МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РЕСПУБЛИКИ КАЗАХСТАН

Казахский национальный исследовательский технический университет
имени К.И.Сатпаева

Институт автоматики и информационных технологий

Кафедра "Программная инженерия"

Расторгуев А. С.

Разработка веб-приложения для проведения онлайн-аукционов

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА
к дипломному проекту

Специальность 6B06102 - Computer Science

Алматы 2022

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РЕСПУБЛИКИ КАЗАХСТАН

Казахский национальный исследовательский технический университет
имени К.И.Сатпаева

Институт автоматики и информационных технологий

Кафедра «Программная инженерия»

ДОПУЩЕН К ЗАЩИТЕ
Заведующая кафедрой ПИ
канд. физ-мат. наук, профессор
_____ А. Н. Молдагулова
"_26_" _____05_____ 2022 г.

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА
к дипломному проекту

На тему: "Разработка веб-приложения для проведения онлайн-аукционов"

Специальность 6B06102 - Computer Science

Выполнил
Расторгуев А.С.

Рецензент
Доктор Phd, доцент
_____ Утегенова А.У.
"_____" _____ 2022 г.

Научный руководитель
Ассоциированный профессор
_____ М.Н. Жекамбаева
"_23_" _____05_____ 2022 г.

Алматы 2022

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РЕСПУБЛИКИ КАЗАХСТАН

Казахский национальный исследовательский технический университет имени К.И.Сатпаева

Институт автоматики и информационных технологий

Кафедра "Программная инженерия"

6B06102 - Computer Science

УТВЕРЖДАЮ
Заведующая кафедрой ПИ
канд. физ-мат. наук, профессор
_____ А. Н. Молдагулова
"26" 05 2022 г.

ЗАДАНИЕ
на выполнение дипломного проекта

Обучающемуся *Расторгуеву Алексею Сергеевичу*
Тема: *Разработка веб-приложения для проведения онлайн-аукционов*
Утверждена *приказом проректора по академической работе* № 489-П/О *от «24» 12 2021 г.*
Срок сдачи законченного проекта *«26» 05 2022 г.*

Исходные данные к дипломному проекту: *Техническая документация по применению технологии, техническое задание, описание БД для хранения информации в виде ER-диаграммы.*
Перечень подлежащих разработке в дипломном проекте вопросов:
а) Разработка дизайна веб-приложения с последующим его преображением во front-end с использованием фреймворка React;
б) Создание модели базы данных и ее реализация в СУБД PostgreSQL;
в) Разработка серверной части сайта с использованием NodeJS
г) Тестирование полученного проекта и его выгрузка на хостинг
Перечень графического материала (с точным указанием обязательных чертежей): *представлены 20 слайдами презентации.*
Рекомендуемая основная литература: *из 21 наименования.*

ГРАФИК

подготовки дипломного

Наименование разделов, перечень разрабатываемых вопросов	**Сроки представления научному руководителю и консультантам**	**Примечание**
1. Анализ предметной области, разработка технического задания на проектирование проекта	16.01.2022	Выполнено
2. Выбор технологий для разработки	07.02.2022	Выполнено
3. Разработка дизайна для приложения	12.02.2022	Выполнено
4. Реализация дизайна	20.02.2022	Выполнено
5. Разработка базы данных	01.03.2022	Выполнено
6. Разработка серверной части сайта	10.03.2022	Выполнено
7. Тестирования приложения	11.04.2022	Выполнено
8. Написание пояснительной записки к дипломному проекту	29.04.2022	Выполнено

Подписи

консультантов и нормоконтролера на законченный дипломный проект с указанием относящихся к ним разделов проекта

Наименования разделов	Консультанты, Ф.И.О. (уч. степень, звание)	Дата подписания	Подпись
Нормоконтролер	Жекамбаева М.Н. Доктор Ph.D., ассоциированный профессор	23.05.22	
Программное обеспечение	Марғұлан Қ. Магистр техн.наук, лектор	23.05.22	

Научный руководитель _____ Жекамбаева М.Н.

Задание принял к исполнению обучающийся _____ Расторгуев А.С.

Дата "17" 11 2021 г.

АННОТАЦИЯ

Данный дипломный проект посвящен разработке веб-приложения проведения онлайн-аукционов.

Различные вещи, время которых, казалось бы, давно прошло, хранятся у многих дома, в подвалах, на полках шкафов, куда никто не заглядывает и так далее. И ведь с возрастом, эта вещь может приобрести статус раритета и классики. И, возможно, многие люди захотят обладать этой вещью. Для этого они начинают искать это на аукционах, где подобные ценности порой уходят за внушительные суммы. Но сегодня, когда все делается через интернет, уже мало кто ходит на живые аукционы и многие выбирают их онлайн аналоги.

В представленном проекте, для разработки front-end использовалась библиотека React, крайне эффективный инструмент для создания пользовательских интерфейсов. Для хранения всей необходимой информации, была использована реляционная СУБД PostgreSQL. Для серверной части сайта использовалась технология NodeJS, позволяющая создавать надежный back-end и имеющая широкую поддержку в IT сообществе.

АНДАПТА

Бұл бітіру жобасы онлайн аукциондарға арналған веб-қосымшаны әзірлеуге арналған.
Уақыты әлдеқашан өтіп кеткен сияқты, көптеген үйлерде, жертөлелерде, ешкім көрмейтін шкафтардың сөрелерінде және т.б. Өйткені, жасы ұлғайған сайын бұл зат сирек және классика мәртебесіне ие болуы мүмкін. Және, мүмкін, көптеген адамдар бұл нәрсеге ие болғысы келеді. Мұны істеу үшін олар оны аукциондарда іздей бастайды, онда мұндай құндылықтар кейде әсерлі сомаларға кетеді. Бірақ бүгінде, барлығы интернет арқылы жасалған кезде, аз адам тікелей аукциондарға барады және көпшілігі өздерінің онлайн әріптестерін таңдайды.

Ұсынылған жобада интерфейсті әзірлеу үшін пайдаланушы интерфейстерін құрудың өте тиімді құралы болып табылатын React кітапханасы пайдаланылды. Барлық қажетті ақпаратты сақтау үшін PostgreSQL реляциялық ДҚБЖ пайдаланылды. Сайттың серверлік бөлігі үшін сенімді бэк-энд құруға мүмкіндік беретін және АТ қауымдастығында кең қолдауға ие NodeJS технологиясы қолданылды.

ANNOTATION

This graduation project is devoted to the development of a web application for online auctions.

Various things, the time of which, it would seem, have long passed, are stored in many houses, in basements, on the shelves of cabinets where no one looks, and so on. And after all, with age, this thing can acquire the status of a rarity and a classic. And, perhaps, many people will want to own this thing. To do this, they begin to look for it at auctions, where such values sometimes go for impressive amounts. But today, when everything is done via the Internet, few people go to live auctions and many choose their online counterparts.

In the presented project, the React library, an extremely effective tool for creating user interfaces, was used to develop the front-end. To store all the necessary information, the PostgreSQL relational DBMS was used. For the server part of the site, NodeJS technology was used, which allows you to create a reliable back-end and has wide support in the IT community.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Сегодня в интернете продается всё, что можно представить, и чаще всего по фиксированной, установленной изначально цене. Но бывают случаи, когда человек не знает, сколько стоит та, или иная вещь, находящаяся у него в руках. А эта вещь может быть достаточно ценной. В таком случае он может выставить ее на аукцион, или же, как это более привычно сегодня, на онлайн-аукцион. Но имеющиеся на данный момент площадки, могут оказаться сложными в использовании для неподготовленного пользователя и не имея желания разбираться он может попытаться найти другую площадку, в которой будет более дружелюбный и понятный интерфейс, будут иметься только те функции, которые необходимы. Создание именно такой площадки является целью данного проекта.

1 Исследовательский раздел

1.1 Цель разработки

Основной задачей данного проекта является реализация и предоставление конечным пользователям возможности участия в онлайн-аукционах.

Веб-приложение является интернет-площадкой, где любой человек может принять участие в аукционе, побороться на торгах за интересующий его товар, путем перебивания ставок других участников аукциона. Помимо этого, любой пользователь может устроить свои собственные торги, и выставить на аукцион имеющийся у него товар. Ключевой целью является удобство пользователя, и упрощение самой процедуры участия в онлайн-аукционах.

1.2 Определения, термины и сокращения

Таблица 1 – Сокращения, термины и их определения

Сокращение или термин	Определение
СУБД	Система управления базами данных
БД	База данных
HTML	Язык гипертекстовой разметки
CSS	Каскадная таблица стилей
JavaScript	Мультипарадигменный язык программирования, основной язык в этом проекте
Node.js	Программная платформа
React.js	JavaScript-библиотека для разработки пользовательских интерфейсов
Фреймворк	Программный продукт, имеющий большое количество встроенных функций, которые упрощают создание больших и нагруженных проектов
Activity diagram	Диаграмма активности – наглядно показывает возможный сценарий использования приложения
ER-диаграмма	Entity relationship - Схематическое изображение модели базы данных, с указанием всех сущностей и связей между ними

1.3 Аналоги и изучаемая область

Изучая область, с которой предстоит работать, я просмотрел некоторые известные интернет-платформы, которые предлагают функционал, связанный с онлайн-аукционами. Важный момент в том, что они предоставляют подобный функционал, вместе с другими различными функциями. И пользователь, которому необходимо именно выставить свою вещь на аукцион, может не разобраться в большом количестве предлагаемых возможностей. Основным аналогом на сегодняшний день является компания Ebay и их одноименный интернет-магазин. На данном сайте помимо всех основных функций интернет-магазина есть так же и такая опция как выставить какую-либо вещь на торги в формате онлайн-аукциона. Имеется большое количество различных настроек, которые никак не относятся к изучаемой области, которые могут отпугнуть пользователя. И было принято решение в создании платформы, единственной специализацией которой, будут именно интернет-аукционы.

2 Технологический раздел

2.1 Графический редактор

В рамках данной дипломной работы, для создания дизайна разрабатываемого веб-приложения использовался онлайн-сервис Figma. Он позволяет не использовать громоздкие программы по типу Adobe Photoshop, и имеет в себе все необходимые функции и инструменты для создания интерфейсов. Основой инструментарий представлен на рисунке 2.1.1

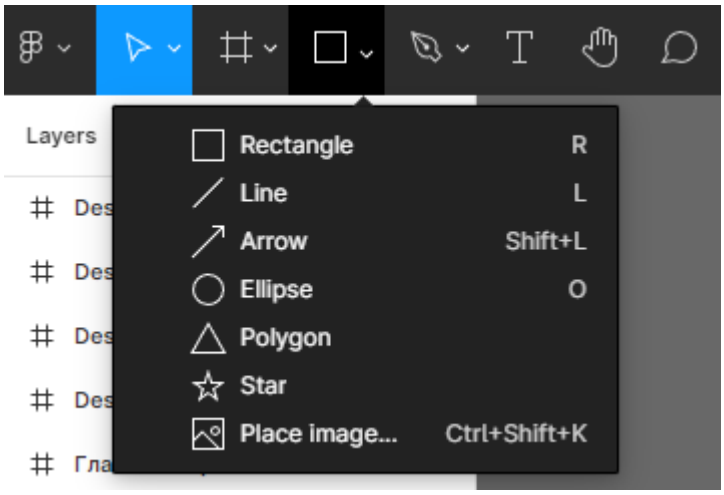

Рисунок 2.1.1 - Основные инструменты в сервисе Figma

2.2 Редактор кода

Для написания кода был выбран редактор Microsoft Visual Code, он совмещает в себе гибкий инструментарий, большое количество настроек, позволяющих настроить его под себя и работать в максимальном комфорте, а также имеется обширная библиотека различных расширений, способных помочь разработчику в работе. Основные расширения, используемые для создания проекта:

- Prettier – позволяет повышать читаемость написанного кода простым сочетанием клавиш.
- SaSS – Расширение позволяющее работать с файлами, в формате .scss, которые создаются при работе с препроцессором SASS.

- SCSS Formatter – Данное расширение автоматически связывает все файлы с расширением scss и преобразует их в стандартный, воспринимаемый всеми браузерами css.
- Brackets Pair Colorizer – Окрашивает в различные цвета пары скобок, что помогает лучше ориентироваться в коде.

2.3 React.js

После того как был отрисован дизайн сайта в виде макета, его необходимо превратить в front-end, для этого использовалась библиотека React.js

React.js – Это библиотека, написанная на языке JavaScript. Ее задача, это проектирование и создание пользовательских интерфейсов, причем она позволяет работать на совершенно разных платформах. Но для поставленной задачи меня интересовало именно ее применение в Web-разработке. Основные два столпа, на которых держится React и из-за чего данная библиотека приобрела такую популярность в сообществе разработчиков это компоненты и состояния.

Компонент представляет из себя определенную часть кода, которая отвечает за отрисовку конкретного компонента на странице. И поскольку компоненты являются независимыми частями кода, появляется возможность с помощью минимальных усилий отрисовать данный компонент на любой странице нашего приложения.

Вторая особенность React.js это состояния. Под состояниями понимается хранение динамических данных определенного компонента. Помимо хранения, состояния постоянно проверяют компонент на наличие изменений, для своевременного внесения необходимых корректировок.

2.4 Node.js

Для серверной части сайта, или же, как её называют Back-end, была выбранная программная платформа Node.js

Изначальной, JavaScript, на момент его создания, был языком узкого направления, который выполнялся исключительно на клиентской стороне. Но шло время, популярность данного языка росла и стали появляться различные улучшения и дополнения для JavaScript. Одним из самых значимых улучшений стало появление специальной платформы, которая позволила вывести JavaScript из разряда языков узкого направления и перевести его в языки общего назначения. Эта платформа и получила называние Node.js

Данную платформу часто называют однопоточной, это не совсем верное утверждение. Все дело в том, что

К особенностям Node.js можно отнести:

Библиотека libuv – Про Node.js часто говорят, что эта платформа однопоточная, это не совсем верное утверждение, все дело как раз в библиотеке libuv. Для начала рассмотрим многопоточные приложения. Когда пользователь обращается к серверу, чтобы получить какие-либо данные, то многопоточная платформа выделяет ему отдельный поток, в котором находятся все возможные обработчики. Если подключается второй пользователь, то уже создается второй поток, созданный именно для этого пользователя, таким образом работа с пользователями распараллеливается и работает со всеми одновременно. Проблема возникает на тех стадиях, когда пользователей становится слишком много, создается слишком много отдельных потоков, а вычислительная мощность сервера не позволяет справляться с такими нагрузками. В такой ситуации сервер рискует войти в состояние “Потокового голодания”. Суть этого состояния в том, что когда потоков становится слишком много, то сервер занят только тем, что переключается между потоками и проверяет их состояние. Теперь возвращаемся к библиотеке lebuv. Данная библиотека предоставляет набор различных потоков, которые позволяют работать с сокетами, функциями ввода и вывода и т. д. и все это происходит асинхронно. Все эти потоки работают в окружении параллельно и не зависимо друг от друга. Предположим, что к приложению подключились несколько пользователей, и всем им необходима какая-то информация, располагаемая в базе данных. Node.js берет эти данные с базы, помещает их в общий поток, там с данными происходят определенные манипуляции, которые обусловлены функционалом приложения и после этого передает данные пользователям. Это называется асинхронным вводом-выводом. Вместо того чтобы каждому пользователю выделять отдельный поток, мы просто выделяем им один. Почему же подобный подход лучше, чем стандартная многопоточность? Потому что потоки, которые предоставляет библиотека lebuv всегда производят какие-либо манипуляции в системе, и никогда не простаивают. Так же их всегда ограниченное количество, что избавляет нас от возможной проблемы “Потокового голодания”.

Продолжая перечисление плюсов платформы Node.js, нельзя не выделить встроенный пакетный менеджер NPM. Node Packet Manager – За то время что существует обсуждаемая платформа, появилось большое количество различных библиотек, позволяющих решать самые разные задачи. Это показывает уровень сплоченности сообщества, ведь в данном пакетном менеджере представлены библиотеки не только от именитых команд разработчиков, но и от простых пользователей, которые посчитали что их библиотека сможет сделать процесс создания приложений немного проще.

Еще один весомый плюс Node.js уже частично упоминался выше, но важно сказать, что по Node.js имеется большое количество документации в самых разных форматах, помимо официальной. И вся она создается неравнодушными

специалистами, которые делятся своим накопленным опытом, и позволяют быстрее находить ответы на имеющиеся вопросы.

2.5 База данных

Для хранения всей необходимой для проекта информации использовалась реляционная система управления базами данных PostgreSQL. Данная система поддерживает базы данных неограниченных размеров, имеет надежные механизмы транзакций и репликации, поддерживает наследование, и легко расширяется. На рисунке 2.5.1 представлена схема базы данных.

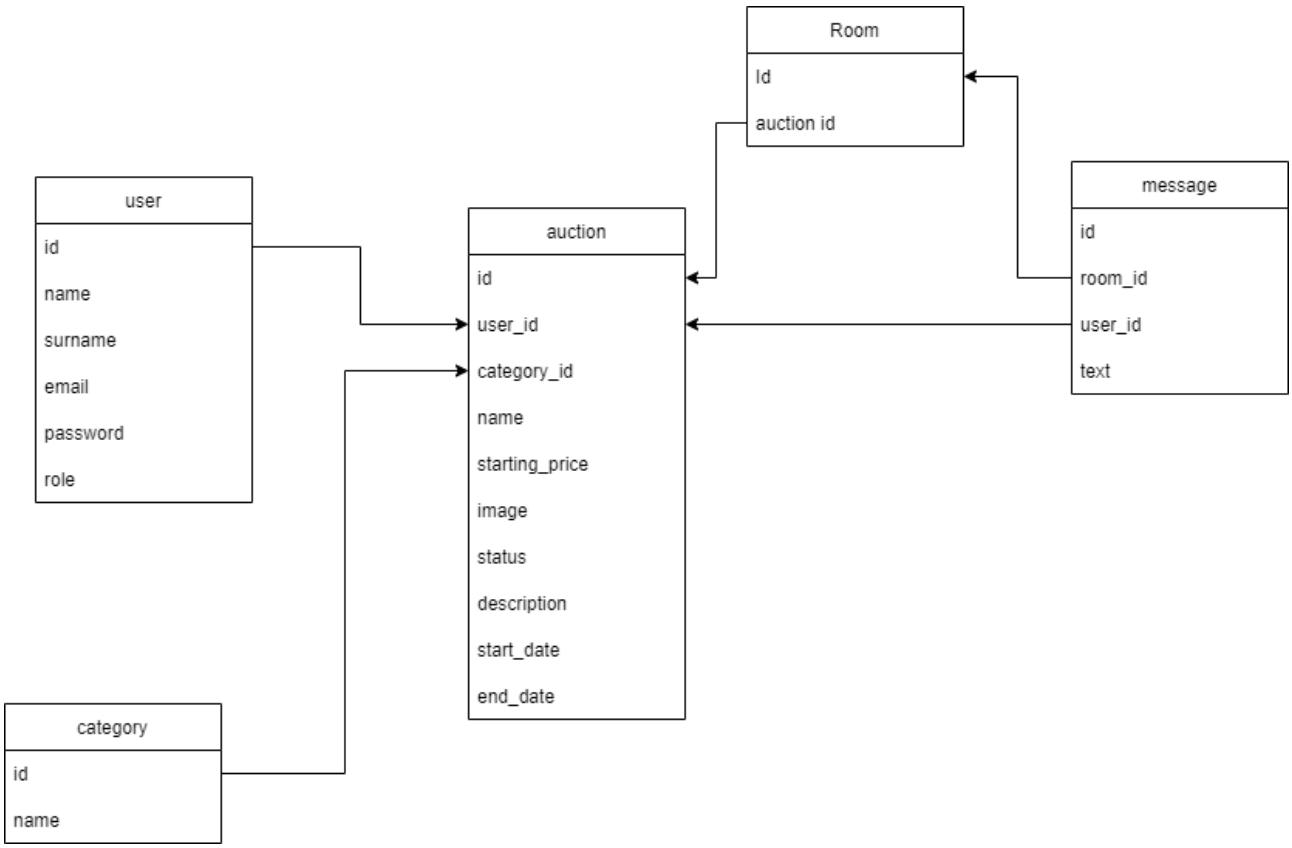

Рисунок 2.5.1 – Схема базы данных

3 Проектная часть

3.1 Этапы разработки веб-приложения

Для облегчения процесса разработки и более четкого понимания поставленной цели, работу над веб-приложением необходимо разделить на определенные этапы.

Разработку веб-приложения можно разделить на следующие этапы:

1. Подготовительный
2. Проектирование
3. Дизайн
4. Верстка
5. Разработка серверной части сайта
6. Тестирование
7. Запуск

На подготовительном этапе необходимо досконально изучить предметную область, с которой предстоит работать. Выделить сильные и слабые стороны конкурентов, и сформировать примерные ожидания от функционала разрабатываемого приложения.

На этапе проектирования составляется четкое техническое задание, идет процесс формирования структуры приложения, и обговаривается конечный функционал. Без изначально составленного и утвержденного технического задания разработка может сильно затянуться или вовсе не дойти до стадии запуска, так как не будет определена конечная цель

Далее идет процесс создания дизайна сайта в формате макета. Важно отрисовать все страницы, причем отрисовать полностью, чтобы на этапе верстки не возвращаться к вопросам по типу “В какой цвет кнопку покрасить”.

Следующим по списку идет этап верстки. На данном этапе нарисованный макет превращается в набор документов, совокупность которых позволяет вывести картинку в браузере и каким-либо образом с ней взаимодействовать.

Без этапа разработки серверной части, сверстанный макет не будет иметь никаких функций. Для того чтобы формы передавали какую-либо информацию, работали кнопки, функции авторизации и так далее, необходимо реализовать серверную часть. На этом же этапе приложение обзаводится собственной базой данных.

После окончания разработки необходимо провести полное тестирование разработанного приложения, так как при разработке некоторые моменты могли быть упущены и это приведет к возникновению различных ошибок.

Когда все предыдущие шаги пройдены, идет запуск проекта, покупка доменного имени, оплата хостинга, настройка резервного копирования.

3.2 Архитектура проекта

Для создания веб-приложения использовался PERN стек. Эта аббревиатура от первых букв из названий основных технологи. PostgreSQL, ExpressJS, React, NodeJS. Помимо данных технологий использовались: HTML, CSS(препроцессор SASS). На рисунке показаны 3.2.1 показана схема взаимодействия основных технологий.

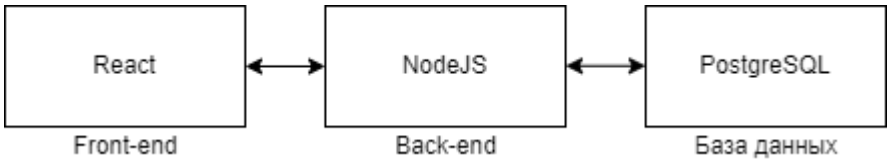

Рисунок 3.2.1 – Схема взаимодействия технологий

3.3 Сценарий использования

Данный тип диаграмм еще называют диаграммами активности. С их помощь можно схематически понять, как будет происходить взаимодействие с готовым проектом. Диаграмма изображена на рисунке 3.3.1.

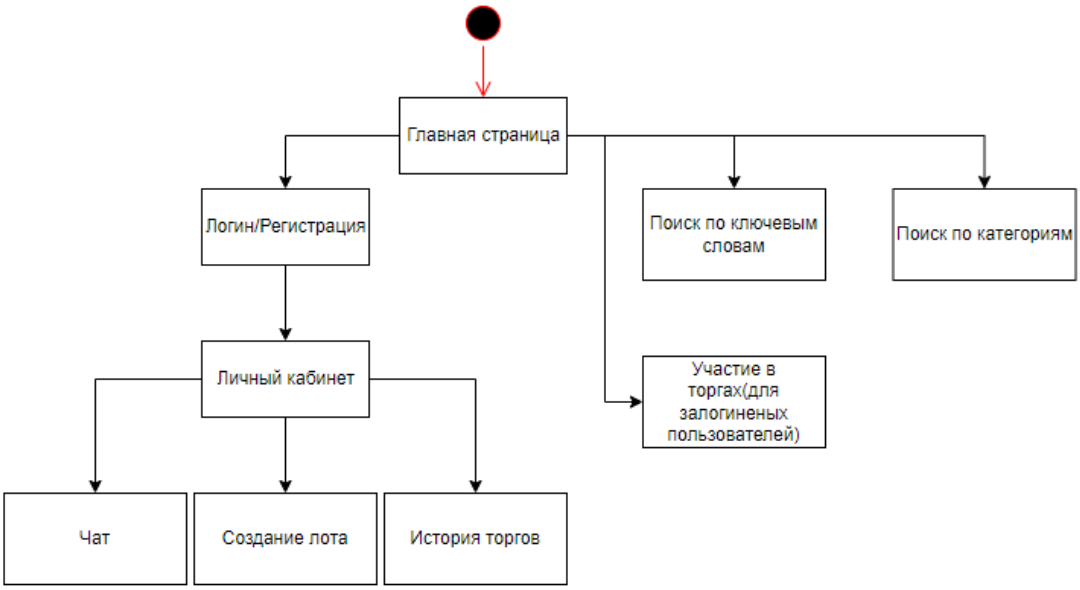

Рисунок 3.3.1 – Диаграмма активности

При входе пользователь попадает на главную страницу, откуда он может просмотреть некоторые имеющиеся лоты и отсортировать их по категориям. Еще пользователю доступна функция поиска. Если пользователь ищет что-то конкретное, то он может попытаться это найти. Главная страница сайта изображена на рисунке 3.3.2.

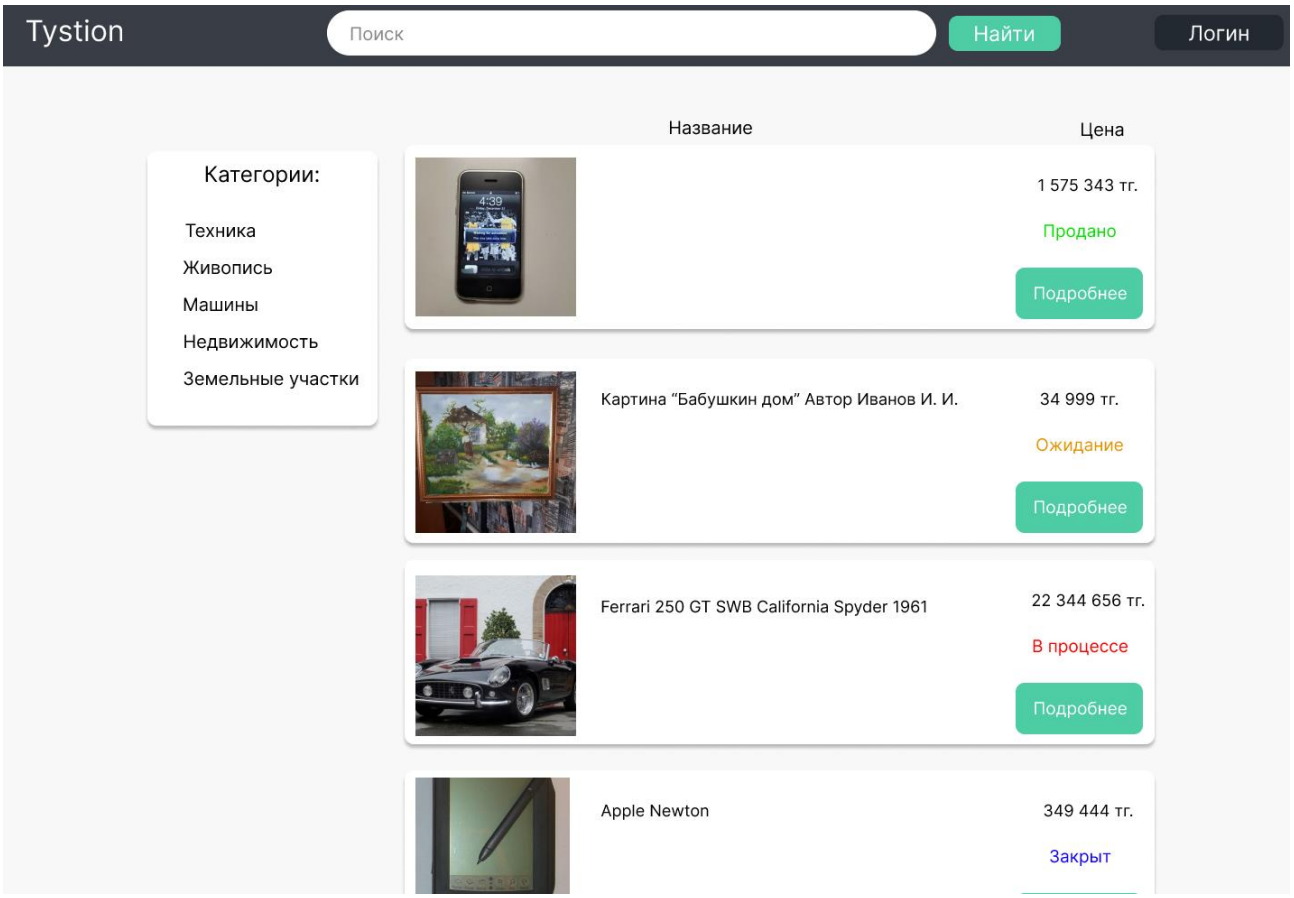

Рисунок 3.3.2 – Главная страница

При просмотре торгов у потенциального клиента может появиться желание поучаствовать в аукционе, для этого ему необходимо войти в свой аккаунт. Страница авторизации изображена на рисунке 3.3.3.

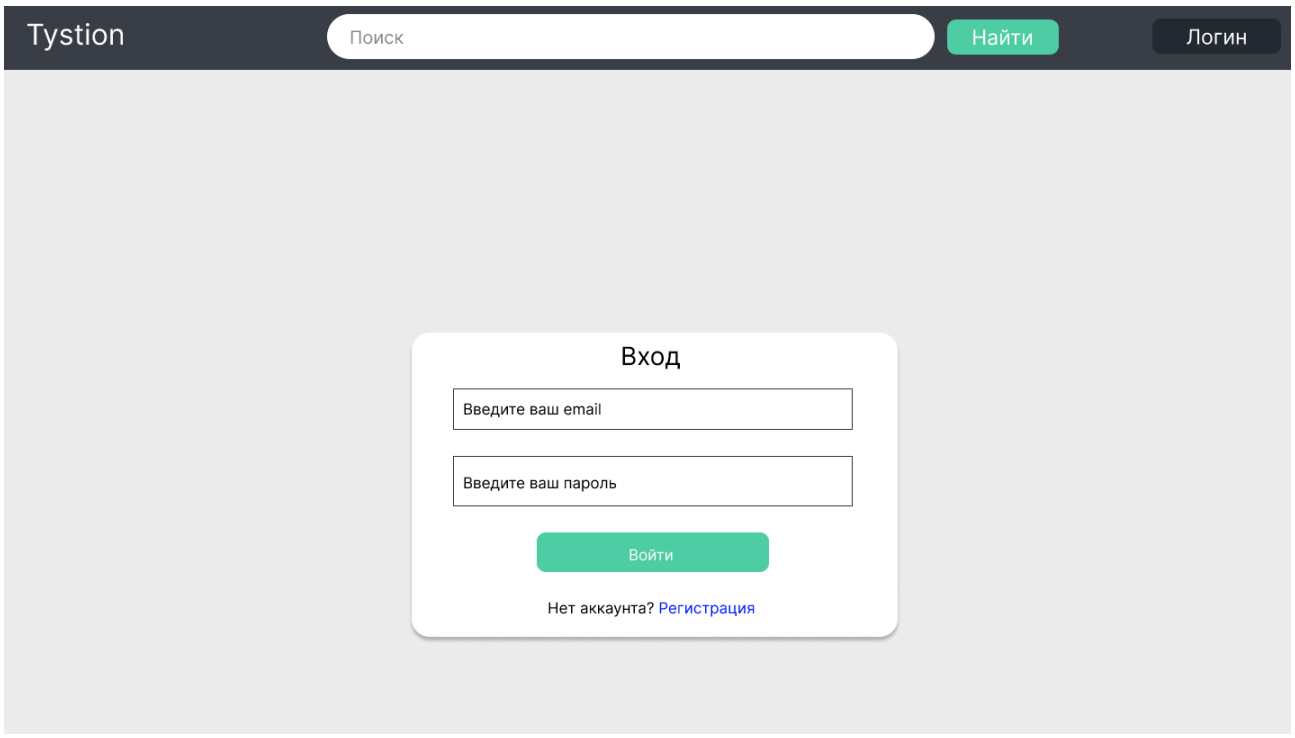

Рисунок 3.3.3 – Страница авторизации

Если у пользователя аккаунт отсутствует, то он может его создать перейдя на страницу регистрации. Страница регистрации изображена на рисунке 3.3.4.

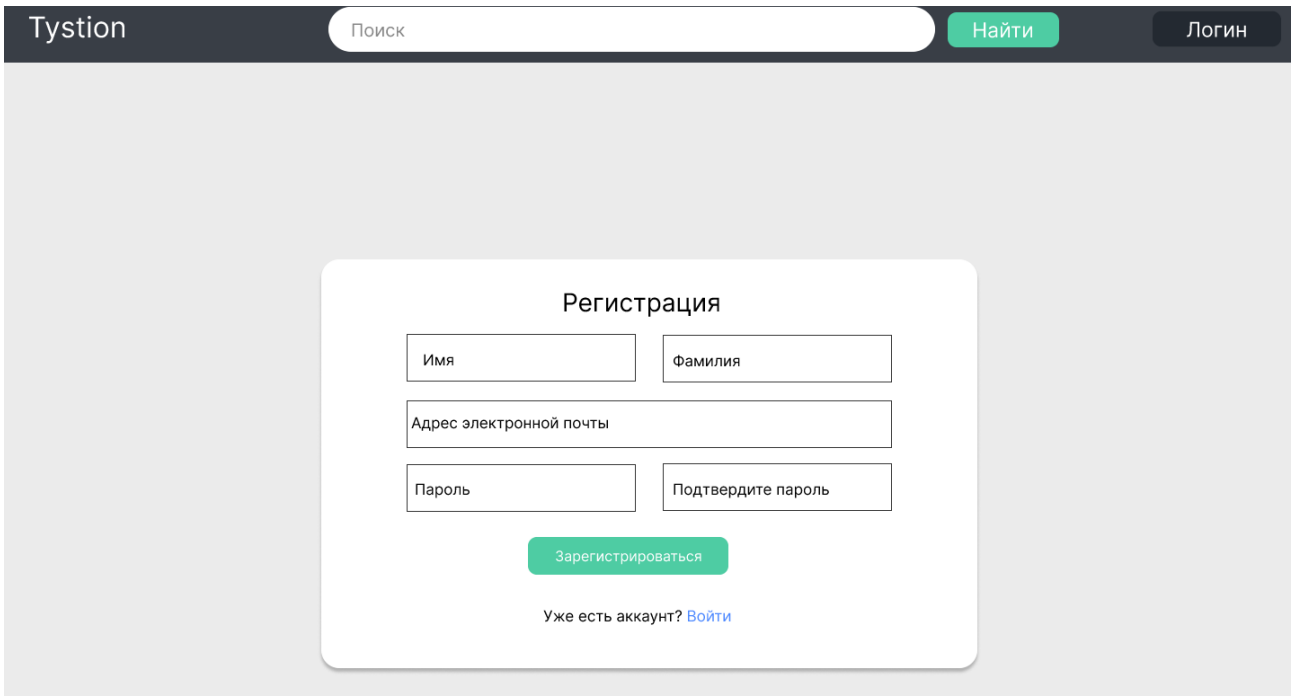

Рисунок 3.3.4 – Страница регистрации

После прохождения регистрации, войдя на страницу любого лота, пользователю будет открыта возможность сделать ставку, вписав сумму в специальное окно. Страница аукциона со всей необходимой информацией изображена на рисунке 3.3.5.

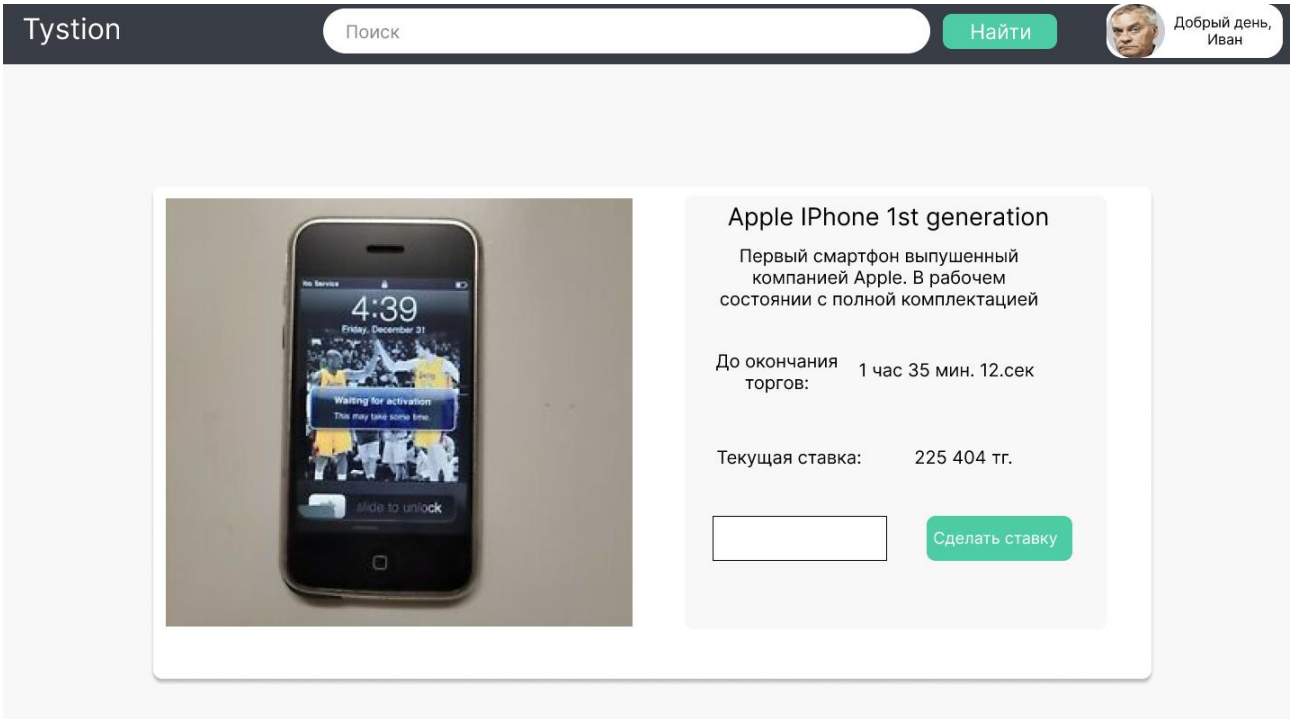

Рисунок 3.3.5 – Страница аукциона

В личном кабинете каждого пользователя имеется история торгов, в которых он принимал участие. Данная информация отображается списком и позволяет в любой момент просмотреть подробности прошедшего аукциона. Личный кабинет с историей поиска изображен на рисунке 3.3.6.

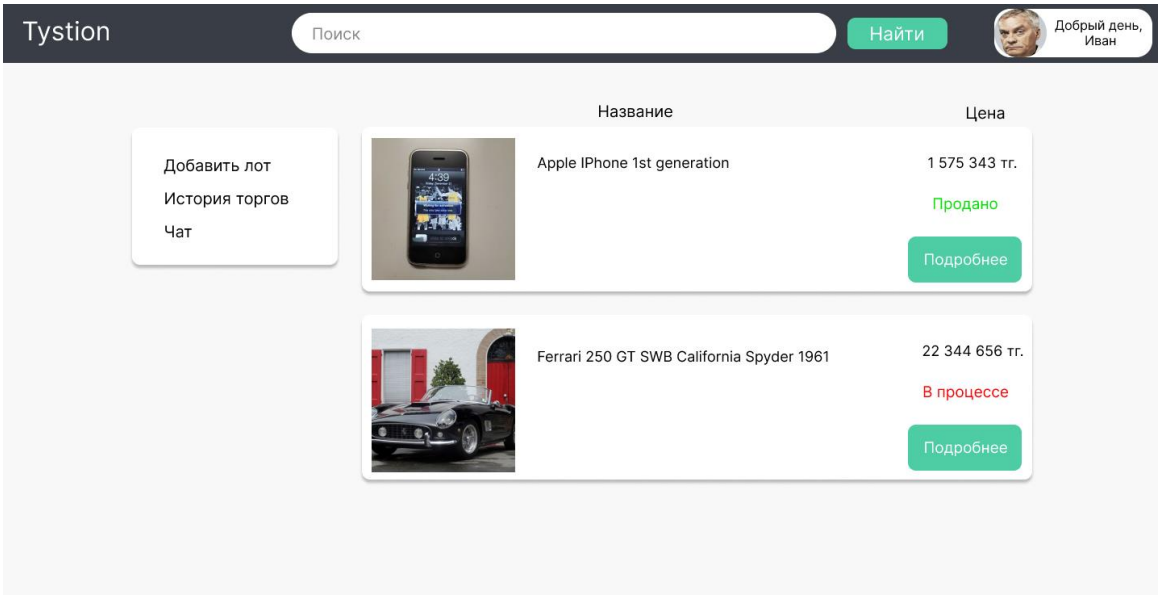

Рисунок 3.3.6 – Личный кабинет и история торгов.

В случае победы на торгах, в личном кабинете появляется чат, в котором связываются победитель аукциона и человек, который устроил эти торги. Личный чат изображен на рисунке 3.3.7.

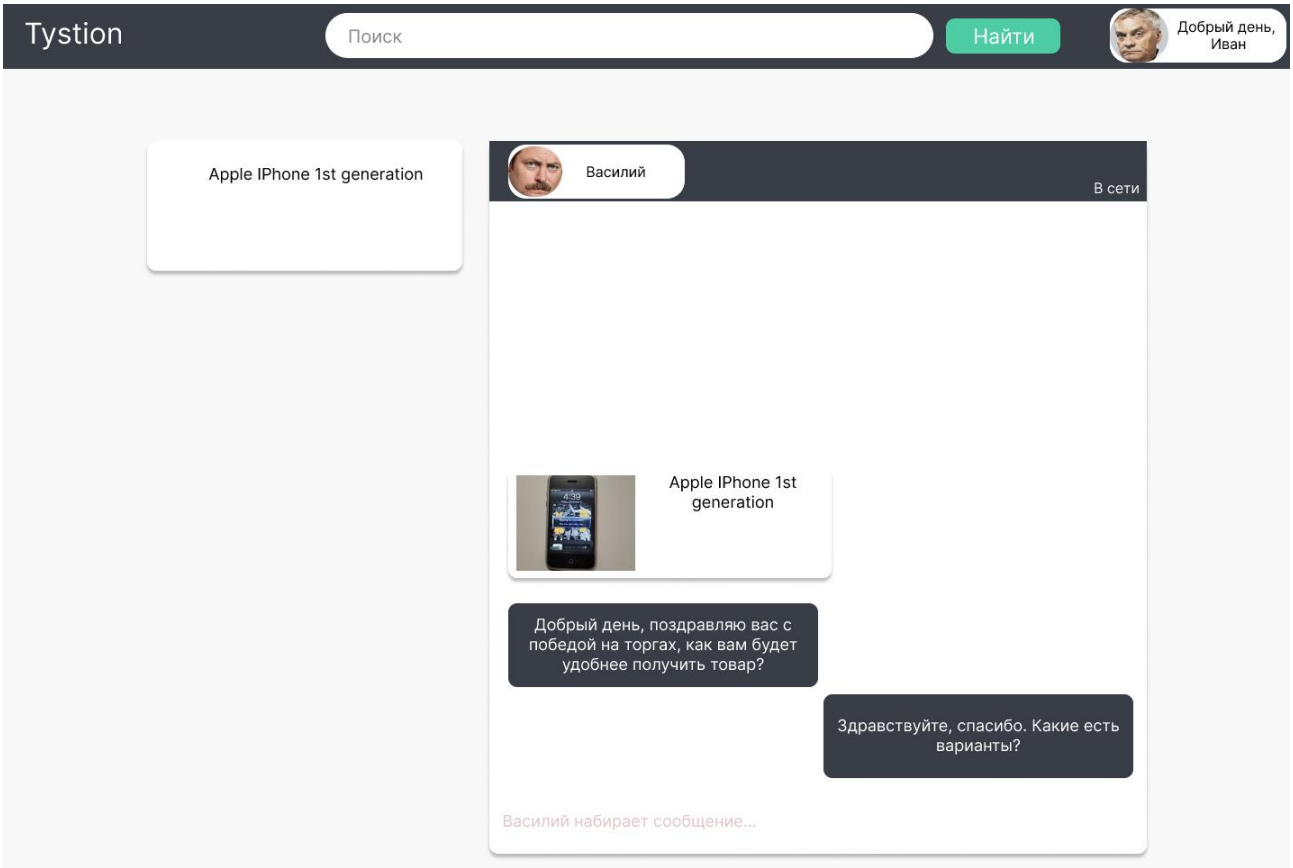

Рисунок 3.3.7 – Чат

Одной из особенностей данного приложения, является то, что любой пользователь может как и поучаствовать в торгах, так их и устроить. Для этого необходимо создать лот, через специальную форму в личном кабинете. Форма создания нового лота изображена на рисунке 3.3.8.

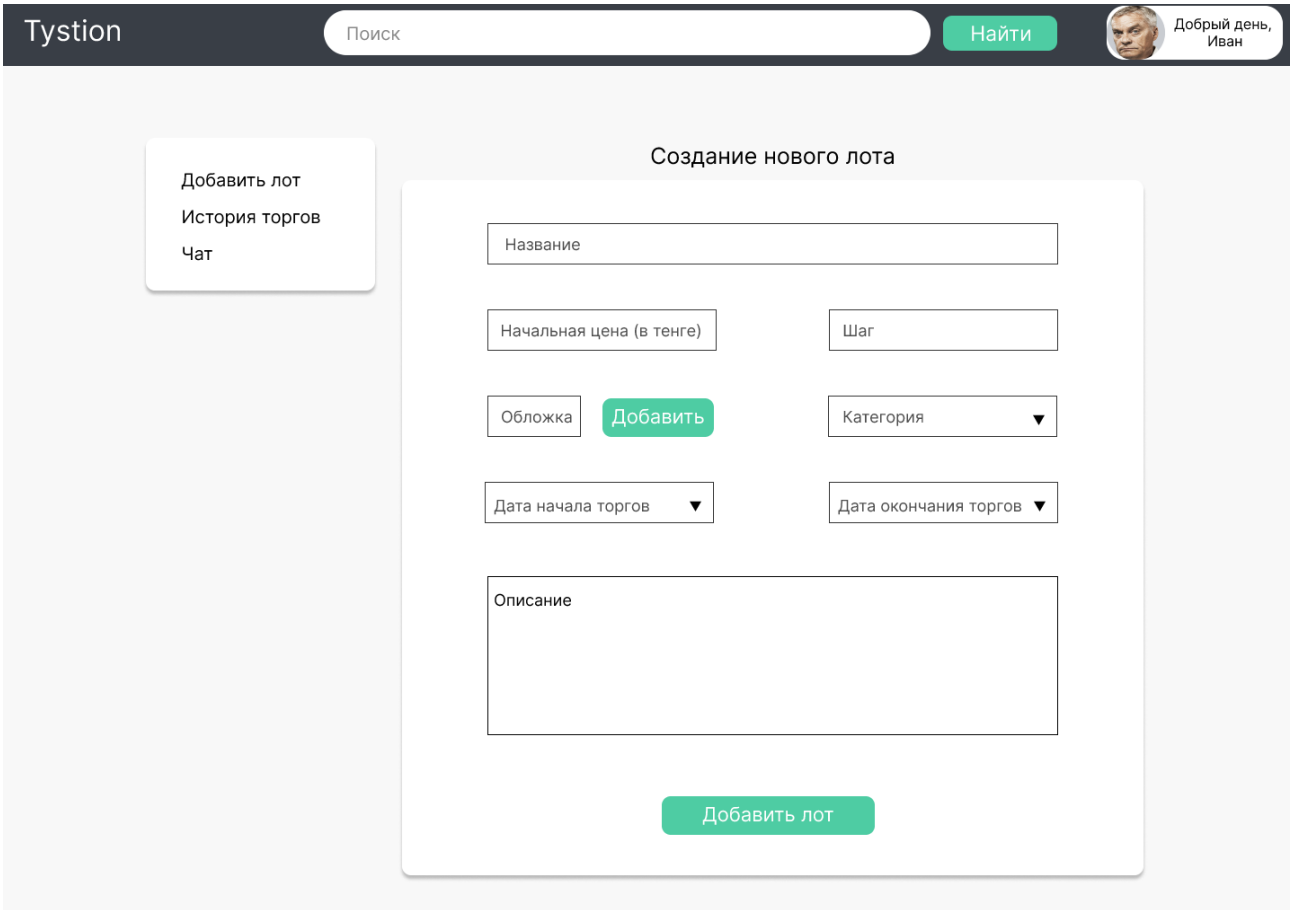

Рисунок 3.3.8 – Форма создания нового лота.

3.4 Клиентская часть

Весь front-end реализован на React, в связи с его надежностью, широкой поддержкой, из-за чего имеется большое количество различный библиотек и с его возможностями к масштабированию. То, как реализовано навигационное меню можно увидеть на рисунке 3.5.1

```
return (
    <Navbar bg="dark" variant="dark">
        <Container>
            <NavLink style={{color:'white'}} to={SHOP_ROUTE}>Тустион</NavLink>
            {user.isAuth ?
                <Nav className="ml-auto" style={{color: 'white'}}>
                    <Button
                        variant={"outline-light"}
                        onClick={() => history.push(ADMIN_ROUTE)}
                    >
                    </Button>
                    <Button
                        variant={"outline-light"}
                        onClick={() => logOut()}
                        className="ml-2"
                    >
                        Выйти
                    </Button>
                </Nav>
                :
                <Nav className="ml-auto" style={{color: 'white'}}>
                    <Button variant={"outline-light"} onClick={() => history.push(LOGIN_ROUTE)}>Авториза
                </Nav>
            }
        </Container>
    </Navbar>

);
```

Рисунок 3.5.1 – Навигация

```
<Modal.Header closeButton>
    <Modal.Title id="contained-modal-title-vcenter">
        Добавить устройство
    </Modal.Title>
</Modal.Header>
<Modal.Body>
    <Form>
        <Dropdown className="mt-2 mb-2">
            <Dropdown.Toggle>{auction.selectedType.name || ""}</Dropdown.Toggle>
            <Dropdown.Menu>
                {auction.types.map(type =>
                    <Dropdown.Item
                        onClick={() => auction.setSelectedType(type)}
                        key={type.id}
                    >
                        {type.name}
                    </Dropdown.Item>
                )}
            </Dropdown.Menu>
        </Dropdown>
        <Dropdown className="mt-2 mb-2">
            <Dropdown.Toggle>{auction.selectedcategory.name || ""}</Dropdown.Toggle>
            <Dropdown.Menu>
                {auction.categorys.map(category =>
                    <Dropdown.Item
                        onClick={() => auction.setSelectedcategory(category)}
                        key={category.id}
                    >
                        {category.name}
                    </Dropdown.Item>
                )}
            </Dropdown.Menu>
        </Dropdown>
```

Рисунок 3.5.2 – Форма создания аукцион

3.5 Серверная часть

Для работы с серверной частью используется Node.js и его пакетный менеджер NPM. С помощью NPM мы устанавливаем все необходимые библиотеки и компоненты. На рисунке 3.4.1 показана установка фреймворка Express.js, который в данном проекте используется для маршрутизации. Делается это путем написания пары слов.

Рисунок 3.4.1 – Установка фреймворка Express.js

Фреймворк Express.js является важной составляющей данного проекта, так как имеется большое количество страниц, между которыми необходимо перемещаться, и эта проблема решается путем установки и настройки этого фреймворка. На рисунке 3.4.2 показана настройка навигации для главной страницы. Так же там изображена настройка, которая будет переводить пользователя на страницу конкретного лота.

```
const Router = require('express')
const router = new Router()
const auctionController = require('../controllers/auctionController')

router.post('/', auctionController.create)
router.get('/', auctionController.getAll)
router.get('/:id',auctionController.getOne)


module.exports = router
```

Рисунок 3.4.2 – Настройка маршрутизации главной страницы

В данном проекте предусмотрена функция регистрации и авторизации. Как и в любой другом проекте, необходимо задумываться о безопасности, в представленном проекте это реализовано посредством JWT токена, для этого было установлена специальная библиотека “jsonwebtoken” и написана специальная проверка, которая при попытке зайти на страницу где нужна авторизация, проверяет наличие токена, в случае его отсутствия, система выдает сообщение “Не авторизован”. Код данной функции можно увидеть на рисунке 3.4.3.

```
const jwt = require('jsonwebtoken')

module.exports = function (req, res, next) {
    if (req.method === "OPTIONS") {
        next()
    }
    try {
        const token = req.headers.authorization.split(' ')[1]
        if (!token) {
            return res.status(401).json({message: "Не авторизован"})
        }
        const decoded = jwt.verify(token, process.env.SECRET_KEY)
        req.user = decoded
        next()
    } catch (e) {
        res.status(401).json({message: "Не авторизован"})
    }
};
```

Рисунок 3.4.3 – Проверка токена

Негласным правилом является хранение паролей пользователей в зашифрованном виде. Ведь, в случае если злоумышленник получит доступ к базе данных, он не сможет воспользоваться полученными паролями, так как ключи для расшифровки хранятся отдельно. Код для регистрации пользователя и хэширования его пароля изображен на рисунке 3.4.4

```
class UserController {
    async registration(req, res, next) {
        const {email, password, role} = req.body
        if (!email || !password) {
            return next(ApiError.badRequest('Некорректный email или password'))
        }
        const candidate = await User.findOne({where: {email}})
        if (candidate) {
            return next(ApiError.badRequest('Пользователь с таким email уже существует'))
        }
        const hashPassword = await bcrypt.hash(password, 5)
        const user = await User.create({email, role, password: hashPassword})
        const token = generateJwt(user.id, user.email, user.role)
        return res.json({token})
    }
```

Рисунок 3.4.4 – Регистрация пользователя

3.6 База данных

Для работы с выбранной базой данных использовалась библиотека sequelize. С ее помощью можно легко взаимодействовать с базой данных, создавать и удалять зависимости, вносить, изменять и удалять данные из базы данных. На рисунке 3.4.1 показан алгоритм, добавляющий таблицы в базы данных.

```
const Auction = sequelize.define('auction',{
    id:{type:DataTypes.INTEGER,primaryKey:true,autoIncrement:true},
    name:{type:DataTypes.STRING,unique:false,allowNull:false},
    starting_price:{type:DataTypes.INTEGER,allowNull:false},
    image:{type:DataTypes.STRING,allowNull:false},
    status:{type:DataTypes.STRING,allowNull:false},
    description:{type:DataTypes.STRING,allowNull:false},
```

Рисунок 3.4.1 – Алгоритм создания таблицы

Подгружая созданную модель в файл auctionController.js, мы получается возможность взаимодействия с таблицей, на рисунке 3.4.1 показан код для создания лота.

```
class AuctionController{
    async create(req, res, next) {
        try{
            const {name, starting_price,categoryId,userId,description} = req.body
            const {image} = req.files
            let status = "В ожидании"
            let fileName = uuid.v4() + ".jpg"
            image.mv(path.resolve(__dirname, '..', 'static', fileName))

            const auction = await Auction.create({name,starting_price,categoryId,userId,description,image:fi

            return res.json(auction)
        } catch(e){
            next(ApiError.badRequest(e.message))
        }
```

Рисунок 3.4.2 – Создание лота

ЗАКЛЮЧЕНИЕ

Результатом данной дипломной работы является подробное изучение рынка, выделение конкретных моментов, которые можно исправить, дополнить, где-то упростить. По итогу исследовательской работы было разработано веб-приложение «Tystion», имеющее дружелюбный пользовательский интерфейс, позволяющее участвовать в интернет-аукционах и устраивать свои собственные торги. Были реализованы следующие функции:

– регистрация и авторизация;
– просмотр имеющихся лотов;
– функция поиска;
– выборка по категориям;
– личный кабинет;
– создание собственных лотов;
– чат.

Самое большое внимание было уделено пользовательскому интерфейсу, для того чтобы сделать его простым, дружелюбным и одновременно достаточно информативным, и функциональным.

Решение использовать такие технологии как Node.js и React.js даёт большие возможности для масштабирования приложения и позволяет не беспокоиться о поддержке этих технологий со стороны их разработчиков.

Все поставленные задачи были выполнены в полном объёме. Проект разработан, протестирован и загружен на хостинг. Были улучшены собственные навыки как Full-stack разработчика.

СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ

1. Современный учебник по JavaScript // Электронная версия на сайте https://learn.javascript.ru/
2. Документация по Node.js // Электронная версия на сайте https://nodejs.org/ru/docs/
3. Документация по React.js // Электронная версия на сайте https://ru.reactjs.org/
4. Документация по Visual Studio Code // Электронная версия на сайте https://code.visualstudio.com/docs
5. Головайчук TJ, Кантелон Майк. ”NodeJS в действии” – Питер, 2014 г
6. Пауэрс Шерли “Изучаем Node.js” – Питер, 2013 г.
7. Документация по Moment.js // Электронная версия на сайте https://momentjs.com/
8. Документация по Express.js // Электронная версия на сайте https://expressjs.com/ru/
9. Флэнаган Дэвид, “JavaScript. Подробное руководство ”, - Символ-Плюс, 2013 г.
10. Документация до PostgreSQL // Электронная версия на сайте https://www.postgresql.org/docs/
11. А.Ю. Васильев “Работа с PostgreSQL: Настройка и масштабирование” – 2017
12. Энтони Молинаро “SQL, сборник рецептов” – 2009
13. Алекс Маккоу “JavaScript Web Application” -2011
14. Документация по react-router // Электронная версия на сайте https://reactrouter.com/
15. Документация по mobx.js // Электронная версия на сайте https://mobx.js.org/
16. Стоян Стефанов “React: Up & Running: Building web applications ” – Питер, 2016 г.
17. Люк Вроблевски “Сначала мобильные” – Мани 2012 г.
18. Мэтью Макдональд “Веб-разработкаю. Исчерпывающее руководство” – Питер, 2016 г.
19. Кесенбери Уитни, “Сторителлинг в проектировании интерфейсов” – Мани 2013 г.
20. Документация по Sequelize // Электронная версия на сайте https://sequelize.org/
21. Документация по Cors // Электронная версия на сайте https://developer.mozilla.org/ru/docs/Web/HTTP/CORS

Приложение А
(обязательное)

Техническое задание

А.1.1 Техническое задание на разработку веб-приложения для проведения онлайн-аукционов.

Настоящее техническое задание распространяется на разработку веб-приложения для проведения онлайн аукционов, для предоставления пользователям простого и надежного инструмента для проведения и участия в онлайн-торгах.

А.1.2 Основание для разработки

Веб-приложение разрабатывается, на основе устного согласия, выданного научным руководителем с выбранной темой дипломной работы.

А.1.3 Назначение

Основное назначение для данного веб-приложения - это предоставление основного и полного функционала интернет-аукциона.

А1.4 Требования к функциональным характеристикам

С помощью таких технологий как React, NodeJS, Express.js, PostgreSQL – реализовать веб-приложение для проведения онлайн-аукционов. У приложения должны быть следующие функции:

- регистрация и авторизация;
- просмотр имеющихся лотов;
- функция поиска;
- выборка по категориям;
- личный кабинет;
- создание собственных лотов;
- чат между победителем аукциона и тем, кто устраивал аукцион.

Приложение Б

Models.js

```
const sequelize = require('../db')
const {DataTypes} = require('sequelize')

const User = sequelize.define('user',{
   id:{type:DataTypes.INTEGER,primaryKey:true,autoIncrement:true},
   name:{type:DataTypes.STRING,allowNull:false},
   surname:{type:DataTypes.STRING,allowNull:false},
   email:{type:DataTypes.STRING, unique:true},
   password:{type:DataTypes.STRING,allowNull:false},
})

const Auction = sequelize.define('auction',{
   id:{type:DataTypes.INTEGER,primaryKey:true,autoIncrement:true},
   name:{type:DataTypes.STRING,unique:false,allowNull:false},
   starting_price:{type:DataTypes.INTEGER,allowNull:false},
   image:{type:DataTypes.STRING,allowNull:false},
   status:{type:DataTypes.STRING,allowNull:false},
   description:{type:DataTypes.STRING,allowNull:false},

})

const Category = sequelize.define('category',{
   id:{type:DataTypes.INTEGER,primaryKey:true,autoIncrement:true},
   name:{type:DataTypes.STRING,unique:true,allowNull:false},
})


User.hasMany(Auction)
Auction.belongsTo(User)

Category.hasMany(Auction)
Auction.belongsTo(Category)


module.exports = {
   User, Auction, Category
}
```

auctionController.js

```
const uuid = require('uuid')
const path = require('path');
const {Auction} = require('../models/models');
const ApiError = require('../error/ApiError');

class AuctionController{
    async create(req, res, next) {
        try{
            const {name, starting_price,categoryId,userId,description} = req.body
            const {image} = req.files
            let status = "В ожидании"
            let fileName = uuid.v4() + ".jpg"
            image.mv(path.resolve(__dirname, '..', 'static', fileName))

            const auction = await Auction.create({name,starting_price,categoryId,userId,description,image:fileName,status})

            return res.json(auction)
        } catch(e){
            next(ApiError.badRequest(e.message))
        }


    }
    async getAll(req, res) {
        const {categoryId} = req.query
        let auctions;
        if(!categoryId){
            auctions = await Auction.findAll()
        }
        if(categoryId){
            auctions = await Auction.findAll({where:{categoryId}})
        }
        return res.json(auctions)
    }
    async getOne(req, res) {
        const {id} = req.params
        const auction = await Auction.findOne(
            {
                where: {id},
```

```
            }


        )
        return res.json(auction)
    }
}

module.exports = new AuctionController()
```

categoryController.js

```
const {Category} = require('../models/models')
const ApiError = require('../error/ApiError')

class CategoryController{
    async create(req, res) {
        const {name} = req.body
        const category = await Category.create({name})
        return res.json(category)
    }
    async getAll(req, res) {
        const categories = await Category.findAll()
        return res.json(categories)
    }
}

module.exports = new CategoryController()
```

userController.js

```
const ApiError = require('../error/ApiError')
const bcrypt = require('bcrypt')
const jwt = require('jsonwebtoken')
const {User} = require('../models/models')

const generateJwt = (id, name, surname, email)=>{
        return jwt.sign(
        {id, name, surname, email},
        process.env.SECRET_KEY,
        {expiresIn:'24h'}

        )
```

```
}

class UserController{
    async registration(req, res, next) {
        const {name, surname, email, password} = req.body
        if(!name || !surname || !email || !password){
            return next(ApiError.badRequest('Заполните все поля'))
        }
        const candidate = await User.findOne({where: {email}})
        if (candidate) {

            return next(ApiError.badRequest('Этот email уже занят'))

        }

        const hashPassword = await bcrypt.hash(password, 5)
        const user = await User.create({name, surname, email, password:hashPassword})
        const token = generateJwt(user.id, user.name, user.surname, user.email )
            return res.json({token})


    }

    async login(req, res, next) {
        const {email, password} = req.body
        const user = await User.findOne({where:{email}})
        if (!user) {
            return next(ApiError.internal('Пользователь не найден'))
        }
        let comparePassword = bcrypt.compareSync(password, user.password)
        if (!comparePassword) {
            return next(ApiError.internal('Указан не верный пароль'))
        }
        const token = generateJwt(user.id, user.name, user.surname, user.email)
        return res.json({token})
    }

    async check(req, res, next) {
      const token = generateJwt(req.user.id, req.user.name, req.user.surname,
req.user.email)
      return res.json({token})
    }
}
```

```
module.exports = new UserController()
```

apiError.js

```
class ApiError extends Error{
    constructor(status,message){
        super();
        this.status = status
        this.message = message
    }
    static badRequest(message){
        return new ApiError(404,message)
    }
    static internal(message){
        return new ApiError(500, message)
    }
    static forbidden(message){
        return new ApiError(403, message)
    }
}

module.exports = ApiError
```

authMiddleware.js

```
const jwt = require('jsonwebtoken')

module.exports = function (req, res, next) {
    if (req.method === "OPTIONS"){
        next()
    }
    try{
        const token = req.headers.authorization.split(' ')[1]
        if (!token){
            return res.status(401).json({message:"Не авторизован"})
        }
        const decoded = jwt.verify(token, process.env.SECRET_KEY)
        req.user  = decoded
        next()
        }catch(e) {
        res.status(401).json({message:"Не авторизован"})
    }
}
```

Index.js

```
require('dotenv').config()
const express = require('express')
const sequelize = require('./db')
const models = require('./models/models')
const cors = require('cors')
const fileUpload = require('express-fileupload')
const router = require('./routes/index')
const errorHandler = require('./middleware/ErrorHandlingMiddleware')
const path = require('path')

const PORT = process.env.PORT || 5000

const app = express()
app.use(cors())
app.use(express.json())
app.use(express.static(path.resolve(__dirname, 'static')))
app.use(fileUpload({}))
app.use('/api', router)

// Обработка ошибок, последний Middleware
app.use(errorHandler)

const start = async () => {
    try {
        await sequelize.authenticate()
        await sequelize.sync()
        app.listen(PORT, () => console.log(`Server started on port ${PORT}`))
    } catch (e) {
        console.log(e)
    }
}


start()
```

userRouter.js

```
const Router = require('express')
const router = new Router()
const userController = require('../controllers/userController')
const authMiddleware = require('../middleware/authMiddleware')

router.post('/registration', userController.registration)
```

```
router.post('/login', userController.login)
router.get('/auth', authMiddleware, userController.check)

module.exports = router
```

auctionItem.js

```
import React from 'react';
import {Card, Col} from "react-bootstrap";
import Image from "react-bootstrap/Image";
import star from '../assets/star.png'
import {useHistory} from "react-router-dom"
import {DEVICE_ROUTE} from "../utils/consts";

const DeviceItem = ({device}) => {
    const history = useHistory()
    return (
        <Col md={3} className={"mt-3"} onClick={() => history.push(DEVICE_ROUTE + '/' + device.id)}>
            <Card style={{width: 150, cursor: 'pointer'}} border={"light"}>
                <Image width={150} height={150} src={process.env.REACT_APP_API_URL + device.img}/>
                <div className="text-black-50 mt-1 d-flex justify-content-between align-items-center">
                    <div>Samsung...</div>
                    <div className="d-flex align-items-center">
                        <div>{device.rating}</div>
                        <Image width={18} height={18} src={star}/>
                    </div>
                </div>
                <div>{device.name}</div>
            </Card>
        </Col>
    );
};

export default DeviceItem;
```

navbar.js

```
import React, {useContext} from 'react';
import {Context} from "../index";
import Navbar from "react-bootstrap/Navbar";
import Nav from "react-bootstrap/Nav";
```

```
import {NavLink} from "react-router-dom";
import {ADMIN_ROUTE, LOGIN_ROUTE, SHOP_ROUTE} from "../utils/consts";
import {Button} from "react-bootstrap";
import {observer} from "mobx-react-lite";
import Container from "react-bootstrap/Container";
import {useHistory} from 'react-router-dom'
const NavBar = observer(() => {
    const {user} = useContext(Context)
    const history = useHistory()

    const logOut = () => {
        user.setUser({})
        user.setIsAuth(false)
    }

    return (
        <Navbar bg="dark" variant="dark">
            <Container>
                <NavLink style={{color:'white'}} to={SHOP_ROUTE}>КупиДевайс</NavLink>
                {user.isAuth ?
                    <Nav className="ml-auto" style={{color: 'white'}}>
                        <Button
                            variant={"outline-light"}
                            onClick={() => history.push(ADMIN_ROUTE)}
                        >
                            Админ панель
                        </Button>
                        <Button
                            variant={"outline-light"}
                            onClick={() => logOut()}
                            className="ml-2"
                        >
                            Выйти
                        </Button>
                    </Nav>
                    :
                    <Nav className="ml-auto" style={{color: 'white'}}>
                        <Button variant={"outline-light"} onClick={() => history.push(LOGIN_ROUTE)}>Авторизация</Button>
```

```
                </Nav>
            }
        </Container>
    </Navbar>

  );
});

export default NavBar;

import React, {useContext, useEffect, useState} from 'react';
import Modal from "react-bootstrap/Modal";
import {Button, Dropdown, Form, Row, Col} from "react-bootstrap";
import {Context} from "../../index";
import {createauction, fetchBrands, fetchauctions, fetchTypes} from
"../../http/auctionAPI";
import {observer} from "mobx-react-lite";

const CreateAuction = observer(({show, onHide}) => {
  const {auction} = useContext(Context)
  const [name, setName] = useState('')
  const [price, setPrice] = useState(0)
  const [file, setFile] = useState(null)
  const [info, setInfo] = useState([])

  useEffect(() => {
    fetchTypes().then(data => auction.setTypes(data))
    fetchBrands().then(data => auction.setBrands(data))
  }, [])

  const addInfo = () => {
    setInfo([...info, {title: '', description: '', number: Date.now()}])
  }
  const removeInfo = (number) => {
    setInfo(info.filter(i => i.number !== number))
  }
  const changeInfo = (key, value, number) => {
    setInfo(info.map(i => i.number === number ? {...i, [key]: value} : i))
  }

  const selectFile = e => {
```

```
        setFile(e.target.files[0])
    }

    const addauction = () => {
        const formData = new FormData()
        formData.append('name', name)
        formData.append('price', `${price}`)
        formData.append('img', file)
        formData.append('brandId', auction.selectedBrand.id)
        formData.append('typeId', auction.selectedType.id)
        formData.append('info', JSON.stringify(info))
        createauction(formData).then(data => onHide())
    }

    return (
        <Modal
            show={show}
            onHide={onHide}
            centered
        >
            <Modal.Header closeButton>
                <Modal.Title id="contained-modal-title-vcenter">
                    Добавить устройство
                </Modal.Title>
            </Modal.Header>
            <Modal.Body>
                <Form>
                    <Dropdown className="mt-2 mb-2">
                        <Dropdown.Toggle>{auction.selectedType.name                ||
""}</Dropdown.Toggle>
                        <Dropdown.Menu>
                            {auction.types.map(type =>
                                <Dropdown.Item
                                    onClick={() => auction.setSelectedType(type)}
                                    key={type.id}
                                >
                                    {type.name}
                                </Dropdown.Item>
                            )}
                        </Dropdown.Menu>
                    </Dropdown>
```

```
                <Dropdown className="mt-2 mb-2">
                  <Dropdown.Toggle>{auction.selectedBrand.name ||
""}</Dropdown.Toggle>
                  <Dropdown.Menu>
                    {auction.brands.map(brand =>
                      <Dropdown.Item
                        onClick={() => auction.setSelectedBrand(brand)}
                        key={brand.id}
                      >
                        {brand.name}
                      </Dropdown.Item>
                    )}
                  </Dropdown.Menu>
                </Dropdown>
                <Form.Control
                  value={name}
                  onChange={e => setName(e.target.value)}
                  className="mt-3"
                  placeholder=""
                />
                <Form.Control
                  value={price}
                  onChange={e => setPrice(Number(e.target.value))}
                  className="mt-3"
                  placeholder=""
                  type="number"
                />
                <Form.Control
                  className="mt-3"
                  type="file"
                  onChange={selectFile}
                />
                <hr/>
                <Button
                  variant={""}
                  onClick={addInfo}
                >
                  Добавить новое свойство
                </Button>
                {info.map(i =>
                  <Row className="mt-4" key={i.number}>
```

```
                    <Col md={4}>
                      <Form.Control
                        value={i.title}
                        onChange={(e) => changeInfo('title', e.target.value,
i.number)}
                        placeholder=""
                      />
                    </Col>
                    <Col md={4}>
                      <Form.Control
                        value={i.description}
                        onChange={(e) => changeInfo('description',
e.target.value, i.number)}
                        placeholder=""
                      />
                    </Col>
                    <Col md={4}>
                      <Button
                        onClick={() => removeInfo(i.number)}
                        variant={"outline-danger"}
                      >
                        Удалить
                      </Button>
                    </Col>
                  </Row>
                )}
              </Form>
            </Modal.Body>
            <Modal.Footer>
              <Button                                    variant="outline-danger"
onClick={onHide}>Закрыть</Button>
              <Button                                    variant="outline-success"
onClick={addauction}>Добавить</Button>
            </Modal.Footer>
          </Modal>
        );
});

export default Createauction;
```