользователь может завершить создание аукциона. Обязательные поля включают: тип аукциона, дату начала и окончания, название и описание товара, его цену, а также выбор категории.

После этого необходимо нажать кнопку «Создать аукцион». Если вся информация введена корректно, форма будет отправлена на сервер. На сервере создастся соответствующая запись, и аукцион станет готовым для работы и участия в торгах.

Рисунок 5.15 показывает форму аукциона со всеми необходимыми полями. Также пользователь может добавить баннера для аукциона и для товара, чтобы другие покупатели видели состояние товара.

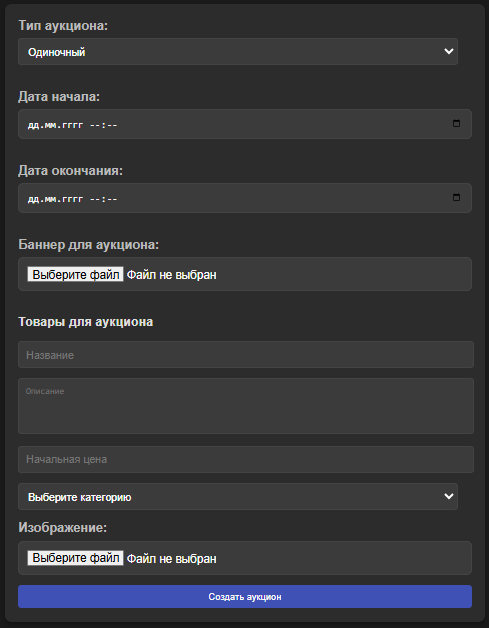


Рисунок 5.15 – Форма аукциона

В случае если пользователь выберет тип аукциона как «множественный», тогда ниже появится дополнительная кнопка «Добавить товар», которая добавляет еще одну форму для товара к уже существующим. Рисунок 5.16 показывает данную кнопку.

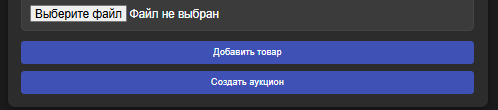


Рисунок 5.16 – Кнопка добавления еще одного товара

После создания пользователя отправит на страницу созданного аукциона, где, если время начала еще не пришло, владелец может отредактировать аукцион, нажав на кнопку «Редактировать аукцион». На рисунке 5.17 показана кнопка редактирования аукциона.

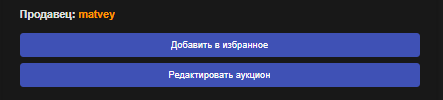


Рисунок 5.17 – Кнопка редактирования

После нажатия на кнопку «Редактировать аукцион» перед владельцем появится форма, в которой можно поменять данные аукциона или удалить, нажав «Удалить аукцион». Рисунок 5.18 изменение аукциона владельцем.

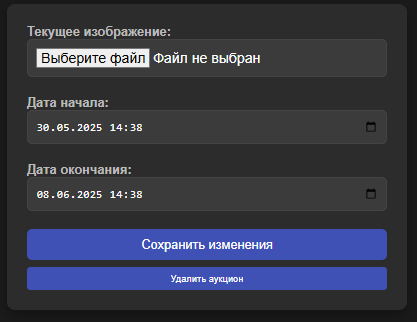


Рисунок 5.18 – Форма редактирваония аукциона

Также у владельца аукциона есть возможность отредактировать товар, нажав на кнопку «Редактировать товар» на странице аукциона. Рисунок 5.19 показывает данную кнопку под товаром.



Рисунок 5.19 – Кнопка редактивароние товара

После нажатия на кнопку «Редактировать товар» пользователь должен ввести новые данные и нажать на кнопку «Сохранить изменения» или выбрать опцию «Удалить продукт». В случае, если товар был единственным в аукционе, сам аукцион также будет удалён. На рисунке 5.20 представлена форма редактирования товара.

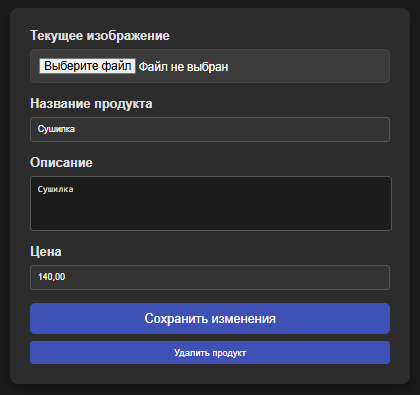


Рисунок 5.20 – Форма редактирования товара

Последним этапом является определение победителя аукциона. В этот момент владелец может просмотреть все ставки, сделанные на текущем аукционе, и начать переписку с победителем прямо на странице аукциона. На рисунке 5.21 представлены функциональные возможности, связанные с этим процессом.

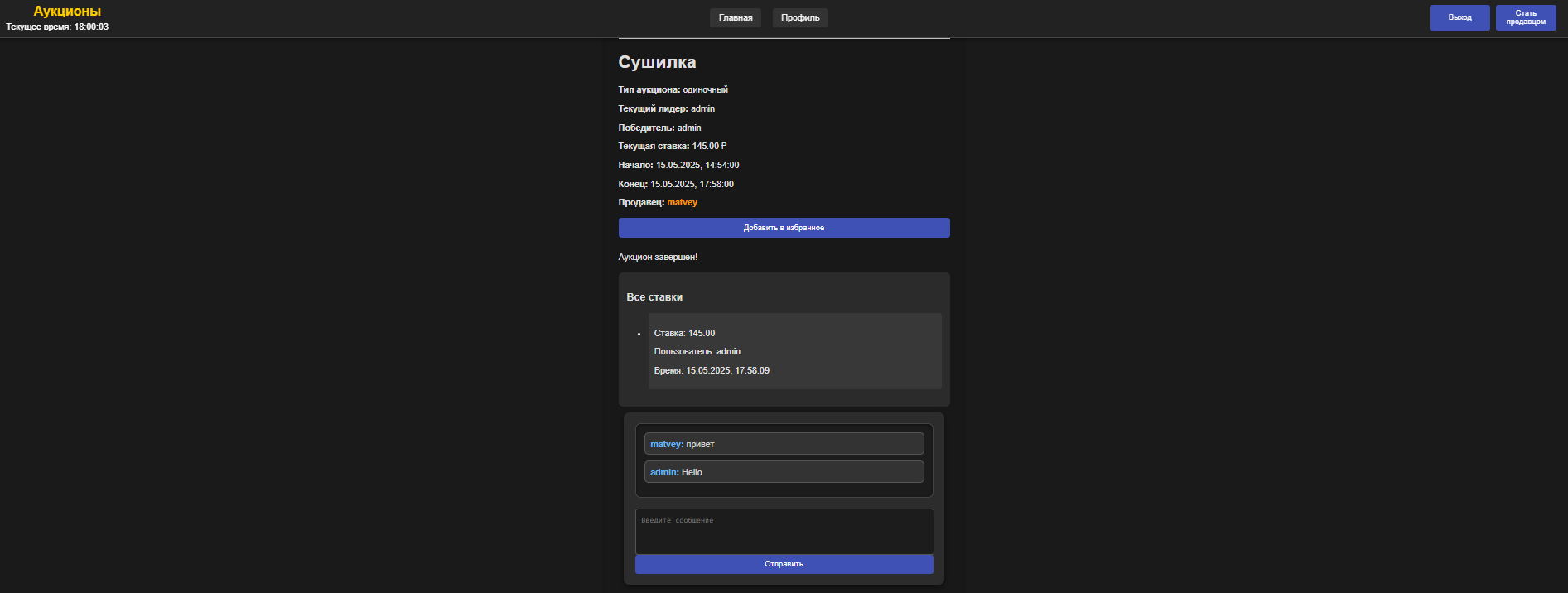


Рисунок 5.21 – Страница ауциона для владельца после завершения торгов

В итоге можно отметить, что роль продавца – это самостоятельная и уникальная функция с собственным набором возможностей, которые недоступны обычному покупателю. Однако, как и в любой роли, у продавца есть свои ограничения, которые делают взаимодействие между участниками аукциона более сбалансированным.

5.1.3 Руководство для администратора

После авторизации администратор попадает на домашнюю страницу, которая содержит важный элемент такой, как кнопка «Cтатистика».

На рисунке 5.22 представлена домашняя страница для администратора.

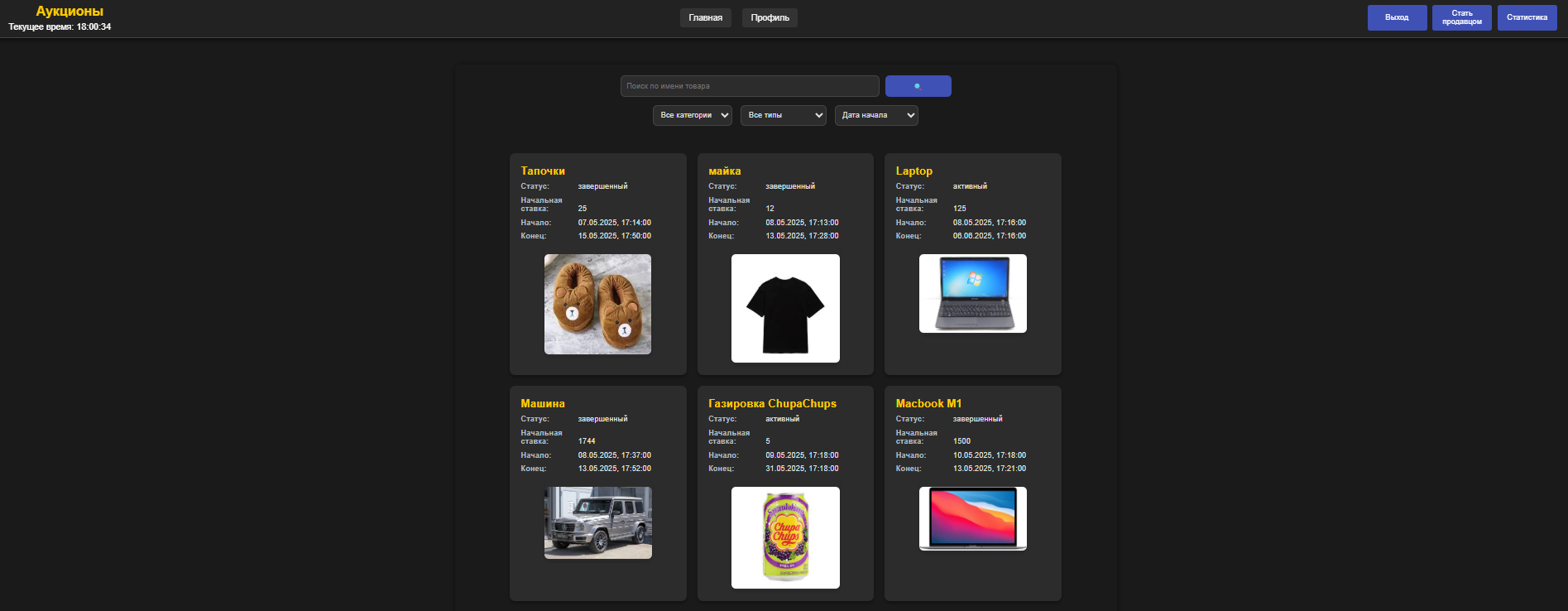


Рисунок 5.22 – Домашняя страница для администратора

Следующей идет страница статистики, где администратор получает доступ к общей информации о работе веб-приложения. Данные представлены в виде наглядных графиков и диаграмм, что позволяет легко анализировать основные показатели и делать выводы о состоянии веб-приложения.

На рисунке 5.23 представлена данная страница статистики.



Рисунок 5.23 – Страница статистики

Также для администратора есть страница с админ панелью для управления сайтом, нажав на каждый из вариантов администратор получает полный перечень данной сущности. На рисунке 5.24 представлена страница админ панели.

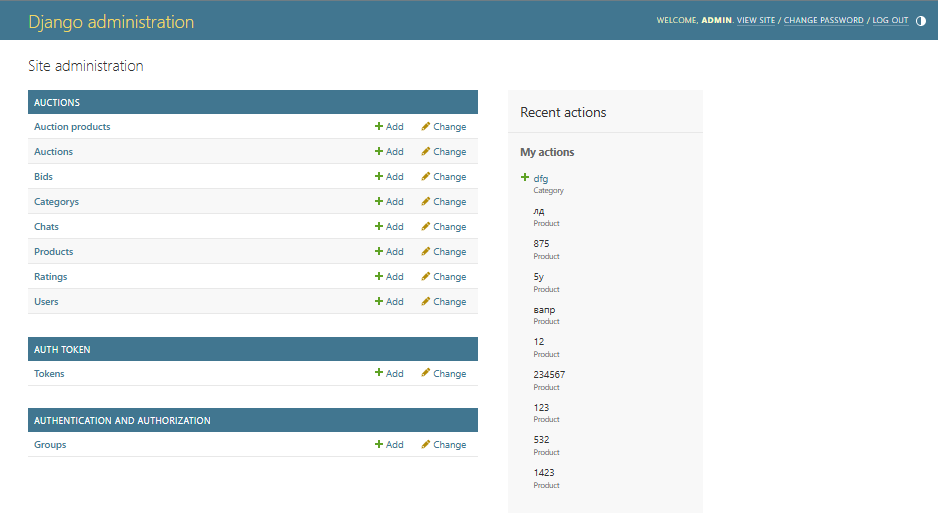


Рисунок 5.24 – Страница админки

Перейдя на страницу с пользователями нажав на «Users», администратор может заблокировать пользователя или наоборот разблокировать, или посмотреть информацию касающуюся любого продукта категории или аукциона.

В случае с пользователем нажав на выпадающее меню где администратор может выбрать «Delete selected user», «Заблокировать выбранного пользователя» и «Разблокировать выбранного пользователя» а также поставить галочку напротив пользователя. В первом случае пользователь пропадет после нажатия на «Go», во втором и третьем после срабатывания пользователь или заблокируется, или разблокируется. Каждая блокировка записывает логи на сервере дабы избежать неправомерных действий со стороны администрации.

На рисунке 5.25 представлена страница блокировки/разблокировки/удаления пользователя с помощью админ панели

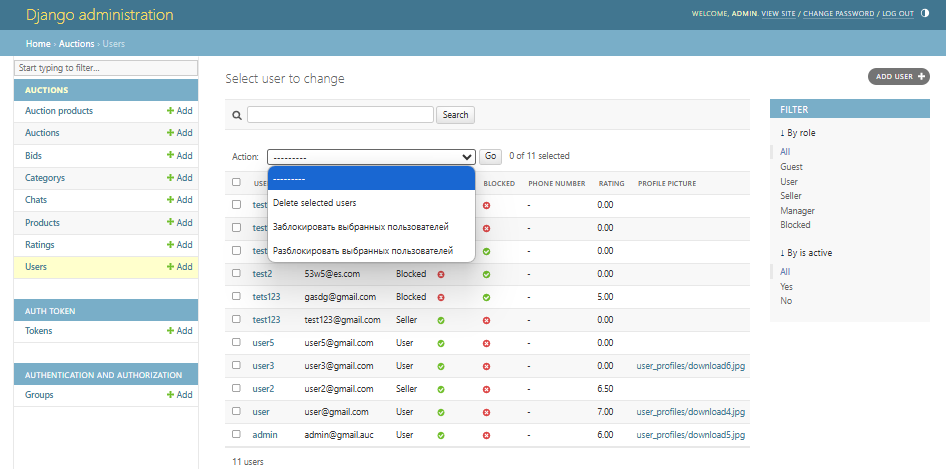


Рисунок 5.25 – Страница управления пользователем

Таким образом, выглядит руководство пользователя для администратора и его взаимодействие с админ панелью.

5.2 Руководство по развертыванию приложения

Самым удобным способом, благодаря которому можно осуществить развертывание приложения – это Docker. Docker позволяет создавать контейнеры, в которых приложении могут существовать независимо от других приложений. Для более удобного развертывания воспользуемся Dockerfile – инструкциями по контейнеризации самого приложения. Dockerfile необходимо добавить на клиентскую и серверную часть. Dockerfile для серверной части представлен в листинге 5.1.

|  |
| --- |
| FROM python:3.9-alpine  RUN apk add --no-cache build-base libffi-dev postgresql-dev  WORKDIR /app  COPY requirements.txt .  RUN pip install --no-cache-dir -r requirements.txt  COPY . .  EXPOSE 8000  CMD ["daphne", "-p", "8000", "auction.asgi:application"] |

Листинг 5.1 – Dockerfile серверной части проекта

Здесь можно выделить следующий порядок действий:

* в начале загружается базовый образ Python 3.9 на основе Alpine Linux из DockerHub, обеспечивающий легкость и минимальный размер;
* затем устанавливаются необходимые пакеты для сборки и поддержки работы приложения: build-base, libffi-dev и postgresql-dev;
* создается рабочая директория, куда будет скопирован код приложения;
* копируется файл requirements.txt для установки зависимостей, и запускается команда для их установки;
* для запуска используется сервер daphne с указанием ASGI.

Следующим шагом будет создание файла Dockerfile для развертки клиентской части приложения. Он отображен в листинге 5.2.

|  |
| --- |
| FROM node:18-alpine  WORKDIR /app  COPY package\*.json ./  RUN npm install  COPY . .  EXPOSE 3000  CMD ["npm", "start"] |

Листинг 5.2 – Dockerfile клиентской части проекта

Порядок действий следующий:

* в начале загружаем образ из DockerHub платформы NodeJS типа;
* делаем каталог «app» текущим рабочим каталогом;
* копируем оба файла «package.json» и «package-lock.json» если есть;
* устанавливаем зависимости проекта;
* копируем файлы и каталоги проекта в текущий рабочий каталог;
* собираем приложение.

На данном этапе были созданы все необходимые docker-файлы, которые нужны для создания и запуска контейнеров Docker. Что бы не запускать все части проекта отдельно, был создан файл docker-compose.yml. В нем будет происходить процесс создания сервисов, настройка их окружения, конфигурационных файлов. Данный файл позволит запустить все части приложения через одну точку входа одной командой с помощью непосредственно докера.

Далее необходимо зайти в папку с файлом «docker-compose.yml», открыть консоль или PowerShell в этой папке и прописать команду «docker compose up». Через некоторое время приложение развернется на докере. После чего можно начинать работу с приложением. Результат развертывания можно увидеть на рисунке 5.26.

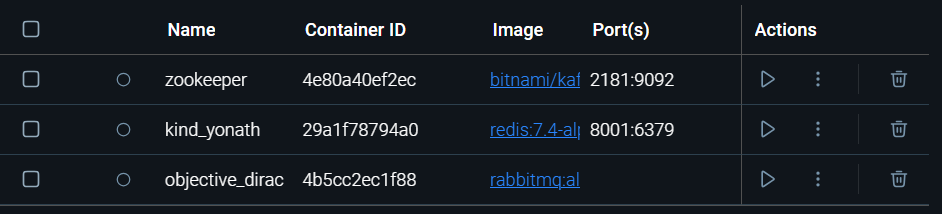


Рисунок 5.26 – Контейнер в Docker Desktop

Как видно, все 3 контейнера есть, критических ошибок при запуске обнаружен не было, значит развертывание прошло успешно.

5.3 Выводы по разделу

Руководство к приложению играет важнейшую роль в обеспечении удобства использования любого онлайн-аукциона. Даже при наличии интуитивно понятного интерфейса пользователи могут столкнуться с трудностями, связанными с незнанием специфики работы платформы или особенностей проведения торгов. Такие сложности могут замедлить процесс освоения функционала, привести к ошибкам или снизить удовлетворенность от использования приложения. Именно поэтому разработка подробных и доступных инструкций является необходимым этапом при создании любого веб-приложения, особенно в сфере электронной коммерции и аукционов.

Для покупателя предусмотрена инструкция, подробно описывающая ключевые этапы работы с системой. Пользователи могут узнать, как создать учетную запись, войти в систему и настроить свой профиль для удобства использования. Также руководство объясняет, как эффективно искать интересующие лоты, используя фильтры и параметры сортировки. Уделено внимание процессу участия в аукционах. Дополнительно рассмотрены аспекты взаимодействия с продавцом, включая обсуждение условий сделки и согласование доставки.

Руководство для продавца освещает все основные этапы работы с приложением, начиная с размещения лотов. В нем подробно описывается процесс добавления нового товара, включая загрузку фотографий, описание, установку начальной цены и сроков торгов. Также продавец может найти информацию о редактировании уже размещенных лотов, например, в случае необходимости обновления данных или изменения условий продажи.

Для администратора подготовлено руководство, которое фокусируется на управлении пользователями, анализе статистики и мониторинге работы приложения. Администратор получает доступ к инструментам для контроля всех действий в системе, включая управление учетными записями, просмотр логов и оперативное устранение возникающих проблем. Благодаря этим возможностям администратор может поддерживать стабильную работу платформы, отслеживать потенциальные угрозы и повышать эффективность работы приложения.

Отдельный раздел посвящен развертыванию приложения с использованием Docker. Это руководство описывает, как настроить контейнерную среду для автоматизации процессов запуска и управления приложением. Пользователи могут найти пошаговые инструкции по созданию и конфигурации контейнеров, что позволяет упростить процесс развертывания, обеспечить изоляцию окружений и повысить масштабируемость системы. Docker также помогает сократить время на настройку и сделать процесс обновления приложения более эффективным.

6 Технико-экономическое обоснование проекта

ФИО

Подпись

Дата

Лист

1

ДП 06.00.ПЗ

Разраб.

*Корень М.А.*

Пров.

Комкова А.В.

Н. контр.

Нистюк О.А.

Утв.

Блинова Е.А.

6 Технико-экономическое

Обоснование проекта

Лит.

Листов

9

БГТУ 1-40 05 01, 2025

Главной целью экономического раздела является экономическое обоснование целесообразности разработки программного средства (ПС), представленного в дипломном проекте. В данном разделе пояснительной записки проводится расчет затрат на всех стадиях разработки и расчет экономии основных видов ресурсов в связи с использованием данного веб-приложения.

Разработка проектов ПС требует разнообразных затрат и нередко значительных объемов ресурсов (трудовых, материальных, финансовых). В связи с этим, разработка и реализация каждого проекта должна быть обоснована, как с технической, так и экономической точки зрения.

6.1 Общая характеристика разрабатываемого программного средства

В рамках данного дипломного проекта разработано веб-приложение для организации и управления аукционами, доступное с любых устройств. Оно упрощает процессы создания и проведения аукционов.

Для администраторов и организаторов приложение предоставляет инструменты эффективного управления лотами, пользователями и торговыми процессами. В отличие от традиционных решений с ручным вводом данных или использованием электронных таблиц, разработка автоматически запускает чат между продавцом и покупателем сразу после завершения аукциона, облегчая обсуждение условий сделки и повышая прозрачность взаимодействия.

Для пользователей – продавцов и покупателей – приложение предоставляет удобные инструменты для создания аукционов, поиска интересующих лотов, участия в торгах посредством ставок, а также общения через встроенный чат, что повышает прозрачность и удобство проведения сделок. После завершения аукциона чат автоматически активируется для согласования условий передачи товара и оплаты.

Программное средство реализовано с использованием Django и Python на серверной стороне, React на клиентской, с базой данных PostgreSQL.

В функционал программного средства входят следующие возможности:

* регистрация в веб-приложении;
* авторизация в веб-приложении;
* поиск аукционов по заданным критериям;
* создание аукционов с описанием;
* участие в аукционах посредством ставок;
* использование чата для общения между продавцом и покупателем.

*Соболевский А. С.*

Консульт.

У

Способ монетизации – продажа разработанного программного средства и смежных прав на него заказчику.

6.2 Исходные данные для проведения расчетов и маркетинговый анализ

Исходные данные для расчета представлены в таблице 6.1. Данные взяты из нормативов, действующих на момент написания дипломного проекта.

Таблица 6.1 – Исходные данные для расчета

|  |  |  |
| --- | --- | --- |
| Наименование показателя | Условные  обозначения | Норматив |
| Численность разработчиков | Чр | 1 |
| Норматив дополнительной заработной платы | Ндз | 15 |
| Ставка отчислений в Фонд социальной защиты населения | Нфсзн | 34 |
| Ставка отчислений в БРУСП «Белгосстрах» | Нбгс | 0,6 |
| Норматив прочих прямых затрат | Нпз | 30 |
| Норматив накладных расходов | Нобп*,* обх | 30 |
| Ставка НДС | Нндс | 20 |
| Налог на прибыль | Нп | 20 |

Далее эти данные понадобятся для расчета всех необходимых параметров в процессе экономического анализа приложения.

6.3 Обоснование цены программного средства

Цена любого программного средства формируется из набора факторов, которые требуют различные трудовые, материальные и финансовые затраты.

Широкое применение вычислительных технологий требует постоянного обновления и совершенствования программных средств. Выбор эффективных проектов программных средств связан с их экономической оценкой и расчетом экономического эффекта, который может определяться как для разработчика, так и для пользователя.

У разработчика экономический эффект выступает в виде чистой прибыли от реализации ПС, остающейся в распоряжении организации, а у пользователя – в виде экономии трудовых, материальных и финансовых ресурсов, получаемой за счет следующих факторов:

* снижения трудоемкости выполнения расчетов, алгоритмизации процесса программирования и дальнейшей отладки разработанных программ;
* сокращения расходов на оплату машинного времени и других ресурсов на отладку программных средств;
* снижения расходов на материалы;
* оптимизации программных средств;
* улучшения показателей основной деятельности в результате использования передовых программных средств.

Стоимостная оценка программных средств у разработчиков предполагает

определение затрат, что включает следующие статьи:

* заработная плата исполнителей – основная и дополнительная;
* отчисления в фонд социальной защиты населения;
* отчисления по обязательному страхованию от несчастных случаев на производстве и профессиональных заболеваний;
* прочие прямые затраты;
* накладные расходы.

На основании затрат рассчитывается себестоимость и отпускная цена конечного программного средства.

6.3.1 Расчет затрат рабочего времени на разработку

Для начала необходимо определить время, затраченное на разработку программного средства. Этот показатель включает три ключевых этапа: анализ и проектирование, непосредственная разработка приложения, а также финальное тестирование. Все этапы учитываются в равной пропорции, так как каждый из них является неотъемлемой частью процесса создания качественного и конкурентоспособного программного продукта. Затраты рабочего времени на разработку программного средства представлены в таблице 6.2.

Таблица 6.2 – Затраты рабочего времени на разработку ПС

|  |  |
| --- | --- |
| Содержание работ | Затраты рабочего времени, часов |
| Рассмотрение аналогов | 8 |
| Проектирование базы данных | 24 |
| Проектирование серверной части приложения | 32 |
| Проектирование клиентской части приложения | 48 |
| Разработка серверной части приложения | 168 |
| Разработка клиентской части приложения | 256 |
| Тестирование приложения | 40 |
| Всего | 576 |

Наиболее трудоемкими затратами рабочего времени являются разработка серверной и клиентской части приложения. Их разработка заняла в сумме 424 часов времени, в клиентской части включена работа по верстке всего приложения, из-за чего клиентская часть обычно занимает на процент больше времени, чем серверная часть. Также дополнительно было выделено время на тестирование всего приложения для выявления неточностей в логик и работе приложения. Всего времени на все этапы ушло 576 часа рабочего времени.

6.3.2 Расчет основной заработной платы

Для определения величины основной заработной платы было проведено исследование уровня оплаты труда специалистов в сфере разработки на Django и React. Такие специалисты являются Full-stack разработчиками, что предполагает повышенную оплату труда благодаря специализации как на клиентской, так и на серверной части разработки. Источниками данных послужили открытые веб-порталы, профессиональные форумы, отчёты по рынку труда, а также общий средний уровень заработной платы в сфере информационных технологий.

По результатам анализа установлено, что средняя месячная заработная плата для junior-разработчиков составляет около 1600 рублей, что соответствует часовой ставке 9,52 руб./час.

Согласно таблице 6.2, проект проектировался, разрабатывался и тестировался одним специалистом на протяжении 576 часов. Основная заработная плата будет рассчитываться по формуле (6.1):

|  |  |  |
| --- | --- | --- |
|  | , | (6.1) |

где Соз – основная заработная плата, руб.;

Траз – время раработки, месяцев;

Краз – количество разработчиков, человек;

Сзп – средняя месячная заработная плата.

|  |
| --- |
| (руб.) |

Основная заработная плата разработчику составила 5483,52 руб.

6.3.3 Расчет дополнительной заработной платы

Дополнительная заработная плата на конкретное программное средство включает выплаты, предусмотренные законодательством о труде, и определяется по нормативу в процентах к основной заработной плате по формуле (6.2):

|  |  |
| --- | --- |
|  | (6.2) |

где Соз – основная заработная плата, руб.;

Ндз – норматив дополнительной заработной платы, %.

(руб.).

Дополнительная заработная плата разработчику составила руб.

6.3.4 Отчисления в Фонд социальной защиты населения

Отчисления в Фонд социальной защиты населения (ФСЗН) и по обязательному страхованию от несчастных случаев на производстве, и профессиональных заболеваний в БРУСП «Белгосстрах» определяются в соответствии с действующими законодательными актами по нормативу в процентном отношении к фонду основной и дополнительной зарплаты исполнителей и вычисляются по формуле (6.3):

|  |  |
| --- | --- |
|  | (6.3) |

где Соз – основная заработная плата, руб.;

Сдз – дополнительная заработная плата на конкретное ПС, руб.;

Нфсзн – норматив отчислений в Фонд социальной защиты, %.

Отчисления в БРУСП «Белгосстрах» вычисляются по формуле (6.4):

|  |  |
| --- | --- |
|  | (6.4) |

Таким образом, общие отчисления в БРУСП «Белгосстрах» составили 37,84 руб., а в фонд социальной защиты населения – 2144,06 руб.

6.3.5 Расчет суммы прочих прямых затрат

В таблице 6.3 представлена информация о прямых затратах в белорусских рублях по курсу Национального банка на 20 мая 2025 года, требуемых в ходе разработки программного средства.

Таблица 6.3 – Прямые затраты при разработке веб-приложения

|  |  |
| --- | --- |
| Содержание затрат | Значение, руб. |
| Оплата хостинга (1 год) | 290,00 |
| Покупка доменного имени (1 год) | 33,00 |
| Покупка сертификата безопасности | 60,00 |
| Оплата лицензии интегрированной среды разработки (3 месяца) | 126,00 |
| Покупка изображений блюд | 80,00 |

Затраты на хостинг, домен, а также разработку веб-приложения для аукционов рассчитываются с учётом годовой периодичности. Помимо этого, для визуального оформления приложения потребовались графические материалы, такие как изображения лотов и другие элементы дизайна для привлечения внимания пользователей. Все затраты были определены с учётом времени, затраченного на разработку, и рассчитаны на определённый период времени.

Сумма прочих затрат составит:

Спз = 290,00 + 33,00 + 60,00 + 126,00 + 80,00 = 589,00 руб.

Таким образом, сумма прочих прямых затрат составила 589,00 рублей.

6.3.6 Расчет суммы накладных расходов

Сумма накладных расходов Собп,обх – произведение основной заработной платы исполнителей на конкретное программное средство Соз на норматив накладных расходов в целом по организации Нобп,обх, по формуле 6.5.

|  |  |  |
| --- | --- | --- |
|  |  | (6.5) |

Сумма накладных расходов составит:

= 50 / 100 = 2741,76 руб.

Таким образом, сумма накладных расходов составила 2741,76 рублей.

6.3.7 Сумма расходов на разработку

Сумма расходов на разработку программного средства Ср определяется как сумма основной и дополнительной заработных плат исполнителей на конкретное программное средство, отчислений на социальные нужды, суммы на оплату машинного времени, суммы прочих затрат и суммы накладных расходов, по формуле (6.6).

|  |  |  |
| --- | --- | --- |
|  |  | (6.6) |

Все данные необходимые для вычисления есть, поэтому можно определить сумму расходов на разработку программного средства.

.

Сумма расходов на разработку программного средства была вычислена на основе данных, рассчитанных ранее, и составила 11818,71 рублей.

6.3.8 Определение цены, оценка эффективности

Отпускная цена рассчитывается на основании цены разработчика, которая формируется на основе показателя рентабельности продукции.

Рентабельность и прибыль по создаваемому программному средству определяются исходя из результатов анализа рыночных условий, переговоров с заказчиком (потребителем) и согласования с ним отпускной цены, включающей дополнительно налог на добавленную стоимость.

Так как даже приблизительные цены аналогов не выкладываются в публичный доступ, а в основном остаются в тайне, то для расчета отпускной цены воспользуемся вариантом на основе желаемого уровня рентабельности.

При маркетинговом анализе рынка аналогичных программных средств для организации и проведения аукционов можно сделать вывод, что рынок насыщен подобными решениями. Однако многие из них либо недостаточно известны, либо не обеспечивают удобство и прозрачность взаимодействия между продавцами и покупателями. Поскольку разрабатываемое программное средство создаётся по индивидуальному заказу и учитывает недостатки существующих аналогов, уровень рентабельности данного программного средства устанавливается на уровне 25%.

Прибыль от реализации вычисляется по формуле (6.7).

|  |  |  |
| --- | --- | --- |
|  |  | (6.7) |

где Урент – уровень рентабельности, %;

Сп – полная себестоимость программного средства, руб.

Чистая прибыль от реализации вычисляется по формуле (6.8).

|  |  |
| --- | --- |
|  | (6.8) |
| где Ппс – прибыль от реализации, руб.;  Нп – налог на прибыль, %. |  |

Цена разработчика программного средства без налогов находится по следующей формуле (6.9).

|  |  |  |
| --- | --- | --- |
|  |  | (6.9) |

Сумма налога на добавленную стоимость рассчитывается из соотношения (6.10).

|  |  |  |
| --- | --- | --- |
|  |  | (6.10) |

где Цр – цена разработки программного средства, руб.;

Нндс – ставка НДС, %.

Исходя из вышеописанных данных рассчитаем прибыль от реализации программного средства, чистую прибыль на основе прибыли от реализации, цену разработчика без налогов, сумму налогов на добавленную стоимость, а также планируемую отпускную цену с учетом НДС.

Ппс = 11818,71 25 / 100 = 2954,68 руб.

Пчист = 2954,68 0,8 = 2363,74 руб.

Цр = 11818,71 + 2954,68 = 14773,39 руб.

НДС = 14773,39 20 / 100 = 2954,68 руб.

Ц с НДС = 14773,39 + 2954,68 = 17728,07 руб.

Как итог, рыночная цена программного средства составляет 14773,39 руб., прибыль от реализации составила 2954,68 руб., чистая прибыль от разработки программного средства с учетом уровня рентабельности равным 25% составила 2363,74 руб., а его цена с учетом НДС – 17728,07 руб.

6.4 Выводы по разделу

В рамках данного раздела были проведены экономические расчеты, на основе которых была определена себестоимость разрабатываемого программного средства, а также прогнозируемая отпускная цена всего продукта путем продажи его заказчику и передаче прав на владение продуктом. Анализ такого вида позволяет определить целесообразность разработки приложения, издержки при разработке приложения и определить итоговую прибыль от продажи программного средства.

В таблице 6.4 представлены результаты расчетов для основных показателей данной главы в краткой форме, а также в приложении Е.

Таблица 6.4 – Результаты расчетов

|  |  |
| --- | --- |
| Наименование показателя | Значение |
| Время разработки, ч. | 576 |
| Количество разработчиков, чел. | 1 |
| Основная заработная плата, руб. | 5483,52 |
| Дополнительная заработная плата, руб. | 855,23 |
| Отчисления в Фонд социальной защиты населения, руб. | 2144,06 |
| Отчисления в БРУСП «Белгосстрах», руб. | 37,84 |
| Прочие прямые затраты, руб. | 589,00 |
| Накладные расходы, руб. | 2741,76 |
| Полная себестоимость, руб. | 11818,71 |
| Чистая прибыль, руб. | 2363,74 |
| Цена реализации (без НДС), руб. | 14773,39 |

Время разработки программного средства составило 576 часов, при этом работал один разработчик. Себестоимость программного средства составила 11818,71 рублей, при этом на заработную плату разработчика пришлось 5483,52 рублей, а на дополнительную заработную плату ­– 855,23 рублей. Цена реализации программного средства без НДС составила 14773,39 рублей. Чистая прибыль от продажи программного средства составила 2363,74 рублей. Исходя из этих данных, можно сделать вывод о целесообразности разработки программного средства, так как проект принесет чистую прибыль в умеренном размере.

Заключение

ФИО

Подпись

Дата

Лист

1

ДП 00.00.ПЗ

Разраб.

*Корень М.А.*

Пров.

Комкова А.В.

Н. контр.

Нистюк О.А.

Утв.

Блинова Е.А.

Заключение

Лит.

Листов

1

БГТУ 1-40 05 01, 2025

В ходе выполнения дипломного проекта по созданию веб-приложения для онлайн-аукционов были последовательно и тщательно решены все задачи, обозначенные во введении. На первом этапе была проведена комплексная и детальная аналитическая работа по изучению существующих аналогичных платформ и решений на рынке. В результате анализа были выявлены как сильные стороны конкурентов, так и их недостатки, что позволило сформировать чёткое понимание того, какие функции и особенности необходимо реализовать в собственном проекте для достижения конкурентных преимуществ и улучшения пользовательского опыта.

На основании полученных данных была спроектирована архитектура веб-приложения, отвечающая современным требованиям по масштабируемости, безопасности и удобству эксплуатации.

Реализация проекта включала поэтапную разработку серверной и клиентской частей приложения. Серверная часть была построена с учетом использования современных архитектурных паттернов, обеспечивающих гибкость и безопасность, в том числе с применением токенов для аутентификации и авторизации пользователей. Клиентская часть разработана с использованием React и сопутствующих библиотек, что позволило создать удобный, отзывчивый и интуитивно понятный интерфейс, адаптированный для различных категорий пользователей и устройств.

Особое внимание в ходе разработки уделялось удобству и функциональности для всех ролей пользователей, включая покупателей, продавцов и администраторов платформы. Проведённое тестирование охватило все ключевые сценарии работы системы, позволило выявить и устранить возможные ошибки и недочёты, гарантируя тем самым стабильность и надёжность работы приложения в реальных условиях.

Помимо технической реализации, была подготовлена подробная документация и руководство пользователя, которые предоставляют полное описание функционала и помогают новым пользователям быстро освоиться с интерфейсом и возможностями системы. Это обеспечивает высокий уровень поддержки и облегчает эксплуатацию программного продукта.

В результате выполненной работы было создано полнофункциональное, надежное и удобное в использовании веб-приложение для проведения онлайн-аукционов, полностью соответствующее целям дипломного проекта. Все поставленные во введении задачи были достигнуты в полном объёме, что подтверждает качество и эффективность проделанной работы.

Список использованной литературы

ФИО

Подпись

Дата

Лист

1

ДП 00.00.ПЗ

Разраб.

*Корень М.А.*

Пров.

Комкова А.В.

Н. контр.

Нистюк О.А.

Утв.

Блинова Е.А.

Список   
используемой литературы

Лит.

Листов

1

БГТУ 1-40 05 01, 2025

1. E-auction [Электронный ресурс] – Режим доступа: https://e-auction.by/­ – Дата доступа: 13.01.2025.
2. Elot [Электронный ресурс] – Режим доступа: https://elot.by/ – Дата доступа: 13.01.2025.
3. Torgi [Электронный ресурс] – Режим доступа: https://torgi.gov.by/ – Дата доступа: 13.01.2025.
4. Python [Электронный ресурс] – Режим доступа: https://docs.python.org/ ­– Дата доступа: 23.01.2025.
5. React – Getting Started [Электронный ресурс] – Режим доступа: https://reactjs.org/docs/getting-started.html – Дата доступа: 20.03.2025.
6. Axios [Электронный ресурс] – Режим доступа: https://axios-http.com/ru/docs/intro – Дата доступа: 24.03.2025.
7. React Router [Электронный ресурс] – Режим доступа: https://reactrouter.com/en/main – Дата доступа: 23.03.2025.

У

ПРИЛОЖЕНИЕ А.  
Трехуровневая архитектура веб-приложения

ФИО

Подпись

Дата

Лист

1

ДП 00.00.ПЗ

Разраб.

*Корень М.А.*

Пров.

Комкова А.В.

Н. контр.

Нистюк О.А.

Утв.

Блинова Е.А.

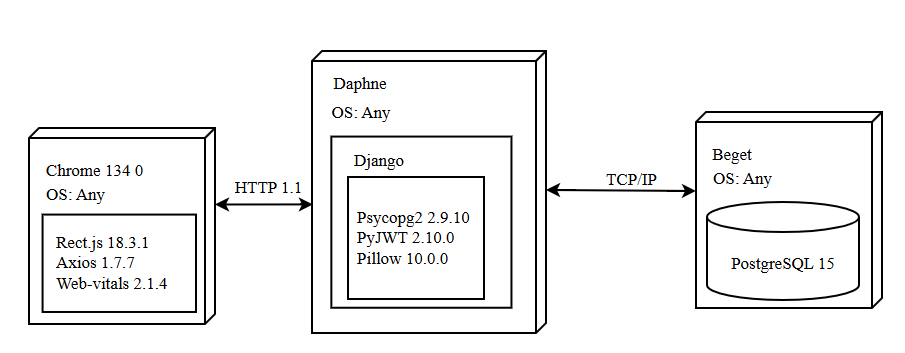
Приложение А

Лит.

Листов

1

БГТУ 1-40 05 01, 2025



У

ПРИЛОЖЕНИЕ Б.  
Логическая схема базы данных

ФИО

Подпись

Дата

Лист

1

ДП 00.00.ПЗ

Разраб.

*Корень М.А.*

Пров.

Комкова А.В.

Н. контр.

Нистюк О.А.

Утв.

Блинова Е.А.

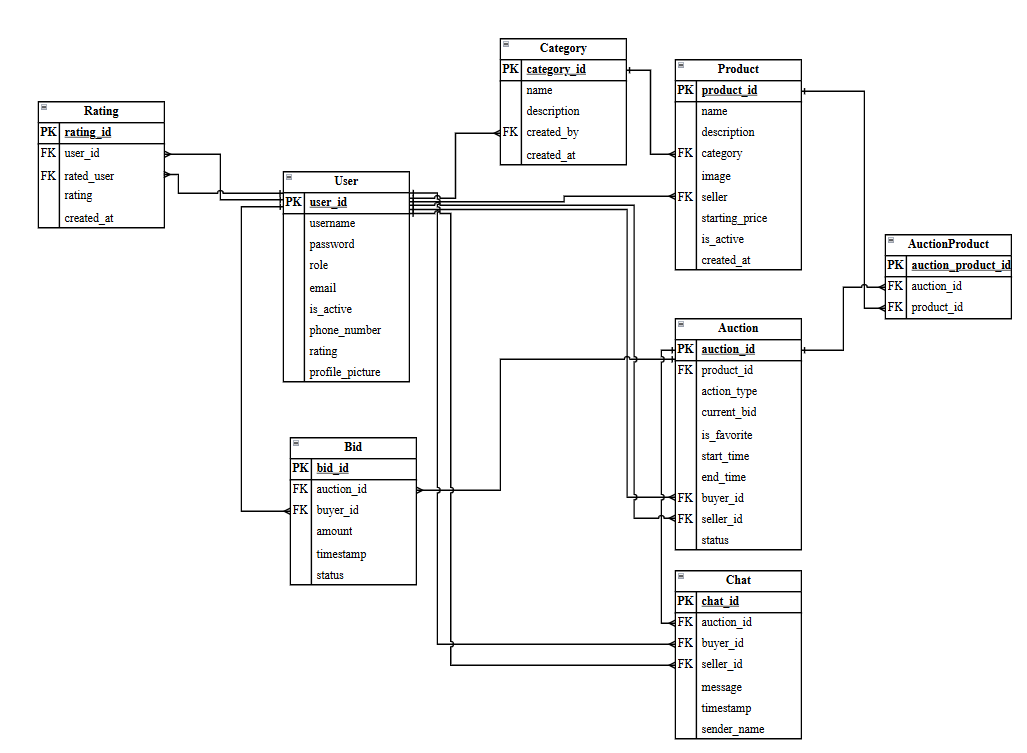
Приложение Б

Лит.

Листов

1

БГТУ 1-40 05 01, 2025



У

ПРИЛОЖЕНИЕ В.  
Диаграмма вариантов использования

ФИО

Подпись

Дата

Лист

1

ДП 00.00.ПЗ

Разраб.

*Корень М.А.*

Пров.

Комкова А.В.

Н. контр.

Нистюк О.А.

Утв.

Блинова Е.А.

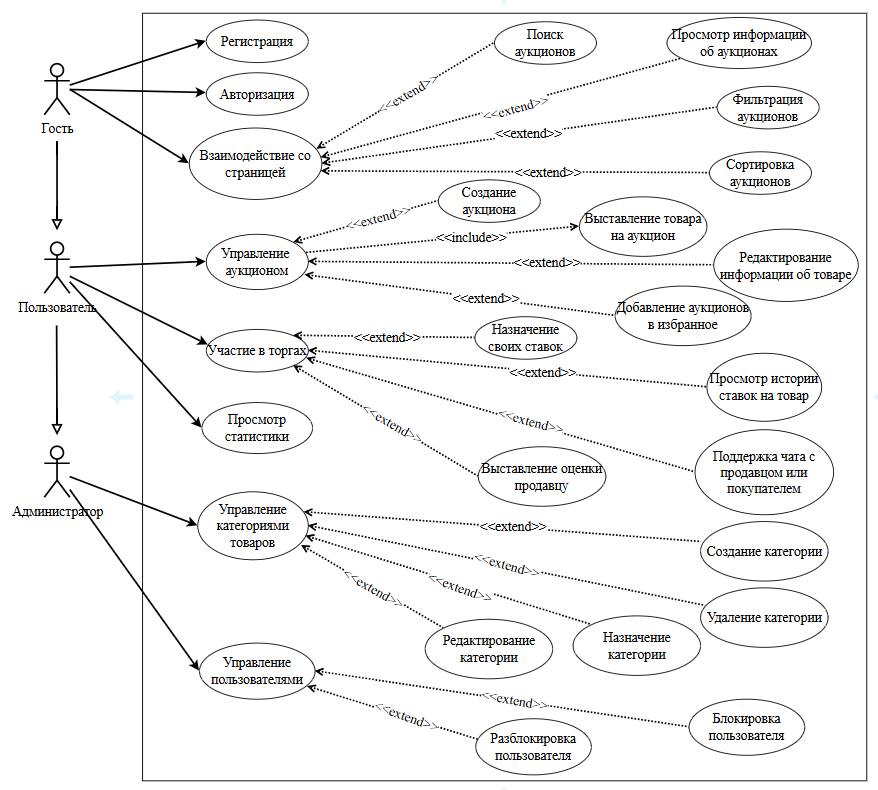
Приложение В

Лит.

Листов

1

БГТУ 1-40 05 01, 2025



У

ПРИЛОЖЕНИЕ Г.  
Процесс создания аукциона

ФИО

Подпись

Дата

Лист

1

ДП 00.00.ПЗ

Разраб.

*Корень М.А.*

Пров.

Комкова А.В.

Н. контр.

Нистюк О.А.

Утв.

Блинова Е.А.

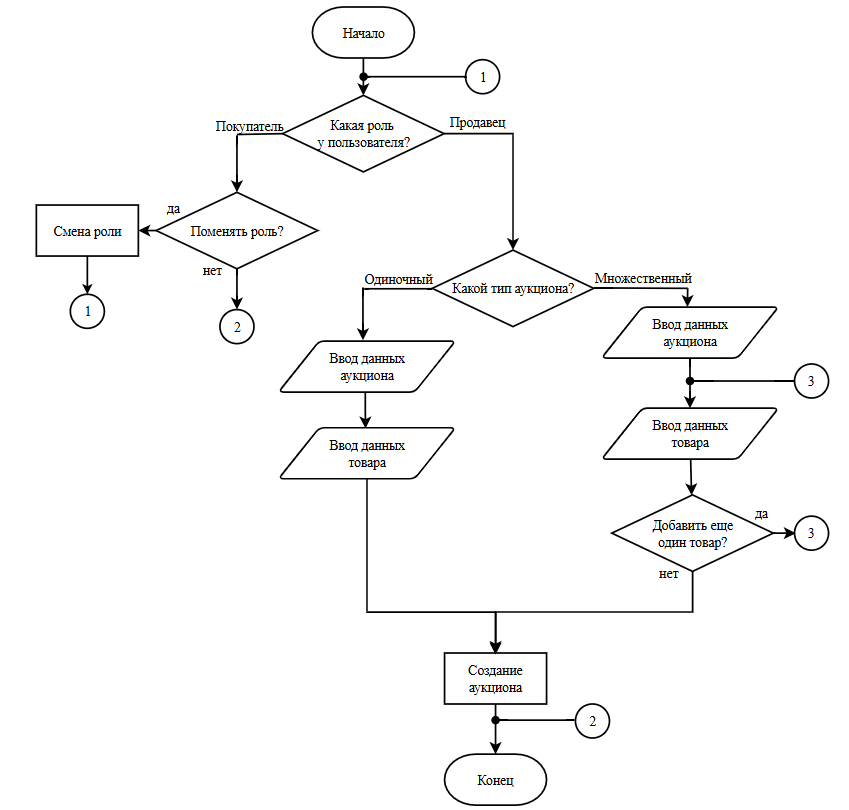
Приложение Г

Лит.

Листов

1

БГТУ 1-40 05 01, 2025



У

ПРИЛОЖЕНИЕ Д.  
Процесс проведения аукциона

ФИО

Подпись

Дата

Лист

1

ДП 00.00.ПЗ

Разраб.

*Корень М.А.*

Пров.

Комкова А.В.

Н. контр.

Нистюк О.А.

Утв.

Блинова Е.А.

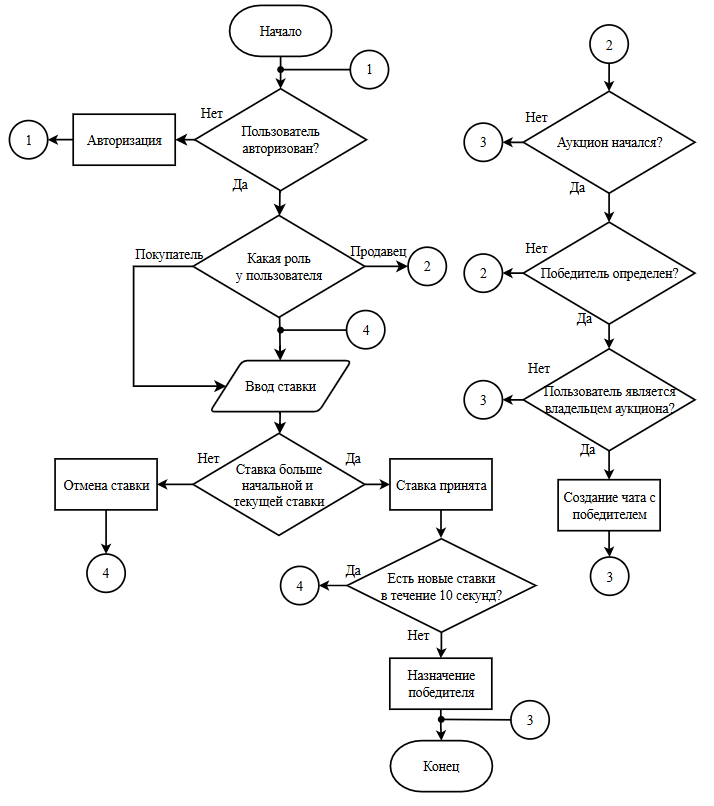
Приложение Д

Лит.

Листов

1

БГТУ 1-40 05 01, 2025



У

ПРИЛОЖЕНИЕ Е.  
Результаты подсчетов экономического обоснования

ФИО

Подпись

Дата

Лист

1

ДП 00.00.ПЗ

Разраб.

*Корень М.А.*

Пров.

Комкова А.В.

Н. контр.

Нистюк О.А.

Утв.

Блинова Е.А.

Приложение Е

Лит.

Листов

1

БГТУ 1-40 05 01, 2025

|  |  |
| --- | --- |
| Наименование показателя | Значение |
| Время разработки, ч. | 576 |
| Количество разработчиков, чел. | 1 |
| Основная заработная плата, руб. | 5483,52 |
| Дополнительная заработная плата, руб. | 855,23 |
| Отчисления в Фонд социальной защиты населения, руб. | 2144,06 |
| Отчисления в БРУСП «Белгосстрах», руб. | 37,84 |
| Прочие прямые затраты, руб. | 589,00 |
| Накладные расходы, руб. | 2741,76 |
| Полная себестоимость, руб. | 11818,71 |
| Чистая прибыль, руб. | 2363,74 |
| Цена реализации (без НДС), руб. | 14773,39 |

У