Содержание

Реферат

Бакалаврская работа состоит из 43 страниц, содержащих 10 источников,

12 рисунков, и 12 таблиц.

Объект исследования: Разработка ифнормационной системы торгов

Дипломная работа выполнена в текстовом редакторе LibreOffice Writer,

представлена в программе Adobe Acrobat в формате PDF.

Результат работы: Клиент серверное веб приложение с полным функцианалом для проведения торгов нумизматикой

Введение

Актуальность темы

Развитие онлайн-аукционов в последние годы стало глобальным трендом, однако в России подобные площадки, специализирующиеся на нумизматике, остаются малоразвитыми. Большинство торгов монетами происходит через форумы, соцсети или локальные клубы, что не обеспечивает прозрачности, безопасности и удобства для участников. Создание специализированной информационной системы для онлайн-торгов позволит структурировать этот рынок, предоставив коллекционерам и инвесторам надежный инструмент для покупки и продажи монет.

Кроме того, такая система может стать важным ценовым ориентиром для розничных продавцов. Поскольку аукционные цены отражают реальный спрос, магазины и частные продавцы смогут использовать данные о завершенных торгах для формирования справедливой стоимости монет. Это особенно актуально для редких экземпляров, где цена часто определяется аукционной конкуренцией. Также система даст нумизматическому сообществу возможность приобретать монеты до их попадания в массовую розницу, что особенно ценно для коллекционеров, ищущих раритеты.

Глава 1. Анализ предметной области и постановка задачи

1.1 Общая характеристика онлайн-торгов

Онлайн-торги представляют собой процесс купли-продажи товаров или услуг в формате аукциона с использованием интернет-технологий. Данный формат становится все более популярным благодаря высокой доступности, прозрачности и возможности привлечения широкой аудитории участников вне зависимости от их географического положения.

Современные онлайн-аукционы подразделяются на несколько видов:

- **Английский аукцион** — классическая форма, при которой участники поочередно повышают ставки до тех пор, пока не останется один победитель;
- **Голландский аукцион** — цена начинается с высокой и постепенно снижается до тех пор, пока кто-то не согласится на покупку;
- **Обратный аукцион** — используется чаще в сфере услуг, когда заказчики публикуют запрос, а исполнители соревнуются, снижая стоимость выполнения;
- **Закрытый аукцион** — участники делают ставки вслепую, не видя предложений конкурентов.

На рынке электронной коммерции существует множество платформ, реализующих аукционные механизмы. Наиболее известными являются:

- **eBay** — глобальный онлайн-аукцион с множеством категорий товаров, включая антиквариат и коллекционные предметы;
- **Catawiki** — специализированная платформа для торгов предметами коллекционирования, включая монеты, марки, произведения искусства;

– **Yahoo! Auctions**, **Heritage Auctions**, и другие — более узкоспециализированные платформы, часто ориентированные на локальные рынки.

Онлайн-аукционы обладают рядом преимуществ:

– Упрощение логистики и документооборота;

– Автоматизация всех этапов торгов;

– Возможность точной фиксации ставок и времени;

– Обширный выбор лотов для покупателей и широкая аудитория для продавцов.

Тем не менее, у данного подхода существуют и определённые недостатки:

– Вероятность мошенничества при отсутствии верификации участников;

– Проблемы с подлинностью товаров;

– Зависимость от технической стабильности платформы;

– Необходимость обеспечения кибербезопасности.

В сфере нумизматики аукционы особенно востребованы, так как они позволяют формировать справедливую рыночную цену за счёт конкурентных ставок. Однако при отсутствии специализированной платформы торги часто происходят стихийно — на форумах, в социальных сетях или через личные сообщения, что снижает уровень доверия к сделкам и затрудняет привлечение новых участников.

Разработка специализированной онлайн-системы торгов, учитывающей особенности нумизматического рынка, становится актуальной задачей для цифровизации данной ниши.

1.2 Особенности и проблемы проведения торгов по монетам

Торги нумизматическими объектами — это узкоспециализированный сегмент аукционного рынка, который имеет ряд уникальных характеристик, отличающих его от других видов онлайн-торгов. Монеты являются не только предметом коллекционирования, но и потенциальным объектом инвестиций, что требует от участников рынка высокой точности, прозрачности и доверия.

Особенности торгов по монетам:

1. **Уникальность лотов**

   Каждая монета обладает определённой степенью уникальности, зависящей от года выпуска, номинала, состояния (грейда), наличия браков, редкости и исторической ценности. Это делает ценообразование особенно чувствительным к качеству описания и визуальному представлению.

2. **Необходимость экспертной оценки**

   В отличие от массовых товаров, определить стоимость и подлинность монеты может только специалист. Поэтому доверие к платформе сильно зависит от возможности привлечения экспертов или предоставления сертификатов.

3. **Высокая вероятность подделок**

   На рынке нумизматики распространены фальшивки, реплики и копии, которые могут быть визуально неотличимы от оригиналов без глубокого анализа. Это создаёт дополнительные риски для покупателей.

4. **Часто — ограниченный круг участников.**

   Торги монетами, особенно редкими, проводятся в узком кругу коллекционеров и заинтересованных лиц. Такие торги требуют тонкой настройки интерфейса и коммуникаций для профессиональной аудитории.

5. **Ценность истории объекта.**

   Происхождение монеты, её владельцы, участие в предыдущих аукционах могут значительно влиять на её стоимость.

Проблемы, возникающие при существующих подходах:

1. **Отсутствие централизованной платформы.**

   Большинство торгов в России происходит через форумы, Telegram-чаты, социальные сети, что не позволяет обеспечить контроль, безопасность и единые стандарты.

2. **Недостаточная прозрачность.**

   Часто ставки размещаются вручную, без фиксации времени, что создаёт возможность манипуляций.

3. **Отсутствие автоматизации.**

   Закрытие торгов, уведомления участникам, определение победителя — всё это зачастую делается вручную, что снижает эффективность и вызывает споры.

4. **Нет гарантий безопасности сделки.**

   В отсутствие механизмов верификации и защиты интересов сторон, сделки могут срываться, а участники — сталкиваться с мошенничеством.

5. **Слабая правовая защищенность.**

   Так как большинство торгов проводится вне регулируемых площадок, отсутствует юридическая база для решения споров.

Таким образом, несмотря на высокий интерес к нумизматическим торгам, текущие методы их проведения не соответствуют современным требованиям безопасности, автоматизации и пользовательского удобства. Эти проблемы

могут быть решены за счёт внедрения специализированной информационной системы, учитывающей особенности предметной области

1.3 Требования к системе

Для эффективного функционирования информационной системы онлайн-торгов нумизматическими объектами необходимо определить функциональные и нефункциональные требования. Они формируют основу для проектирования архитектуры, интерфейсов и бизнес-логики системы.

Функциональные требования

1. **Регистрация и авторизация пользователей**

- Возможность регистрации с подтверждением электронной почты;
- Вход в систему с использованием логина и пароля;
- Хранение паролей в зашифрованном виде;
- Восстановление доступа.

2. **Работа с лотами**

- Создание лота с возможностью добавления изображений, описания, стартовой цены и срока завершения торгов;
- Просмотр всех активных лотов;
- Поиск и фильтрация по параметрам (категория, цена, дата окончания);
- Редактирование и удаление лота до начала торгов (только владельцем или администратором).

3. **Система ставок**

- Возможность делать ставки зарегистрированным пользователям;
- Автоматическое повышение текущей цены и сохранение истории ставок;

- Уведомление пользователей при перебитии ставки и завершении аукциона;
- Определение победителя по истечении времени торгов.

4. **Личный кабинет**

- Просмотр истории ставок и побед;
- Управление собственными лотами;
- Настройки профиля.

5. **Административная панель**

- Управление пользователями (блокировка, удаление);
- Модерация лотов (в том числе подозрительных или фальшивых);
- Управление категориями монет.

6. **История торгов и аналитика**

- Доступ к завершённым торгам и их результатам;
- Возможность отслеживания рыночной динамики цен на конкретные монеты;
- Статистика по активности пользователей и категориям лотов.

Нефункциональные требования

1. **Производительность**

- Система должна обеспечивать одновременную работу не менее 100 активных пользователей без существенного падения скорости.

2. **Безопасность**

- Использование защищённого протокола HTTPS;
- Токены для авторизации;

− Защита от SQL-инъекций и XSS-атак;

− Ограничение частоты запросов (rate limiting).

3. **Надёжность и устойчивость**

− Обработка ошибок на серверной и клиентской части;

− Сохранение данных в случае сбоя;

− Резервное копирование БД.

4. **Масштабируемость**

− Возможность дальнейшего увеличения нагрузки;

− Возможность интеграции с платёжными системами и сервисами оценки.

5. **Удобство использования**

− Удобный и интуитивно понятный интерфейс;

− Адаптация под мобильные устройства;

− Быстрая навигация по категориям и лотам.

6. **Локализация**

− Русский язык по умолчанию;

− Возможность дальнейшей реализации мультиязычного интерфейса.

Таким образом, требования к системе охватывают все основные аспекты её функционирования — от пользовательского взаимодействия до обеспечения безопасности и устойчивости работы. На основе этих требований в следующей главе будет разработана архитектура будущей платформы.

Глава 2. Проектирование информационной системы

2.1 Архитектура системы

Информационная система онлайн-торгов монетами реализована по принципу клиент-серверной архитектуры с чётким разделением backend- и frontend-частей. Это обеспечивает гибкость, масштабируемость и возможность дальнейшего развития системы. Для обеспечения более высокой степени информационной безопасности панель администратора вынесена в отдельный сервис, который никак не связан с основным кодом. В таком случае, злоумышленнику потребуется как минимум знать, где расположен адресс админ панели, но это можно предотвратить, если настроить сервер с обработкой админ запросов обрабатывать данные только от знакомых ip адресов.

Более подробно о реализации системы написано в Главе 3

Общие принципы архитектуры:

- **Backend:** реализован на языке программирования Python с использованием современного фреймворка FastAPI, который обеспечивает высокую производительность и поддержку асинхронных запросов.
- **Frontend:** представляет собой веб-клиент, написанный с использованием HTML/CSS/JavaScript
- **База данных:** MongoDB используется в качестве СУБД для хранения информации о пользователях, лотах, ставках и результатах торгов.
- **Хранение изображений:** изображения лотов загружаются и сохраняются в базе данных в виде закодированной строки в формате b64

- **Аутентификация и безопасность:** применяется Bearer токен для авторизации пользователей, с поддержкой ролей (пользователь, администратор).

Основные компоненты системы:

1. **Клиентская часть (Frontend):**

- Форма регистрации и входа;
- Интерфейс просмотра лотов;
- Страница лота с возможностью сделать ставку;
- Личный кабинет пользователя;
- Административная панель (доступна только админам).
- Страница с информацией о компании

2. **Серверная часть (Backend):**

REST API, реализующий бизнес-логику:

- Авторизация/аутентификация;
- Управление пользователями;
- Работа с лотами;
- Обработка ставок;
- Логика завершения торгов и определения победителя.
- Middleware для обработки ошибок и логирования;
- Планировщик, для автоматического завершения торгов в нужное время.

3. **База данных:**

- Таблицы: users, admins, auctions, tokens
- Индексы на ключевые поля для повышения производительности.

4. **Механизм уведомлений (опционально):**

- Отправка email-уведомлений при перебитии ставки или завершении аукциона;

Взаимодействие компонентов (в виде последовательности):

- Пользователь отправляет HTTP-запрос (например, POST /login);
- FastAPI обрабатывает запрос, проверяет данные, возвращает Bearer-токен;
- Клиент использует токен для последующих действий (например, POST /bid);
- Сервер проверяет валидность токена, обновляет данные в БД;
- В случае события (завершение торгов) система уведомляет участников и записывает результат.

Преимущества выбранной архитектуры:

- Высокая производительность благодаря FastAPI и асинхронной обработке;
- Гибкость расширения функционала (возможность добавления платёжных модулей, модерации, чата и пр.);
- Удобство сопровождения и отладки за счёт модульной структуры;
- Совместимость с различными клиентскими приложениями (в том числе мобильными).

2.2 Модель данных

В системе используется MongoDB — документно-ориентированная нереляционная база данных. Это позволяет гибко хранить и масштабировать данные без строгой схемы, что особенно удобно в системах с активным пользовательским взаимодействием, как в случае онлайн-аукционов.

В системе определены следующие основные коллекции:

1. clients — Пользователи

Содержит данные зарегистрированных клиентов, участвующих в торгах.

Таблица 1 — Пользователи

Поле	**Тип**	**Описание**
_id	ObjectId	Уникальный ID MongoDB
id	int	Внутренний числовой ID
nickname	string	Отображаемое имя
email	string	Адрес электронной почты
phone_number	string	Телефон
password	string (SHA256)	Захешированный пароль
avito_url	string	Ссылка на профиль/портфолио
email_verified	boolean	Подтверждение email
get_mails, mail_receive_	boolean	Настройки уведомлений

Связь: один пользователь может участвовать в неограниченном числе торгов.

2. auctions — Аукционы / Лоты

Представляет монету или набор монет, выставленных на торги.

Таблица 2 — Аукционы

Поле	**Тип**	**Описание**
_id	ObjectId	Уникальный ID
a_id	int	Числовой ID аукциона
short_name	string	Название лота
description	string	Подробности

Поле	Тип	Описание
start_price	float (string)	Стартовая цена
min_bid_step	float (string)	Минимальный шаг ставки
start_datetime	datetime	Время начала
end_datetime	datetime	Время окончания
bets	array	Вложенные документы ставок
bank	string (URL)	Ссылка на ЦБ/инфу о выпуске
photo	string (base64/path)	Фото монеты
is_active	boolean	Идут ли торги
deleted	boolean	Удалён ли лот

Связь: один аукцион содержит много ставок. Каждая ставка — это объект с nickname, bet_cost, clients_id, created_at.

3. admins — Администраторы

Таблица 3 — Аыдминистраторы

Поле	Тип	Описание
_id	ObjectId	ID администратора
email	string	Email администратора

Используется для разграничения доступа, например, к модерации лотов.

4. tokens — Токены сброса пароля / подтверждения

Таблица 4 — Токены

Поле	Тип	Описание
_id	ObjectId	ID документа
email	string	Email, к которому привязан токен

Поле	Тип	Описание
token	string	Сам токен
expire_at	datetime	Время истечения срока действия

Используется для восстановления доступа.

5. id_counters — Счетчики идентификаторов

Таблица 5 — Счетчики

MongoDB не использует автоинкременты, поэтому в системе реализован ручной счетчик.

Поле	Тип	Описание
_id	ObjectId	ID документа
a_id	int	Последний ID для аукционов
clients_id	int	Последний ID для пользователей
c_id	int	Дополнительный счётчик (например, ставок)

Связи между коллекциями (логические):

- clients.id → используется в auctions.bets.clients_id — чтобы связать пользователя и его ставку.
- auctions.a_id — можно использовать как внешний ID для ссылок, истории, аналитики.
- admins.email ↔ clients.email — пересечение возможно для прав администратора.

2.3 Диаграмма компонентов

Информационная система состоит из трех основных компонентов:

1. Веб-клиент (Frontend)

Назначение: предоставляет пользовательский интерфейс для участия в торгах, регистрации, авторизации, просмотра лотов и ставок.

- Технологии: HTML/CSS/JS (или Vue/React, если применимо)

Основной функционал:

- Регистрация и авторизация
- Просмотр текущих и завершённых аукционов
- Участие в торгах (ставки)
- Уведомления о победе или перебитой ставке

2. Backend-сервер (FastAPI)

Назначение: обработка бизнес-логики, маршрутизация запросов, управление пользователями, лотами и ставками.

- Технологии: Python, FastAPI

REST API эндпоинты:

- /auth/register, /auth/login
- /auctions/ — список активных аукционов
- /auctions/{id} — информация о лоте
- /bets/ — создание ставок
- /account— личный кабинет пользователя

Дополнительно:

- Проверка ролей (админ/пользователь)
- Расчёт победителя по окончании аукциона
- Уведомления на email

3. База данных (MongoDB)

Назначение: хранение информации о пользователях, лотах, ставках, сессиях, настройках уведомлений и ID-счетчиках.

Коллекции:

- clients — пользователи

– auctions — аукционы (с вложенными ставками)

– admins — список администраторов

– tokens — временные токены (например, сброс пароля)

– id_counters — ручное управление ID

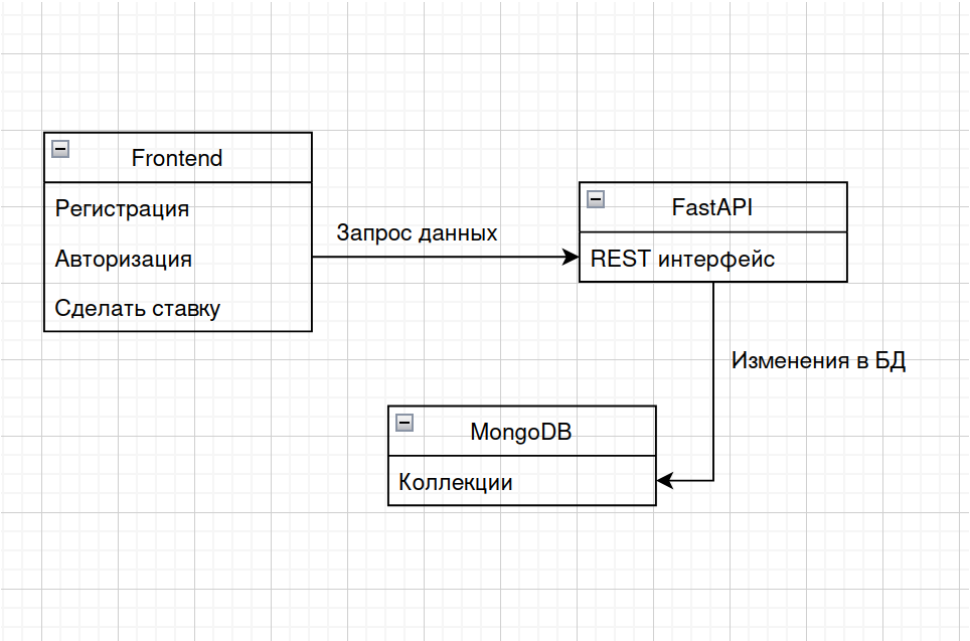


Рисунок 1 — Диаграмма компонентов

2.4 Бизнес-логика и пользовательские сценарии

Бизнес-логика системы онлайн-торгов монетами реализует ключевые процессы, обеспечивающие корректное проведение аукционов, взаимодействие пользователей и безопасность сделок. Основные компоненты:

1. Управление аукционами
   - Создание лота: продавец заполняет данные (название, описание, стартовая цена, сроки торгов), загружает фото. Система проверяет обязательные поля и присваивает уникальный ID.
   - Автоматическое завершение торгов: по истечении времени end_datetime система определяет победителя (участник с максимальной ставкой) и уведомляет всех заинтересованных.

2. Система ставок
   - Проверка минимального шага ставки: новая ставка должна быть выше текущей на min_bid_step
   - Обновление цены лота в реальном времени через WebSocket или polling.
   - Фиксация истории ставок для аудита (сохранение bet_cost, clients_id, created_at).

3. Безопасность и валидация
   - Только авторизованные пользователи могут делать ставки.
   - Только администраторы могут редактировать и создавать лоты
   - Панель администратора не связана с главным сайтом

4. Уведомления
   - Email-оповещения о:

- Перебитии ставки.
- Победе в аукционе.
- Начале/завершении торгов.

Пользовательские сценарии :

1. Сценарий 1: Регистрация и вход

Действия пользователя:

- Переходит на страницу регистрации.
- Заполняет форму (email, пароль, никнейм).
- Получает письмо с подтверждением.
- Входит в систему через логин/пароль.

Ответ системы:

- Создаёт запись в коллекции clients.
- Генерирует токен для авторизации.

2. Сценарий 2: Создание лота

Действия пользователя:

- Администратор в админ панели нажимает "Создать лот".
- Загружает фото монеты, указывает стартовую цену и сроки.
- Подтверждает размещение.

Ответ системы:

- Проверяет данные (например, start_price > 0).
- Сохраняет лот в коллекции auctions с is_active=true.

3. Сценарий 3: Участие в торгах

Действия пользователя:

- Просматривает активные лоты.
- Выбирает монету, вводит ставку (например, 10 000 руб.).

Ответ системы:

- Проверяет, что ставка >= current_price + min_bid_step.
- Обновляет цену лота и добавляет ставку в массив bets.
- Отправляет уведомление предыдущему лидеру.

4. Сценарий 4: Завершение аукциона

Действия системы:

- При достижении end_datetime выбирает победителя.
- Помечает лот как is_active=false.
- Отправляет победителю письмо с инструкциями по оплате.

5. Сценарий 5: Администрирование

Действия администратора:

- Просматривает список лотов через /admin/auctions.
- Удаляет лот по усмотрению.

Ответ системы:

- Меняет статус лота на deleted=true.
- Уведомляет владельца о причине удаления.

2.5 Безопасность и авторизация

Когда пользователь регистрируется в системе, данные попадают в базу данных и хранятся в зашифрованном виде. Если злоумышленник получит доступ к базе, он не сможет получить первоначальные данные пароля, так как те зашифрованны необратимым шифрованием SHA256. Трафик пересылаемый между клиентом и сервером зашифрован с использование TLS сертификатов и протоколу HTTPS.

Глава 3. Реализация системы

3.1 Используемые технологии

Приложение поделено на две части: клиентская и администраторская, которая работает отдельно от системы торгов. Для разработки веб приложения были выбранны следующие инструменты:

Серверная часть написана на python 3.12 с испольщованием фреймворка FastAPI. Выбор языка обоснован практичностью использования разработчиком. В этом наборе инструментов реализованы практически все необходимые сценарии, а недостающие компоненты можно импортировать благодаря полной совместимости пакетов. Также на серверной части используется JinjaTemplates — пакет для более комфортной индексации шаблонных файлов и используется для отправки статического контента на веб клиент. В качестве ORM для базы данных выбран ассинхронный движок Motor для MongoDB из пакета Pymongo. Для поддержки токенов авторизации используется FastAPILogin.

В качестве базы данных выбранна документоориентированная MongoDB. Ключевым решением для выбора этой СУБД стал формат хранения данных в базе. Традиционно в MongoDB используется BSON, который при ответе на запросы преобразовывается в JSON, родной формат для браузера. Вторым поводом выбрать эту СУБД стал тот, факт, что архитектура базы данных подразумевает в себе отношения между документами только в единичных случаях, поэтому возможность SQL-like баз в данном случае избыточны.

В качестве Фронтенд реализации я остановился на классическом наборе HTML, CSS, JavaScript. Фреймворки хоть и дают эффективность при разработке, но всегда проигрывают в скорости обработки браузером, ведь для того, чтобы фреймворк отработал, на клиент нужно также отправить и пакеты с фреймворком, которые часто могут в объеме превосходить написанный код в десятки раз. Отдельное внимание уделено безопасности, так как невнимательность при разработке клиентской части зачастую становится лазейкой для угрозы кибер атаки. Все поля ввода и запросы экранируются как на сервере, так и на клиенте, чтобы избежать атаки. На клиентскую часть не поступает никакой информации, которая могла бы описать размеры базы данных и данные пользователей.

Также выбранный подход написания дает возможность распределить ключевые части по Docker контейнерам и впоследствии подключить метрики. Такое развертывание является эталонным в современной разработке.

3.2 Основные модули системы

Основные модули серверной чатси:

- Точка входа приложения
- Инициализация параметров, с которыми веб сервер запускает обработчик
- Подключение всех сценариев запросов от клиента
- модуль для управления пользователями в БД
- модуль для управления аукционами в БД
- модуль отправки по протоколу SMTP
- модуль обеспечивающий авторизацию
- модуль для отправки статического контента
- модуль для CEO оптимизации для поисковых систем
- модуль для реализации панели администратора

3.3 Примеры REST-запросов

Для того, чтобы клиент мог взаимодействовать с сервером, необходимо определить, по каким адресам клиент сможет запрашивать информацию и какие данные возвращать. Для удобной документации API я использую стандарт OpenAPI. В таблице 6 приведу все эндпоинты сервера

Таблица 6 — Эндпоинты приложения

Метод	Путь	Требует токен?	Описание	Параметры / Тело запроса
POST	/auth/token	Нет	Логин (получение токена)	grant_type, username, password (form)
GET	/auth/get_data	Да	Получить данные пользователя	—
POST	/auth/logout	Да	Выход из аккаунта	—
POST	/auth/register	Нет	Регистрация	phone_number, password, nickname, email, avito_url
POST	/auth/check_mail	Нет	Проверить почту	mail (query param)
GET	/mongo/auction/get_all	Нет	Получить все аукционы	—
GET	/mongo/auction/get	Нет	Получить один аукцион	a_id (query param)
POST	/mongo/auction/add	Да	Добавить аукцион	JSON тело
POST	/mongo/auction/ add_bet_to_auction	Да	Добавить ставку в аукцион	a_id, bet_cost (query params)
POST	/mongo/auction/ update_auction	Да	Обновить аукцион	JSON тело
DELETE	/mongo/auction/delete	Да	Удалить аукцион	a_id (query param)
GET	/mongo/auction/ get_bets	Нет	Получить ставки по аукциону	a_id (query param)
GET	/mongo/auction/ get_time	Нет	Получить оставшееся	a_id (query param)

Метод	Путь	Требует токен?	Описание	Параметры / Тело запроса
			время аукциона	
GET	/mongo/clients/get_all	Да	Получить всех клиентов	—
GET	/mongo/clients/get	Нет	Получить одного клиента	clients_id (query param)
POST	/mongo/clients/add	Нет	Добавить клиента	phone_number, password, email, nickname, avito_url
POST	/mongo/clients/update	Да	Обновить клиента	JSON тело
POST	/mongo/clients/ change_password	Да	Сменить пароль клиента	JSON тело
POST	/mongo/clients/ active_clients	Да	Сделать клиентов активными	—
DELETE	/mongo/clients/ban	Да	Забанить клиента	id (query param)
POST	/mongo/clients/unban	Да	Разбанить клиента	id (query param)
POST	/mongo/clients/edit	Да	Редактировать клиента	JSON тело
GET	/robots.txt	Нет	robots.txt для SEO	—
GET	/sitemap.xml	Нет	sitemap.xml для SEO	—
GET	/mailservice/verify	Нет	Подтвердить email	token (query param)

3.4 Скриншоты пользовательского интерфейса

Для того, чтобы пользователю было приятно пользоваться информационной системой, скорости обработки данных недостаточно, также необходимо чтобы навигация по приложению была удобной, а UI/UX дизайн продуманным.

Чтобы решить проблему удобности, я начал опираться на решения в других системах, которые с торгами могут быть даже не связанными. В итоге было выбранно решение следующего типа: малый набор цветовой палитры, разделение активной области на две части, первая из которых вертикальное

навигационное меню в левой части области, и вторая, которая занимает 80% пространства и включает в себя область, в которую от контекста добавляется информация актуальная для конкретного отдела. Выбор навигационного меню вертикального типа обоснован тем, что в процессе жизни системы может потребоваться дополнение новыми элементами управления. Так как вертикальное меню легко пролистывается, его удобство использование с увеличением количества вложенных элементов не изменяется, в отличии от горизонтального меню на верху или внизу рабочего пространства.

Главная старница приложения содержит в себе информацию об аукционах, чернобелым цветом обозначены просшедшие, а активные имеют полную цветовую палитру.

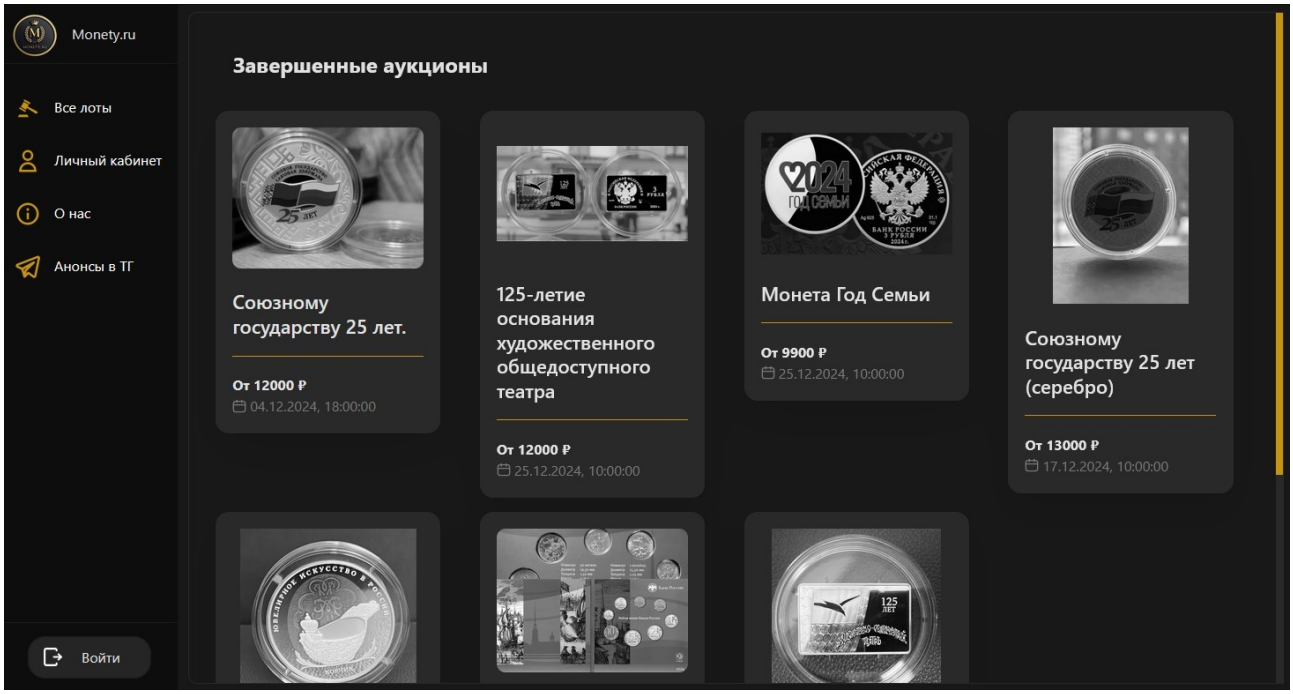


Рисунок 2 — главная страница приложения

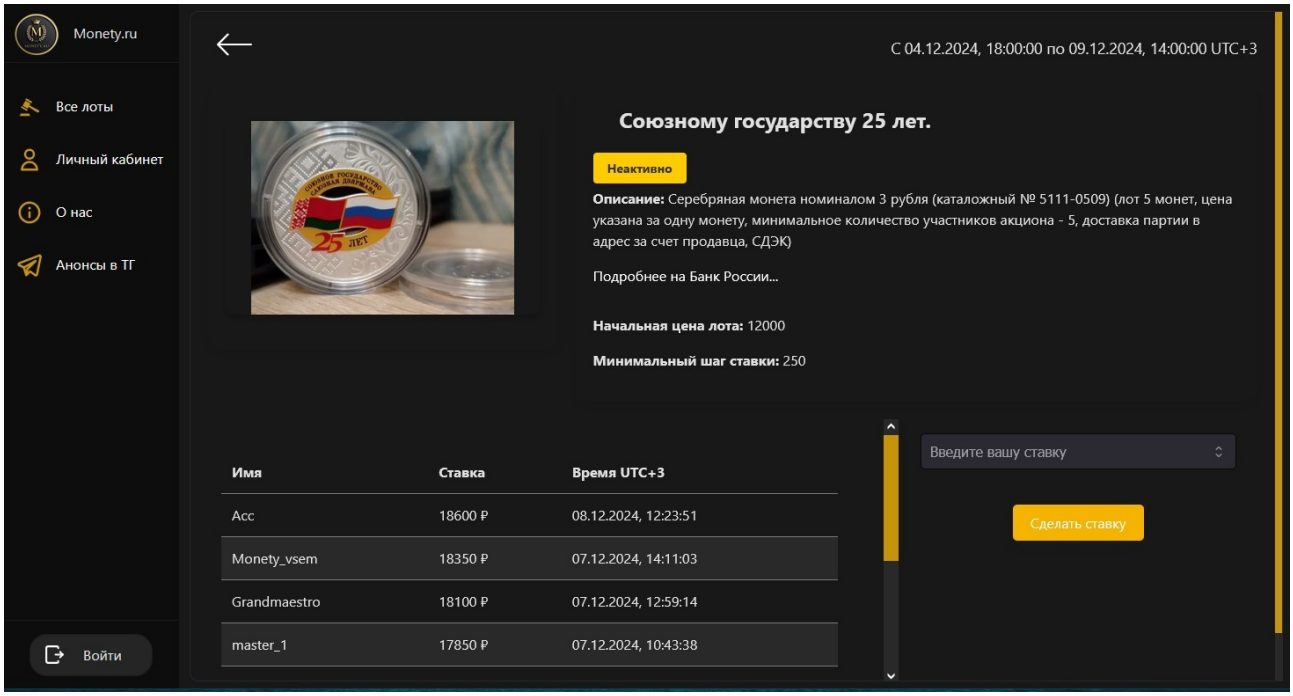


Рисунок 3 — Страница ставок

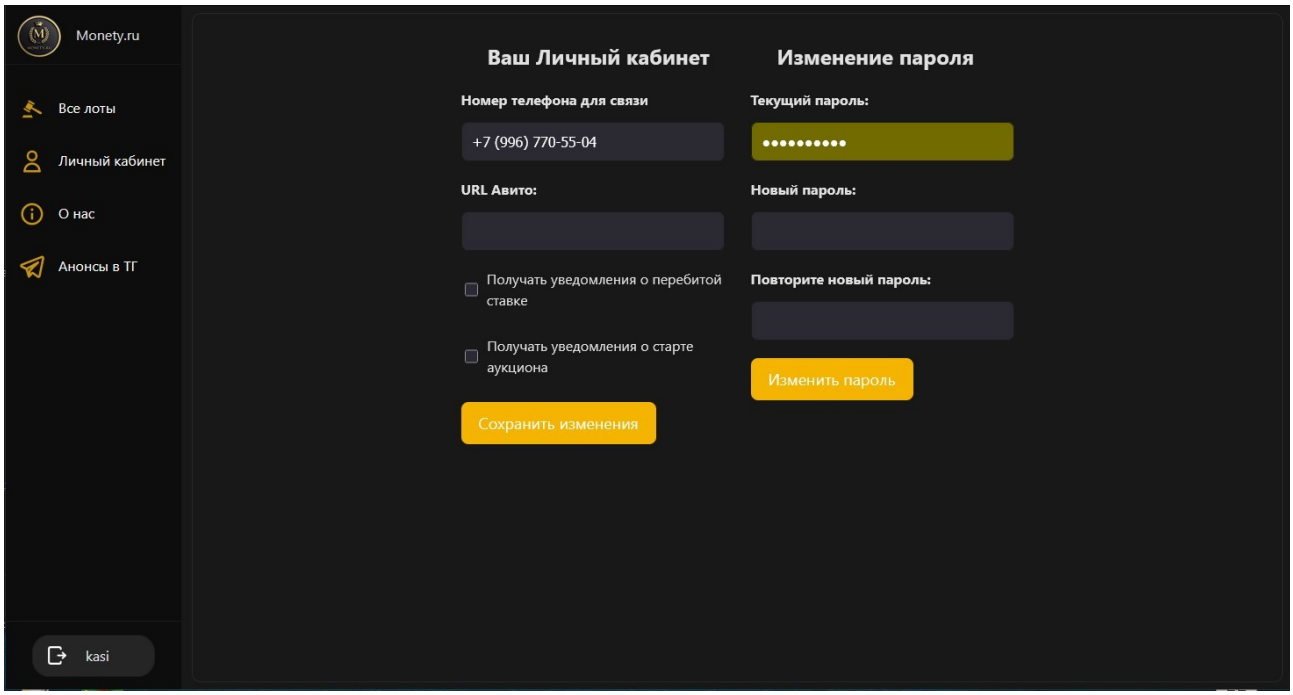


Рисунок 4 — Личный кабинет

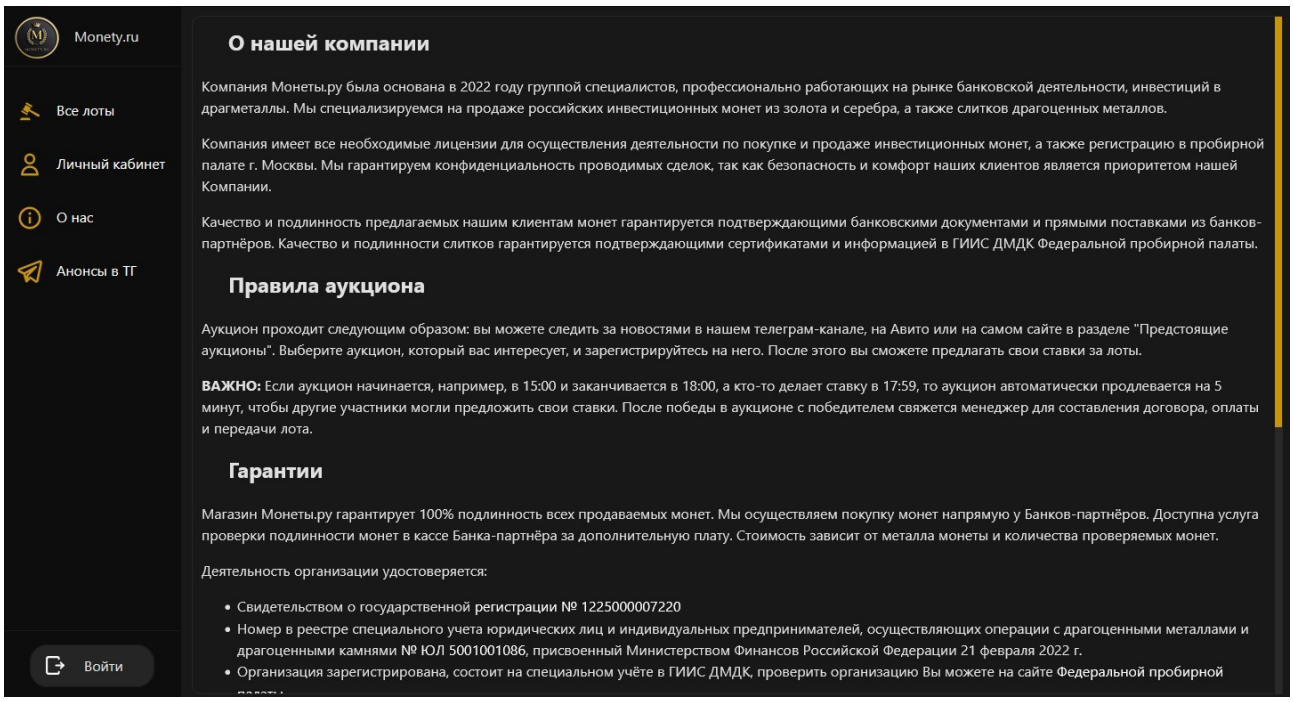
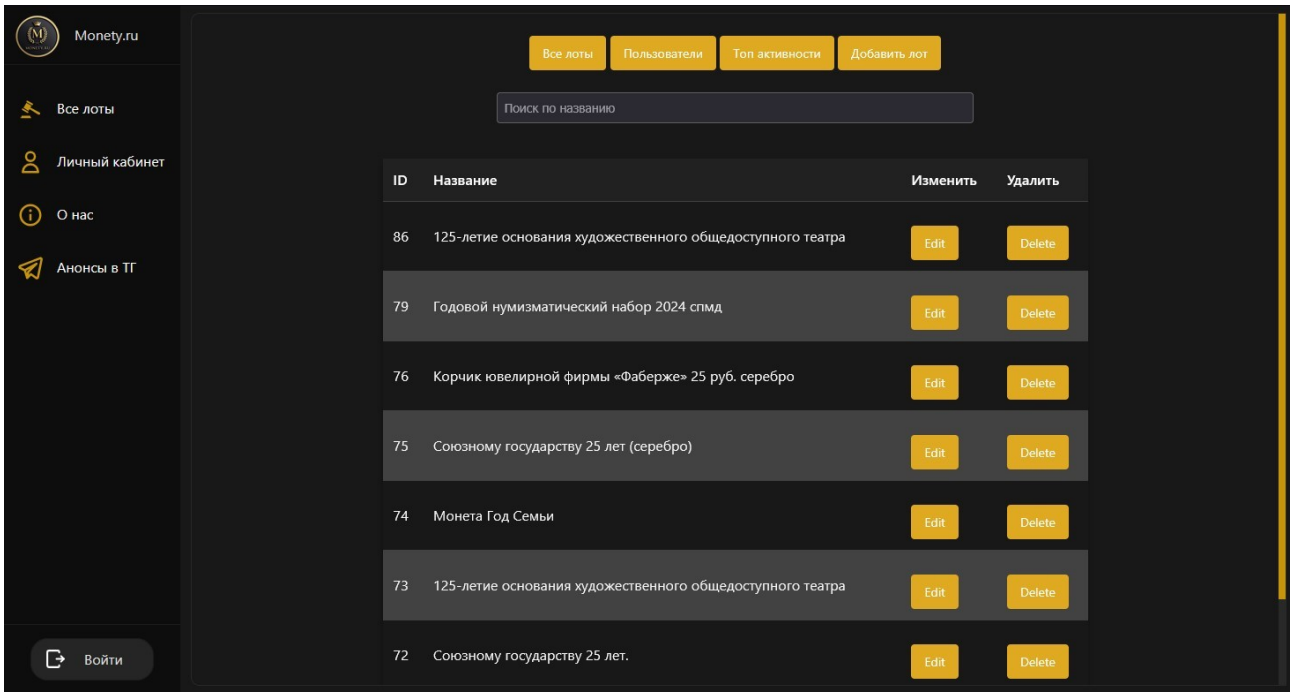


Рисунок 5 — страница с информацией о компании



Страница 6 — Панель администратора

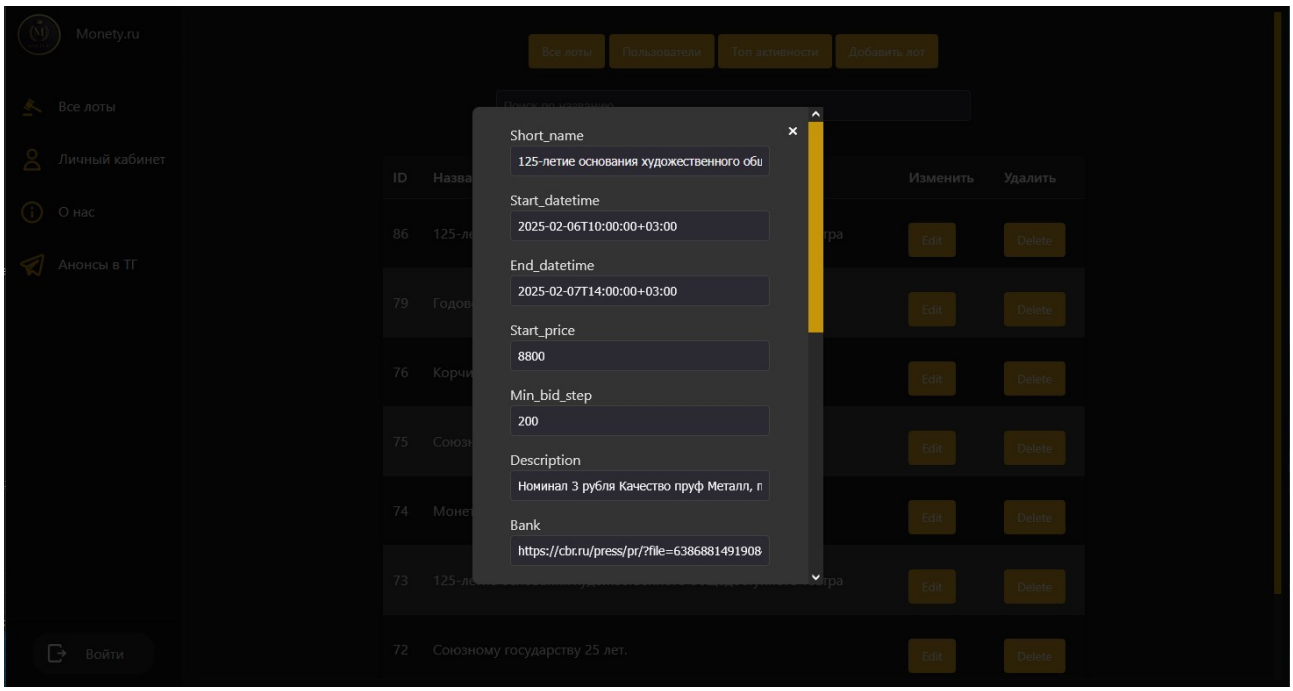


Рисунок 7 — окно для редактирования аукциона

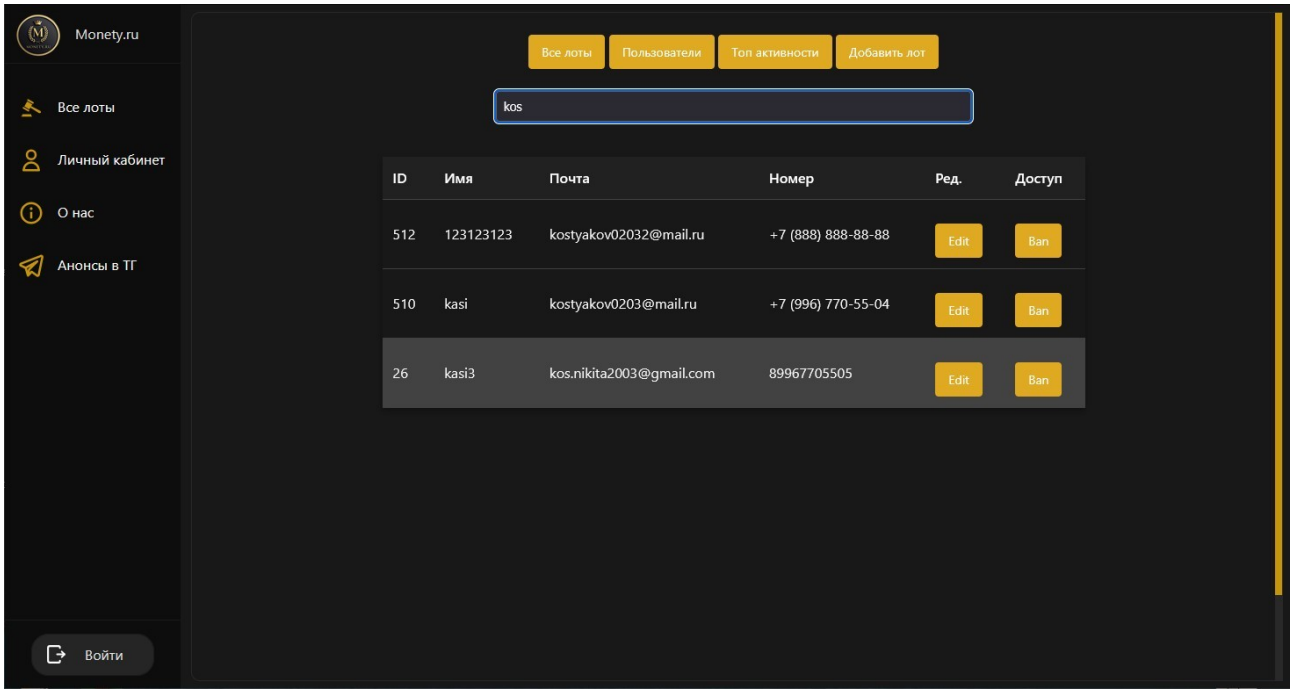


Рисунок 8 — использование поиска по пользователям

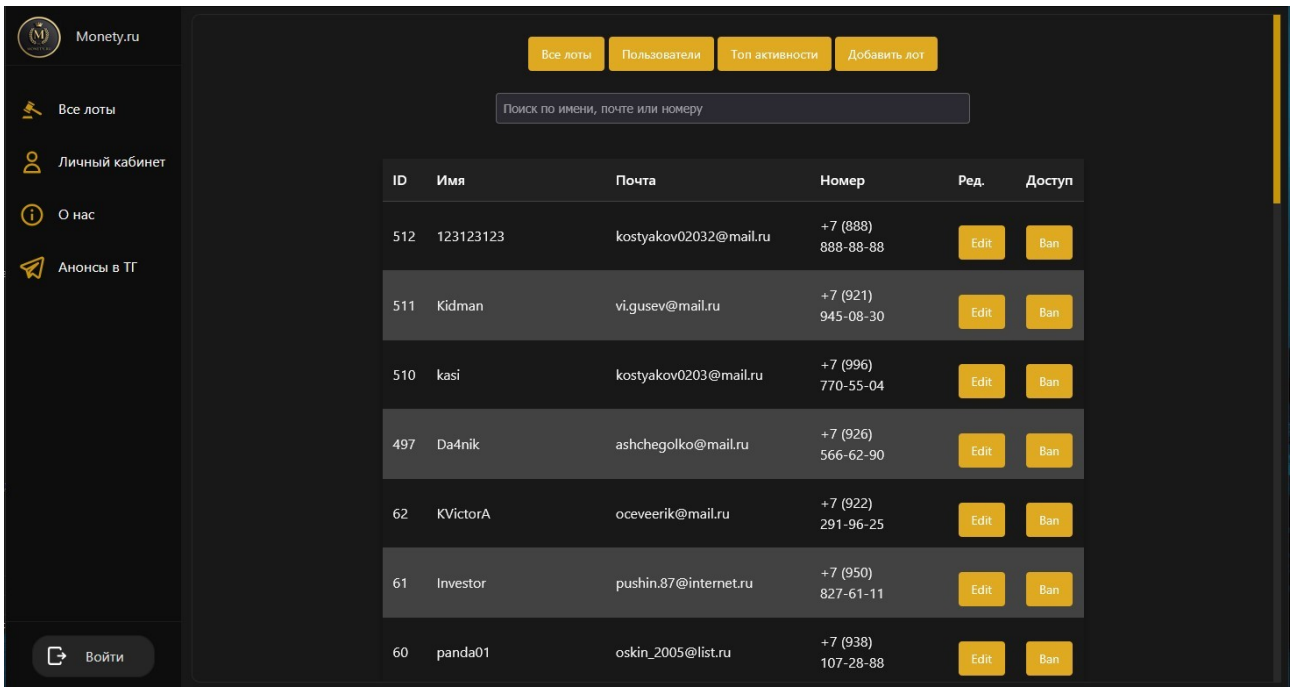


Рисунок 9 — Таблица всех полтзователей

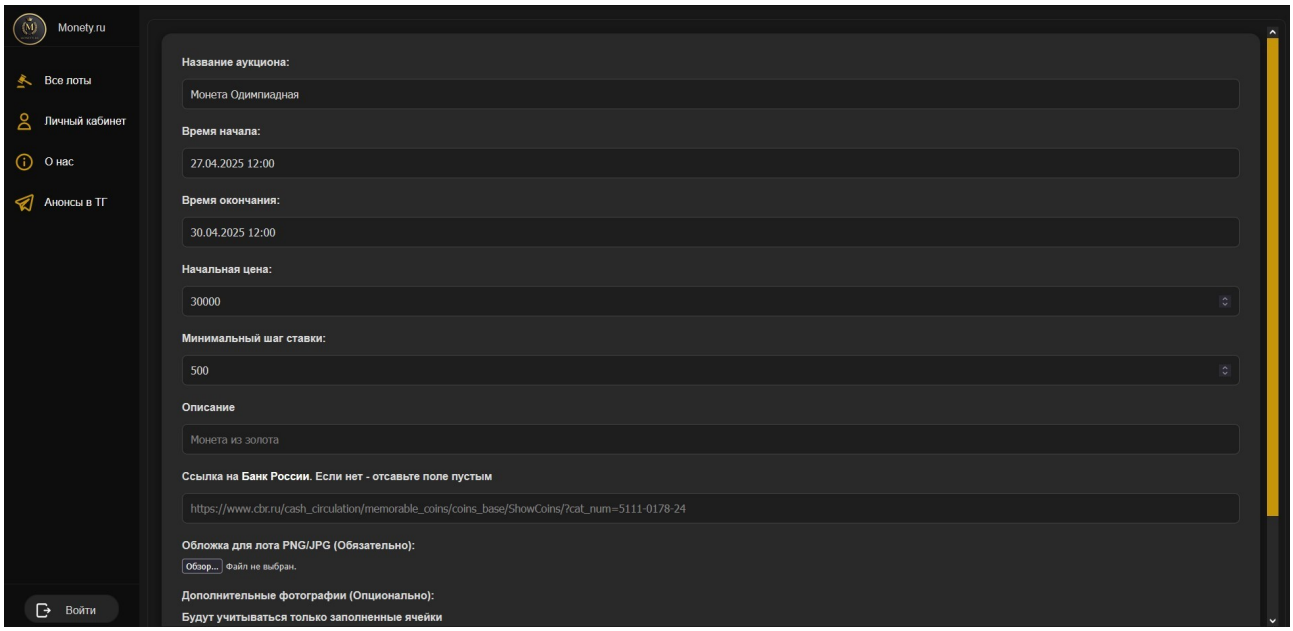


Рисунок 10 — Добавление аукциона

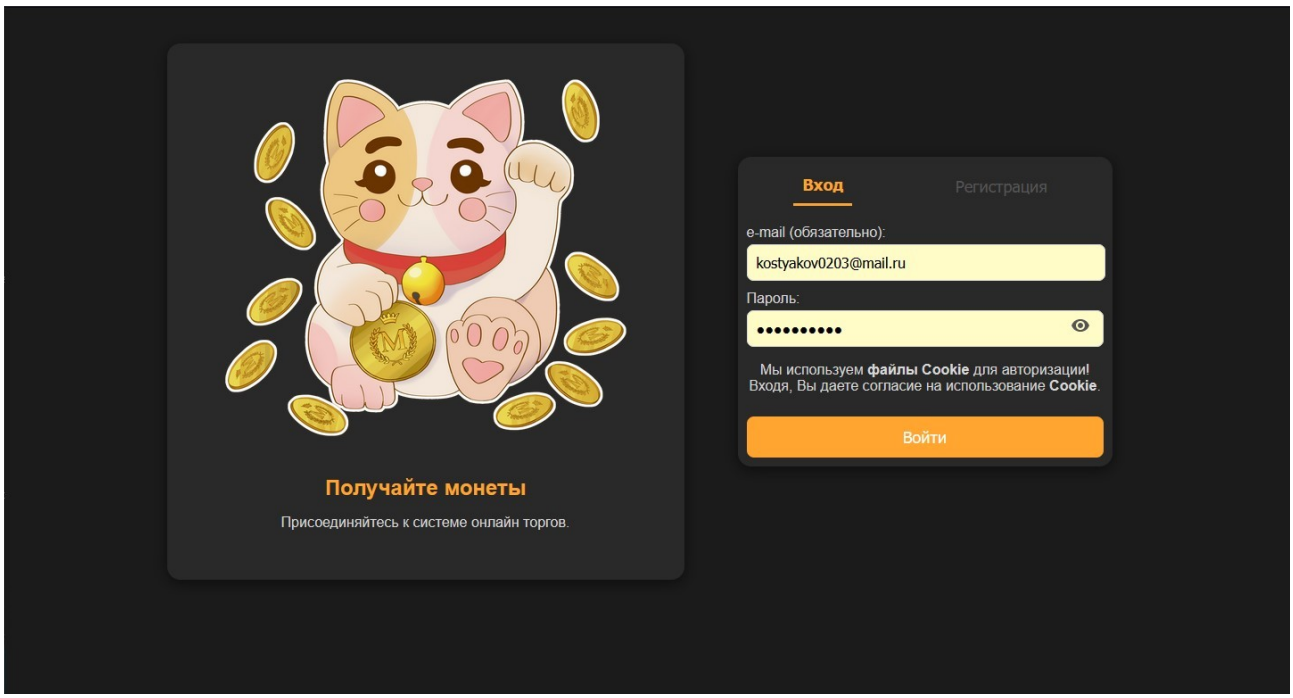


Рисунок 11 — Страница входа в систему

3.5 Тестирование и отладка

Для проверки корректности работы API предусмотрены несколько способов тестирования:

1. Использование Swagger UI

FastAPI автоматически генерирует **интерактивную документацию** по адресу:

http://localhost:8000/docs

В Swagger UI можно:

– Отправлять тестовые запросы к API.

– Проверять ответы и коды статусов.

– Передавать токен авторизации через кнопку **Authorize**.

2. Postman

Для более глубокой отладки рекомендуется использовать **Postman**:

– Импортировать коллекцию запросов.

– Устанавливать токены авторизации.

– Сохранять сценарии тестирования.

3. curl или httpie

Тестировать отдельные запросы из консоли:

Пример с помощью curl:

```
curl -X POST "http://localhost:8000/auth/token" \
-H "Content-Type: application/x-www-form-urlencoded" \
-d "username=user@example.com&password=string"
```

Пример с помощью httpie:

```
http POST http://localhost:8000/auth/token username=user@example.com
password=string
```

4. Логи сервера

Для отладки ошибок следует:

– Проверять вывод консоли, где запущен FastAPI-сервер.

– Логи содержат информацию об ошибках валидации, неправильных маршрутах и статусах ответов.

Глава 4. Экономическое обоснование

4.1 Расчёт затрат на разработку

Рассчет затрат на разработку основывается на следующих параметрах

Таблица 7 — Затраты на разработку

Этап работы	**Описание этапа**	**Оценка времени**	**Ставка (руб/час)**	**Стоимость (руб)**
Проектирование API	Разработка структуры маршрутов, авторизация, безопасности	8 ч	1500	12 000
Реализация функционала	Написание эндпоинтов, логики работы аукционов и клиентов	60 ч	1500	90 000
Разработка административной панели	Интерфейс управления аукционами и пользователями	15 ч	1500	22 500
Тестирование и отладка	Unit-тесты, интеграционное тестирование, исправление багов	10 ч	1500	15 000
Подготовка документации	Руководство пользователя, описание API	5 ч	1500	7 500
Развёртывание на сервере	Настройка сервера, деплой проекта	5 ч	1500	7 500

Итого:

- Общее время: 103 часа
- Общая стоимость: 154 500 руб

4.2 Оценка стоимости системы

Стоимость системы складывается из нескольких ключевых факторов:

Таблица 8 — Оценка стоимости системы

Компонент системы	**Описание**	**Ориентировочная стоимость (руб)**
Разработка программного обеспечения	Стоимость создания серверной части и админ-панели	154 500
Серверное оборудование / хостинг	Аренда VPS-сервера (12 мес. по 1000 руб/мес)	12 000
Системное и сервисное ПО	Подписки на сторонние сервисы (почтовые API, домен)	3 000
Техническая поддержка и сопровождение	Ежемесячная поддержка и обновления (12 мес. по 15000 руб/мес)	180 000

Итого:

- Начальная стоимость разработки и запуска: 158 000 руб
- Дополнительные расходы на 12 месяцев эксплуатации: 180 000 руб
- Полная стоимость системы за первый годовой цикл:
  338 000 руб

Примечания:

- В дальнейшем стоимость эксплуатации может быть снижена за счёт оптимизации серверных мощностей.
- Расходы на поддержку могут варьироваться в зависимости от количества доработок и пользовательской активности.
- При расширении функционала (например, интеграция платёжных систем) стоимость возрастёт.

4.3 Сравнение с альтернативами

При выборе решения для реализации системы управления аукционами были рассмотрены следующие альтернативные варианты:

Таблица 9 — Сравнение альтернативных вариантов разработки

Вариант	**Преимущества**	**Недостатки**	**Оценка стоимости (руб)**
Разработка на заказ (мой проект)	Полная адаптация под бизнес-процессы, гибкость, расширяемость	Требует времени на разработку и тестирование	154 000
Готовые SaaS-решения (например, AuctionSoftware, WeAuction)	Быстрый запуск, техподдержка включена	Высокая абонентская плата, ограниченные возможности кастомизации	от 15 000 руб/мес → 180 000 за 12 мес
Опенсорс-платформы (например, Sharetribe, OpenAuction)	Бесплатное ПО, доступ к коду	Требуются значительные доработки и сопровождение	70 000 (доработки + внедрение)
Разработка на конструкторах сайтов (Tilda, Wix + доп. скрипты)	Быстрая разработка интерфейса, низкий порог входа	Ограничения по функциональности, сложная интеграция со сторонними системами	40 000

Вывод:

- **Разработка собственной системы** является наиболее целесообразным вариантом при необходимости высокой гибкости, масштабируемости и контролируемых затрат на долгосрочную перспективу.
- **Готовые решения** целесообразны только для очень маленьких проектов с минимальными требованиями.
- **Опенсорс** — хорош для старта, но требует ресурсов для поддержки и развития.
- **Конструкторы** удобны для витринных сайтов, но не подходят для сложной бизнес-логики (аукционы, ставки, лоты).

4.4 Риски и способы их минимизации

Таблица 10 — Риски и способы их минимизации

Риск	**Возможные последствия**	**Способы минимизации**
Технические ошибки при разработке	Сбои в работе системы, потеря данных	Многоуровневое тестирование (юнит-, интеграционное), ревью кода
Недостаточная защита данных пользователей	Утечка персональной информации, штрафы	Использование SSL/TLS, шифрование данных, аудит безопасности
Невозможность масштабирования системы	Ограничение роста бизнеса	Проектирование масштабируемой архитектуры, использование облачных решений
Рост затрат на поддержку	Увеличение эксплуатационных расходов	Документирование системы, автоматизация мониторинга и развертывания
Низкая вовлеченность пользователей	Малая активность на платформе	Упрощение пользовательского интерфейса, маркетинговые кампании
Проблемы с интеграцией сторонних сервисов	Ограничение функциональности	Выбор стабильных API-провайдеров, создание резервных планов
Задержки в сроках разработки	Срыв запуска проекта	Agile-методология, регулярные планерки, контроль выполнения задач

Основная стратегия минимизации рисков:

- Использование **гибких методов управления проектами** (Scrum, Kanban);
- Проведение **регулярных внутренних проверок качества**;
- Постоянная **обратная связь с тестовой группой пользователей**;
- Создание **резервных копий базы данных** и планов аварийного восстановления.

Глава 5. Защита данных и безопасность

5.1 Актуальные угрозы

Таблица 11 — актуальные угрозы

Угроза	**Возможные последствия**	**Способы защиты**
Несанкционированный доступ	Кража данных, управление аккаунтами	Аутентификация, авторизация, логирование
Атаки на отказ в обслуживании (DDoS)	Недоступность сервиса	Использование анти-DDoS сервисов, фильтрация трафика
Утечка персональных данных	Нарушение закона о защите данных, штрафы	Шифрование хранения и передачи данных
Вредоносное ПО на стороне клиента	Кража данных пользователей	Защита от XSS и CSRF атак
Взлом API-интерфейсов	Неавторизованный доступ к функциям системы	Ограничение доступа, использование токенов
Фишинговые атаки на пользователей	Потеря данных для входа	Обучение пользователей, двухфакторная аутентификация
Ошибки конфигурации серверов и баз данных	Потеря целостности или доступности системы	Регулярный аудит конфигураций, автоматизированные проверки
Уязвимости сторонних библиотек и фреймворков	Компрометация всей системы	Постоянное обновление зависимостей, мониторинг CVE

5.2 Средства защиты

Важные направления защиты:

- Принцип минимально необходимого доступа;
- Мониторинг событий безопасности в реальном времени;

– Регулярное обновление ПО и систем безопасности;

– Резервное копирование данных и план восстановления после инцидентов.

5.3 Регламент хранения и обработки данных

Для обеспечения безопасности, целостности и законности хранения данных в системе вводятся следующие правила:

Таблица 12 — Регламенты обработки данных

Параметр	**Регламент**
Категории обрабатываемых данных	Персональные данные (ФИО, email, телефон), данные о ставках и активности пользователей, служебные данные системы
Срок хранения персональных данных	Минимально необходимый для целей обработки. Обычно 1 год после удаления аккаунта, если иное не требуется законодательством
Хранилище данных	Защищённые базы данных на сервере с шифрованием на уровне хранения (например, MongoDB с включённым шифрованием)
Резервное копирование	Ежедневное автоматическое резервное копирование с хранением копий в зашифрованном виде на отдельном сервере в течение 30 дней
Передача данных	Только через защищённые каналы (HTTPS, SSL/TLS)
Доступ к данным	Ограниченный: только авторизованные сотрудники с необходимым уровнем доступа
Удаление данных	По запросу пользователя или по истечении срока хранения, с использованием безопасных методов удаления
Обработка данных третьими лицами	Только на основании договоров, гарантирующих соблюдение требований законодательства о защите данных
Журналирование событий	Ведение логов доступа и изменения данных с сохранением в течение 6 месяцев для анализа инцидентов безопасности
Соответствие законодательству	GDPR (при наличии пользователей из ЕС), ФЗ-152 (о персональных данных), другие применимые нормы

Дополнительные положения:

– Пользователи имеют право на доступ к своим данным, их исправление и удаление.

– Регулярные проверки и аудит процедур обработки данных.

– Обработка специальных категорий данных (например, биометрических) запрещена без явного согласия пользователя.

Заключение

Выводы по результатам работы

В ходе выполнения дипломной работы была разработана современная веб-система для управления аукционами и клиентскими данными, оснащённая административной панелью. Основные итоги работы можно сформулировать следующим образом:

- Реализован полный цикл REST API для управления аукционами и пользователями, включая создание, обновление, удаление и получение данных.
- Разработана безопасная система аутентификации и авторизации с использованием протокола OAuth2 и менеджера сессий.
- Создана удобная и функциональная административная панель для управления ресурсами системы.
- Проведено тестирование API-запросов для проверки корректности всех реализованных функций.
- Оценена стоимость разработки и эксплуатации системы, произведено сравнение с альтернативными решениями.
- Проанализированы риски и разработаны мероприятия по их минимизации.
- Установлены регламенты хранения и обработки персональных данных в соответствии с современными стандартами безопасности и требованиями законодательства.

В результате система отвечает требованиям надёжности, безопасности и масштабируемости, а её архитектура позволяет быстро расширять функциональность при необходимости.

Возможности развития проекта

В дальнейшем проект можно развивать не только в сторону улучшения аукционной составляющей, но и добавлять новые способы торговли монетами, например разработать раздел для розничной продажи. Также помимио реализации английской системы, можно добавить и альтернативные способы. Когда клиентская база достигнет достаточного количества пользователей и камуникация с ними будет достаточно объемной, можно подключить шлюз для работы с CRM системами, а также разработать внутриплатформенный чат или единый инструмент для общения с клиентами через популярные мессенджеры

Список литературы

1. **Документация по FastAPI**:
    - Sebastián Ramírez. *FastAPI Documentation*. URL: https://fastapi.tiangolo.com/
2. **Python 3.12**:
    - Python Software Foundation. *Python 3.12 Documentation*. URL: https://docs.python.org/3.12/
3. **Jinja2**:
    - Armin Ronacher. *Jinja2 Documentation*. URL: https://jinja.palletsprojects.com/
4. **MongoDB**:
    - MongoDB, Inc. *MongoDB Manual*. URL: https://www.mongodb.com/docs/
5. **Motor (Асинхронный драйвер для MongoDB)**:
    - Mark J. Bostic, David Beasley. *Motor Documentation*. URL: https://motor.readthedocs.io/
6. **Docker**:
    - Docker, Inc. *Docker Documentation*. URL: https://docs.docker.com/
7. **Официальная документация Pymongo**:
    - Python Software Foundation. *PyMongo Documentation*. URL: https://pymongo.readthedocs.io/
8. **Javasript и базовые веб-технологии**:
    - MDN Web Docs. *HTML, CSS, and JavaScript Guide*. URL: https://developer.mozilla.org/en-US/docs/Web
9. **Использование MongoDB в веб-приложениях**:
    - Kristina Chodorow. *MongoDB: The Definitive Guide*. O'Reilly Media, 2013.
10. **Проектирование RESTful API с использованием FastAPI**:
    - *Building RESTful APIs with FastAPI*