

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РФ
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»
ПОЛИТЕХНИЧЕСКИЙ ИНСТИТУТ

Факультет
вычислительной техники

Кафедра
МОиПЭВМ

Направление подготовки 09.03.04 «Программная инженерия»

БАКАЛАВРСКАЯ РАБОТА
на тему
Программные средства проведения
автомобильного аукциона

Студент _____
(подпись, дата) Просвирнин Денис Константинович
(ФИО полностью)

Руководитель _____
(подпись, дата) Балашова И. Ю.
(фамилия, инициалы)

Нормоконтролёр _____
(подпись, дата) Такташкин Д.В.
(фамилия, инициалы)

Работа допущена к защите (протокол заседания кафедры от _____ № _____)

Заведующий кафедрой _____
(подпись) Козлов А.Ю.
(фамилия, инициалы)

Работа защищена с отметкой _____ (протокол заседания ГЭК от _____ № _____)

Секретарь ГЭК _____
(подпись) Попова Н.А.
(фамилия, инициалы)

Пенза, 2023

Федеральное государственное бюджетное образовательное учреждение
высшего образования

ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

ПОЛИТЕХНИЧЕСКИЙ ИНСТИТУТ

Факультет

Вычислительной техники

Кафедра

МО и ПЭВМ

«Утверждаю»

Заведующий кафедрой

_____ А.Ю. Козлов

«__» _____ 2023 г

**ЗАДАНИЕ
НА ВЫПУСКНУЮ КВАЛИФИКАЦИОННУЮ РАБОТУ
БАКАЛАВРА**

1. Студент Просвирнин Денис Константинович гр. 19ВП2 факультета
<Фамилия, имя, отчество полностью>

вычислительной техники

2. Тема работы Программные средства проведения автомобильного аукциона

Тема утверждена приказом ПГУ № _____ от " _____ " апреля 2023 года

2. Руководитель работы Балашова Ирина Юрьевна

3. Задание на работу (назначение разработки, исходные данные и т.п.)

1) *Назначение: web-приложение для проведения интернет-аукциона автомобилей*

2) *Среда выполнения: операционная система Windows 10, Web-браузер*

Языковые средства разработки: JavaScript, CSS, HTML, SQL.

Инструментальные средства: Webstorm, MySQL, Draw.io,

NodeJS, Google Chrome

Технологии разработки: ООП, принципы SOLID, шаблоны проектирования, архитектура REST API.

3) Основные функции приложения:

а) просмотр списка лотов;

б) просмотр подробной информации о выбранном лоте;

в) возможность сделать ставку на выбранный лот;

г) авторизация и регистрация;

д) создание своих лотов;

е) возможность получить информацию о победителе.

4. Перечень подлежащих разработке вопросов:

1) Анализ предметной области и постановка задачи;

2) Проектирование;

3) Реализация;

4) Контроль качества программного обеспечения;

5) План разработки бюджета;

6) Оформление пояснительной записки.

5. Календарный график выполнения работы

№ п/п	Наименование этапов работы	Объем работы	Срок выполнения	Подпись руководителя
1	Анализ предметной области и требований к системе	10%	12.04.2023 – 16.04.2023	
2	Проектирование программных средств	20%	17.04.2023 – 26.04.2023	
3	Реализация программных средств	30%	27.04.2023 – 11.05.2023	
4	Тестирование программных средств	20%	12.05.2023 – 21.05.2023	
5	Оформление пояснительной записки	20%	22.05.2023 – 01.06.2023	

Дата выдачи задания «22» декабря 2022 г.

Руководитель бакалаврской работы _____
(подпись, дата)

Задание к исполнению принял студент _____
(подпись, дата)

Работу к защите допустить

Декан ФВТ

(подпись, дата)

Л.Р. Фионова

Реферат

Пояснительная записка содержит 115 листов, 41 рисунков, 8 таблиц, 17 использованных источников и 4 приложения.

ИНТЕРНЕТ-АУКЦИОН, WEB-ПРИЛОЖЕНИЕ, КЛИЕНТ, СЕРВЕР, БАЗА ДАННЫХ, REST API, СУБД, UML, MVC, СЕРВЕР БАЗЫ ДАННЫХ.

Объект исследования – средства автоматизации рабочих процессов интернет-аукциона автомобилей.

Цель выпускной квалификационной работы – разработать web-приложение, позволяющее проводить интернет-аукцион автомобилей.

Технологии разработки – разработка REST API-сервера проводилась на языке программирования JavaScript в среде WebStorm. Клиентское приложение разработано на языке JavaScript с использованием фреймворка React для web-приложений. Для создания базы данных использовался сервер MySQL.

Среда исполнения – Windows, Web-браузер.

Результаты работы – в ходе выполнения выпускной квалификационной работы были созданы клиентская и серверная часть web-приложения интернет-аукциона автомобилей. Все требования к разработке приложения соблюдены.

					ПГУ 09.03.04– 8БР192.22 ПЗ						
Изм.	Лист	№ докум.	Подпись	Дата	Программные средства проведения автомобильного аукциона. Пояснительная записка			Лит.	Лист	Листов	
Разраб.		Просвирнин Д. К									
Пров.		Балашова И.Ю.								4	115
								Группа 19ВП2			
Н. Контр.		Такташкин Д.В									
Утв.		Козлов А.Ю.									

Содержание

Введение	7
1 Анализ программных средств для проведения интернет-аукциона автомобилей	8
1.1 Анализ предметной области.....	8
1.2 Обзор продуктов-аналогов	8
1.2.1 Сайт-аукцион «Carlink».....	9
1.2.2 Сайт-аукцион «Auto-mama».....	10
1.2.3 Сайт-аукцион «Exleasingcar»	10
1.3 Сравнительный анализ и критериальная оценка аналогов.....	11
1.4 Исследование вариантов использования.....	14
1.5 Анализ требований	19
1.6 Постановка задачи.....	20
2 Проектирование web-приложения аукциона автомобилей	21
2.1 Архитектура приложения	21
2.2 Проектирование структур данных.....	22
2.2.1 Концептуальная модель базы данных.....	22
2.2.2 Логическая модель базы данных	24
2.3 Проектирование программных средств	25
2.4 Проектирование интерфейса	27
3 Реализация web-приложение для интернет-аукциона автомобилей	32
3.1 Реализация серверной части приложения.....	32
3.1.1 Реализация базы данных	32
3.1.2 Реализация серверных компонентов.....	33
4 Контроль качества программного обеспечения	36
4.1 Выбор режима тестирования.....	36

4.2 Тестирование серверной части web-приложения интернет-аукциона автомобилей	37
4.3 Тестирование клиентской части web-приложения для интернет-аукциона автомобилей	42
4.4 Планирование разработки и оценка бюджета	43
Заключение.....	46
Список использованных источников.....	48
Приложение А. Глоссарий.....	50
Приложение Б. Листинг программы.....	53
Приложение В. Описание сущностей.....	108
Приложение Г. Результаты тестирования	111

Введение

В современном мире автомобиль является неотъемлемой частью жизни многих людей. Каждый год продается огромное количество автомобилей, и среди них есть как новые, так и подержанные. Сейчас на рынке представлено множество способов покупки автомобиля, одним из которых являются интернет-аукционы. Интернет-аукционы автомобилей становятся все более популярными среди покупателей, благодаря простоте и удобству совершения покупки онлайн [1]. В отличие от традиционных салонов, на интернет-аукционах можно найти большой выбор автомобилей и купить автомобиль по более привлекательной цене. Разработка программного обеспечения (ПО) для автомобильного аукциона является актуальной задачей, так как на рынке представлено мало приложений-аналогов и большинство из них имеют ограниченный функционал.

Объект исследования – средства автоматизации рабочих процессов интернет-аукциона автомобилей.

Целью выпускной квалификационной работы разработка web-приложения для проведения интернет-аукциона автомобилей.

Для достижения поставленной цели необходимо выполнить следующие задачи:

- провести анализ требований;
- выполнить проектирование программных средств, структур данных и интерфейса;
- реализовать web-приложения для интернет-аукциона автомобилей;
- отладить и протестировать полученное приложение;
- составить план разработки и оценить бюджет разработки.

В работе использованы методы сравнительного анализа, методы проектирования, разработки и тестирования программного обеспечения.

1 Анализ программных средств для проведения интернет-аукциона автомобилей

1.1 Анализ предметной области

Для проведения анализа требований необходимо предварительно проанализировать предметную область.

Разработка интернет-аукциона автомобилей – это процесс создания web-платформы [2], которая позволяет пользователям производить покупки и продажи автомобилей в режиме онлайн. Данный вид бизнеса является одним из самых динамично развивающихся в сфере электронной коммерции, т.к. позволяет сэкономить время и деньги клиентов, а также упрощает процесс продаж и покупок автомобилей.

Потенциальные пользователи данного сервиса – это как физические, так и юридические лица, которые заинтересованы в покупке или продаже автомобилей. Для пользователей сервис предлагает удобный и быстрый способ поиска и просмотра доступных автомобилей, а также функцию участия в аукционе, что позволяет значительно снизить цену на приобретаемый автомобиль.

Итак, разработка интернет-аукциона автомобилей является перспективным направлением в сфере электронной коммерции, которое позволяет сократить время и затраты клиентов на покупку или продажу автомобилей.

1.2 Обзор продуктов-аналогов

В качестве аналогов разрабатываемого приложения были выбраны следующие сервисы:

- сайт аукцион «Carlink» (<https://carlink.ru/>);
- сайт аукцион «Auto-mama» (<https://automama.ru/express-auction>);
- сайт аукцион «Exleasingcar» (<https://www.exleasingcar.com/ru>);

Данные решения имеют положительные оценки пользователей и являются популярными. Первые два варианта представляют собой сервисы российских аукционов. Это особенно важно, поскольку необходимо иметь представление о существующих аналогах в стране.

1.2.1 Сайт-аукцион «Carlink»

«CarLink» – это сайт, который предоставляет возможность покупки и продажи автомобилей в различных городах. Интерфейс приложения представлен на рисунке 1.

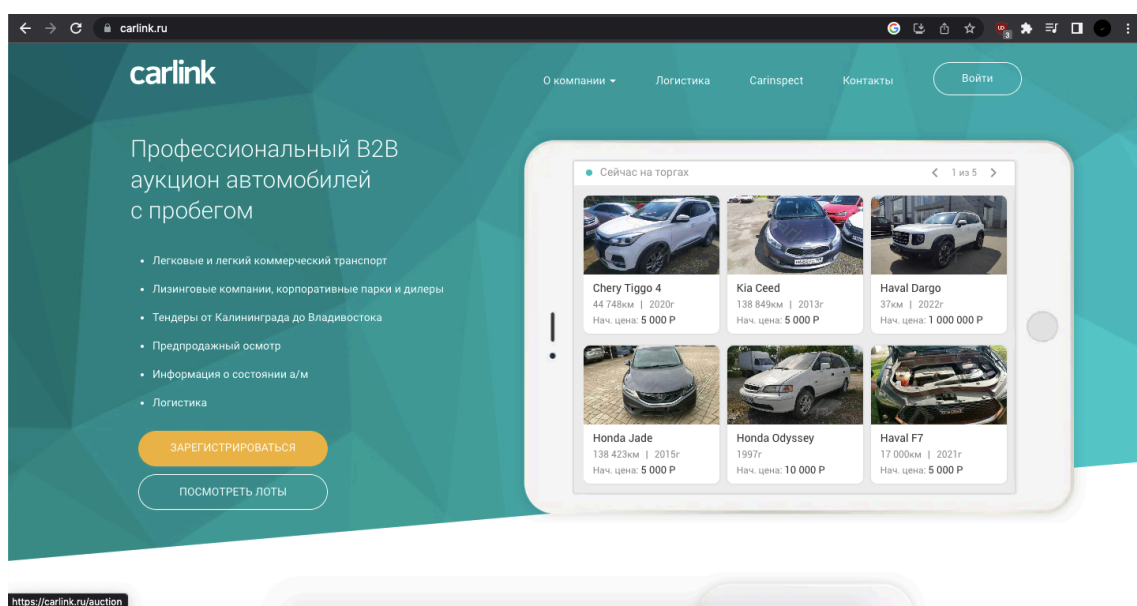


Рисунок 1 – Интерфейс сайта «Carlink»

Преимуществами данного сайта являются: удобный интерфейс, наличие фильтров, связь с продавцом.

Недостатками данного сайта являются: ограниченный функционал, наличие обязательной регистрации, неполная информация об автомобилях.

Как видно из проведенного анализа, сайт «Carlink» имеет больше преимуществ, чем недостатков, однако наличие обязательной регистрации для просмотра лотов отталкивает новых пользователей.

1.2.2 Сайт-аукцион «Auto-mama»

Это сайт, на котором можно купить или продать автомобиль. Сайт специализируется на продаже автомобилей с пробегом и является одним из крупнейших автомобильных порталов в России. Интерфейс приложения представлен на рисунке 2.

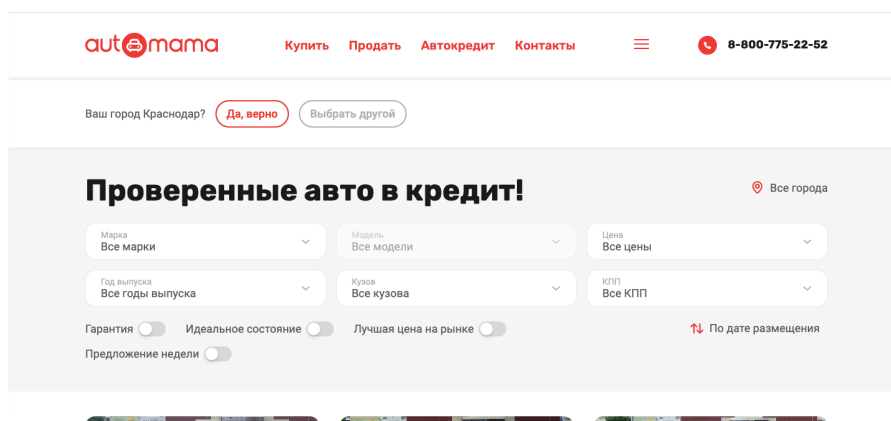


Рисунок 2 – Интерфейс сайта «Auto-mama»

Преимуществами данного сайта являются: огромная база автомобилей, наличие версии для мобильных устройств, наличие страницы новостей, которая постоянно обновляется.

Недостатками данного сайта являются: навязчивые рекламные объявления, неудобный интерфейс.

Интерфейс неудобен для пользователя и слишком много рекламных объявлений.

1.2.3 Сайт-аукцион «Exleasingcar»

Это сайт, на котором можно купить бывший в лизинге автомобиль. Сайт специализируется на продаже автомобилей премиум-класса с пробегом из лизинговых компаний и является одним из крупнейших автомобильных порталов на рынке СНГ. Интерфейс приложения представлен на рисунке 3.

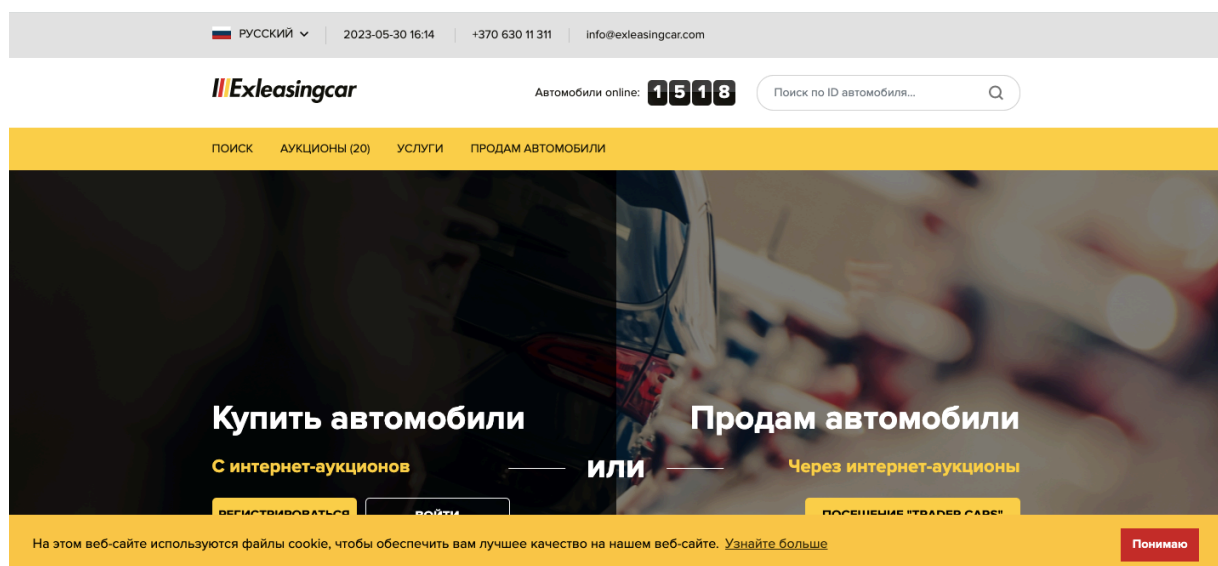


Рисунок 3 – Интерфейс сайта «Exleasingcar»

Преимуществами данного сайта являются: большой выбор автомобилей, удобный поиск, удобное представление информации, интуитивно понятный интерфейс, информативные статьи.

Недостатками данного сайта являются: ограниченный выбор марок и моделей, нет возможности добавить свои лоты.

Сайт «Exleasingcar» является хорошим примером сайта для проведения интернет-аукциона автомобилей. Имеет красивый и интуитивно понятный интерфейс, всю необходимую информацию для выбора автомобиля. Недостатком является отсутствие возможности добавить свои лоты.

1.3 Сравнительный анализ и критериальная оценка аналогов

Для составления оценки оформим таблицу, в которой будут отражены критерии значимости. Сравнительный анализ заключается в том, что несколько аналогичных решений рассматриваются одновременно, а их основные параметры и характеристики сравниваются друг с другом. Данная информация представлена в таблице 1.

Таблица 1 – Выделение критериев значимости

№	Название критерия
	Функциональные возможности
1	Просмотр списка лотов
2	Просмотр информации о лоте
3	Создание лотов
4	Просмотр информации о автомобиле
5	Наличие форм регистрации и авторизации
6	Добавление лотов в избранное
	Технические возможности
7	Возможность комфортного использования на слабом устройстве
	Необязательные функциональные возможности
8	Бесплатность продукта
	Эргономика
9	Наличие привлекательного интерфейса
10	Наличие интуитивно понятного интерфейса
11	Отсутствие лишней информации, в которой легко запутаться

Для оценки приведенных выше критериев используется трехбалльная система оценки, в которой значения изменяются от 0 до 2. В таблице 2 приведены более подробные характеристики оценки.

Таблица 2 – Шкала баллов

Балл	Характеристика оценки
0	функция (качество) отсутствует, реализовать ее невозможно
1	похожие, но отличные от требуемых, функции или качества системы находятся в стадии разработки
2	функция или качество безоговорочно реализовано

Для выставления оценок необходимо подробнее рассмотреть каждое приложение.

Оценка проводится по следующей формуле:

$$O_{\text{пп}} = \sum_{i=1}^n (Z_i \cdot K_i),$$

где $O_{\text{пп}}$ – интегральная оценка программного продукта;

n – количество критериев сравнения заявленных продуктов согласно таблице критериев;

Z_i – оценка критерия;

K_i – коэффициент значимости критерия сравнения.

На основании полученной в ходе анализа информации составим таблицу, в которой будут наглядно видны преимущества и недостатки каждого продукта и общая их оценка. Результат отображен в таблице 3.

Таблица 3 – Критериальная оценка программных продуктов

№	Критерий	Программные средства для проведения интернет-аукционов		
		«CarLink»	«Auto-mama»	«Exleasingcar»
1	Просмотр списка лотов	2	1	2
2	Просмотр информации о лоте	1	2	2
3	Создание лотов	2	0	0
4	Просмотр информации о автомобиле	1	1	2
5	Наличие форм регистрации и авторизации	1	1	2
6	Добавление лотов в избранное	1	1	1
7	Возможность комфортного использования на слабом устройстве	1	2	2
8	Бесплатность продукта	2	2	2
9	Наличие привлекательного интерфейса	2	1	2
10	Наличие интуитивно понятного интерфейса	2	1	2
11	Отсутствие лишней информации, в которой легко запутаться	2	1	2
	Интегральная оценка	17	13	19

На основании полученных оценок была построена столбчатая диаграмма, отображенная на рисунке 4.

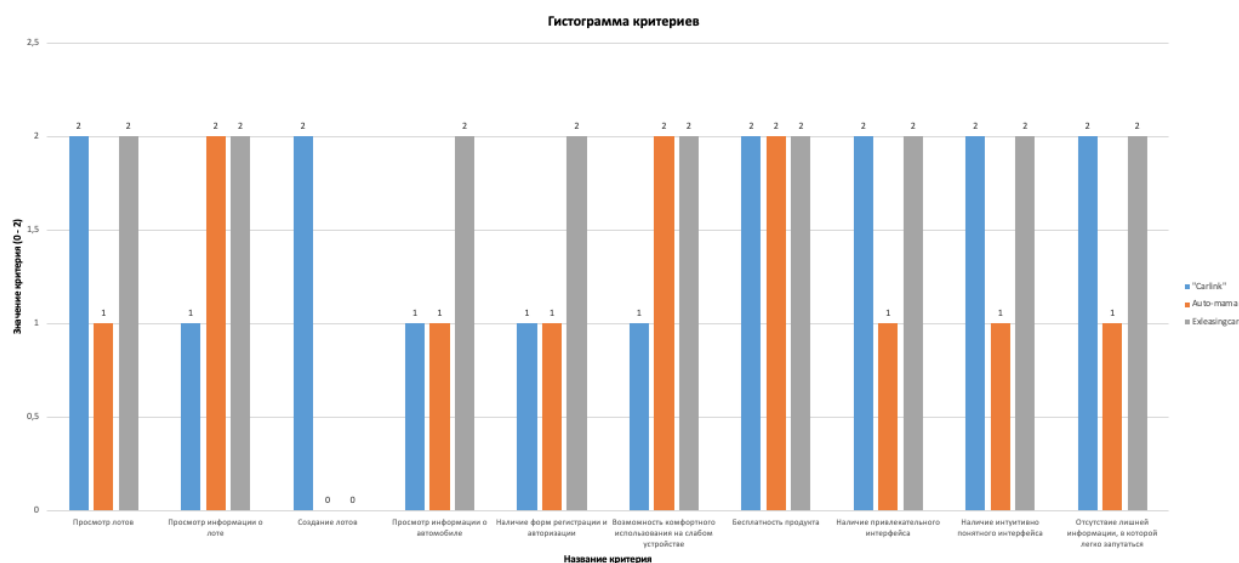


Рисунок 4 – Диаграмма оценок по критериям

Данная диаграмма представляет собой данные в виде разноцветных столбцов. Значение цвета указано в легенде.

На рисунке 5 отображена столбчатая диаграмма итоговых оценок.

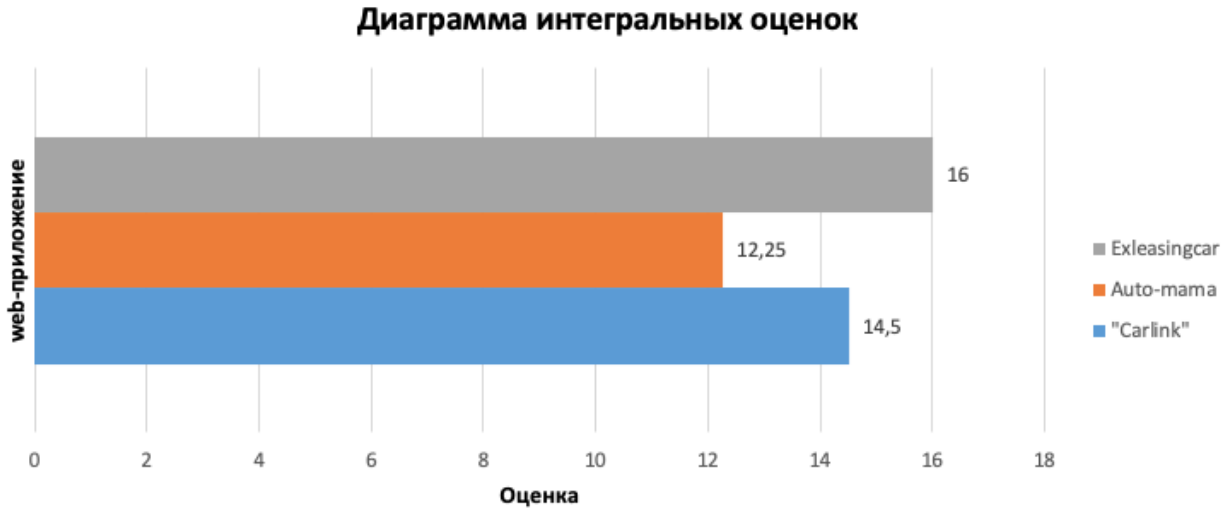


Рисунок 5 – Столбчатая диаграмма интегральных оценок

Подводя итог сравнения, можно сделать вывод, что ни один из сервисов не отвечает всем критериям в полной мере. Сайт «Carlink» имеет удобный дизайн, но не предоставляет пользователю информацию в нужной мере. Сайт «Auto-mama» имеет всю нужную информацию, но не имеет интуитивно понятного и привлекательного дизайна. Наибольшее количество баллов набрал сайт «Exleasingcar», однако он не имеет функции добавления своих лотов.

1.4 Исследование вариантов использования

Варианты использования UML используются для отображения функциональности, которую должна предоставлять система для пользователей и других систем. Она показывает, как система взаимодействует с актерами, чтобы выполнять конкретные задачи в определенных сценариях использования [3].

Web-приложение предназначено для взаимодействия пользователя с интернет аукционом в удаленном формате. Пользователи могут получить

полную информацию о лотах и сделать ставку на выбранный лот, а также добавить свой лот, используя данную систему.

Web-приложение должно обеспечить пользователя следующими возможностями:

- просмотр информации о лотах;
- просмотр списка лотов
- добавлять лоты в избранные;
- создание лотов;
- авторизация в приложении.

На основе выделенных задач была сформирована диаграмма вариантов использования, представленная на рисунке 6.

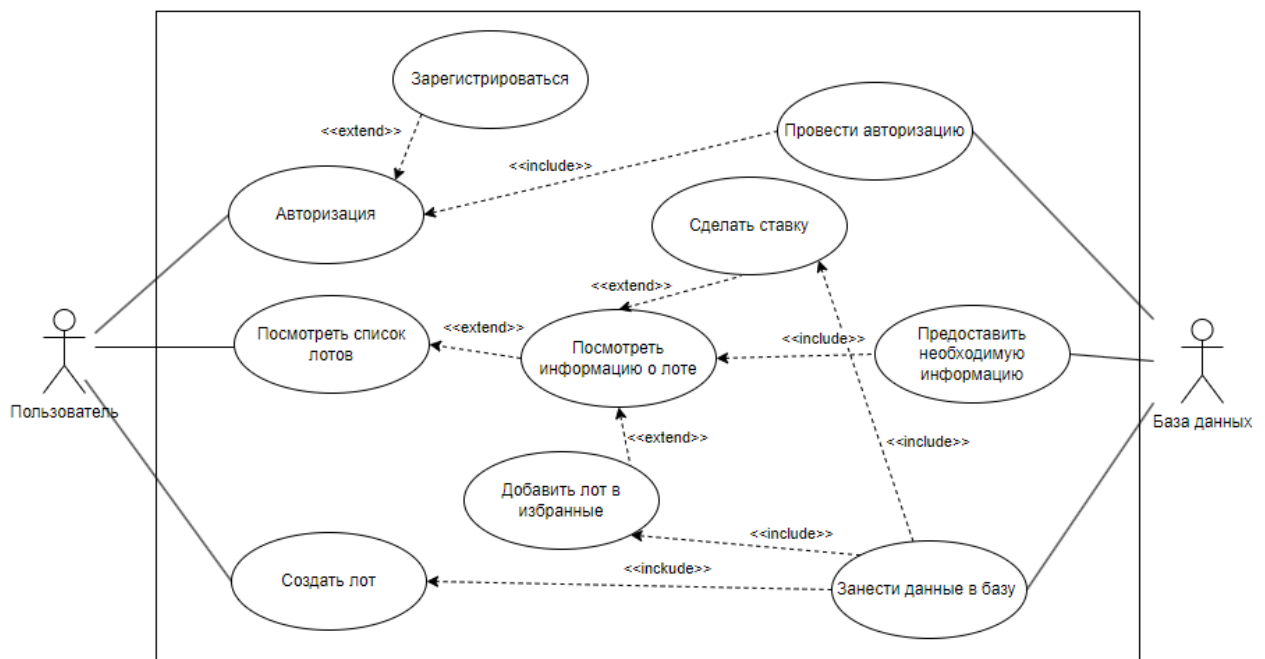


Рисунок 6 – Диаграмма вариантов использования для web-приложения интернет-аукциона автомобилей

Для вариантов использования «Авторизоваться» и «Посмотреть список лотов» были составлены спецификации, приведенные на рисунках 7-8.

Название: авторизоваться
ID: 1
Краткое описание: Система предоставляет пользователю возможность войти в свою учетную запись.
Главные актеры: Пользователь, База данных.
Второстепенные актеры: Нет.
Предусловие: База данных подключена к приложению, система загружена.
Основной поток: Система предоставляет пользователю форму для ввода данных. Пользователь вводит логин. Пользователь вводит пароль. Пользователь жмёт кнопку «Войти». Система проверяет наличие пользователя с введенными логином и паролем в базе. Система открывает пользователю доступ к основному функционалу web-приложения.
Постусловие: Пользователь вошел в свою учетную запись.
Альтернативные потоки: Пользователь не был найден в базе, система выводит сообщение о необходимости повторить попытку входа.

Рисунок 7 – Спецификация варианта использования «Авторизоваться»

Вариант использования «Авторизоваться» будет реализован добавлением в приложение страницы с авторизацией и регистрацией пользователя.

Название: просмотреть список лотов
ID: 2
Краткое описание: Система предоставляет пользователю список всех лотов.
Главные актеры: Пользователь, База данных.
Второстепенные актеры: Нет.
Предусловие: База данных подключена к приложению, система загружена.
Основной поток: Система предоставляет пользователю список лотов.
Постусловие: Пользователь просмотрел информацию о лотах.
Альтернативные потоки: 2.1. Пользователь выбирает просмотр информации о лоте. 3.1. Система отображает информацию о лоте.

Рисунок 8 – Спецификация варианта использования «Просмотреть список лотов»

Для реализации варианта использования «Просмотреть список лотов» будет создана специальная страница.

Для вариантов использования «Посмотреть информацию о лоте» и «Создать лот» были составлены спецификации, приведенные на рисунках 9-10.

Название: посмотреть информацию о лоте
ID: 3
Краткое описание: Система предоставляет пользователю информацию о выбранном лоте.
Главные актеры: Пользователь, База данных.
Второстепенные актеры: Нет.
Предусловие: База данных подключена к приложению, система загружена.
Основной поток: Система предоставляет пользователю кликабельную ссылку на лот. Пользователь выбирает интересующий лот. Система отображает страницу информации с лотом Пользователь возвращается на страницу со списком лотов.
Постусловие: Пользователь просмотрел информацию об лоте.
Альтернативные потоки: Пользователь выбирает сделать ставку.

Рисунок 9 – Спецификация варианта использования «Посмотреть информацию о лоте»

Чтобы просмотреть информацию о лоте, пользователю необходимо нажать на лот в списке. После этого сформируется страница, на которой будет представлена информация.

Название: создать лот
ID: 4
Краткое описание: Пользователь создает новый лот, а система вносит информацию в базу.
Главные актеры: Пользователь, База данных.
Второстепенные актеры: Нет.
Предусловие: База данных подключена к приложению, система загружена.
Основной поток: Система предоставляет пользователю пункт меню с названием «Создать лот». Пользователь жмет на кнопку «Создать лот». Система отображает форму для создания лота. Пользователь вводит информацию о лоте. Система обновляет информацию в базе данных
Постусловие: Пользователь создал лот, информация в базе данных обновилась.
Альтернативные потоки: Пользователь удаляет лот.

Рисунок 10 – Спецификация варианта использования «Создать лот»

Для создания лота в приложении предполагается реализация страницы «Создать лот».

Для вариантов использования «Сделать ставку» и «Добавить лот в избранное» были составлены спецификации, приведенные на рисунках 11-12.

Название: сделать ставку
ID: 5
Краткое описание: Система предоставляет пользователю возможность сделать ставку на лот.
Главные актеры: Пользователь, База данных.
Второстепенные актеры: Нет.
Предусловие: База данных подключена к приложению, система загружена.
Основной поток: Пользователь выбирает интересующий лот. Система отображает страницу лота с кнопкой «Сделать ставку». Пользователь жмет на кнопку «Сделать ставку». Пользователь вводит сумму ставки. Система обновляет информацию в базе данных.
Постусловие: Пользователь сделал ставку на лот, информация в базе данных обновилась.
Альтернативные потоки: Пользователь отказывается от ставки.

Рисунок 11 – Спецификация варианта использования «Сделать ставку»

Для того чтобы сделать ставку, пользователям необходимо будет авторизоваться в системе, выбрать интересующий их лот, определить размер ставки и подтвердить ее, нажав на кнопку «Сделать ставку».

Название: добавить лот в избранное
ID: 6
Краткое описание: Система предоставляет пользователю добавить лот в список избранных.
Главные актеры: Пользователь, База данных.
Второстепенные актеры: Нет.
Предусловие: База данных подключена к приложению, система загружена.
Основной поток: Пользователь выбирает интересующий лот. Система отображает страницу лота с кнопкой «добавить в избранное». Пользователь жмет на кнопку «добавить в избранное». Система обновляет информацию в базе данных.
Постусловие: Пользователь добавил лот в избранное, информация в базе данных обновилась.
Альтернативные потоки: Пользователь убирает лот из избранного.

Рисунок 12 – Спецификация варианта использования «Добавить лот в избранное»

Для того чтобы добавить лот в избранное, пользователям необходимо будет авторизоваться в системе, выбрать интересующий их лот, и нажать на кнопку «Добавить в избранное». После этого информация в базе данных обновится, и лот будет закреплен за пользователем.

1.5 Анализ требований

Для формирования функциональных требований, необходимых для основных вариантов использования, необходимо провести анализ полученных результатов и составить соответствующие требования. Явный набор требований крайне важен по ряду причин. Одна из них заключается в том, что такие требования позволяют гарантировать, что функциональность системы определяется исключительно пользователем, а не программистом. Ясно сформулированные требования дают пользователю возможность провести анализ и утвердить их, если они соответствуют его потребностям [4]. Выделим функциональные требования к продукту. Приложение интернет-аукциона автомобилей должно позволять пользователю выполнять следующие функции:

- проводить авторизацию или регистрацию в системе;
- просмотр списка лотов
- добавлять лоты в избранные;
- просматривать информацию о лотах;
- создавать лоты.

Для того чтобы иметь доступ к возможности делать ставки и создавать свои лоты, пользователь должен создать аккаунт. В случае отсутствия аккаунта, данный функционал будет недоступен.

База данных должна быть разработана и функционировать в среде MySQL. Для взаимодействия с базой данных необходимо использовать приложение, запущенное в браузере, которое будет принимать запросы от пользователя, обрабатывать и направлять их в MySQL. В данном случае,

персональный компьютер будет использоваться в качестве сервера-посредника.

1.6 Постановка задачи

Разрабатываемое приложение должно соответствовать указанным требованиям и предоставлять пользователям возможность просмотра интересующей информации о лотах. В таблице 4 описаны категории пользователей информационной системы.

Таблица 4 – Категория пользователей информационной системы

№	Категория пользователей	Наименование приложения	Фрагмент структуры предметной области	Информационные задачи
1	Администратор	Администрирование	Список лотов; Информация о лотах	Модерация лотов
2	Пользователь	Интернет-аукцион автомобилей	Список лотов; Информация о лотах; Создание лотов; Информация о ставках	Просмотр информации; Создание лотов

Различные категории пользователей имеют различные права. Так, пользователю из категории «Администратор» доступны все привилегии категории «Пользователь», а также управление данными.

Для реализации разделения прав будет предусмотрена система авторизации.

2 Проектирование web-приложения аукциона автомобилей

2.1 Архитектура приложения

Web-приложение для аукциона автомобилей включает в себя несколько компонентов, включая базу данных и сервер, а также сервер приложений (API), который основан на архитектуре REST API, а так же оно содержит клиентскую часть с реализацией интерфейса пользователя. В зависимости от конкретной реализации, архитектура web-приложения может включать в себя различные компоненты и протоколы, но основная идея заключается в том, чтобы разделить приложение на четко определенные компоненты, соответствующие своим функциям. Для обеспечения безопасности данных и упрощения последующей поддержки, сервер приложений и клиентская часть должны спроектированы на основе шаблона проектирования MVC. Данный шаблон проектирования, который разделяет систему на три компонента: модель данных (Model), пользовательский интерфейс (View) и управление потоком данных и обработки пользовательских действий (Controller). Это позволяет улучшить организацию кода, уменьшить зависимость между компонентами системы, упростить тестирование и обеспечить более легкую поддержку [5]. На рисунке 13 изображена схема взаимодействия web-приложения с REST API и сервером базы данных.

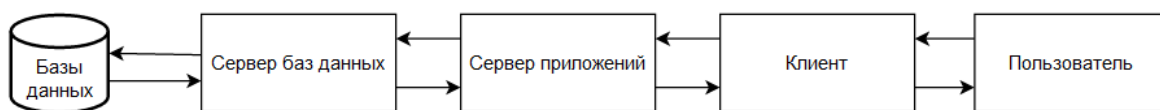


Рисунок 13 – Схема архитектуры разрабатываемого приложения

Как видно из представленного рисунка, каждый компонент обменивается данными с другим, за счет чего система реагирует на действия пользователя и отправляет необходимую информацию. Так, компонент «Web-приложение» обрабатывает действия компонента «Пользователь», после чего при необходимости посылает запросы к «REST API» с помощью HTTP-

запросов. Сервер приложения (API) соединяется с компонентом «MySQL» и отправляет SQL-запросы к компоненту «База данных». После обработки запроса сервер БД возвращает данные в обратном порядке, и пользователю отображается запрашиваемая информация.

Web-приложение используется для отображения данных. Этот компонент преобразует действия пользователя в HTTP-запросы. Основная бизнес-логика содержится в сервере приложения и базе данных.

2.2 Проектирование структур данных

Проектирование структуры данных является одним из важнейших этапов при разработке информационных систем и может повлиять на успешность и производительность системы в целом.

Для организации и хранения наборов информации необходимо использовать базу данных. Системы баз данных позволяют пользователям эффективно управлять данными, сохранять их в структурированной форме, обрабатывать и анализировать. Базу данных можно рассматривать как электронную картотеку, которая представляет собой хранилище или контейнер для файлов с данными, хранящимися в компьютере [6].

2.2.1 Концептуальная модель базы данных

Концептуальная модель представляет собой описание основных сущностей (таблиц) и связей между ними независимо от ограничений, вытекающих из моделей данных, поддерживаемая той или иной СУБД. Концептуальная модель является базой для разработки логической модели, которая более детально описывает структуру хранения данных [7].

Графическое отображение концептуальной модели базы данных, представляющее диаграмму вида «сущность-связь», представлено на рисунке 14.

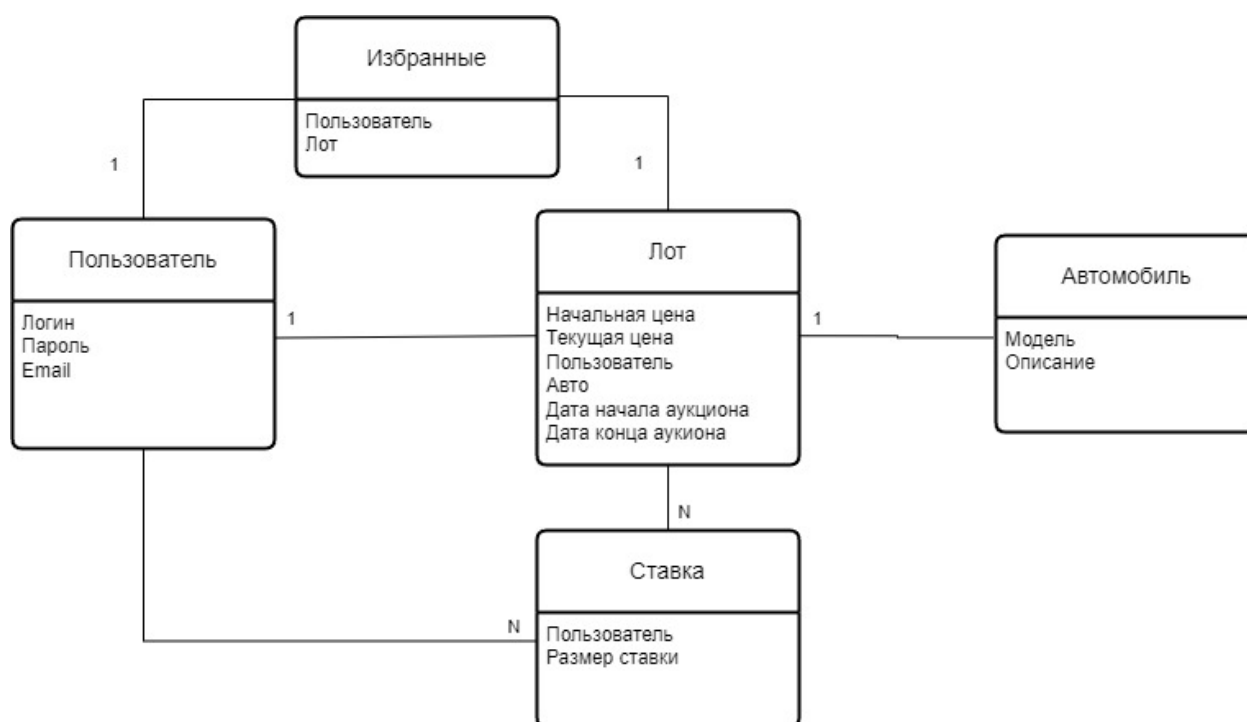


Рисунок 14 – Концептуальная модель базы данных приложения

Как видно из рисунка, база данных будет содержать тринадцать сущностей. В базе данных сущности представляют реальные или абстрактные объекты, которые имеют существенное значение для решаемой проблемы. Они описываются своими атрибутами и могут быть взаимосвязаны между собой через отношения [8]. Их описание представлено в таблице 5.

Таблица 5 – Описание сущностей

№	Наименование	Описание	Категория	Количество экземпляров	Режим обновления
1	Лот	Информация о лоте.	Стержневая	5000	Раз в день или чаще
2	Автомобиль	Информация о автомобиле	Стержневая	1000	Раз в день или чаще
3	Ставка	Информация о ставках	Стержневая	10000	Несколько раз в час
4	Пользователь	Информация о пользователях	Стержневая	1000	Несколько раз в час
5	Избранные	Информация о избранных лотах	Стержневая	1000	Несколько раз в час

Как видно из таблицы, база данных содержит четыре стержневых таблиц и семь вспомогательных. Вспомогательные таблицы используются для уточнения данных.

2.2.2 Логическая модель базы данных

После выделения необходимых сущностей нужно построить логическую модель базы данных. Логическая модель отражает суть предметной области и позволяет указать первичные и внешние ключи.

Логическая модель базы данных – это диаграмма, которая описывает структуру хранения данных в базе данных, но не указывает на конкретные технологии реализации. Логическая модель позволяет определить сущности и их атрибуты, а также связи между ними [9].

Логическая модель базы данных обычно создается на основе бизнес-требований и процессов предприятия, и она служит основой для создания физической модели данных. Логическая модель помогает определить, какие данные нужны для обработки бизнес-процессов, какие связи между данными нужны и как организовать данные в базе данных. Это позволяет создать эффективную базу данных, которая довольно надежна и легко обновляема.

Графическое отображение логической модели базы данных в виде диаграммы вида «сущность-связь» представлено на рисунке 15.

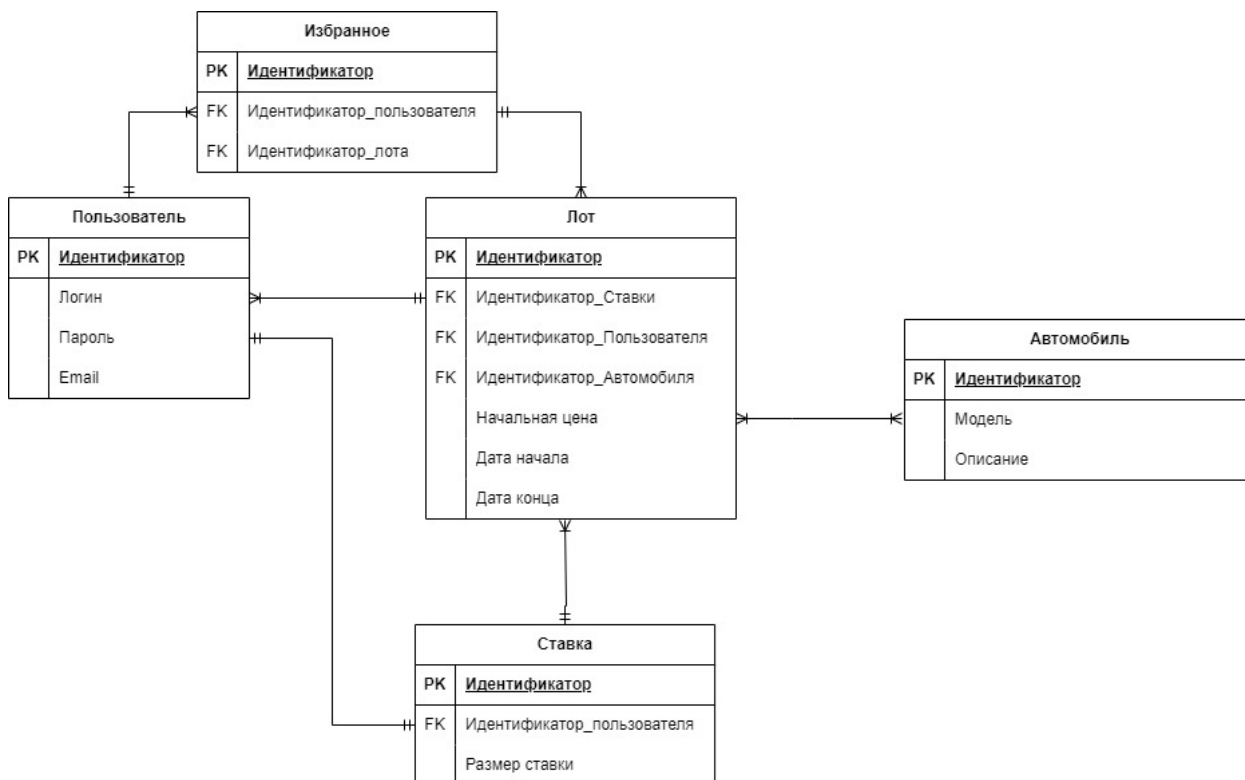


Рисунок 15 – Логическая модель базы данных

На основании построенной логической модели были выделены характеристики атрибутов каждой сущности. Атрибуты и их описание представлены в приложении Б.

На данном этапе можно считать проектирование структур данных завершенным.

2.3 Проектирование программных средств

Для представления сущностей в ходе программирования необходимо выделить соответствующие классы. Выделение классов помогает организовать программу с точки зрения структуры и упростить ее поддержку. Для этого в проекте должен быть создан каталог «Сущности». Не каждая сущность нуждается в своей модели. Итоговая диаграмма классов представлена на рисунке 16.

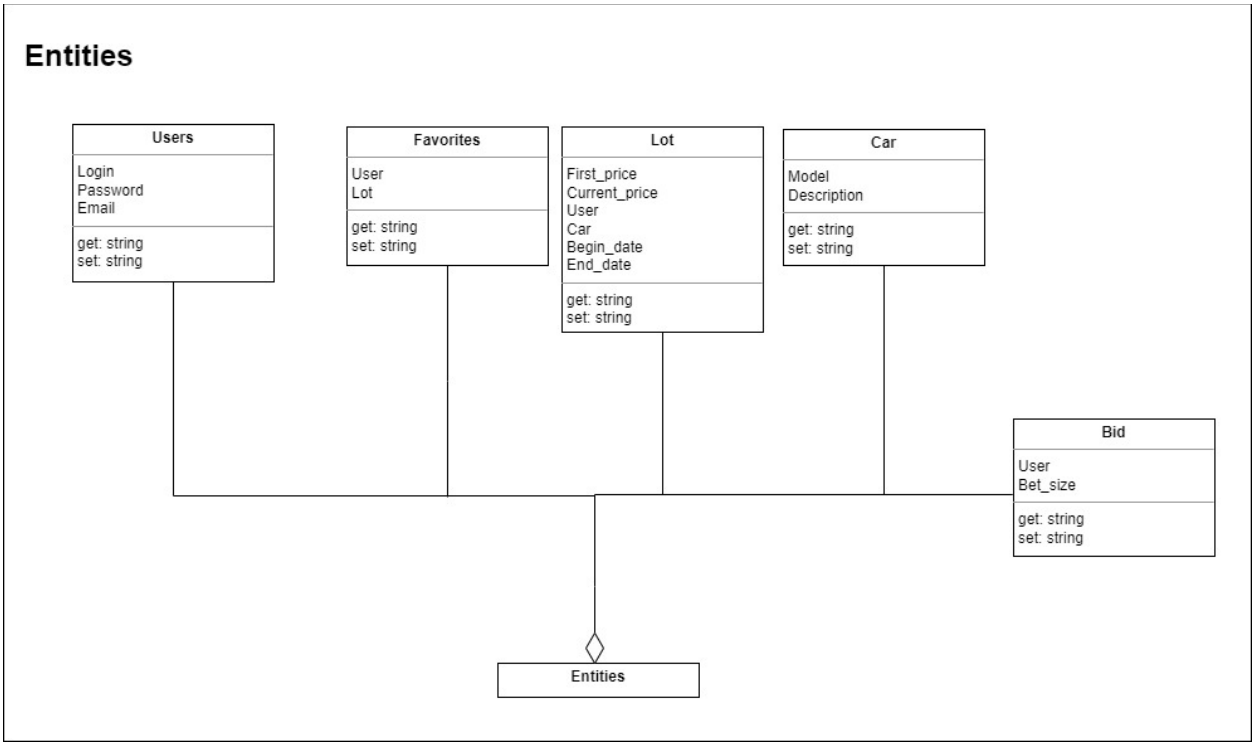


Рисунок 16 – Диаграмма классов приложения

Представленная диаграмма подходит как к архитектуре клиентского приложения, так и к архитектуре REST API-приложения, так как они получают и отправляют одни и те же данные.

Для варианта использования «Сделать ставку» была построена диаграмма деятельности. Инициатором действия является пользователь, поэтому его необходимо включить в диаграмму. Клиентское приложение обрабатывает все действия пользователя, такие как переход на страницу с списком лотов, переход на страницу лота. В нем также реализована часть логики, а именно отображение информации о необходимости авторизации и сообщение о том, что необходимо ввести размер ставки больше текущей. Занесение информации в базу проходит в приложении-сервере. Полученный результат представлен на рисунке 18.

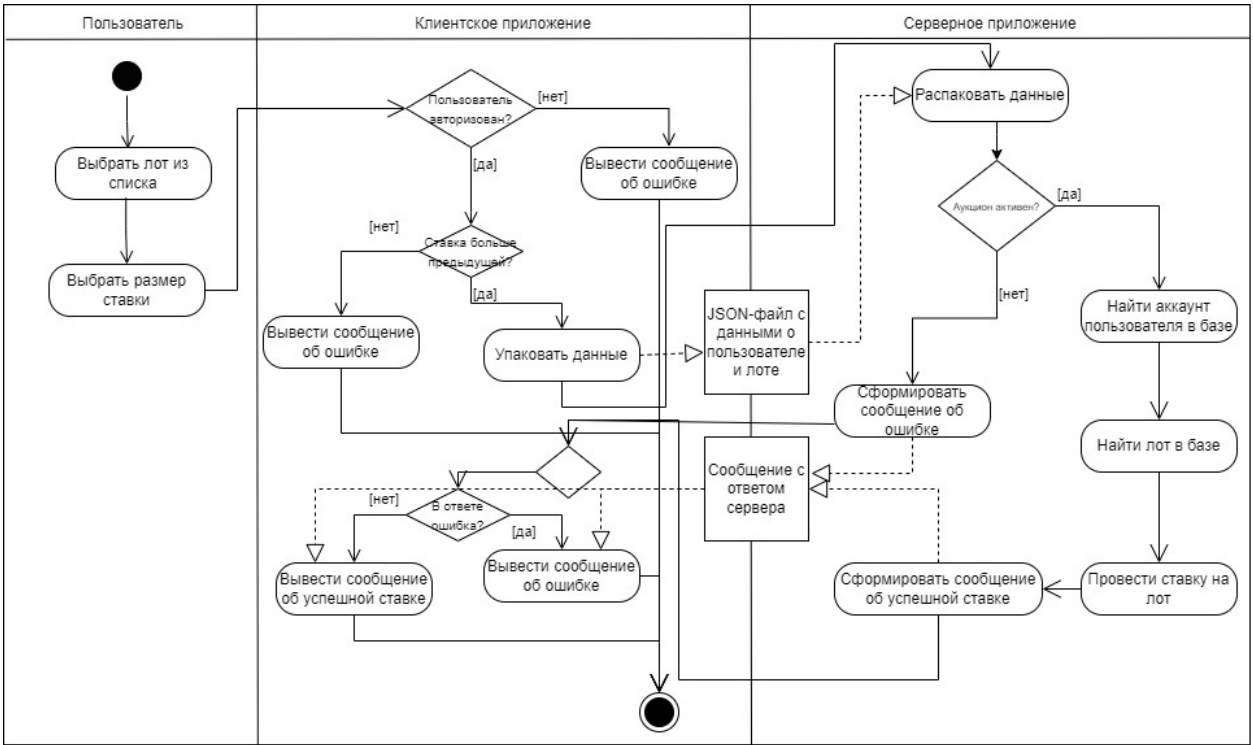


Рисунок 17 – Диаграмма деятельности web-приложения для проведения интернет-аукциона автомобилей

Диаграмма деятельности в нотации UML отображает последовательность действий, которые должны быть выполнены, чтобы достичь конечной цели. Она позволяет показать, какие шаги необходимо сделать для выполнения задачи, а также какие ресурсы необходимы для ее выполнения [10].

2.4 Проектирование интерфейса

Проектирование интерфейса – это процесс создания понятного, эффективного и пользовательски удобного взаимодействия между пользователями и системой [11].

Интерфейс приложения должен быть максимально удобен в использовании и соответствовать потребностям пользователей. Также интерфейс должен соответствовать выделенным требованиям в полном объеме. Должны присутствовать следующие страницы:

- страница регистрации и авторизации;
- страница со списком лотов;
- страница лота;
- страница создания лота;
- страница с избранными лотами.

Для правильной реализации интерфейса необходимо составить макеты. Макет главной страницы со списком лотов представлен на рисунке 18.

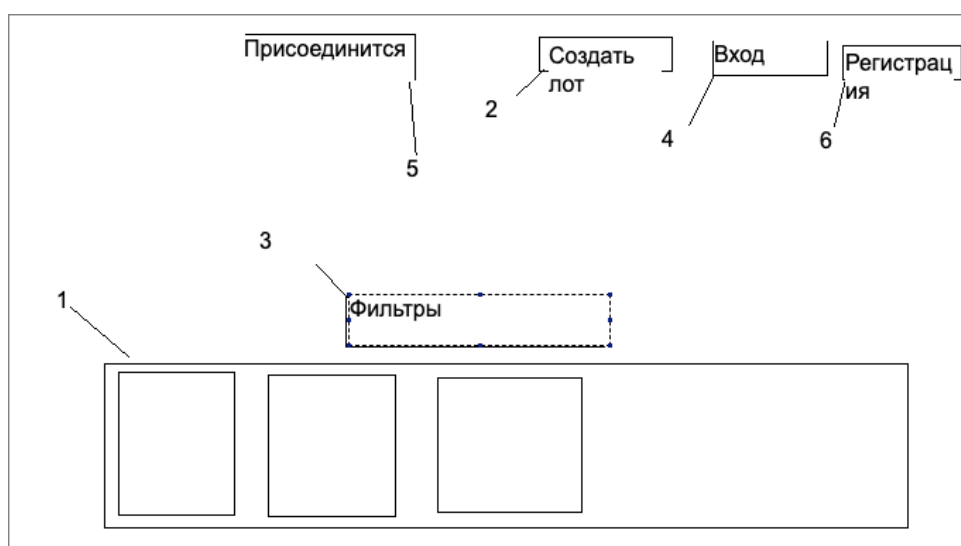


Рисунок 18 – Макет интерфейса главной страницы со списком лотов

На рисунке выделены несколько пронумерованных элементов. Элемент 1 представляет собой список лотов. Элемент 2 представляет собой кнопку для перехода на страницу создания лота. 3 – форма позволяющая настроить фильтрацию. Элемент 4 на рисунке представляет собой кнопку,

которая используется для авторизации. Элемент 5 на рисунке представляет собой кнопку, которая используется для авторизации, а элемент 6 – кнопка для регистрации, если у пользователя нет аккаунта. В случае неверного ввода данных или попытки регистрации с занятым логином, система выдаст уведомление о неверном вводе.

На рисунке 19 представлен макет интерфейса страницы входа.

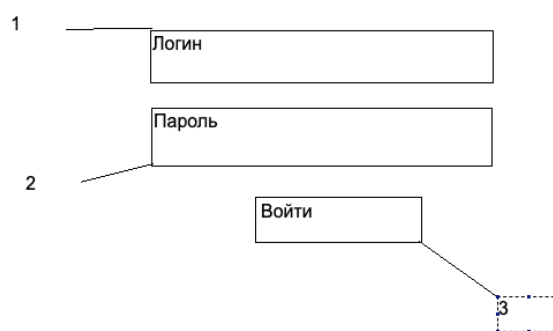


Рисунок 19 – Макет интерфейса страницы «Авторизации»

Элемент 1 на рисунке является поле для ввода логина пользователя. Элемент 2 является поле для ввода пароля пользователя.

На рисунке 20 представлен макет интерфейса страницы регистрации.

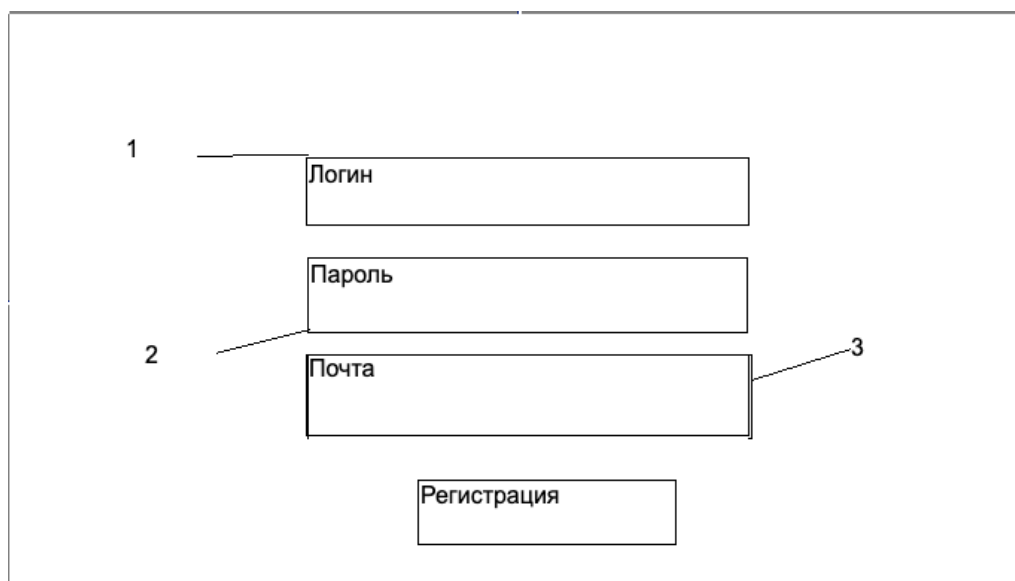


Рисунок 20 – Макет интерфейса страницы «Регистрация»

Элемент 1 на рисунке является поле для ввода логина пользователя. Элемент 2 является поле для ввода пароля пользователя. Элемент 3 на рисунке является поле для ввода почты пользователя.

Макет интерфейса списка созданных лотов представлен на рисунке 21.

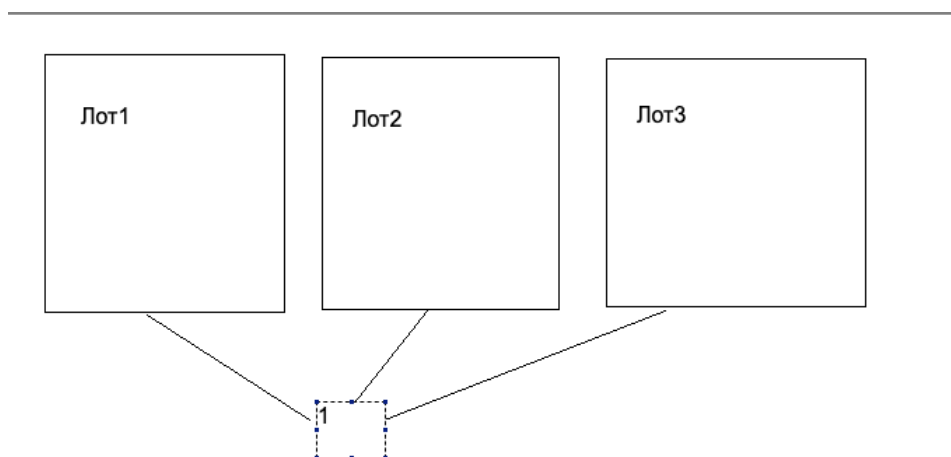


Рисунок 21 – Макет интерфейса страницы созданных лотов

Элемент 1 представляет собой список лотов.

На рисунке 22 представлен макет интерфейса страницы с информацией о лоте.

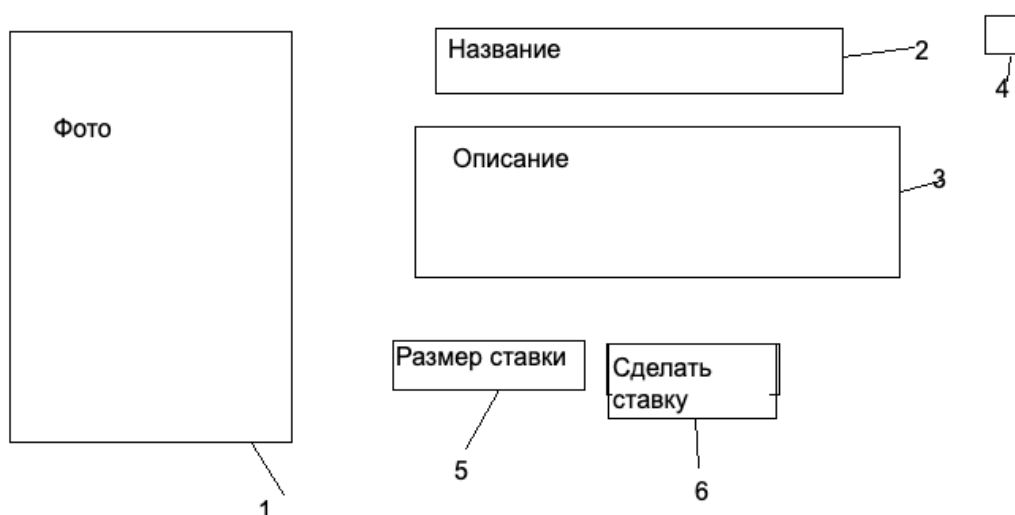


Рисунок 22 – Макет интерфейса страницы с информацией о лоте

На рисунке выделены несколько пронумерованных элементов. Элемент 1 представляет собой картинку лота. Элемент 2 представляет собой название лота. 3 – описание выбранного лота. Элемент 4 кнопка для добавления лота в избранное. Элемент 5 поле для ввода размера ставки, а элемент 6 – кнопка сделать вставку. В случае ввода ставки меньше текщей система покажет сообщение об ошибке.

На рисунке 23 представлен макет интерфейса страницы избранных лотов.

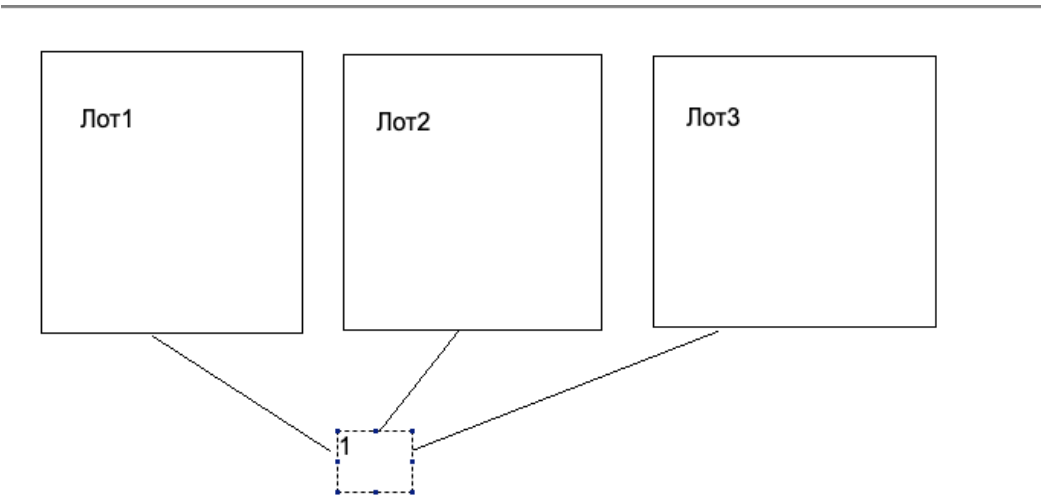


Рисунок 23 – Макет интерфейса страницы избранных лотов

Элемент 1 представляет собой список лотов.

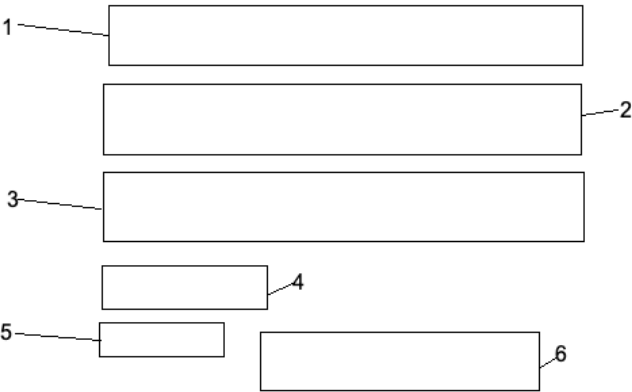


Рисунок 24 – Макет интерфейса страницы создания лотов

На рисунке 24 выделены несколько пронумерованных элементов. Элемент 1 представляет собой поле для ввода модели автомобиля. Элемент 2 представляет собой поле для ввода описания автомобиля. 3 – поле для ввода начальной цены лота. Элемент 4 поле для ввода даты конца аукциона. Элемент 5 поле для загрузки картинки, а элемент 6 – кнопка подтвердить.

3 Реализация web-приложение для интернет-аукциона автомобилей

3.1 Реализация серверной части приложения

3.1.1 Реализация базы данных

Реализацию базы данных следует начать с построения физической модели базы данных.

Создание Физической модели базы данных – это процесс создания схемы базы данных и описание способа хранения данных на физическом устройстве на основе логической модели базы данных. Она включает в себя определение таблиц и отношений между ними, выбор типов данных для полей таблицы, определение первичных и внешних ключей, индексов и других параметров, влияющих на производительность базы данных [12].

Используя построенную ранее логическую модель, были выделены характеристики атрибутов каждой сущности. Атрибуты и их описание представлены в приложении В.

Согласно составленным таблицам была построена физическая модель базы данных. Результат представлен на рисунке 25.

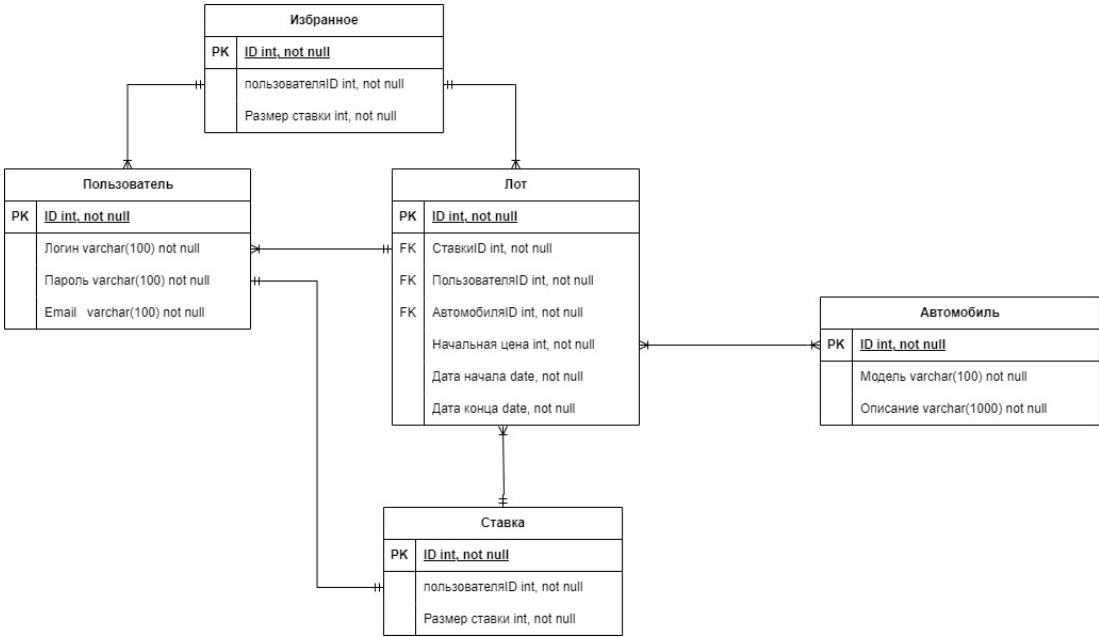


Рисунок 25 – Физическая модель базы данных приложения

Физическая модель отражает особенности, которые свойственны выбранному серверу базы данных. Так, в MySQL можно указывать размерность символьных атрибутов, но нельзя указать размер целочисленных.

3.1.2 Реализация серверных компонентов

Реализация серверных компонентов обычно включает в себя написание программного кода, который выполняет определенные функции на сервере. Эти компоненты могут обрабатывать запросы от клиентов, осуществлять доступ к базе данных, производить вычисления и многое другое.

Серверная часть приложения отвечает за обмен информацией между клиентом и базой данных. Она включает в себя классы, соответствующие задействованным сущностям в базе данных.

Серверная часть реализована в среде WebStorm. Взаимодействие с базой данных происходит с использованием библиотеки Sequelize.

На рисунке 26 представлена диаграмма классов приложения-сервера.

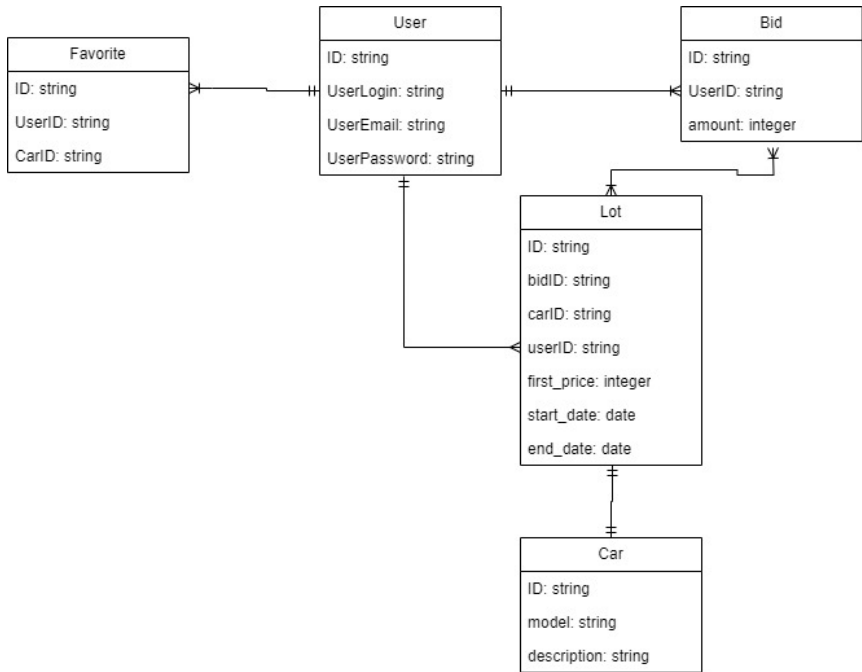


Рисунок 26 – Диаграмма классов приложения-сервера

Сервер приложения реализует архитектуру REST API с различными URL для обращения. REST API позволяет передавать данные между клиентом и сервером в формате JSON, используя различные HTTP методы, такие как

GET, POST, PUT и DELETE. Предоставляемые методы представлены в таблице 6.

Таблица 6 – Спецификация REST API приложения

№	Назначение сервиса	HTTP метод	URI сервиса	Класс контроллера
1	Получение списка лотов	GET	/auctions	AuctionsController
2	Получение подробной информации о выбранном лоте	GET	/ auctions /{ID}/	AuctionsController
3	Получение списка избранных лотов	GET	/favorites/{ID}	FavoriteController
5	Создание нового лота	POST	/lot	LotController
6	Получение списка созданных пользователем лотов	POST	/ auctions /{ID}/	AuctionsController
7	Сделать ставку	POST	/Bids/{ID}	BidsController
9	Регистрация пользователя в системе	POST	/Registration	AuthController
10	Авторизация пользователя в системе	POST	/Authorization	AuthController

На рисунке 27 представлена диаграмма компонентов сервера web-приложения интернет-аукциона автомобилей.

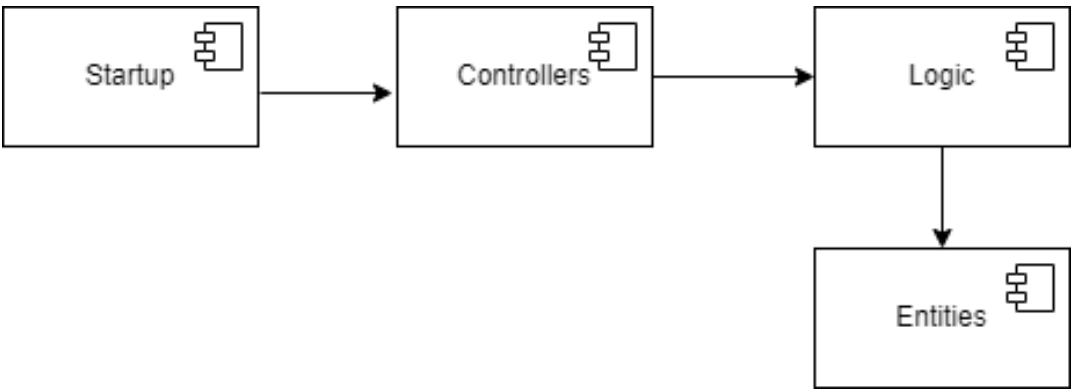


Рисунок 27 – Диаграмма компонентов сервера web-приложения интернет-аукциона автомобилей

Ниже представлено описание каждого компонента:

- компонент Startup – отвечает за логику запуска сервера в локальной сети; благодаря нему приложения-клиент может получать доступ по URL;

- компонент **Controllers** – промежуточный компонент, предназначенный для связи с логикой приложения; сюда включены контроллеры, перечисленные в таблице 6;
- компонент **Logic** – реализует логику приложения, а именно доступ к базе данных;
- компонент **Entities** – отвечает за представление сущностей в приложении; соответствует необходимым сущностям в базе данных.

Для корректной работы web-приложения в связке с сервером и базой данных, необходимо правильно развернуть все компоненты. Для этого была разработана диаграмма разворачивания, представленная на рисунке 28.

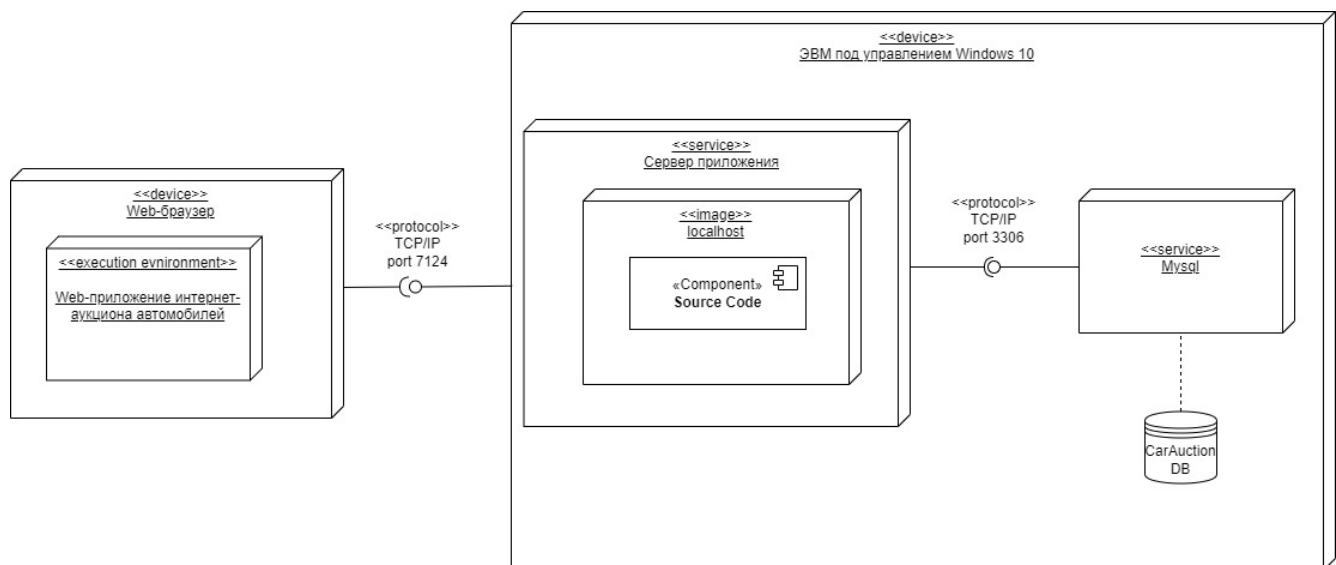


Рисунок 28 – Диаграмма разворачивания приложения

Согласно диаграмме, приложение соответствует трехуровневой архитектуре.

Клиентское приложение запускается на устройстве пользователя. Для корректной работы клиент должен быть подключен к локальной сети. Приложение-сервер запускается на персональном компьютере под управлением Windows 10 и связывается с сервером базы данных.

4 Контроль качества программного обеспечения

4.1 Выбор режима тестирования

Тестирование программного обеспечения – процесс проверки соответствия приложения требованиям и спецификациям. Основными целями тестирования является выявление и устранение ошибок и дефектов в работе приложения и проверка соответствия функциональных требований к приложению [13]

Для проверки отдельных компонентов web-приложения существует множество методов тестирования. В данном случае для разработанного приложения был выбран метод функционального тестирования черного ящика, который позволяет оценить работу приложения с точки зрения конечного пользователя. Функциональное тестирование позволяет удостовериться в том, что система работает согласно ожиданиям и реагирует правильно на действия пользователя. Учитывая размеры приложения, функциональное тестирование методом черного ящика является достаточно эффективным выбором.

Были рассмотрены следующие виды тестирования:

- функциональное тестирование;
- нефункциональное тестирование;
- интеграционное тестирование.

Функциональное тестирование – это метод тестирования программного обеспечения, который проверяет, соответствует ли приложение ожиданиям пользователя и выполняет ли оно требования функциональности. Цель функционального тестирования – проверить, работает ли приложение так, как должно работать, и выполняет ли оно функции согласно спецификациям и требованиям. Данный метод тестирования является одним из самых применяемых, так как позволяет наиболее полно проверить программу на соответствие требованиям [14].

Нефункциональное тестирование – это метод тестирования программного обеспечения, который проверяет, соответствуют ли нефункциональные требования приложения, такие как производительность, надежность, безопасность, удобство использования и т.д. В отличие от функционального тестирования, которое проверяет функциональность приложения, нефункциональное тестирование сфокусировано на тестировании тех аспектов приложения, которые не связаны непосредственно с его функционалом [15].

Интеграционное тестирование – это процесс тестирования, который проверяет, как работают различные части приложения вместе после их интеграции. Целью интеграционного тестирования является обнаружение ошибок взаимодействия между отдельными компонентами приложения, которые могут возникнуть в результате их интеграции. Интеграционное тестирование также может проводиться как в автоматическом, так и вручную, чтобы убедиться, что все компоненты работают должным образом и интеграция не приводит к ошибкам [16].

На основании анализа различных методов было принято решение провести функциональное тестирование, так как оно во многом имитирует работу конечного пользователя. В ходе тестирования будет выявлено соответствие реализованных требований ожидаемым.

4.2 Тестирование серверной части web-приложения интернет-аукциона автомобилей

Для основных вариантов использования были проведены функциональные тесты. Тестирование проводилось с использованием программы Postman, позволяющей отправлять различные запросы по заданному URL-адресу. Назначение, описание, ожидаемые и полученные результаты приведены в таблице 7.

Таблица 7 – Функциональные тесты серверной части приложения

№ теста	Назначение	Описание	Входные данные	Ожидаемый результат	Актуальный результат
1	Проверка корректности выдачи списка о лотов	По заданному URL отправляется GET-запрос	URL: /Auctions/	Список лотов, представленный в виде json-файла	Пройден
2	Проверка корректности авторизации	По заданному URL отправляется POST-запрос	URL: /Authorization Содержание json-файла: "userLogin": "pppppp", "userPass": "ataler97"	ID пользователя из базы данных	Пройден
3	Проверка регистрации с неверно указанными логином и паролем (данный пользователь уже зарегистрирован)	По заданному URL отправляется POST-запрос	URL: /Registration Содержание json-файла: "userID": null, "userLogin": "pppppp@mai.com", "userPass": "admin", "	ID -1, так как такой пользователь уже существует	Пройден
4	Проверка корректности получения информации о выбранном лоте	По заданному URL отправляется GET-запрос	URL: /Auction/{ID} ID: 8	Информация о выбранном лоте	Пройден
5	Проверка добавления нового лота	По заданному URL отправляется POST-запрос	URL: /Registration Содержание json-файла: "lotID": 3, "carID": 4, "userID": "1", "start_date": null, "end_date": null, "firstprise": null	Создан новый лот	Пройден
6	Проверка добавления лота в избранное	По заданному URL отправляется POST-запрос	URL: /Registration Содержание json-файла: "userID": null, "auctionID": "admin",	Лот добавлен в список избранных	Пройден

В листинге 1 представлен результат теста №1, ответ соответствует ожидаемому. Это означает, что сервер корректно реализует необходимую логику.

Листинг 1 – Часть полученного тела ответа, представляющая собой json-файл с СПИСКОМ ЛОТОВ

```

{
    "lotID": "1",
    "bidID": "1",
    "UserID": "1",
    "carID": "1"
    "first_price": "150000"
    "start_date": "15.06.2023"
    "end_date": "30.06.2023"
},
{
    "lotID": "2",
    "bidID": "2",
    "UserID": "1",
    "carID": "3"
    "first_price": "15000"
    "start_date": "18.06.2023"
    "end_date": "20.06.2023"
},
{
    "lotID": "3",
    "bidID": "4",
    "UserID": "2",
    "carID": "2"
    "first_price": "50000000"
    "start_date": "05.05.2023"
    "end_date": "25.06.2023"
},
{
    "lotID": "4",
    "bidID": "6",
    "UserID": "1",
    "carID": "4"
    "first_price": "20000000"
    "start_date": "21.06.2023"
    "end_date": "25.06.2023"
},
{
    "lotID": "4",
    "bidID": "6",
    "UserID": "1",
    "carID": "4"
    "first_price": "20000000"
    "start_date": "21.06.2023"
    "end_date": "25.06.2023"
}
,
{
    "lotID": "3",
    "bidID": "4",
    "UserID": "2",
    "carID": "2"
    "first_price": "50000000"
    "start_date": "05.05.2023"
    "end_date": "25.06.2023"
}

```

На рисунке 29 представлен результат теста №2, в качестве ответа возвращен ID, равный «1». Данный ID соответствует пользователю с логином «admin» и паролем «admin», значит, сервер корректно реализует логику авторизации.

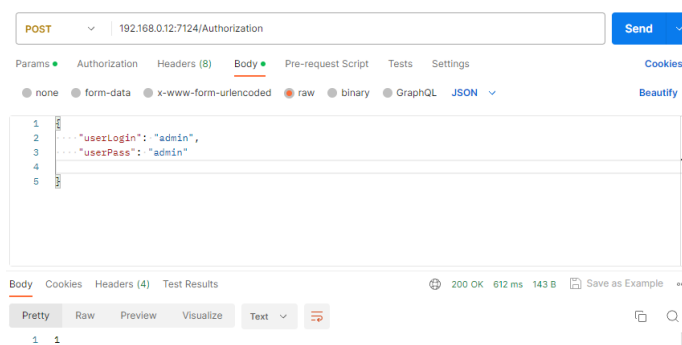


Рисунок 29 – Результат тестирования запроса авторизации (тест №2)

На рисунке 30 отображен результат теста №3.

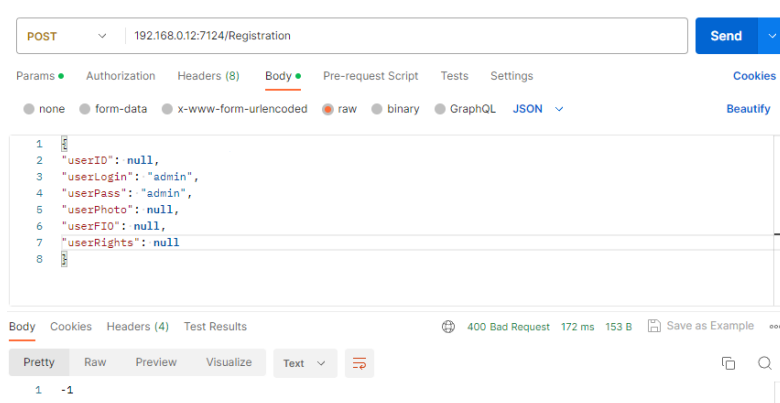


Рисунок 30 – Результат тестирования запроса на регистрацию (тест №3)

На нем видно, что в качестве ответа возвращен ID, равный -1. Такой ответ предусмотрен логикой сервера и означает, что пользователь с заданным логином уже зарегистрирован. Исходя из этого можно сделать вывод, что сервер корректно обрабатывает попытку регистрации с уже занятым логином.

В листинге 2 представлен результат теста №4.

Листинг 2 – Тело ответа, содержащее информацию об выбранном лоте

```
[
  {
    "id": "10",
    "model": "Skoda Rapid 2020",
    "description": "Автомобиль в состоянии нового. Пробег реальный
подтвержден на обслуживании у дилера Дистанционный запуск двигателя! Лазерные
фары Активный круиз контроль Обогрев передних и задних сидений Обогрев руля
Камеры 360гр Музыка Harman/Kardon ",
  }
]
```

Согласно листингу 2, в качестве ответа вернулась коллекция, состоящая из информации о выбранном лоте. На основании ответа можно сделать

вывод, что сервер корректно реализует логику получения информации об выбранном лоте.

На рисунке 31 отображен результат теста №5.

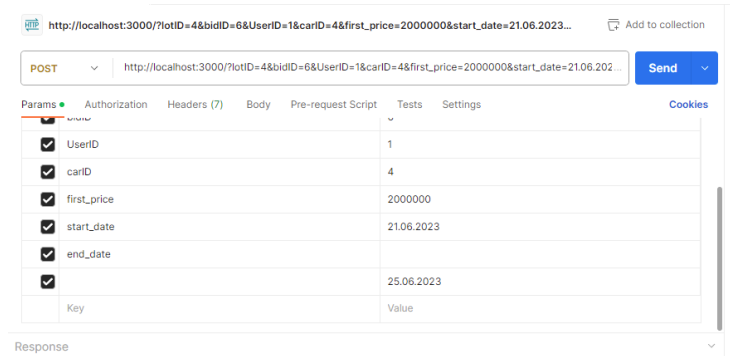


Рисунок 31 – Результат тестирования запроса на добавление лота (тест №5)

Исходя из результата, полученного в ходе выполнения теста №5 и представленного на рисунке 31, можно сделать вывод, что сервер корректно обрабатывает запросы на получение информации о лоте.

На рисунке 32 отображен результат теста №6.

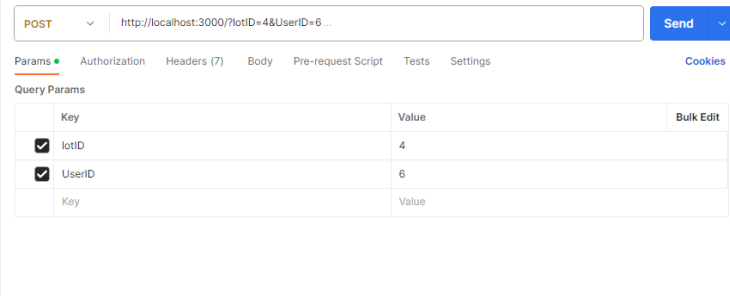


Рисунок 32 – Результат тестирования запроса на добавление лота в избранное (тест №6)

Исходя из результата, полученного в ходе выполнения теста №6 и представленного на рисунке 32, можно сделать вывод, что сервер корректно обрабатывает запросы на получение информации о избранном лоте.

По результатам проведения тестов можно увидеть, что сервер корректно обрабатывает все запросы, и в том числе предусматривает ответы в случае некорректного ввода.

4.3 Тестирование клиентской части web-приложения для интернет-аукциона автомобилей

Для основных функций, представленных на диаграмме вариантов использования, были проведены функциональные тесты. Их перечень и описание представлены в таблице 8.

Таблица 8 – Перечень тестов клиентской части

№ теста	Назначение	Описание	Входные данные	Ожидаемый результат	Актуальный результат
1	Проверка корректности вывода списка лотов	Пользователь переходит на главную страницу	–	Список лотов	Пройден (рисунок Г.1)
2	Проверка корректности вывода информации о лоте	Пользователь выбирает лот из списка	–	Информация о выбранном лоте	Пройден (рисунок Г.2)
3	Проверка корректности вывода списка лотов по фильтрам	Пользователь переходит на главную страницу и выбирает нужный фильтр	–	Список лотов соответствующий фильтру	Пройден (рисунок Г.3)
4	Проверка реакции приложения на попытку сделать ставку на выбранный лот	Пользователь не проходит авторизацию и пробует сделать ставку	–	Сообщение об ошибке, пользователь не может сделать ставку	Пройден (рисунок Г.4)
5	Проверка реакции системы на добавление лота в избранное	Пользователь проходит авторизацию и пробует добавить выбранный лот в список избранных	Логин: admin Пароль: admin	Сообщение об успешном добавлении лота в список избранных, лот добавлен в избранное	Пройден (рисунок Г.5)
6	Проверка реакции приложения на попытку сделать ставку на выбранный лот	Пользователь авторизован и пробует сделать ставку	–	Сообщение об успешном добавлении ставки, пользователь не может сделать ставку	Пройден (рисунок Г.6)
7	Проверка реакции системы создание нового лота	Пользователь авторизован жмет на кнопку создать лот,	Модель: Skoda Rapid 2020, Описание: хорошем состоянии Цена: 150000 Дата конца аукциона: 19.07,23	Новый лот будет добавлен в базу данных	Пройден (рисунок Г.7)

Продолжение таблицы 7

№ теста	Назначение	Описание	Входные данные	Ожидаемый результат	Актуальный результат
8	Проверка реакции системы создание нового лота	Пользователь неавторизован жмет на кнопку создать лот,		Откроется страница авторизации и регистрации	Пройден (рисунок Г.8)

Исходя из результатов, полученных в ходе тестирования, можно сделать вывод, что функциональные требования реализованы в полной мере, в том числе и обработка исключительных ситуаций.

4.4 Планирование разработки и оценка бюджета

Планирование разработки и оценка бюджета была выполнена при помощи программы для управления проектами Microsoft Project [13]. Проект был начат 27 марта 2023 года.

При планировании разработки было выделено 33 задачи. Список задач с общей информацией (длительность, дата начала, дата окончания и др.) представлены на рисунке 24.

№	Название задачи	Длительность	Начало	Окончание	Предшественники	Названия ресурсов
1	Начало проекта	0 дней	Пн 27.02.23	Пн 27.02.23		
2	Анализ требований	14 дней	Пн 27.02.23	Чт 16.03.23		
3	Изучение предметной области	3 дней	Пн 27.02.23	Ср 01.03.23	1	Аналитик;Компьютер[
4	Анализ аналогов	4 дней	Чт 02.03.23	Вт 07.03.23	3	Аналитик;Компьютер[
5	Анализ целевой аудитории	3 дней	Ср 08.03.23	Пт 10.03.23	4	Аналитик;Интернет;Кс
6	Исследование вариантов использования	3 дней	Пн 13.03.23	Ср 15.03.23	5	Аналитик;Компьютер[
7	Анализ требований	1 день	Чт 16.03.23	Чт 16.03.23	6	Аналитик;Компьютер[
8	Анализ требований завершен	0 дней	Чт 16.03.23	Чт 16.03.23	7	
9	Выбор средств	6,67 дней	Пт 17.03.23	Пн 27.03.23	8	
10	Выбор целевой платформы	2 дней	Пт 17.03.23	Вт 21.03.23	8	Программист;Компьют
11	Выбор аппаратного обеспечения	1 день	Вт 21.03.23	Ср 22.03.23	10	Программист;Аналити
12	Выбор программного обеспечения	3 дней	Ср 22.03.23	Пн 27.03.23	11	Программист;Компьют
13	Выбор средств завершен	0 дней	Пн 27.03.23	Пн 27.03.23	12	
14	Проектирование приложения	15 дней	Пн 27.03.23	Пн 17.04.23	13	
15	Проектирование структур данных	2 дней	Пн 27.03.23	Ср 29.03.23		
16	Разработка концептуальной модели	1 день	Пн 27.03.23	Вт 28.03.23	13	Программист;Компьют
17	Разработка логической модели	1 день	Вт 28.03.23	Ср 29.03.23	16	Программист;Компьют
18	Проектирование программных средств	4 дней	Ср 29.03.23	Вт 04.04.23	17	Программист;Компьют
19	Проектирование бизнес-логики приложения	4 дней	Вт 04.04.23	Пн 10.04.23	18	Программист;Компьют
20	Проектирование пользовательского интерфейса	5 дней	Пн 10.04.23	Пн 17.04.23	19	Дизайнер;Компьютер[
21	Проектирование завершено	0 дней	Пн 17.04.23	Пн 17.04.23	20	
22	Реализация приложения	22 дней	Пн 17.04.23	Пт 19.05.23	21	
23	Создание и наполнение базы данных	8 дней	Пн 17.04.23	Чт 27.04.23	21	Программист;Компьют
24	Разработка сервера приложения	7 дней	Чт 27.04.23	Ср 10.05.23	23	Программист;Компьют
25	Разработка приложения-клиента	7 дней	Ср 10.05.23	Пт 19.05.23	24	Программист;Компьют
26	Реализация приложения завершена	0 дней	Пт 19.05.23	Пт 19.05.23	25	
27	Тестирование приложения	7 дней	Пт 19.05.23	Вт 30.05.23	26	
28	Отладка приложения	2 дней	Пт 19.05.23	Вт 23.05.23	26	Программист;Тестиро
29	Выбор методов тестирования	1 день	Вт 23.05.23	Ср 24.05.23	28	Тестировщик;Интерне
30	Функциональное тестирование	2 дней	Ср 24.05.23	Пт 26.05.23	29	Тестировщик;Интерне
31	Модульное тестирование	2 дней	Пт 26.05.23	Вт 30.05.23	30	Тестировщик;Интерне
32	Тестирование завершенно	0 дней	Вт 30.05.23	Вт 30.05.23	31	
33	Конец проекта	0 дней	Вт 30.05.23	Вт 30.05.23	32	

Рисунок 38 – Список задач проектирования

Для того чтобы успешно выполнить проект, необходимо использовать определенные ресурсы. Описание ресурсов представлено на рисунке 25.

	1	Название ресурса	Тип	Единицы измерения материалов	Краткое название	Группа	Макс. единиц	Стандартная ставка	Ставка сверхурочных	Затраты на исполыз.	Начисление	Базовый календарь
1		Программист	Трудовой		П		100%	70 000,00 Р/мес	500,00 Р/ч	0,00 Р	Пропорциональное	Стандартный
2		Аналитик	Трудовой		А		100%	40 000,00 Р/мес	0,00 Р/ч	0,00 Р	Пропорциональное	Стандартный
3		Тестировщик	Трудовой		Т		100%	40 000,00 Р/мес	0,00 Р/ч	0,00 Р	Пропорциональное	Стандартный
4		Дизайнер	Трудовой		Д		100%	55 000,00 Р/мес	0,00 Р/ч	0,00 Р	Пропорциональное	Стандартный
5		Компьютер	Материальный		К			0,00 Р		0,00 Р	Пропорциональное	
6		Интернет	Затраты		И						Пропорциональное	

Рисунок 39 – Список ресурсов проекта

После назначения задач была получена полноценная диаграмма Ганта (рисунок 26). К ней представлены продолжительность задач и взаимосвязи их друг с другом.

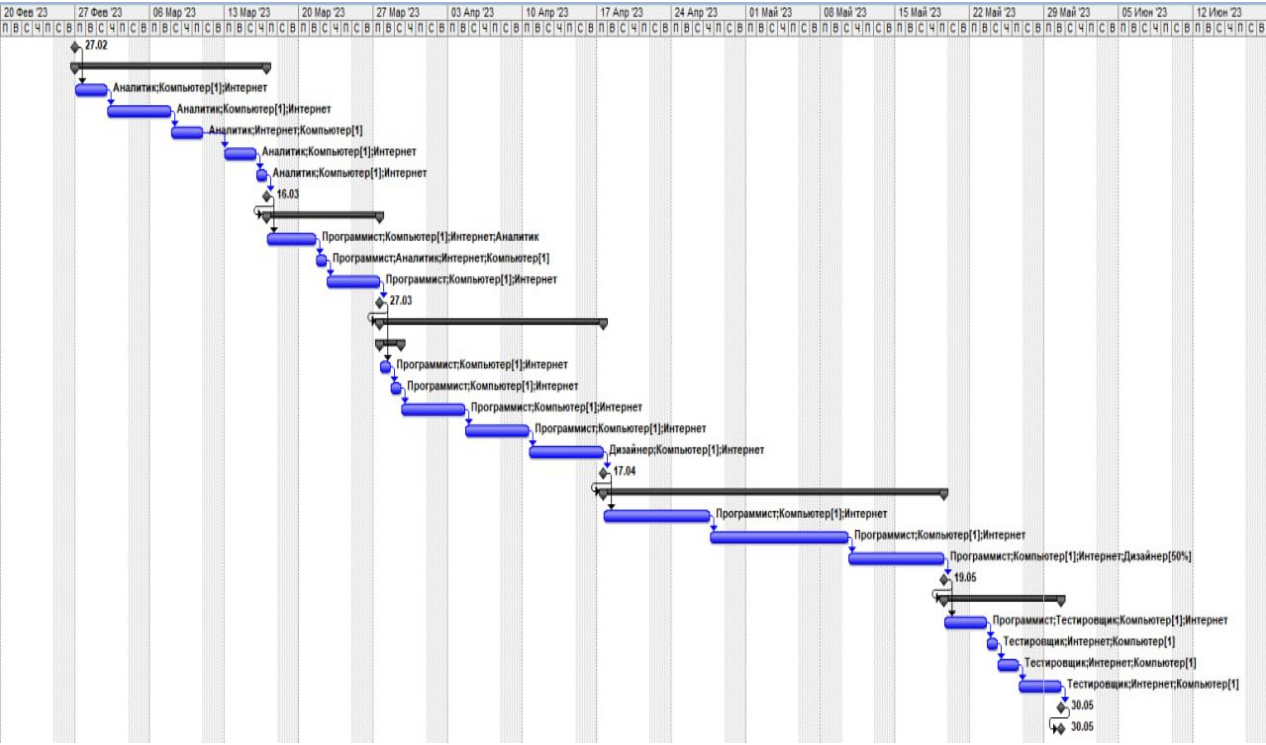


Рисунок 40 – Диаграмма Ганта проекта

После составления проекта, определения дат и используемых ресурсов, был построен отчет, представленный на рисунке 27.

<u>Даты</u>			
Начало:	Пн 27.02.23	Окончание:	Вт 30.05.23
Баз. начало:	НД	Баз. окончание:	НД
Факт. начало:	НД	Факт. окончание:	НД
Отклон. начала:	0 дней	Отклон. окончания:	0 дней
<u>Длительность</u>			
Текущий план:	64,67 дней	Оставшаяся:	64,67 дней
Базовый план:	0 дней?	Фактическая:	0 дней
Отклонение:	64,67 дней	Процент завершения:	0%
<u>Трудозатраты</u>			
Текущий план:	569,33 ч	Оставшаяся:	569,33 ч
Базовый план:	0 ч	Фактические:	0 ч
Отклонение:	569,33 ч	Процент завершения:	0%
<u>Затраты</u>			
Текущий план:	208 708,33 Р	Оставшаяся:	208 708,33 Р
Базовый план:	0,00 Р	Фактические:	0,00 Р
Отклонение:	208 708,33 Р		
<u>Состояние задачи</u>		<u>Состояние ресурса</u>	
Еще не начатые задачи	33	Трудовые ресурсы:	5
Выполняющиеся задачи	0	Трудовые ресурсы с прав. дост	0
Завершённые задачи:	0	Материальные ресурсы:	1
Итого задач:	33	Итого ресурсов:	6

Рисунок 41 – Отчет по проекту

На основании этого отчета и времени выполнения проекта можно сделать вывод о целесообразности разработки.

Заключение

Результатом выполнения выпускной квалификационной работы является разработанное и отлаженное web-приложение для проведения интернет-аукциона автомобилей.

В ходе выполнения этапа анализа требований была обоснована актуальность разрабатываемого приложения. Рассмотрены приложения-аналоги и осуществлен их обзор, выделены преимущества и недостатки. На основании этих критериев проведена оценка. Результатом стало получение интегральных оценок каждого аналога. Также выявлены функциональных требования и проведен их анализ.

В ходе выполнения этапа проектирования приложения определена архитектура приложения. Принято решение использовать клиент-серверное взаимодействие между web-приложением и базой данных. Для обмена данными между ними спроектировано дополнительное приложение, использующее шаблон REST API. Для его реализации, а также реализации приложения-клиента, выбран шаблон MVC. Также спроектированы структуры данных и созданы макеты интерфейса. Проектирование структур данных включило в себя построение концептуальной, логической и физической моделей данных, нацеленных на приложение MySQL.

В ходе выполнения этапа реализации была разработана база данных и сервер приложения, соответствующий архитектуре REST API. Реализация как приложения-сервера, так и web-приложения проходила в среде разработки Webstorm на языке JavaScript с использованием различных компонентов, в том числе фреймворка React. Взаимодействие с базой данных осуществлено с использованием библиотеки Sequelize.

Приложение было протестировано двумя видами тестирования. Серверная часть проверена с помощью модульного тестирования, а клиентская – с помощью функционального тестирования. В результате все тесты были пройдены.

Был составлен план и проведена оценка бюджета. В ходе выполнения данного этапа выявлены временные и финансовые затраты на разработку и необходимые ресурсы. Результатом стала диаграмма Ганта и отчет.

Таким образом, поставленная по реализации web-приложения для проведения интернет-аукциона автомобилей задача выполнена в полной мере.

Список использованных источников

1. Аукционная торговля в России. // CYBERLENINKA [Электронный ресурс] URL: <https://cyberleninka.ru/article/n/auksionnaya-torgovlya-v-rossii>(дата обращения: 22.05.2023).
2. Что такое веб-приложение простыми словами: виды и алгоритм разработки // VC [Электронный ресурс]. – 2020. – 21 сентября. – URL: <https://vc.ru/dev/397220-chto-takoe-veb-prilozhenie-prostymi-slovami-vidy-i-algoritm-razrabotki> (дата обращения: 22.05.2023).
3. Ричи, Г. UML. Руководство по моделированию, 3-е издание / Г. Ричи; А. Слинкин. – М.: Издательский дом «Питер», 2010. – 412 с.
4. Макконнелл, С. Совершенный код, 2-е издание / С. Макконнелл; пер. с англ.: А. Слинкин. – М.: Издательский дом "Русская редакция", 2018. – 887 с.
5. Гамма, Э. Приемы объектно-ориентированного программирования / Э. Гамма; пер. с англ.: А. Слинкин. – М.: Издательский дом "Питер", 2001. – 343 с.
6. Дейт, К. Дж. Введение в системы баз данных, 8-е издание / К. Дж. Дейт; пер. с англ. – М.: Издательский дом «Вильямс», 2005. – 1328 с.
7. Костюкова, А. П. Базы данных: общий курс: учебное пособие / А. П. Костюкова. – Москва: Акад. естествознания, 2018. – Текст: непосредственный. Ч. 1. – 81 с.
8. Зильбершнац, А. Database System Concepts, 6-е издание / А. Зильбершнац, Г. Корт, С. Сударшан. – Издательство «Connect Learn Succeed», 2010. – 1349 с.
9. Эрнандес, М. Database design for Mere Mortals: A Hands-On guide to Relational database design / М. Эрнандес. – М.: Издательский дом «Addison-Wesley», 2013. – 672 с.
10. Буч, Г. Язык UML. Руководство пользователя, 2-е издание / Г. Буч, Дж. Рамбо, И. Якобсон; пер. с англ.: Н. Мухин. – М.: Издательский дом «ДМК Пресс», 2006. – 496 с.

11. Тидвелл, Дж. Дизайн интерфейса: Лучшие практики для взаимодействия с пользователем / Дж. Тидвелл; пер. с англ.: О. Березкина. – М.: Издательский дом «Питер», 2012. – 352 с.
12. Конноли, Т. Введение в системы базы данных, 8-е издание / Т. Конноли. – М.: Издательский дом «Pearson Education Limited», 2008. – 1272 с.
13. Куликов, С. Тестирование программного обеспечения. Базовый курс, 3-е издание / С. Куликов. – М.: Издательство МГТУ им. Н. Э. Баумана, 2020. – 301 с.
14. Функциональное тестирование программного обеспечения // Лаборатория программирования [Электронный ресурс]. URL: <https://daglab.ru/funkcionalnoe-testirovanie-programmnogo-obespechenija/> (дата обращения 25.05.2023).
15. Нефункциональное тестирование // ИксБи Софтваре [Электронный ресурс]. URL: <https://xbsoftware.ru/testirovanie-po/polnij-tsykl/nefunctional-test/> (дата обращения 25.05.2023).
16. Для чего нужно интеграционное тестирование? // Сообщество IT-специалистов [Электронный ресурс]. URL: <https://habr.com/ru/post/556002/> (дата обращения 25.05.2023).
17. Смелов, В. Управление проектами в Microsoft Project: учеб.-метод. пособие для студентов по направлению специальности «Информационные системы и технологии (издательско-полиграфический комплекс)» / В. Смелов. – Минск: БГТУ, 2011. – 160 с.

ПРИЛОЖЕНИЕ А

ГЛОССАРИЙ

(справочное)

API (Application Programming Interface) – набор готовых классов, функций, процедур, констант и структур, предоставляемых приложением или операционной системой для использования во внешних программных продуктах.

REST API – архитектурный стиль взаимодействия компонентов распределённого приложения в сети. REST представляет собой согласованный набор ограничений, учитываемых при проектировании распределённой гипермедиа-системы.

SQL (Structured Query Language) – декларативный язык программирования, применяемый для создания, модификации и управления данными в реляционной базе данных, управляемой соответствующей системой управления базами данных.

HTTP (HyperText Transfer Protocol) – протокол прикладного уровня передачи данных предназначенный для передачи гипертекстовых документов.

UML (Unified Modeling Language) – язык графического описания для объектного моделирования в области разработки программного обеспечения, моделирования бизнес-процессов, системного проектирования и отображения организационных структур.

СУБД – система управления базами данных.

Node.js — программная платформа, основанная на движке V8, превращающая JavaScript из узкоспециализированного языка в язык общего назначения.

MySQL — свободная реляционная система управления базами данных. Разработку и поддержку MySQL осуществляет корпорация Oracle.

WebStorm — интегрированная среда разработки на JavaScript, CSS & HTML от компании JetBrains, разработанная на основе платформы IntelliJ IDEA.

React — JavaScript-библиотека с открытым исходным кодом для разработки пользовательских интерфейсов. React разрабатывается и

поддерживается Facebook, Instagram и сообществом отдельных разработчиков и корпораций.

PK – Primary Key (первичный ключ), в реляционной модели данных один из потенциальных ключей отношения, выбранный в качестве основного ключа (или ключа по умолчанию).

FK – Foreign Key (внешний ключ), понятие теории реляционных баз данных, относящееся к ограничениям целостности базы данных.

JSON – JavaScript Object Notation, текстовый формат, полностью независимый от языка реализации.

ПРИЛОЖЕНИЕ Б
ЛИСТИНГ ПРОГРАММЫ
(рекомендуемое)

Серверная часть.

Листинг Б.1 – Файл User.js

```
// User Model
module.exports = (sequelize, DataTypes) => {
  const User = sequelize.define("User", {
    username: {
      type: DataTypes.STRING(255),
      unique: true
    },
    password: {
      type: DataTypes.STRING,
    },
    name: {
      type: DataTypes.STRING,
    },
    email: {
      type: DataTypes.STRING,
    },
  });
  // Associated through foreign keys automatically
  User.associate = (models) => {
    User.hasMany(models.Car, {
      onDelete: "cascade",
    });
    User.hasMany(models.Bid, {
      onDelete: "cascade",
    });
    User.hasMany(models.Favorites, {
      onDelete: "cascade",
    });
    User.hasMany(models.Lot, {
      onDelete: "cascade",
    });
  };
};
return User;
};
```

Листинг Б.2 – Файл Car.js

```
// Item Model
module.exports = (sequelize, DataTypes) => {
  const Car = sequelize.define("Item", {
    model: {
      type: DataTypes.STRING,
      allowNull: false,
    },
    description: {
      type: DataTypes.STRING,
      allowNull: false,
    },
    coverPhoto: {
      type: DataTypes.STRING(255),
      defaultValue: "https://localhost:33123/images/placeholder.png",
    }
  });
  // Associated through foreign keys automatically
  Item.associate = (models) => {
    Item.hasMany(models.Bid, {
      onDelete: 'cascade',
    });
  };
};
```

```

        Item.hasMany(models.Photo, {
            onDelete: 'cascade',
        });
        Item.hasMany(models.UserData, {
        });
    };
    return Car;
};
};

```

Листинг Б.3 – Файл index.js

```

'use strict';
const fs = require('fs');
const path = require('path');
const Sequelize = require('sequelize');
const basename = path.basename(__filename);
const env = process.env.NODE_ENV || 'development';
const config = require(__dirname + '/../config/config.json')[env];
const db = {};
let sequelize;
if (config.use_env_variable) {
    sequelize = new Sequelize(process.env[config.use_env_variable], config);
} else {
    sequelize = new Sequelize(config.database, config.username,
config.password, config);
}
fs
    .readdirSync(__dirname)
    .filter(file => {
        return (file.indexOf('.') !== 0) && (file !== basename) && (file.slice(-
3) === '.js');
    })
    .forEach(file => {
        const model = require(path.join(__dirname, file))(sequelize,
Sequelize.DataTypes);
        db[model.name] = model;
    });
Object.keys(db).forEach(modelName => {
    if (db[modelName].associate) {
        db[modelName].associate(db);
    }
});
db.sequelize = sequelize;
db.Sequelize = Sequelize;
// db.sequelize.sync({ force: false, alter:true});
module.exports = db;

```

Листинг Б.4 – Файл Bid.js

```

// Bid Model
module.exports = (sequelize, DataTypes) => {
    const Bid = sequelize.define("Bid", {
        amount: {
            type: DataTypes.integer(15,2),
        },
        userID: {
            type: DataTypes.integer(15,2),
        },
    });
    return Bid;
};

```

Листинг Б.5 – Файл Favorites.cs

```
// Favorites Model
module.exports = (sequelize, DataTypes) => {
  const Item = sequelize.define("Item", {
    userID: {
      type: DataTypes.integer(15,2),
    },
    lotID: {
      type: DataTypes.integer(15,2),
    },
  });

  // Associated through foreign keys automatically
  Favorites.associate = (models) => {
    Item.hasMany(models.Bid, {
      onDelete: 'cascade',
    });
    Item.hasMany(models.Photo, {
      onDelete: 'cascade',
    });
    Item.hasMany(models.UserData, {
    });
  };

  return Favorites;
};
```

}; Листинг Б.6 – Файл Lot.js

```
// Lot Model
module.exports = (sequelize, DataTypes) => {
  const Lot = sequelize.define("Item", {
    userID: {
      type: DataTypes.integer(15,2),
    },
    carID: {
      type: DataTypes.integer(15,2),
    },
    bidID: {
      type: DataTypes.integer(15,2),
    },
    buy_price: {
      type: DataTypes.DOUBLE(15,2),
      allowNull: true,
    },
    first_bid: {
      type: DataTypes.DOUBLE(15,2),
    },
    started: {
      type: DataTypes.DATE,
      defaultValue: DataTypes.NOW,
      allowNull: false,
    },
  },
```

```

        ends: {
            type: DataTypes.DATE,
            allowNull: false,
        },
    });
    // Associated through foreign keys automatically
    Lot.associate = (models) => {
        Lot.hasMany(models.Bid, {
            onDelete: 'cascade',
        });
        Lot.hasMany(models.Photo, {
            onDelete: 'cascade',
        });
        Lot.hasMany(models.UserData, {
        });
    };
    return Lot;
};

```

Листинг Б.7 – Файл ControllerBids.cs

```

const express = require('express');
const router = express.Router();
const { Bid, Item , User} = require('../models');
const {validateTheToken} = require('../middlewares/Authent');
router.get('/:itemId', async (req, res) => {
    const itemId = req.params.itemId;
    const bids = await Bid.findAll({where : { ItemId: itemId}});
    res.json(bids);
});
router.post('/', validateTheToken, async (req, res) => {
    let bid = req.body;
    const bidder = await User.findPk(bid.UserId);
    bid.bidderRating = bidder.bidderRating;
    bid.bidderName = bidder.username;
    await Bid.create(bid);
    // and then update the item currently and highest bidder
    await Item.update({
        currently: bid.amount,
        highestBidder: bid.UserId,
    },
    {where : {
        id: bid.ItemId,
    }
    });
    await Item.increment({number_of_bids: 1}, { where: { id: bid.ItemId } });
});
module.exports = router;

```

Листинг Б.8 – Файл ControllerLot.cs

```

const express = require('express');
const router = express.Router();
const { Item, Bid, Mail , Message, User, Contact, Category, UserData, UserTop
} = require('../models');
const {validateTheToken} = require('../middlewares/Authent');
const { Op } = require("sequelize");
const fs = require('fs');
const js2xml = require('js2xml').Js2Xml;
const xml2js = require('xml2js');
router.get('/', async (req, res) => {
    // A bit of house keeping. Updating the ones that have been expired

```

```

const allItems = await Item.findAll({
  where: {
    state: "AVAILABLE",
    ends: {
      [Op.lt]: new Date()
    }
  },
});
for (var i = 0; i < allItems.length; i++) {
  if( allItems[i].number_of_bids>0){
    await Item.update({
      state: "PURCHASED",
    },
    {where : {
      id: allItems[i].id,
    }
    });
    // Create the mail subsection for them
    const newmail = await Mail.create({
      itemId: allItems[i].id,
      itemName: allItems[i].name,
      bidderId: allItems[i].highestBidder,
      sellerId: allItems[i].UserId,
    }).catch(err => {
      console.log(err);
    });
    const userito = await User.findByPk( allItems[i].UserId);
    const biderito = await User.findByPk(allItems[i].highestBidder);
    // Send confirmation message to buyer
    await Message.create({
      title: "Confirmation",
      body: "Congratulations you won the auction! Feel free to
contact me for further information on shipment and payment!",
      senderId: userito.id,
      recipientId: biderito.id,
      senderName: userito.name,
      senderSurname: userito.surname,
      recipientName: biderito.name,
      recipientSurname: biderito.surname,
      MailId: newmail.id,
      UserId: userito.id,
    }).catch(err => {
      console.log(err);
    });
    // looking for one is enough since its symmetric
    const contactalready = await Contact.findOne({ where: {contactId:
userito.id, UserId: biderito.id}});

    // if they are not already on each others contacts add them
    if(contactalready ===null){
      await Contact.create({
        contactId: userito.id,
        contactName: userito.name,
        contactSurname: userito.surname,
        contactUsername: userito.username,
        UserId: biderito.id,
      }).catch(err => {
        console.log(err);
      });
      await Contact.create({
        contactId: biderito.id,
        contactName: biderito.name,
        contactSurname: biderito.surname,
        contactUsername: biderito.username,

```

```

        UserId: userito.id,
      }).catch(err => {
        console.log(err);
      });
    }
  }
  else{
    await Item.update({
      state: "EXPIRED",
    },
    {where : {
      id: allItems[i].id,
    }
    });
  }
}
// House keeping updating on the ones that they're going to be available
await Item.update({
  state: "AVAILABLE",
},
{where : {
  state: "EXPECTED",
  started: {
    [Op.lte]: new Date()
  }
}
});
// Finally return the actually available items
const availableItems = await Item.findAll({ where: { state: "AVAILABLE"},
});
res.json(availableItems);
});
router.get('/admin/', validateTheToken, async (req, res) => {
  const username = req.header("username");
  if (username=="admin"){

    // A bit of house keeping. Updating the ones that have been expired
    const allItems = await Item.findAll({
      where: {
        state: "AVAILABLE",
        ends: {
          [Op.lt]: new Date()
        }
      },
    });
    for (var i = 0; i < allItems.length; i++) {
      if( allItems[i].number_of_bids>0){
        await Item.update({
          state: "PURCHASED",
        },
        {where : {
          id: allItems[i].id,
        }
        });
        // Create the mail subsection for them
        const newmail = await Mail.create({
          itemId: allItems[i].id,
          itemName: allItems[i].name,
          bidderId: allItems[i].highestBidder,
          sellerId: allItems[i].UserId,
        }).catch(err => {
          console.log(err);
        });
        const userito = await User.findPk( allItems[i].UserId);

```

```

        const biderito = await
User.findByPk(allItems[i].highestBidder);
        // Send confirmation message to buyer
        await Message.create({
            title: "Confirmation",
            body: "Congratulations you won the auction! Feel free to
contact me for further information on shipment and payment!",
            senderId: userito.id,
            recipientId: biderito.id,
            senderName: userito.name,
            senderSurname: userito.surname,
            recipientName: biderito.name,
            recipientSurname: biderito.surname,
            MailId: newmail.id,
            UserId: userito.id,
        }).catch(err => {
            console.log(err);
        });
        // looking for one is enough since its symmetric
        const contactalready = await Contact.findOne({ where:
{contactId: userito.id, UserId: biderito.id}});

        // if they are not already on each others contacts add them
        if(contactalready ===null){
            await Contact.create({
                contactId: userito.id,
                contactName: userito.name,
                contactSurname: userito.surname,
                contactUsername: userito.username,
                UserId: biderito.id,
            }).catch(err => {
                console.log(err);
            });
            await Contact.create({
                contactId: biderito.id,
                contactName: biderito.name,
                contactSurname: biderito.surname,
                contactUsername: biderito.username,
                UserId: userito.id,
            }).catch(err => {
                console.log(err);
            });
        }
    }
    else{
        await Item.update({
            state: "EXPIRED",
        },
        {where : {
            id: allItems[i].id,
        }
        });
    }
}
// House keeping updating on the ones that they're going to be
available
await Item.update({
    state: "AVAILABLE",
},
{where : {
    state: "EXPECTED",
    started: {
        [Op.lte]: new Date()
    }
}
}

```

```

    }
  });

  // get the refreshed list
  const availableItems = await Item.findAll();
  res.json(availableItems);
}
else{
  res.json({message: "forbidden"});
}

});
// recursively find all the subcategories
function myChildren(categoriesList, superCategoryId){
  var categs = [];
  categs.push(superCategoryId);
  for (var i=0; i<categoriesList.length;i++){
    if (categoriesList[i].CategoryId==superCategoryId){
      categs.push.apply(categs, myChildren(categoriesList,
categoriesList[i].id))
    }
  }
  return categs;
}
router.get('/categories/:id', async (req, res) => {
  // A bit of house keeping. Updating the ones that have been expired
  const allItems = await Item.findAll({
    where: {
      state: "AVAILABLE",
      ends: {
        [Op.lt]: new Date()
      }
    },
  });
  for (var i = 0; i < allItems.length; i++) {
    if( allItems[i].number_of_bids>0){
      await Item.update({
        state: "PURCHASED",
      },
      {where : {
        id: allItems[i].id,
      }
    });
      // Create the mail subsection for them
      const newmail = await Mail.create({
        itemId: allItems[i].id,
        itemName: allItems[i].name,
        bidderId: allItems[i].highestBidder,
        sellerId: allItems[i].UserId,
      }).catch(err => {
        console.log(err);
      });
      const userito = await User.findByPk( allItems[i].UserId);
      const biderito = await User.findByPk(allItems[i].highestBidder);
      // Send confirmation message to buyer
      await Message.create({
        title: "Confirmation",
        body: "Congratulations you won the auction! Feel free to
contact me for further information on shipment and payment!",
        senderId: userito.id,
        recipientId: biderito.id,
        senderName: userito.name,
        senderSurname: userito.surname,
        recipientName: biderito.name,

```

```

        recipientSurname: biderito.surname,
        MailId: newmail.id,
        UserId: userito.id,
    }).catch(err => {
        console.log(err);
    });
    // looking for one is enough since its symmetric
    const contactalready = await Contact.findOne({ where: {contactId:
userito.id, UserId: biderito.id}});

    // if they are not already on each others contacts add them
    if(contactalready ===null){
        await Contact.create({
            contactId: userito.id,
            contactName: userito.name,
            contactSurname: userito.surname,
            contactUsername: userito.username,
            UserId: biderito.id,
        }).catch(err => {
            console.log(err);
        });
        await Contact.create({
            contactId: biderito.id,
            contactName: biderito.name,
            contactSurname: biderito.surname,
            contactUsername: biderito.username,
            UserId: userito.id,
        }).catch(err => {
            console.log(err);
        });
    }
}
else{
    await Item.update({
        state: "EXPIRED",
    },
    {where : {
        id: allItems[i].id,
    }
    });
}
}
// House keeping updating on the ones that they're going to be available
await Item.update({
    state: "AVAILABLE",
},
{where : {
    state: "EXPECTED",
    started: {
        [Op.lte]: new Date()
    }
}
});
// now find the items of this category and below
const allCategories = await Category.findAll();
const superCategoryId = req.params.id;
var categs = []
categs.push.apply(categs, myChildren(allCategories, superCategoryId))
const availableItems = [];
for (var i=0; i<categs.length;i++){
    const freshItems = await Item.findAll({ where:
    { furthestmostCategoryId: categs[i],
    state: "AVAILABLE"},
});

```

```

        availableItems.push.apply(availableItems, freshItems);
    }
    res.json(availableItems);
});
router.get('/fetchy/:id', async (req, res) => {
    const myId = req.params.id;

    if (req.body && req.body.userId){
        const userId = req.body.userId;
        var UserData = await User.findOne( { where: { UserId: userId, ItemId:
myId } } );
        if (UserData){
            // if he hasn't clicked on that before
            if ( UserData.rating<2){
                await UserData.update({
                    rating: 2,
                },
                {where : {
                    id: UserData.id,
                }
            });
        }
        else{
            await UserData.create({
                rating: 2,
                UserId: userId,
                ItemId: myId
            });
        }

        const itemito = await Item.findByPk(myId);
        res.json(itemito);
    });
    router.post('/fetchy/:id', async (req, res) => {
        const myId = req.params.id;

        const itemito = await Item.findByPk(myId);
        res.json(itemito);
    });
    router.get('/fetchByUser/:id', async (req, res) => {
        const myId = req.params.id;
        const alltheUserItems = await Item.findAll({
            where: { UserId: myId},
            include: [Bid],
        });
        res.json(alltheUserItems);
    });
    router.get('/mywatchlist/:id', async (req, res) => {
        const myId = req.params.id;
        const allMyBids = await Bid.findAll({
            where: { UserId: myId},
        });
        // Find unique item ids
        var ids = []
        for (var i=0;i<allMyBids.length;i++){
            ids.push(allMyBids[i].ItemId);
        }
        var uniqueIds= [... new Set(ids)]
        var watchlist = []
        for (var i=0;i<uniqueIds.length;i++){
            const itemito = await Item.findByPk(uniqueIds[i]);

```

```

        if (itemito.state=='AVAILABLE'){
            watchlist.push(itemito);
        }
    }
    res.json(watchlist);
});
router.post('/', validateTheToken, async (req, res) => {
    const item = req.body;
    const userId = req.user.id;
    item.UserId = userId;
    await Item.create(item);
    res.json(item);
});
router.put('/purchase/:id', validateTheToken, async (req, res)=>{
    const myId = req.params.id;
    const bidderId = req.body.userId;
    const sellerId = req.body.sellerId;
    const finalAmount = req.body.purchaseAmount;

    await Item.update({
        state: "PURCHASED",
        highestBidder: bidderId,
        currently: finalAmount,
    },
    {where : {
        id: myId,
    }
    });
    await Item.increment({number_of_bids: 1}, { where: { id: myId } });
    const itemito = await Item.findByPk(myId);
    // Finally create the mail subsection for them
    const newmail = await Mail.create({
        itemId: myId,
        itemName: itemito.name,
        bidderId: bidderId,
        sellerId: sellerId,
    }).catch(err => {
        console.log(err);
    });
    const userito = await User.findByPk(sellerId);
    const biderito = await User.findByPk(bidderId);
    // Send confirmation message to buyer
    await Message.create({
        title: "Confirmation",
        body: "Congratulations you won the auction! Feel free to contact me
for further information on shipment and payment!",
        senderId: sellerId,
        recipientId: bidderId,
        senderName: userito.name,
        senderSurname: userito.surname,
        recipientName: biderito.name,
        recipientSurname: biderito.surname,
        MailId: newmail.id,
        UserId: sellerId,
    }).catch(err => {
        console.log(err);
    });
    // looking for one is enough since its symmetric
    const contactalready = await Contact.findOne({ where: {contactId:
sellerId, UserId: bidderId}});

    // if they are not already on each others contacts add them
    if(contactalready ===null){

```

```

        await Contact.create({
            contactId: userito.id,
            contactName: userito.name,
            contactSurname: userito.surname,
            contactUsername: userito.username,
            UserId: biderito.id,
        }).catch(err => {
            console.log(err);
        });
        await Contact.create({
            contactId: biderito.id,
            contactName: biderito.name,
            contactSurname: biderito.surname,
            contactUsername: biderito.username,
            UserId: userito.id,
        }).catch(err => {
            console.log(err);
        });
    }
});
// Update the auction Name
router.put('/name/:id', validateTheToken, async (req, res)=>{
    const myId = req.params.id;
    const item = req.body;

    // further check for the user even though i checked in the frontend
    const userId = req.header("userId");

    await Item.update({
        name: item.name,
    },
    {where : {
        id: myId,
        UserId: userId
    }
    });
});
// Update the auction Starting date
router.put('/started/:id', validateTheToken, async (req, res)=>{
    const myId = req.params.id;
    const item = req.body;

    // further check for the user that is not someone else
    const userId = req.header("userId");

    await Item.update({
        started: item.started,
    },
    {where : {
        id: myId,
        UserId: userId
    }
    });
});
// Update the auction Ending date
router.put('/ends/:id', validateTheToken, async (req, res)=>{
    const myId = req.params.id;
    const item = req.body;

    // further check for the user that is not someone else
    const userId = req.header("userId");

    await Item.update({
        ends: item.ends,
    },
    {where : {
        id: myId,
        UserId: userId
    }
    });
});

```

```

    },
    {where : {
      id: myId,
      UserId: userId
    }
  });
});
// Update the buyprice
router.put('/buyprice/:id', validateTheToken, async (req, res)=>{
  const myId = req.params.id;
  const item = req.body;

  // further check for the user that is not someone else
  const userId = req.header("userId");

  await Item.update({
    buy_price: item.buy_price,
  },
  {where : {
    id: myId,
    UserId: userId
  }
  });
});
// Update the currently and first bid values
router.put('/currently/:id', validateTheToken, async (req, res)=>{
  const myId = req.params.id;
  const item = req.body;

  // further check for the user that is not someone else
  const userId = req.header("userId");

  await Item.update({
    currently: item.currently,
    first_bid: item.currently,
  },
  {where : {
    id: myId,
    UserId: userId
  }
  });
});
// Update the location
router.put('/location/:id', validateTheToken, async (req, res)=>{
  const myId = req.params.id;
  const item = req.body;

  // further check for the user that is not someone else
  const userId = req.header("userId");

  await Item.update({
    location: item.location,
  },
  {where : {
    id: myId,
    UserId: userId
  }
  });
});
// Update the country
router.put('/country/:id', validateTheToken, async (req, res)=>{
  const myId = req.params.id;
  const item = req.body;

```

```

// further check for the user that is not someone else
const userId = req.header("userId");

await Item.update({
  country: item.country,
},
{where : {
  id: myId,
  UserId: userId
}
});
});
// Update the description
router.put('/description/:id', validateTheToken, async (req, res)=>{
  const myId = req.params.id;
  const item = req.body;

  // further check for the user that is not someone else
  const userId = req.header("userId");

  await Item.update({
    description: item.description,
  },
  {where : {
    id: myId,
    UserId: userId
  }
  });
});
// Update the coordinates
router.put('/coordinates/:id', validateTheToken, async (req, res)=>{
  const myId = req.params.id;
  const item = req.body;

  // further check for the user that is not someone else
  const userId = req.header("userId");

  await Item.update({
    latitudeLongitude: item.latitudeLongitude,
  },
  {where : {
    id: myId,
    UserId: userId
  }
  });
});
// Update the category
router.put('/category/:id', validateTheToken, async (req, res)=>{
  const myId = req.params.id;
  const item = req.body;

  // further check for the user that is not someone else
  const userId = req.header("userId");

  await Item.update({
    furthestmostCategoryId: item.furthestmostCategoryId,
  },
  {where : {
    id: myId,
    UserId: userId
  }
  });
});
// validate the user too before deleting

```

```

router.delete('/:id', validateTheToken, async (req, res)=>{
  const myId = req.params.id;

  // further check for the user even though i checked in the frontend
  const userId = req.header("userId");

  await Item.destroy({where : {
    id: myId,
    UserId: userId
  }});
});
// Recursively find the hierarchy tree, ie all the parents till the root
function myParents(categoriesList, childId){
  var categs = [];
  for (var i=0; i<categoriesList.length;i++){
    if (categoriesList[i].id==childId){

      const childCategory = categoriesList[i];
      categs.push(childCategory.name);
      for (var j=0; j<categoriesList.length;j++){
        if (categoriesList[j].id==childCategory.CategoryId){
          categs.push.apply(categs, myParents(categoriesList,
categoriesList[j].id))
        }
      }
    }
  }
  return categs;
}
router.get('/downloadJSON', validateTheToken, async (req, res) => {
  const username = req.header("username");
  if (username=="admin"){
    const allItems = await Item.findAll();
    const allCategories = await Category.findAll();
    var itemjsons = []
    for (var i=0;i<allItems.length;i++){
      // Find the item's categories
      var categories = [];
      categories.push.apply(categories, myParents(allCategories,
allItems[i].furthestmostCategoryId))

      // Reverse so its from the general to the specific
      var categoriesInOrder = categories.reverse();

      // Find the seller information
      const seller = await User.findByPk(allItems[i].UserId);
      // Find the bids on the item
      const allBids = await Bid.findAll({
        where: { ItemId: allItems[i].id},
      });
      // Fix the bids format
      var bidsInOrder =[]
      for (var j=0;j<allBids.length;j++){
        const bidder = await User.findByPk(allBids[j].UserId);

        const bid = {
          Bidder: {
            Rating: bidder.bidderRating,
            UserId: bidder.username,
            Location: bidder.location,
            Country: bidder.country,
          },
          Time: allBids[j].time.toLocaleString(),
          Amount: allBids[j].amount,

```

```

        };
        bidsInOrder.push(bid);

    }
    var location = {Location: allItems[i].location}
    if (allItems[i].latitudeLongitude){
        location.Latitude =
allItems[i].latitudeLongitude.coordinates[0];
        location.Longitude =
allItems[i].latitudeLongitude.coordinates[1];
    }
    var item = {}
    if (allItems[i].buy_price){
        item = {
            ItemID: allItems[i].id ,
            Name: allItems[i].name,
            Category: categoriesInOrder,
            Currently: allItems[i].currently,
            Buy_Price: allItems[i].buy_price,
            First_Bid: allItems[i].first_bid,
            Number_Of_Bids: allItems[i].number_of_bids,
            Bids: bidsInOrder,
            Location: location,
            Country: allItems[i].country,
            Started: allItems[i].started.toLocaleString(),
            Ends: allItems[i].ends.toLocaleString(),
            Seller: { Rating: seller.sellerRating, UserID:
seller.username },
            Description: allItems[i].description
        };
    }
    else {
        item = {
            ItemID: allItems[i].id ,
            Name: allItems[i].name,
            Category: categoriesInOrder,
            Currently: allItems[i].currently,
            First_Bid: allItems[i].first_bid,
            Number_Of_Bids: allItems[i].number_of_bids,
            Bids: bidsInOrder,
            Location: location,
            Country: allItems[i].country,
            Started: allItems[i].started.toLocaleString(),
            Ends: allItems[i].ends.toLocaleString(),
            Seller: { Rating: seller.sellerRating, UserID:
seller.username },
            Description: allItems[i].description
        };
    }
    itemjsons.push(item);
}
res.json(itemjsons);
}
else{
    res.json({message: "forbidden"});
}
});
router.get('/downloadXML', validateTheToken, async (req, res) => {
    const username = req.header("username");
    if (username=="admin"){
        const allItems = await Item.findAll();
        const allCategories = await Category.findAll();
        var itemsXMLformat = "<Items>\n"
        for (var i=0;i<allItems.length;i++){

```

```

// Find the item's categories
var categories = [];
categories.push.apply(categories, myParents(allCategories,
allItems[i].furthestmostCategoryId))

// Reverse so its from the general to the specific
var categoriesInOrder = categories.reverse();

// Find the seller information
const seller = await User.findByPk(allItems[i].UserId);

itemsXMLformat += '\t<Item ItemID="'
itemsXMLformat += allItems[i].id.toString()
itemsXMLformat += '">\n'
itemsXMLformat += "\t\t<Name>"
itemsXMLformat += allItems[i].name
itemsXMLformat += '</Name>\n'
for (var jj=0;jj<categoriesInOrder.length;jj++){
    itemsXMLformat += "\t\t<Category>"
    itemsXMLformat += categoriesInOrder[jj]
    itemsXMLformat += "</Category>\n"
}
itemsXMLformat += "\t\t<Currently>$"
itemsXMLformat += allItems[i].currently.toString()
itemsXMLformat += '</Currently>\n'
if (allItems[i].buy_price!=null){
    itemsXMLformat += "\t\t<Buy_Price>$"
    itemsXMLformat += allItems[i].buy_price.toString()
    itemsXMLformat += '</Buy_Price>\n'
}
itemsXMLformat += "\t\t<First_Bid>$"
itemsXMLformat += allItems[i].first_bid.toString()
itemsXMLformat += '</First_Bid>\n'
itemsXMLformat += "\t\t<Number_of_Bids>"
itemsXMLformat += allItems[i].number_of_bids.toString()
itemsXMLformat += '</Number_of_Bids>\n'
// Find the bids on the item
const allBids = await Bid.findAll({
    where: { ItemId: allItems[i].id},
});
if (allBids.length==0){
    itemsXMLformat += '\t\t<Bids />\n'
}
else{
    itemsXMLformat += '\t\t<Bids>\n'
    for (var jj=0;jj<allBids.length;jj++){
        const bidder = await User.findByPk(allBids[jj].UserId);
        itemsXMLformat += "\t\t\t<Bid>\n"
        itemsXMLformat += '\t\t\t\t<Bidder Rating="'
        itemsXMLformat += bidder.bidderRating.toString()
        itemsXMLformat += '" UserID="'
        itemsXMLformat += bidder.username
        itemsXMLformat += '">\n'
        itemsXMLformat += '\t\t\t\t\t<Location>'
        itemsXMLformat += bidder.location
        itemsXMLformat += '</Location>\n'
        itemsXMLformat += '\t\t\t\t\t<Country>'
        itemsXMLformat += bidder.country
        itemsXMLformat += '</Country>\n'
        itemsXMLformat += '\t\t\t\t\t</Bidder>\n'
        itemsXMLformat += '\t\t\t\t\t<Time>'
        itemsXMLformat += allBids[jj].time.toLocaleString()
        itemsXMLformat += '</Time>\n'
        itemsXMLformat += '\t\t\t\t\t<Amount>$'
    }
}

```

```

        itemsXMLformat += allBids[jj].amount.toString()
        itemsXMLformat += '</Amount>\n'
        itemsXMLformat += "\t\t\t</Bid>\n"
    }
    itemsXMLformat += '\t\t</Bids>\n'
}
if (allItems[i].latitudeLongitude) {
    itemsXMLformat += '\t\t<Location Latitude="'
    itemsXMLformat +=
allItems[i].latitudeLongitude.coordinates[0].toString()
    itemsXMLformat += '" Longitude="'
    itemsXMLformat +=
allItems[i].latitudeLongitude.coordinates[1].toString()
    itemsXMLformat += '">'
}
else{
    itemsXMLformat += "\t\t<Location>"
}
itemsXMLformat += allItems[i].location
itemsXMLformat += '</Location>\n'
itemsXMLformat += "\t\t<Country>"
itemsXMLformat += allItems[i].country
itemsXMLformat += '</Country>\n'
itemsXMLformat += "\t\t<Started>"
itemsXMLformat += allItems[i].started.toLocaleString()
itemsXMLformat += '</Started>\n'
itemsXMLformat += "\t\t<Ends>"
itemsXMLformat += allItems[i].ends.toLocaleString()
itemsXMLformat += '</Ends>\n'
itemsXMLformat += '\t\t<Seller Rating="'
itemsXMLformat += seller.sellerRating.toString()
itemsXMLformat += '" UserID="'
itemsXMLformat += seller.username
itemsXMLformat += '" />\n'
itemsXMLformat += "\t\t<Description>"
itemsXMLformat += allItems[i].description
itemsXMLformat += '</Description>\n'
itemsXMLformat += "\t</Item>\n"
}
itemsXMLformat += "</Items>"
res.json(itemsXMLformat);
}
else{
    res.json({message: "forbidden"});
}
});
router.get('/top/:id', async (req, res) => {
    const myId = req.params.id;
    const recommendations = await UserTop.findOne({
        where: { UserId: myId, },
    });
    var items = []
    if (recommendations!=null){
        const item1 = await Item.findByPk(recommendations.p1);
        // check also if it's still available (since during the 24 hours it
might have been purchased)
        if (item1!=null && item1.state=="AVAILABLE" && item1.UserId!= myId){
            items.push(item1)
        }
        const item2 = await Item.findByPk(recommendations.p2);
        if (item2!=null && item2.state=="AVAILABLE" && item2.UserId!= myId){
            items.push(item2)
        }
        const item3 = await Item.findByPk(recommendations.p3);

```

```

    if (item3!=null && item3.state=="AVAILABLE" && item3.UserId!= myId){
        items.push(item3)
    }
    const item4 = await Item.findByPk(recommendations.p4);
    if (item4!=null && item4.state=="AVAILABLE" && item4.UserId!= myId){
        items.push(item4)
    }
    const item5 = await Item.findByPk(recommendations.p5);
    if (item5!=null && item5.state=="AVAILABLE" && item5.UserId!= myId){
        items.push(item5)
    }
    const item6 = await Item.findByPk(recommendations.p6);
    if (item6!=null && item6.state=="AVAILABLE" && item6.UserId!= myId){
        items.push(item6)
    }
}
res.json(items);
});
// this to import the xmls in the database
// comment out so it doesn't run when the site is "online"
router.post('/importxmls', async (req, res) => {
    const parser = new xml2js.Parser();
    var path = require("path");
    // to do it in a bunch
    // for (var i=0;i<30;i++){
    //     await fs.readFile(path.join(__dirname, '..', 'ebayXMLs', `items-
    ${i}.xml`), async (err, data)=>{
    //         await parser.parseString(data, async (err, result)=>{
    //             one by one
    await fs.readFile(path.join(__dirname, '..', 'ebayXMLs', 'items-23.xml'),
    async (err, data)=>{
        await parser.parseString(data, async (err, result)=>{
            if (err){
                throw err;
            }
            for ( var p = 0 ; p < result.Items.Item.length ; p++ ){
                // console.log(result.Items.Item[p].Number_of_Bids[0]);
                const itemito = result.Items.Item[p];
                // Add this and those with zero either don't add them or add
                them for sale
                // since the offer no value as data for the matrix
                factorisation
                if( parseInt(itemito.Number_of_Bids[0]) == 0 ){
                    const myCategories = itemito.Category;
                    var lastcategory = 171;

                    // For each category in the tree
                    for (var i=0; i< myCategories.length; i++){

                        var cat = await Category.findOne( { where: { name:
myCategories[i] } } );

                        if (cat==null){
                            const cat2 = await Category.create({
                                name: myCategories[i],
                                CategoryId: lastcategory,
                            }).catch(err => {
                                console.log(err);
                            });
                        }

                        // As the furthestmost category in the tree
                        lastcategory = cat2.id;
                    }
                }
            }
        }
    }
}

```

```

        else {
            lastcategory = cat.id
        }
    }

    const activeUsers = await User.findAll({ where: {
approved: true, admin: false},
    });

    const pickUserId = activeUsers[Math.floor(Math.random() *
activeUsers.length)].id;

    var item = {

        name: itemito.Name[0],
        currently:
parseFloat(itemito.Currently[0].substring(1)),
        first_bid:
parseFloat(itemito.First_Bid[0].substring(1)),
        number_of_bids: parseInt(itemito.Number_of_Bids[0]),
        country: itemito.Country[0],
        started: new Date(),
        ends: 'Dec-13-23 18:10:40',
        description: itemito.Description[0],
        state: 'AVAILABLE',
        furthestmostCategoryId: lastcategory,
        UserId: pickUserId,
    }

    if (itemito.Buy_Price){
        item.buy_price=
parseFloat(itemito.Buy_Price[0].substring(1))
    }
    if (itemito.Location[0]._) {
        item.location=itemito.Location[0]._
        var point = { type: 'Point', coordinates:
[itemito.Location[0].$.Latitude,itemito.Location[0].$.Longitude]};
        item.latitudeLongitude=point
    }
    else{
        item.location=itemito.Location[0];
    }
    await Item.create(item);
}
else{
    const myCategories = itemito.Category;
    var lastcategory = 171;
    // For each category in the tree
    for (var i=0; i< myCategories.length; i++){

        var cat = await Category.findOne( { where: { name:
myCategories[i] } } );
        if (cat==null){
            const cat2 = await Category.create({
                name: myCategories[i],
                CategoryId: lastcategory,
            }).catch(err => {
                console.log(err);
            });
            lastcategory = cat2.id;
        }
        else {
            lastcategory = cat.id
        }
    }
}

```

```

    }
    const activeUsers = await User.findAll({ where: {
approved: true, admin: false},
    });
    const pickUserId = activeUsers[Math.floor(Math.random() *
activeUsers.length)].id;
    var item = {
        name: itemito.Name[0],
        currently:
parseFloat(itemito.Currently[0].substring(1)),
        first_bid:
parseFloat(itemito.First_Bid[0].substring(1)),
        number_of_bids: parseInt(itemito.Number_of_Bids[0]),
        country: itemito.Country[0],
        started: itemito.Started[0],
        ends: itemito.Ends[0],
        description: itemito.Description[0],
        state: 'PURCHASED',
        furthestmostCategoryId: lastcategory,
        UserId: pickUserId,
    }
    if (itemito.Buy_Price){
        item.buy_price=
parseFloat(itemito.Buy_Price[0].substring(1))
    }
    if (itemito.Location[0]._){
        item.location=itemito.Location[0]._
        var point = { type: 'Point', coordinates:
[itemito.Location[0].$.Latitude,itemito.Location[0].$.Longitude]};
        item.latitudeLongitude=point
    }
    else{
        item.location=itemito.Location[0];
    }
    const newItem = await Item.create(item);
    // Do a random bid from one of the other users
    const activeBuyers = await User.findAll({
        where: { approved: true,
admin: false,
id: {[Op.ne]: pickUserId},
        }
    });
    const pickBuyer =
activeBuyers[Math.floor(Math.random() * activeBuyers.length)];
    var bid = {
        time: itemito.Bids[0].Bid[i].Time[0],
        amount:
parseFloat(itemito.Bids[0].Bid[i].Amount[0].substring(1)),
        ItemId: newItem.id,
    }
    bid.bidderRating = pickBuyer.bidderRating;
    bid.bidderName = pickBuyer.username;
    bid.UserId = pickBuyer.id;
    await Bid.create(bid);
    // Also update the history which is the important
part for the Matrix Factorisation
    var userData = await UserData.findOne(
        { where: {
            UserId: pickBuyer.id,
            ItemId: newItem.id
        }
    });
    if (userData==null){
        await UserData.create({

```

```

                                rating: 5,
                                UserId: pickBuyer.id,
                                ItemId: newItem.id
                            });
                        }
                    }
                }
            });
            res.json("Success");
        });
    });
    module.exports = router;

```

Листинг Б.10 – Файл ControllerUser.cs

```

const express = require('express');
const router = express.Router();
const { User } = require('../models');
const bcrypt = require('bcrypt');
const { sign } = require('jsonwebtoken');
const { validateTheToken } = require('../middlewares/Authent');
const { Op } = require("sequelize");
router.post('/', async (req, res) => {

    const { username , password , name , surname , email , telephone ,
latitudeLongitude , location , country , taxnumber, sellerRating, saleCount,
bidderRating, buyCount } = req.body;
    bcrypt.hash(password, 10).then((hash)=>{
        User.create({
            username: username,
            password: hash,
            name: name,
            surname: surname,
            email: email,
            telephone: telephone,
            latitudeLongitude: latitudeLongitude,
            location: location,
            country: country,
            taxnumber: taxnumber,
            sellerRating: sellerRating,
            saleCount: saleCount,
            bidderRating: bidderRating,
            buyCount: buyCount,
            // this here just for safety
            admin: false,
            approved: false,
        });
        res.json('Successful');
    });
});
router.post('/login', async (req, res) => {

    const { username , password } = req.body;
    const user = await User.findOne({ where: {username: username}});
    // if the user is not in the table
    if (!user){
        res.json({error: "User doesn't exist!"});
    }
    else{
        if (user.approved===false){
            res.json({error: "approval"});
        }
        else{

```

```

        bcrypt.compare(password, user.password).then((matched)=>{
            if(!matched){
                res.json({error: "Wrong User Credentials!"});
            }
            else{
                const accessToken = sign({username: user.username, id:
user.id}, "hereputyoursecret");
                res.json({token: accessToken, username: user.username,
id: user.id });
            }
        });
    }

    });
router.get('/fetchy/:id', async (req, res) => {
    const myId = req.params.id;
    const userito = await User.findByPk(myId);
    // Don't return more information from the seller than is safe
    res.json({ username: userito.username, saleCount: userito.saleCount,
sellerRating: userito.sellerRating, name: userito.name, surname:
userito.surname, id: userito.id });
});
router.get('/exists/:id', async (req, res) => {
    const myId = req.params.id;
    const usr = await User.findOne({ where: { username: myId } });
    if(usr ===null){
        res.json({
            exists: false,
        });
    }
    else{
        res.json({
            exists: true,
        });
    }
});
// This is not asynchronous cause it's important for security
router.get('/validate', validateTheToken, (req, res) => {
    res.json({
        username: req.user.username,
        id: req.user.id,
        status: true,
    });
});
router.get('/fetchyall/:id', async (req, res) => {
    const myId = req.params.id;
    const userito = await User.findByPk(myId, {
        // Don't return the password from the attributes
        attributes: {exclude: ["password"]},
    });
    if (!userito || userito.username==="admin"){
        res.json({message:"Not allowed"})
    }
    else{
        res.json(userito);
    }
});
router.get('/', validateTheToken, async (req, res) => {
    // make sure that it's the admin
    const username = req.user.username;
    if (username==='admin'){
        // excluding here the admin and the password from the info
        const allUsers = await User.findAll( {where: {

```

```

        username: { [Op.ne]: 'admin' }
      },
      attributes: {exclude: ["password"]},
    });
    res.json(allUsers);
  }
});
router.get('/approve', validateTheToken , async (req, res) => {
  // excluding here the admin and the password from the info
  const allUsers = await User.findAll( {where: {
    approved: false,
    username: { [Op.ne]: 'admin' }
  },
  attributes: {exclude: ["password"]},
  });
  res.json(allUsers);

});
// User approval by the admin after validating the token
router.put('/approve', validateTheToken, async (req, res) => {
  const userList = req.body;
  const username = req.user.username;
  if (username==='admin'){
    for (var i = 0; i < userList.length; i++) {
      var userId = userList[i];
      console.log(userId);
      await User.update({
        approved: true
      },
      {where : {
        id: userId
      }
    })
    }
    res.json("Succesfully approved users!");
  }
  else{
    res.json("This is forbidden!")
  }
});
module.exports = router;

```

Листинг Б.11 – Файл Auth.js

```

// For the JSON Web Token Authentication
const {verify } = require('jsonwebtoken');
// Intermediary validation function
const validateTheToken = (req, res, next) =>{
  const accessToken = req.header("accessToken");
  if (!accessToken){
    return res.json({error: "No user is logged in!"})
  }
  else{
    try {
      const validTokenito = verify(accessToken, "hereputyoursecret");
      if(validTokenito){
        req.user = validTokenito;
        return next();
      }
    } catch (error) {
      return res.json({error: error});
    }
  }
}

```

```

    }
};

module.exports = {validateTheToken};

```

Листинг Б.12 – Файл index.js

```

// Initializing the variables
const express = require('express');
const app = express();
const cors = require('cors');
const https = require('https');
const fs = require('fs');
const db = require('./models');
const { User, Category } = require('./models');
const bcrypt = require('bcrypt');
const bodyParser = require("body-parser");
const cookieParser = require("cookie-parser");
const session = require("express-session");
// then express and cors
app.use(express.json());
app.use(cors({
  origin: ["https://localhost:3000"],
  methods: ["GET", "POST", "PUT", "DELETE"],
  credentials: true
}));
// this for the cookies
app.use(cookieParser());
app.use(bodyParser.urlencoded({extended:true }));
app.use(session({
  key: "userId",
  // this for development
  secret: "tempsecret",
  resave: false,
  saveUninitialized:false,
  cookie: {
    // cookie expires in 8 hours
    expires: 60 * 60 * 8,
  },
}));
// Routers
const itemRouter = require('./routes/Items');
app.use("/items", itemRouter);
const bidRouter = require('./routes/Bids');
app.use("/bids", bidRouter);
const userRouter = require('./routes/Users');
app.use("/auth", userRouter);
const mailRouter = require('./routes/Mail');
app.use("/mail", mailRouter);
const categoryRouter = require('./routes/Categories');
app.use("/categories", categoryRouter);
const historyRouter = require('./routes/History');
app.use("/history", historyRouter);
const photographyRouter = require('./routes/Photos');
app.use("/photos", photographyRouter);
app.use('/images/', express.static('images'));
// Create the ssl server
const sslServer = https.createServer({
  key: fs.readFileSync('/Users/yourname/key.pem'),
  cert: fs.readFileSync('/Users/yourname/cert.pem')
}, app);
// listen on port 33123 creating the tables in models in the process
db.sequelize.sync({ force: false, alter:true}).then(()=>{

```

```

console.log("Encrypted server up and running\nConnected to database");
// create the admin, if they do not already exist
const password="1234";
bcrypt.hash(password, 10).then((hash)=>{
  User.create({
    username: "admin",
    password: hash,
    name: "admin",
    admin: true,
    approved: true,
  }).catch(err => {
    console.log("Already exists");
  });
});
// Creating the basic "parent" category of the app
Category.create({
  name: "All Categories",
}).catch(err => {
  console.log("Already set up");
});
sslServer.listen(33123, ()=>{
  console.log("Listening on port: 33123");
});
});

```

Клиентская часть.

Листинг Б.18 – Файл AuctionDetail.js

```

import SimpleCard from "../ui/SimpleCard";
import Button from "@mui/material/Button";
import FavoriteBorderIcon from "@mui/icons-material/FavoriteBorder";
import FavoriteIcon from "@mui/icons-material/Favorite";
import TextField from "@mui/material/TextField";
import { useRef, useState, useEffect } from "react";
import { useRouter } from "next/router";
import { useDispatch, useSelector } from "react-redux";
import { alertActions } from "../../store/alertSlice";
import { useCookies } from "react-cookie";
import { ref, update, remove, set } from "firebase/database";
import database from "../../firebase/firebase";
import calculateRemainingTime from "../../helpers/remainingTimeCalculator";
import daysAndHours from "../../helpers/daysAndHoursConverter";
import classes from "../AuctionDetail.module.css";
const AuctionDetail = (props) => {
  const [priceIsHighlighted, setPriceIsHighlighted] = useState(false);
  const dispatch = useDispatch();
  const router = useRouter();
  const bidInput = useRef();
  const isLoggedIn = useSelector((state) => state.auth.isLoggedIn);
  const [favourite, setFavourite] = useState(false);
  const [lastBidder, setLastBidder] = useState("");
  const [isActive, setIsActive] = useState(true);
  const [isClosed, setIsClosed] = useState(false);
  const [cookie, setCookie, removeCookie] = useCookies();
  const userId = cookie.userId;
  const username = cookie.username;
  const [price, setPrice] = useState(props.auction.price);
  const [cardGridClasses, setCardGridClasses] = useState(
    classes.auctionCardGridActive
  );
  useEffect(() => {
    if (remainingTimeinMs < 0) {

```

```

    if (props.auction.lastBidder === "") {
      setLastBidder("This auction ended with no bids");
      setIsClosed(true);
    } else {
      setLastBidder(props.auction.lastBidder);
    }
    setIsActive(false);
    setCardGridClasses(classes.auctionCardGridClosed);
  }
}, []);
useEffect(() => {
  fetch(
    "https://diplom-c89f3-default-rtdb.europe-
west1.firebaseio.com/favourites.json"
  )
    .then((response) => response.json())
    .then((data) => {
      if (data !== null) {
        for (const value of Object.values(data)) {
          if (
            value.auctionId === router.query.auctionId &&
            value.userId === userId
          ) {
            setFavourite(true);
          }
        }
      }
    });
}, []);
const priceClasses = `${classes.price} ${
  priceIsHighlighted ? classes.bump : ""
}`;
const remainingTimeinMs = calculateRemainingTime(
  parseInt(props.auction.expirationTime)
);
const expirationDate = daysAndHours(remainingTimeinMs);
useEffect(() => {
  if (remainingTimeinMs < 0) {
    setIsActive(false);
  }
}, []);
const onBidHandler = (event) => {
  event.preventDefault();
  const auctionId = router.query.auctionId;
  if (!isLoggedIn) {
    dispatch(
      alertActions.error(
        "Пожалуйста, сначала войдите в аккаунт. Тогда вы сможете делать
ставки"
      )
    );
  };
  setTimeout(() => {
    dispatch(alertActions.close());
  }, 5000);
  return;
} else if (
  bidInput.current.value > parseFloat(props.auction.price.replace(/,/g,
""))
) {
  //Setting price so that change is displayed in-page
  setPriceIsHighlighted(true);
  setTimeout(() => {
    setPriceIsHighlighted(false);
  }, 300);
}

```

```

    setPrice(parseInt(bidInput.current.value).toLocaleString("en-US"));
    //Updating price and last bidder in database:
    function addBid(bid, username) {
      const updates = {};
      updates["/auctions/" + auctionId + "/price/"] = bid;
      updates["/auctions/" + auctionId + "/lastBidder/"] = username;
      bidInput.current.value = "";
      return update(ref(database), updates);
    }
    addBid(
      parseInt(bidInput.current.value).toLocaleString("en-US"),
      username
    );
  } else {
    dispatch(
      alertActions.error("Ваша ставка должна быть выше текущей цены")
    );
    setTimeout(() => {
      dispatch(alertActions.close());
    }, 5000);
  }
};
function addFav() {
  const favData = {
    auctionId: router.query.auctionId,
    userId: userId,
  };
  fetch(
    "https://diplom-c89f3-default-rtdb.europe-west1.firebaseio.com/favourites.json",
    {
      method: "POST",
      headers: {
        "Content-Type": "application/json",
      },
      body: JSON.stringify(favData),
    }
  )
    .then((response) => response.json())
    .then((data) => {
      console.log("Success:", data);
    })
    .catch((error) => {
      console.error("Error:", error);
    });
}
const onFavouriteHandler = () => {
  if (!isLoggedIn) {
    dispatch(
      alertActions.error(
        "Пожалуйста, сначала войдите в аккаунт. После этого вы сможете добавлять аукционы в избранное."
      )
    );
    setTimeout(() => {
      dispatch(alertActions.close());
    }, 5000);
    return;
  }
  // Getting favourites list from database:
  setFavourite(!favourite);
  fetch(
    "https://diplom-c89f3-default-rtdb.europe-west1.firebaseio.com/favourites.json"
  )

```

```

)
.then((response) => response.json())
.then((data) => {
  if (data !== null) {
    let counter = 0;
    for (const [key, value] of Object.entries(data)) {
      if (
        value.auctionId === router.query.auctionId &&
        value.userId === userId
      ) {
        remove(ref(database, "favourites/" + key));
      } else {
        counter++;
      }
      if (counter === Object.keys(data).length) {
        ///// Adding Favourite to Database /////
        addFav();
      }
    }
  } else {
    ///// Adding Favourite to Database /////
    addFav();
  }
});
};
return (
  <section className={classes.auctionCard}>
    <SimpleCard>
      <div className={classes.auctionCardFlex}>
        {!favourite && (
          <FavoriteBorderIcon
            sx={{
              bgcolor: "white",
              color: "black",
              border: 1,
              borderColor: "white",
              borderRadius: 2,
              boxShadow: 3,
            }}
            fontSize="large"
            className={classes.favoriteBtn}
            onClick={onFavouriteHandler}
          />
        )}
        {favourite && (
          <FavoriteIcon
            sx={{
              bgcolor: "white",
              color: "black",
              border: 1,
              borderColor: "white",
              borderRadius: 2,
              boxShadow: 3,
            }}
            fontSize="large"
            className={classes.favoriteBtn}
            onClick={onFavouriteHandler}
          />
        )}
        {isActive && <h3 className={classes.activeBadge}>АКТИВНО</h3>}
        {!isActive && !isClosed && (
          <h3 className={classes.soldBadge}>ПРОДАНО</h3>
        )}
        {!isActive && isClosed && (

```

```

    <h3 className={classes.closedBadge}>ЗАКРЫТО</h3>
  )}
  <img
    className={classes.image}
    src={props.auction.image}
    alt="car picture"
  />
  <div className={cardGridClasses}>
    <h1 className={classes.model}>{props.auction.model}</h1>
    <div className={classes.description}>
      {props.auction.description}
    </div>
    <h1 className={priceClasses}>P{price}</h1>
    {isActive && (
      <form onSubmit={onBidHandler} className={classes.form}>
        <TextField
          inputRef={bidInput}
          id="outlined-number"
          label="P Введите ставку"
          type="number"
          width="100%"
          inputProps={{ min: 0, max: 1000000000000000000 }}
          InputLabelProps={{
            shrink: true,
          }}
        />
        <Button type="submit" variant="contained">
          Сделать ставку
        </Button>
      </form>
    )}
    <div className={classes.ownerAndTimeOrResult}>
      <div>
        Владелец:{" "}
        <span
          className={classes.ownerSpan}>{props.auction.owner}</span>
      </div>
      {isClosed && <div>Этот аукцион завершился без ставок</div>}
      {!isActive && !isClosed && (
        <div>
          Победитель:{" "}
          <span className={classes.lastBidder}>@{lastBidder}</span>
        </div>
      )}
      {isActive && (
        <div>
          Оставшееся время:{" "}
          <span
            className={classes.timeLeftSpan}>{expirationDate}</span>
        </div>
      )}
    </div>
  </div>
</div>
</SimpleCard>
</section>
);
};
export default AuctionDetail;

```

Листинг Б.19 – Файл AuctionItem.js

```
import { useState, useEffect } from "react";
import Link from "next/link";
import calculateRemainingTime from "../../helpers/remainingTimeCalculator";
import daysAndHours from "../../helpers/daysAndHoursConverter";
import SimpleCard from "../../ui/SimpleCard";
import CardMedia from "@mui/material/CardMedia";
import classes from "./AuctionItem.module.css";

const AuctionItem = (props) => {
  const [lastBidder, setLastBidder] = useState("");
  const [isActive, setIsActive] = useState(true);
  const [isClosed, setIsClosed] = useState(false);
  const remainingTimeinMs = calculateRemainingTime(
    parseInt(props.expirationTime)
  );
  const expirationDate = daysAndHours(remainingTimeinMs);
  useEffect(() => {
    if (remainingTimeinMs < 0) {
      if (props.lastBidder === "") {
        setLastBidder("Этот аукцион завершился без ставок");
        setIsClosed(true);
      } else {
        setLastBidder(props.lastBidder);
      }
      setIsActive(false);
    }
  }, []);
  return (
    <Link href={`/auction/${props.id}`}>
      <section className={classes.cardSection}>
        <SimpleCard>
          <CardMedia
            className={classes.image}
            component="img"
            height="200"
            image={props.image}
            alt="car picture"
          />
          <div className={classes.cardContent}>
            {isActive && <h3 className={classes.activeBadge}>АКТИВНЫЙ</h3>}
            {!isActive && !isClosed && (
              <h3 className={classes.soldBadge}>ПРОДАНО</h3>
            )}
            {!isActive && isClosed && (
              <h3 className={classes.closedBadge}>ЗАКРЫТ</h3>
            )}
          </div>
          <h1>
            {props.model}
          </h1>
          {isActive && (
            <div className={classes.timeLeftOrResult}>
              Оставшееся время: {expirationDate}
            </div>
          )}
          {!isActive && !isClosed && (
            <div className={classes.timeLeftOrResult}>
              Победитель: {lastBidder}
            </div>
          )}
          {!isActive && isClosed && (
            <div className={classes.timeLeftOrResult}>
              {lastBidder}
            </div>
          )}
        </SimpleCard>
      </section>
    </Link>
  );
};
```

```

    ))
    <h1 className={classes.price}>P{props.price}</h1>
  </div>
</SimpleCard>
</section>
</Link>
);
};

export default AuctionItem;

```

Листинг Б.20 – Файл AuctionList.js

```

import AuctionItem from "../AuctionItem";
import classes from "../AuctionList.module.css";
const AuctionList = (props) => {
  let auctions = [];
  if (props && props.auctions) {
    auctions = props.auctions;
  }
  return (
    <ul className={classes.list}>
      {auctions.map((auction) => (
        <li key={auction.auctionId}>
          <AuctionItem
            model={auction.model}
            expirationTime={auction.expirationTime}
            price={auction.price}
            image={auction.image}
            id={auction.auctionId}
            lastBidder={auction.lastBidder}
          />
        </li>
      ))}
    </ul>
  );
};

export default AuctionList;

```

Листинг Б.21 – Файл NewAuctionForm.js

```

import SimpleCard from "../../components/ui/SimpleCard";
import TextField from "@mui/material/TextField";
import Button from "@mui/material/Button";
import CircularProgress from "@mui/material/CircularProgress";
import { useRef, useState } from "react";
import { useRouter } from "next/router";
import { useCookies } from "react-cookie";
import { useDispatch } from "react-redux";
import { alertActions } from "../../store/alertSlice";
import classes from "../NewAuctionForm.module.css";
import { storage } from "../../firebase/firebase";
import { ref, uploadBytesResumable, getDownloadURL } from "firebase/storage";
const NewAuctionForm = () => {
  const dispatch = useDispatch();
  const [cookie, setCookie, removeCookie] = useCookies();
  const username = cookie.username;
  const router = useRouter();
  const [image, setImage] = useState(null);
  const [isUploading, setIsUploading] = useState(false);
  const inputModel = useRef();

```

```

const inputDescription = useRef();
const inputPrice = useRef();
const inputDateTime = useRef();
const auctionId = Date.now()
//Getting current formatted Date and Time timestamp to be used in the
expirationTime input selector:
const currentDate = new Date();
const mins = ("0" + currentDate.getMinutes()).slice(-2);
const time = currentDate.getHours() + ":" + mins;
const cDay = ("0" + currentDate.getDate()).slice(-2);
const cMonth = currentDate.getMonth() + 1;
const cYear = currentDate.getFullYear();
const currentFormattedDateTime =
  cYear + "-" + cMonth + "-" + cDay + "T" + time;
// Storing uploaded image in a state
const handleImageChange = (e) => {
  if (e.target.files[0]) {
    setImage(e.target.files[0]);
  }
};
const onSubmitHandler = (event) => {
  event.preventDefault();
  // Checking if a file has been selected
  if (image === null) {
    dispatch(alertActions.error("Пожалуйста, выберите изображение."));
    setTimeout(() => {
      dispatch(alertActions.close());
    }, 5000);
    return;
  }
  const expirationDate = new Date(inputDateTime.current.value);
  const expirationDateInMs = expirationDate.getTime();
  const storageRef = ref(storage, `${image.name}`);
  const uploadTask = uploadBytesResumable(storageRef, image);
  // Checking if the file is a valid image. If not, displaying error
  message.
  if (!image.name.match(/\.(jpg|jpeg|png|gif)$/)) {
    dispatch(alertActions.error("Пожалуйста, выберите изображение."));
    setTimeout(() => {
      dispatch(alertActions.close());
    }, 5000);
    // Cancelling the upload if not valid.
    uploadTask.cancel();
    return;
  }
  uploadTask.on(
    "state_changed",
    (snapshot) => {
      // Get task progress, including the number of bytes uploaded and the
total number of bytes to be uploaded
      setIsUploading(true);
      const progress =
        (snapshot.bytesTransferred / snapshot.totalBytes) * 100;
      console.log("Загрузка на" + progress + "% завершена");
      switch (snapshot.state) {
        case "paused":
          console.log("Загрузка приостановлена");
          break;
        case "running":
          console.log("Выполняется загрузка");
          break;
      }
    },
    (error) => {

```

```

// A full list of error codes is available at
// https://firebase.google.com/docs/storage/web/handle-errors
switch (error.code) {
  case "storage/unauthorized":
    // User doesn't have permission to access the object
    break;
  case "storage/canceled":
    // User canceled the upload
    break;
  // ...
  case "storage/unknown":
    // Unknown error occurred, inspect error.serverResponse
    break;
}
},
() => {
  // Upload completed successfully, now we can get the download URL
  getDownloadURL(uploadTask.snapshot.ref).then((downloadURL) => {
    const newAuctionData = {
      model: inputModel.current.value,
      auctionId: auctionId.toString(),
      expirationTime: expirationDateInMs,
      price: parseInt(inputPrice.current.value).toLocaleString("en-
US"),
      description: inputDescription.current.value,
      owner: username,
      image: downloadURL,
      lastBidder: "",
    };
    // Emptying the inputs
    inputModel.current.value = "";
    inputDescription.current.value = "";
    inputPrice.current.value = "";
    inputDateTime.current.value = "";
    fetch(
      `https://diplom-c89f3-default-rtdb.europe-
west1.firebaseio.com/auctions/${newAuctionData.auctionId}.json`,
      {
        method: "PUT",
        headers: {
          "Content-Type": "application/json",
        },
        body: JSON.stringify(newAuctionData),
      }
    )
      .then((response) => response.json())
      .then((data) => {
        console.log("Успешно:", data);
        setIsUploading(false);
        router.push(`/`);
      })
      .catch((error) => {
        console.error("Ошибка:", error);
      });
  });
}
);
};
return (
  <section className={classes.formSection}>
    <SimpleCard>
      <form onSubmit={onSubmitHandler} className={classes.form}>
        <h1>Новый лот</h1>
        <TextField

```

```

        required
        inputRef={inputModel}
        id="outlined-basic"
        label="Модель"
        variant="outlined"
      />
      <TextField
        required
        inputRef={inputDescription}
        id="outlined-multiline-static"
        label="Описание..."
        multiline
        rows={4}
      />
      <TextField
        required
        inputRef={inputPrice}
        id="outlined-number"
        label="Цена ₴"
        type="number"
        InputLabelProps={{
          shrink: true,
        }}
      />
      <TextField
        required
        inputRef={inputDateTime}
        id="datetime-local"
        label="Дата конца аукциона"
        type="datetime-local"
        defaultValue={currentFormattedDateTime}
        sx={{ width: 250 }}
        InputLabelProps={{
          shrink: true,
        }}
      />
      <div className={classes.imageInput}>
        Загрузить изображение
        <input type="file" accept="image/*" onChange={handleImageChange}
      />
    </div>
    {!isUploading && (
      <Button className={classes.input} type="submit"
variant="contained">
        Подтвердить
      </Button>
    )}
    {isUploading && <CircularProgress className={classes.progress} />}
  </form>
</SimpleCard>
</section>
);
};
export default NewAuctionForm;

```

Листинг Б.22 – Файл LoginForm.js

```

import { useRef, useState } from "react";
import { useRouter } from "next/router";
import { authActions } from "../../store/authSlice";
import { alertActions } from "../../store/alertSlice";
import { useDispatch } from "react-redux";
import { useCookies } from "react-cookie";

```

```

import SimpleCard from "../../components/ui/SimpleCard";
import TextField from "@mui/material/TextField";
import Button from "@mui/material/Button";
import CircularProgress from "@mui/material/CircularProgress";
import classes from "./LoginForm.module.css";
const LoginForm = () => {
  const [isLoading, setIsLoading] = useState(false);
  const router = useRouter();
  const [cookie, setCookie] = useCookies();
  const emailInput = useRef();
  const passwordInput = useRef();
  const dispatch = useDispatch();
  const onSubmitHandler = (event) => {
    event.preventDefault();
    setIsLoading(true);
    const loginData = {
      email: emailInput.current.value,
      password: passwordInput.current.value,
      returnSecureToken: true,
    };
    // Signing in user through Firebase
    fetch(
      "https://identitytoolkit.googleapis.com/v1/accounts:signInWithPassword?key=AIzaSyBJxb-Ayy0nEhS5w467HKmDhSv5aVXlhBA",
      {
        method: "POST",
        headers: {
          "Content-Type": "application/json",
        },
        body: JSON.stringify(loginData),
      }
    )
      .then((response) => response.json())
      .then((data) => {
        if (data.idToken) {
          dispatch(
            authActions.login({ token: data.idToken, userId: data.localId })
          );
        }
        if (data.error) {
          console.log(data.error.message);
          let message = null;
          // Adapting error messages so they are easier to understand
          if (data.error.message === "INVALID_EMAIL") {
            message = "Пожалуйста, введите действительный адрес электронной почты.";
          } else if (data.error.message === "EMAIL_NOT_FOUND") {
            message = "Введенный адрес электронной почты не зарегистрирован. Пожалуйста, сначала зарегистрируйтесь.";
          } else if (data.error.message === "INVALID_PASSWORD") {
            message = "Пароль не правильный.";
          } else if (data.error.message === "MISSING_PASSWORD") {
            message = "Пароль отсутствует.";
          }
          dispatch(alertActions.error(message || data.error.message));
          setTimeout(() => {
            dispatch(alertActions.close());
          }, 5000);
          setIsLoading(false);
          return;
        }
      })
      // We store in cookies the user/session data:

```

```

    setCookie("token", JSON.stringify(data.idToken), {
      path: "/",
      maxAge: 3600, // Expires after 1hr
      sameSite: true,
    });
    setCookie("userId", JSON.stringify(data.localId), {
      path: "/",
      maxAge: 3600, // Expires after 1hr
      sameSite: true,
    });
    setCookie("expirationTime", JSON.stringify(Date.now() + 3600000), {
      path: "/",
      maxAge: 3600, // Expires after 1hr
      sameSite: true,
    });
    const id = data.localId;
    return fetch(
      "https://diplom-c89f3-default-rtdb.europe-west1.firebaseio.com/app/users.json"
    )
      .then((response) => response.json())
      .then((data) => {
        let username = "";
        if (data != null) {
          for (const value of Object.values(data)) {
            if (value.userId === id) {
              username = value.username;
              setCookie("username", username, {
                path: "/",
                maxAge: 3600, // Expires after 1hr
                sameSite: true,
              });
            }
          }
        }
        emailInput.current.value = "";
        passwordInput.current.value = "";
        setIsLoading(false);
        router.push(`/`);
        dispatch(alertActions.success("Добро пожаловать, " + username +
"!"));
        setTimeout(() => {
          dispatch(alertActions.close());
        }, 5000);
      });
  })
  .catch((error) => {
    console.log(error);
  });
};

return (
  <section className={classes.formSection}>
    <SimpleCard>
      <form onSubmit={onSubmitHandler} className={classes.form}>
        <h1>Вход</h1>
        <TextField
          inputRef={emailInput}
          id="outlined-basic"
          label="E-mail"
          type="email"
          variant="outlined"
        />
        <TextField
          inputRef={passwordInput}

```

```

        id="outlined-basic"
        label="Пароль"
        type="password"
        variant="outlined"
      />
      {!isLoading && (
        <Button type="submit" variant="contained">
          Войти
        </Button>
      )}
      {isLoading && <CircularProgress className={classes.progress} />}
    </form>
  </SimpleCard>
</section>
);
};
export default LoginForm;

```

Листинг Б.23 – Файл SignUpForm.js

```

import { useRef, useState } from "react";
import { useRouter } from "next/router";
import SimpleCard from "../../components/ui/SimpleCard";
import TextField from "@mui/material/TextField";
import Button from "@mui/material/Button";
import CircularProgress from "@mui/material/CircularProgress";
import { alertActions } from "../../store/alertSlice";
import { useDispatch } from "react-redux";
import classes from "./SignUpForm.module.css";

const SignUpForm = () => {
  const [isLoading, setIsLoading] = useState(false);
  const router = useRouter();
  const dispatch = useDispatch();
  const emailInput = useRef();
  const passwordInput = useRef();
  const usernameInput = useRef();
  const onSubmitHandler = (event) => {
    event.preventDefault();
    setIsLoading(true);
    const enteredData = {
      email: emailInput.current.value,
      password: passwordInput.current.value,
      returnSecureToken: true,
    };
    const username = usernameInput.current.value;
    emailInput.current.value = "";
    passwordInput.current.value = "";
    usernameInput.current.value = "";
    fetch(
      "https://identitytoolkit.googleapis.com/v1/accounts:signUp?key=AIzaSyBJxb-
      Ayy0nEhS5w467HKmDhSv5aVXlhBA",
      {
        method: "POST",
        headers: {
          "Content-Type": "application/json",
        },
        body: JSON.stringify(enteredData),
      }
    )
      .then((response) => response.json())
      .then((data) => {
        if (data.error) {

```

```

    let message = null;
    console.log(data.error.message);
    if (data.error.message === "EMAIL_EXISTS") {
      message = "Адрес электронной почты уже используется другим
аккаунтом.";
    } else if (data.error.message === "OPERATION_NOT_ALLOWED") {
      message = "Password sign-in is disabled for this project.";
    } else if (data.error.message === "TOO_MANY_ATTEMPTS_TRY_LATER") {
      message =
        "We have blocked all requests from this device due to unusual
activity. Try again later.";
    } else if (data.error.message === "WEAK_PASSWORD") {
      message = "Пароль должен быть не менее 6 символов";
    }
    dispatch(alertActions.error(message || data.error.message));
    setTimeout(() => {
      dispatch(alertActions.close());
    }, 5000);
    setIsLoading(false);
    return;
  }
  const IdAndUsername = { userId: data.localId, username: username };
  fetch(
    `https://diplom-c89f3-default-rtdb.europe-
west1.firebaseio.com/users.json`,
    {
      method: "POST",
      headers: {
        "Content-Type": "application/json",
      },
      body: JSON.stringify(IdAndUsername),
    }
  )
    .then((response) => response.json())
    .then((data) => {
      console.log("Success:", data);
      dispatch(alertActions.success("Аккаунт успешно создан! Теперь вы
можете войти в систему."));
      setTimeout(() => {
        dispatch(alertActions.close());
      }, 5000);
      setIsLoading(false);
      router.push(`/login`);
    })
    .catch((error) => {
      console.error("Error:", error);
    });
  });
  .catch((error) => {
    console.log(error);
  });
});
return (
  <section className={classes.formSection}>
    <SimpleCard>
      <form onSubmit={onSubmitHandler} className={classes.form}>
        <h1>Регистрация</h1>
        <TextField
          required
          inputRef={emailInput}
          id="outlined-basic"
          label="E-mail"
          type="email"
          variant="outlined"

```

```

    />
    <TextField
      required
      inputRef={usernameInput}
      id="outlined-basic"
      label="Имя пользователя"
      type="text"
      variant="outlined"
    />
    <TextField
      required
      inputRef={passwordInput}
      id="outlined-basic"
      label="Пароль"
      type="password"
      variant="outlined"
    />
    {!isLoading && <Button type="submit" variant="contained">
      Зарегистрироваться
    </Button>}
    {isLoading && <CircularProgress className={classes.progress}/>}
  </form>
</SimpleCard>
</section>
);
};

export default SignupForm;

```

Листинг Б.24 – Файл Filter.js

```

import classes from "../Filter.module.css";
import BasicSelect from "../ui/BasicSelect";
const Filter = (props) => {
  return (
    <section className={classes.section}>
      <h1 className={classes.title}>Результаты фильтрации:</h1>
      <BasicSelect status={props.status} setStatus={props.setStatus}
        priceRange={props.priceRange} setPriceRange={props.setPriceRange}/>
    </section>
  );
};
export default Filter;

```

Листинг Б.25 – Файл AccountMenu.js

```

import * as React from "react";
import { useEffect, useState } from "react";
import Box from "@mui/material/Box";
import Avatar from "@mui/material/Avatar";
import Menu from "@mui/material/Menu";
import MenuItem from "@mui/material/MenuItem";
import ListItemIcon from "@mui/material/ListItemIcon";
import Divider from "@mui/material/Divider";
import IconButton from "@mui/material/IconButton";
import AddCircleIcon from "@mui/icons-material/AddCircle";
import Tooltip from "@mui/material/Tooltip";
import Logout from "@mui/icons-material/Logout";
import Link from "next/link";
const AccountMenu = (props) => {
  const [firstLetter, setFirstLetter] = useState("");
  const [username, setUsername] = useState("user");
  const [anchorEl, setAnchorEl] = React.useState(null);

```

```

const open = Boolean(anchorEl);
const handleClick = (event) => {
  setAnchorEl(event.currentTarget);
};
const handleClose = () => {
  setAnchorEl(null);
};
useEffect(() => {
  if (props && props.username) {
    setFirstLetter(props.username[0].toUpperCase());
    setUsername(props.username);
  }
}, [props]);
return (
  <React.Fragment>
    <Box sx={{ display: "flex", alignItems: "center", textAlign: "center"
  >>
    <Tooltip title={props.username}>
      <IconButton onClick={handleClick} size="small" sx={{ ml: 2 }}>
        <Avatar sx={{ width: 32, height: 32, bgcolor: "#017cbf" }}>
          {firstLetter}
        </Avatar>
      </IconButton>
    </Tooltip>
  </Box>
  <Menu
    anchorEl={anchorEl}
    open={open}
    onClose={handleClose}
    onClick={handleClose}
    PaperProps={{
      elevation: 0,
      sx: {
        overflow: "visible",
        filter: "drop-shadow(0px 2px 8px rgba(0,0,0,0.32))",
        mt: 1.5,
        "& .MuiAvatar-root": {
          width: 32,
          height: 32,
          ml: -0.5,
          mr: 1,
        },
        "&:before": {
          content: '""',
          display: "block",
          position: "absolute",
          top: 0,
          right: 14,
          width: 10,
          height: 10,
          transform: "translateY(-50%) rotate(45deg)",
          zIndex: 0,
        },
      },
    }}
    transformOrigin={{ horizontal: "right", vertical: "top" }}
    anchorOrigin={{ horizontal: "right", vertical: "bottom" }}
  >
    <MenuItem>
      <Avatar /> {username}
    </MenuItem>
    <Divider />
    <Link href="/new-auction">
      <MenuItem>

```

```

        <ListItemIcon>
          <AddCircleIcon fontSize="small" />
        </ListItemIcon>
        Добавить новый лот
      </MenuItem>
    </Link>
    <MenuItem onClick={props.logout}>
      <ListItemIcon>
        <Logout fontSize="small" />
      </ListItemIcon>
      Выйти
    </MenuItem>
  </Menu>
</React.Fragment>
);
};
export default AccountMenu;

```

Листинг Б.26 – Файл Footer.js

```

import classes from "../Footer.module.css";
const Footer = () => {
  return <footer className={classes.footer}> </footer>;
};
export default Footer;

```

Листинг Б.27 – Файл Layout.js

```

import { useDispatch } from "react-redux";
import { useSelector } from "react-redux";
import { authActions } from "../../store/authSlice";
import { alertActions } from "../../store/alertSlice";
import { useCookies } from "react-cookie";
import Footer from "../Footer";
import Navbar from "../Navbar";
import Alert from "@mui/material/Alert";
import classes from "../Layout.module.css";
import calculateRemainingTime from "../../helpers/remainingTimeCalculator";
const Layout = (props) => {
  const dispatch = useDispatch();
  const [cookie, setCookie] = useCookies(["expirationTime"]);
  const alertState = useSelector((state) => state.alert);
  ///////////////AUTO-LOGOUT////////////////////////////////////
  const expirationTime = parseInt(cookie.expirationTime);
  setTimeout(() => {
    dispatch(authActions.logout());
  }, calculateRemainingTime(expirationTime));
  ///////////////
  const onCloseHandler = () => {
    dispatch(alertActions.close());
  };
  return (
    <div className={classes.wholePage}>
      <Navbar />
      {alertState.showAlert && (
        <Alert
          className={classes.alert}
          onClose={onCloseHandler}
          severity={alertState.type}
          sx={{ width: "100%" }}
        >

```

```

        {alertState.message}
      </Alert>
    )}
    <main className={classes.main}>{props.children}</main>
    <Footer />
  </div>
);
};
export default Layout;

```

Листинг Б.28 – Файл NavBar.js

```

import { useEffect, useState } from "react";
import { useSelector } from "react-redux";
import { useDispatch } from "react-redux";
import { authActions } from "../../store/authSlice";
import { useCookies } from "react-cookie";
import { useRouter } from "next/router";
import Link from "next/link";
import MenuRoundedIcon from "@mui/icons-material/MenuRounded";
import Button from "@mui/material/Button";
import AccountMenu from "../AccountMenu";
import classes from "../Navbar.module.css";

const NavBar = () => {
  const router = useRouter();
  const dispatch = useDispatch();
  const [cookie, setCookie, removeCookie] = useCookies();
  const [drawerClass, setDrawerClass] = useState(classes.mobileNav);
  const [backdropClass, setBackdropClass] = useState(classes.backdrop);
  const [favouritesBtnClass, setMenuBtnClass] = useState(null);
  const [allAuctionsBtnClass, setAllAuctionsBtnClass] = useState(null);
  const [joinBtnClass, setJoinBtnClass] = useState(null);
  const [sellCarBtnClass, setSellCarBtnClass] = useState(null);
  const [newAuctionBtnClass, setNewAuctionBtnClass] = useState(null);
  const [myAuctionsBtnClass, setMyAuctionsBtnClass] = useState(null);
  const currentPath = router.pathname;
  // Checking in redux store if user is logged in.
  const isLoggedIn = useSelector((state) => state.auth.isLoggedIn);
  // Getting username from cookies and storing it in a variable.
  const username = cookie.username;
  // When dispatching the logout action, we also remove all the user cookies.
  This way, user will be kept logged out if refreshes the page.
  const logoutHandler = (event) => {
    event.preventDefault();
    setDrawerClass(classes.mobileNav);
    setBackdropClass(classes.backdrop);
    removeCookie("token", { path: "/" });
    removeCookie("userId", { path: "/" });
    removeCookie("expirationTime", { path: "/" });
    removeCookie("username", { path: "/" });
    dispatch(authActions.logout());
    router.push(`/`);
  };

  const openDrawerHandler = () => {
    setDrawerClass(classes.mobileNavOpen);
    setBackdropClass(classes.backdropOpen);
  };

  const closeDrawerHandler = () => {
    setDrawerClass(classes.mobileNav);
    setBackdropClass(classes.backdrop);
  };

  // When clicking on a menu button, we want to keep the button selected. So
  this Useffect checks the url route and assing the right class to the button.

```

```

useEffect(() => {
  if (router.pathname.includes("my-favourites")) {
    setMenuBtnClass(classes.selectedBtn);
  } else {
    setMenuBtnClass(null);
  }
  if (router.pathname === "/") {
    setAllAuctionsBtnClass(classes.selectedBtn);
  } else {
    setAllAuctionsBtnClass(null);
  }
  if (router.pathname.includes("sign-up")) {
    setJoinBtnClass(classes.selectedBtn);
  } else {
    setJoinBtnClass(null);
  }
  if (router.pathname.includes("sell-my-car")) {
    setSellCarBtnClass(classes.selectedBtn);
  } else {
    setSellCarBtnClass(null);
  }
  if (router.pathname.includes("new-auction")) {
    setNewAuctionBtnClass(classes.selectedBtn);
  } else {
    setNewAuctionBtnClass(null);
  }
  if (router.pathname.includes("sell-my-car")) {
    setSellCarBtnClass(classes.selectedBtn);
  } else {
    setSellCarBtnClass(null);
  }
  if (router.pathname.includes("my-auctions")) {
    setMyAuctionsBtnClass(classes.selectedBtn);
  } else {
    setMyAuctionsBtnClass(null);
  }
}, [currentPath]);
return (
  <section>
    <div onClick={closeDrawerHandler} className={backdropClass}></div>
    <header className={classes.headerDesktop}>
      <Link href="/">
        <div className={classes.logo}>Аукцион</div>
      </Link>
      <div>
        <ul className={classes.middleButtons}>
          {!isLoggedIn && (
            <Link href="/sign-up">
              <li id={classes.joinNow} className={joinBtnClass}>
                Присоединяйся сейчас
              </li>
            </Link>
          )}
          {isLoggedIn && (
            <Link href="/my-favourites">
              <li className={favouritesBtnClass}>Мое избранное</li>
            </Link>
          )}
          <Link href="/">
            <li className={allAuctionsBtnClass}>Все лоты</li>
          </Link>
          {isLoggedIn && (
            <Link href="/my-auctions">

```

```

        <li className={myAuctionsBtnClass}>Мои лоты</li>
      </Link>
    )}
    {isLoggedIn && (
      <Link href="/new-auction">
        <li className={newAuctionBtnClass}>Создать лот</li>
      </Link>
    )}
    {!isLoggedIn && (
      <Link href="/sell-my-car">
        <li className={sellCarBtnClass}>Создать лот</li>
      </Link>
    )}
  </ul>
</div>
<nav>
  {isLoggedIn && (
    <AccountMenu username={username} logout={logoutHandler} />
  )}
  {!isLoggedIn && (
    <ul>
      {isLoggedIn && (
        <li>
          <Link href="/new-auction">
            <Button className={classes.button} variant="contained">
              New Auction
            </Button>
          </Link>
        </li>
      )}
      {isLoggedIn && (
        <li>
          <Link href="/">
            <Button
              className={classes.button}
              onClick={logoutHandler}
              variant="contained"
            >
              ({cookie.username}) ВЫЙТИ
            </Button>
          </Link>
        </li>
      )}
      {!isLoggedIn && (
        <li>
          <Link href="/login">
            <Button className={classes.button} variant="contained">
              Войти
            </Button>
          </Link>
        </li>
      )}
      {!isLoggedIn && (
        <li>
          <Link href="/sign-up">
            <Button className={classes.button} variant="contained">
              Регистрация
            </Button>
          </Link>
        </li>
      )}
    </ul>
  )}
</nav>

```

```

</header>
<header className={classes.headerMobile}>
  <Link href="/">
    <div className={classes.logo}>Конкретный аукцион</div>
  </Link>
  <MenuRoundedIcon
    onClick={openDrawerHandler}
    fontSize="large"
    className={classes.hamburger}
  />
  <nav className={drawerClass}>
    <ul className={classes.mobileNavButtons}>
      {!isLoggedIn && (
        <li className={classes.mobileNavItem}>
          <Link href="/login">
            <Button
              className={classes.button}
              variant="contained"
              onClick={closeDrawerHandler}
            >
              Войти
            </Button>
          </Link>
        </li>
      )}
      {isLoggedIn && (
        <li className={classes.mobileNavItem}>
          <Link href="/sign-up">
            <Button
              className={classes.button}
              variant="contained"
              onClick={closeDrawerHandler}
            >
              Зарегистрироваться
            </Button>
          </Link>
        </li>
      )}
      {isLoggedIn && (
        <li className={classes.mobileNavItem}>
          <Link href="/my-favourites">
            <Button
              className={classes.button}
              variant="contained"
              onClick={closeDrawerHandler}
            >
              Мое избранное
            </Button>
          </Link>
        </li>
      )}
      <li className={classes.mobileNavItem}>
        <Link href="/">
          <Button
            className={classes.button}
            variant="contained"
            onClick={closeDrawerHandler}
          >
            Все лоты
          </Button>
        </Link>
      </li>
      {isLoggedIn && (
        <li className={classes.mobileNavItem}>

```

```

        <Link href="/my-auctions">
          <Button
            className={classes.button}
            variant="contained"
            onClick={closeDrawerHandler}
          >
            Мои лоты
          </Button>
        </Link>
      </li>
    )}
    {isLoggedIn && (
      <li className={classes.mobileNavItem}>
        <Link href="/new-auction">
          <Button
            className={classes.button}
            variant="contained"
            onClick={closeDrawerHandler}
          >
            <li className={sellCarBtnClass}>Создать лот</li>
          </Button>
        </Link>
      </li>
    )}
    {!isLoggedIn && (
      <li className={classes.mobileNavItem}>
        <Link href="/sell-my-car">
          <Button
            className={classes.button}
            variant="contained"
            onClick={closeDrawerHandler}
          >
            Создать лот
          </Button>
        </Link>
      </li>
    )}
    {isLoggedIn && (
      <li className={classes.mobileNavItem}>
        <Link href="/">
          <Button
            className={classes.button}
            onClick={logoutHandler}
            variant="contained"
          >
            ({username}) Выйти
          </Button>
        </Link>
      </li>
    )}
  </ul>
</nav>
</header>
</section>
);
};
export default Navbar;

```

Листинг Б.29 – Файл AuctionPage.js

```

// our-domain.com/auction/auctionId
import { Fragment, useEffect, useState } from "react";
import { useDispatch } from "react-redux";

```

```

import { authActions } from "../../store/authSlice";
import { useRouter } from "next/router";
import AuctionDetail from "../../components/auctions/AuctionDetail";
import { parseCookies } from "../../helpers";
const Auction = ({ data }) => {
  const dispatch = useDispatch();
  const [auctionId, setAuctionId] = useState("");
  const [auctionInfo, setAuctionInfo] = useState(null);
  const router = useRouter();
  // Checking whether there are authentication cookies available or not. If
  // available -> Login. Otherwise -> Logout.
  if (!data.token || !data.userId || !data.username || !data.expirationTime)
  {
    dispatch(authActions.logout());
  } else {
    dispatch(authActions.login({ token: data.token, userId: data.userId }));
  }
  // Making sure the router params are available:
  useEffect(() => {
    if (router && router.query) {
      setAuctionId(router.query.auctionId);
    }
  }, [router]);
  // Getting the list of auctions:
  useEffect(() => {
    fetch(
      "https://diplom-c89f3-default-rtdb.europe-
west1.firebaseio.com/auctions.json"
    )
      .then((response) => response.json())
      .then((data) => {
        if (data !== null) {
          for (const value of Object.values(data)) {
            if (value.auctionId === auctionId) {
              setAuctionInfo(value);
            }
          }
        }
      });
  }, [auctionId]);
  return (
    <Fragment>
      {auctionInfo && <AuctionDetail auction={auctionInfo} />}
    </Fragment>
  );
};
export default Auction;
Auction.getInitialProps = async ({ req }) => {
  const data = parseCookies(req);
  return {
    data: data && data,
  };
};
};

```

Листинг Б.30 – Файл LoginPage.js

```

// our-domain.com/login
import LoginForm from "../../components/authentication/LoginForm";
const LoginPage = () => {
  return <LoginForm />;
};
export default LoginPage;

```

Листинг Б.31 – Файл MyAuctionPage.js

```
// our-domain.com/my-auctions
import { useEffect, useState } from "react";
import { parseCookies } from "../../helpers";
import { useDispatch } from "react-redux";
import { useSelector } from "react-redux";
import { authActions } from "../../store/authSlice";
import Spinner from "../../components/ui/Spinner";
import AuctionList from "../../components/auctions/AuctionList";
import classes from "../../styles/myAuctions.module.css";
const MyAuctions = ({ data }) => {
  const [isLoading, setIsLoading] = useState(false);
  const dispatch = useDispatch();
  const [myAuctions, setMyAuctions] = useState([]);
  const [isEmpty, setIsEmpty] = useState(true);
  // Checking whether there are authentication cookies available or not. If
  // available -> Login. Otherwise -> Logout.
  if (!data.token || !data.userId || !data.username || !data.expirationTime)
  {
    dispatch(authActions.logout());
  } else {
    dispatch(
      authActions.login({
        token: data.token,
        userId: data.userId,
        username: data.username,
      })
    );
  }
  const username = useSelector((state) => state.auth.username);
  let myAuctionsData = [];
  // Getting the list of auctions and filtering userId:
  useEffect(() => {
    setIsLoading(true);
    fetch(
      "https://diplom-c89f3-default-rtdb.europe-
      west1.firebaseio.com/auctions.json"
    )
      .then((response) => response.json())
      .then((data) => {
        if (data !== null) {
          for (const value of Object.values(data)) {
            if (value.owner === username) {
              myAuctionsData.push(value);
            }
          }
          if (myAuctionsData.length !== 0) {
            setIsEmpty(false);
            setMyAuctions(myAuctionsData);
          }
          setIsLoading(false);
        }
      });
  }, [dispatch]);
  return (
    <section className={classes.section}>
      {isLoading && <Spinner />}
      {!isEmpty && <AuctionList auctions={myAuctions} />}
      {isEmpty && <h1>Nothing to show. No auctions were created yet.</h1>}
    </section>
  );
};
```

```

};
export default MyAuctions;
MyAuctions.getInitialProps = async ({ req }) => {
  const data = parseCookies(req);
  return {
    data: data && data,
  };
};
};

```

Листинг Б.32 – Файл MyFavPage.js

```

// our-domain.com/my-favourites
import { useEffect, useState } from "react";
import { useDispatch } from "react-redux";
import { authActions } from "../../store/authSlice";
import { useSelector } from "react-redux";
import { parseCookies } from "../../helpers";
import AuctionList from "../../components/auctions/AuctionList";
import Spinner from "../../components/ui/Spinner";
const Favourites = ({ data }) => {
  const [isLoading, setIsLoading] = useState(false);
  const dispatch = useDispatch();
  const [favourites, setFavourites] = useState([]);
  // Checking whether there are authentication cookies available or not. If
  // available -> Login. Otherwise -> Logout.
  if (!data.token || !data.userId || !data.username || !data.expirationTime)
  {
    dispatch(authActions.logout());
  } else {
    dispatch(authActions.login({ token: data.token, userId: data.userId }));
  }
  const userId = useSelector((state) => state.auth.userId);
  let favouritesIds = [];
  let favouriteAuctions = [];
  // Getting the list of favourites and filtering username:
  useEffect(() => {
    setIsLoading(true);
    fetch(
      "https://diplom-c89f3-default-rtdb.europe-
west1.firebaseio.com/favourites.json"
    )
      .then((response) => response.json())
      .then((data) => {
        if (data !== null) {
          for (const value of Object.values(data)) {
            if (JSON.stringify(value.userId) === userId) {
              favouritesIds.push(value.auctionId);
            }
          }
        }
        // Getting the list of auctions, filtering favourites and adding to
        final array:
        fetch(
          "https://diplom-c89f3-default-rtdb.europe-
west1.firebaseio.com/auctions.json"
        )
          .then((response) => response.json())
          .then((data) => {
            if (data !== null) {
              for (const value of Object.values(data)) {
                if (favouritesIds.includes(value.auctionId)) {
                  favouriteAuctions.push(value);
                }
              }
            }
          })
        }
      }
    }
  }

```

```

        setFavourites(favouriteAuctions);
        setIsLoading(false);
      }
    });
  }
});
}, [userId]);
return (
  <section>
    {isLoading && <Spinner />}
    <AuctionList auctions={favourites} />;
  </section>
);
};
export default Favourites;
Favourites.getInitialProps = async ({ req }) => {
  const data = parseCookies(req);
  return {
    data: data && data,
  };
};
};

```

Листинг Б.33 – Файл NewAuctionPage.js

```

// our-domain.com/new-auction
import { useDispatch } from "react-redux";
import { authActions } from "../../store/authSlice";
import { parseCookies } from "../../helpers";
import NewAuctionForm from "../../components/auctions/NewAuctionForm";
const NewAuctionPage = ({ data }) => {
  const dispatch = useDispatch();
  // Checking whether there are authentication cookies available or not. If
  // available -> Login. Otherwise -> Logout.
  if (!data.token || !data.userId || !data.username || !data.expirationTime)
  {
    dispatch(authActions.logout());
  } else {
    dispatch(authActions.login({ token: data.token, userId: data.userId }));
  }
  return <NewAuctionForm />;
};
export default NewAuctionPage;
NewAuctionPage.getInitialProps = async ({ req }) => {
  const data = parseCookies(req);
  return {
    data: data && data,
  };
};
};

```

Листинг Б.34 – Файл SellCarPage.js

```

// our-domain.com/sell-my-car

import { useDispatch } from "react-redux";
import { authActions } from "../../store/authSlice";
import Link from "next/link";

import Button from "@mui/material/Button";

import { parseCookies } from "../../helpers";

import classes from "../../styles/SellCarPage.module.css";

```

```

const SellCarPage = ({ data }) => {
  const dispatch = useDispatch();

  // Checking whether there are authentication cookies available or not. If
  // available -> Login. Otherwise -> Logout.
  if (!data.token || !data.userId || !data.username || !data.expirationTime)
  {
    dispatch(authActions.logout());
  } else {
    dispatch(authActions.login({ token: data.token, userId: data.userId }));
  }

  return (
    <section className={classes.section}>
      <h1>
        Если вы хотите продать автомобиль, вам сначала нужно создать аккаунт
      </h1>
      <Link href="/sign-up">
        <Button className={classes.button} variant="contained">
          Зарегистрироваться
        </Button>
      </Link>
      <h1>Уже есть аккаунт?</h1>
      <Link href="/login">
        <Button className={classes.button} variant="contained">
          Войти
        </Button>
      </Link>
    </section>
  );
};

export default SellCarPage;

SellCarPage.getInitialProps = async ({ req }) => {
  const data = parseCookies(req);

  return {
    data: data && data,
  };
};

```

Листинг Б.35 – Файл SingUp.js

```

// our-domain.com/sign-up
import SignupForm from "../../components/authentication/SignupForm";
const SignupPage = () => {
  return <SignupForm />;
};
export default SignupPage;

```

Листинг Б.36 – Файл App.js

```

import Script from "next/script";
import { Provider } from "react-redux";
import { CookiesProvider } from "react-cookie";
import store from "../../store/index";
import Layout from "../../components/layout/Layout";
import "../../styles/globals.css";
function MyApp({ Component, pageProps }) {
  return (
    <CookiesProvider>

```

```

    <Provider store={store}>
      <Layout>
        { /* Global Site Tag (gtag.js) - Google Analytics */ }
        <Script
          strategy="lazyOnload"
          src={"https://www.googletagmanager.com/gtag/js?id=G-38WZF6734L"}
        />
        <Script id="gtag-init" strategy="lazyOnload">
          {`
            window.dataLayer = window.dataLayer || [];
            function gtag(){dataLayer.push(arguments);}
            gtag('js', new Date());
            gtag('config', 'G-38WZF6734L');`}
          </Script>
          <Component {...pageProps} />
        </Layout>
      </Provider>
    </CookiesProvider>
  );
}
export default MyApp;

```

Листинг Б.37 – Файл index.js

```

// our-domain.com/
import { useEffect, useState } from "react";
import { useDispatch } from "react-redux";
import { authActions } from "../store/authSlice";
import { parseCookies } from "../helpers/";
import filteringFunction from "../helpers/filteringFunction";
import AuctionList from "../components/auctions/AuctionList";
import Spinner from "../components/ui/Spinner";
import HomepageImage from "../components/ui/HomepageImage";
import Filter from "../components/filters/Filter";
const HomePage = ({ data }) => {
  const [isLoading, setIsLoading] = useState(false);
  const [isEmpty, setIsEmpty] = useState(false);
  const [filteredList, setFilteredList] = useState([]);
  const [status, setStatus] = useState("");
  const [priceRange, setPriceRange] = useState("");
  const dispatch = useDispatch();
  // Checking whether there are authentication cookies available or not. If
  // available -> Login. Otherwise -> Logout.
  if (!data.token || !data.userId || !data.username || !data.expirationTime)
  {
    dispatch(authActions.logout());
  } else {
    dispatch(authActions.login({ token: data.token, userId: data.userId }));
  }
  // Getting the list of all auctions from database:
  useEffect(() => {
    setIsLoading(true);
    fetch(
      "https://diplom-c89f3-default-rtdb.europe-
west1.firebaseio.com/auctions.json"
    )
      .then((response) => response.json())
      .then((data) => {
        let list = [];
        // Storing response values in a temporal list for later filtering
        if (data != null) {
          for (const value of Object.values(data)) {
            list.push(value);
          }
        }
      })
  }

```

```

    }
    // Filtering results by auction state and price range
    const filtered = filteringFunction(status, priceRange, list);
    //Checking if the filtered list is empty
    if (filtered.length === 0) {
        setIsEmpty(true);
    } else {
        setIsEmpty(false);
    }
    // Storing filtered list in a state:
    setFilteredList(filtered);
    // Setting isLoading to false so that the loading spinner is not
rendered anymore
        setIsLoading(false);
    });
}, [status, priceRange]);
return (
    <section>
        <HomepageImage />
        <Filter
            status={status}
            setStatus={setStatus}
            priceRange={priceRange}
            setPriceRange={setPriceRange}
        />
        {isLoading && <Spinner />}
        {isEmpty && <h1 id="noResults">С этими фильтрами ничего не
найдено</h1>}
        <AuctionList auctions={filteredList} />
    </section>
);
};
export default HomePage;
HomePage.getInitialProps = async ({ req }) => {
    const data = parseCookies(req);
    return {
        data: data && data,
    };
};
};

```

ПРИЛОЖЕНИЕ В
ОПИСАНИЕ СУЩНОСТЕЙ
(рекомендуемое)

В таблице В.1 представлено описание атрибутов сущности «Лот».

Таблица В.1 – Описание атрибутов сущности «Лот»

Наименование	Описание	Тип данных	Размерность	Область допустимых значений	Возможность значения NULL	Роль	Примечание	Примечание
Идентификатор	Уникальный номер актера в таблице	Целый	4	1 – 9999	Нет	PK	1553	–
Идентификатор ставки	Уникальный номер актера в таблице	Целый	4	1 – 9999	Нет	FK	1553	–
Идентификатор пользователя	Уникальный номер актера в таблице	Целый	4	1 – 9999	Нет	FK	1553	–
Идентификатор автомобиля	Уникальный номер актера в таблице	Целый	4	1 – 9999	Нет	FK	1553	–
Начальная цена	Начальная цена лота	Целый	8	1 – 99999999	нет			
Дата начала	Дата начала аукциона	Дата						
Дата конца	Дата конца аукциона	Дата						

В таблице В.2 представлено описание атрибутов сущности «Ставка».

Таблица В.2 – Описание атрибутов сущности «Ставка»

Наименование	Описание	Тип данных	Размерность	Область допустимых значений	Возможность значения NULL	Роль	Примечание	Примечание
Идентификатор	Уникальный номер ставки в таблице	Целый	4	1 – 9999	Нет	PK	1553	–
Идентификатор пользователя	Уникальный номер актера в таблице	Целый	4	1 – 9999	Нет	FK	1553	–
Текущая ставка	Текущая ставка на лот	Целый	8	1 – 99999999	нет			

В таблице В.3 представлено описание атрибутов сущности «Пользователь».

Таблица В.3 Описание атрибутов сущности «Пользователь»

Наименование	Описание	Тип данных	Размерность	Область допустимых значений	Возможность значения NULL	Роль	Пример	Примечание
Идентификатор	Уникальный пользователь в таблице	Целый	4	1 – 9999	Нет	PK	1553	–
Логин	Логин пользователя	Символьный	100	Любые символы	Нет	–		–
Пароль	Пароль пользователя	Символьный	100	Любые символы	Нет	–		–
Email	Электронная почта пользователя	Символьный	100	Любые символы	Нет	–		–

В таблице В.4 представлено описание атрибутов сущности «Автомобиль».

Таблица В.4 Описание атрибутов сущности «Автомобиль»

Наименование	Описание	Тип данных	Размерность	Область допустимых значений	Возможность значения NULL	Роль	Пример	Примечание
Идентификатор	Уникальный номер автомобиля в таблице	Целый	4	1 – 9999	Нет	PK	1553	–
Модель	Название модели автомобиля	Символьный	100	Любые символы	Нет	–	3+16+	–
Описание	Описание автомобиля	Символьный	1000	Любые символы	Нет	–	3+16+	–

В таблице В.5 представлено описание атрибутов сущности «Избранное».

Таблица В.5 Описание атрибутов сущности «Избранное»

Наименование	Описание	Тип данных	Размерность	Область допустимых значений	Возможность значения NULL	Роль	Пример	Примечание
Идентификатор	Уникальный номер избранного в таблице	Целый	4	1 – 9999	Нет	PK	1553	–
Идентификатор пользователя	Уникальный номер пользователя в таблице	Целый	4	1 – 9999	Нет	FK	1553	–
Идентификатор автомобиля	Уникальный номер автомобиля в таблице	Целый	4	1 – 9999	Нет	FK	1553	–

ПРИЛОЖЕНИЕ Г
РЕЗУЛЬТАТЫ ТЕСТИРОВАНИЯ
(рекомендуемое)

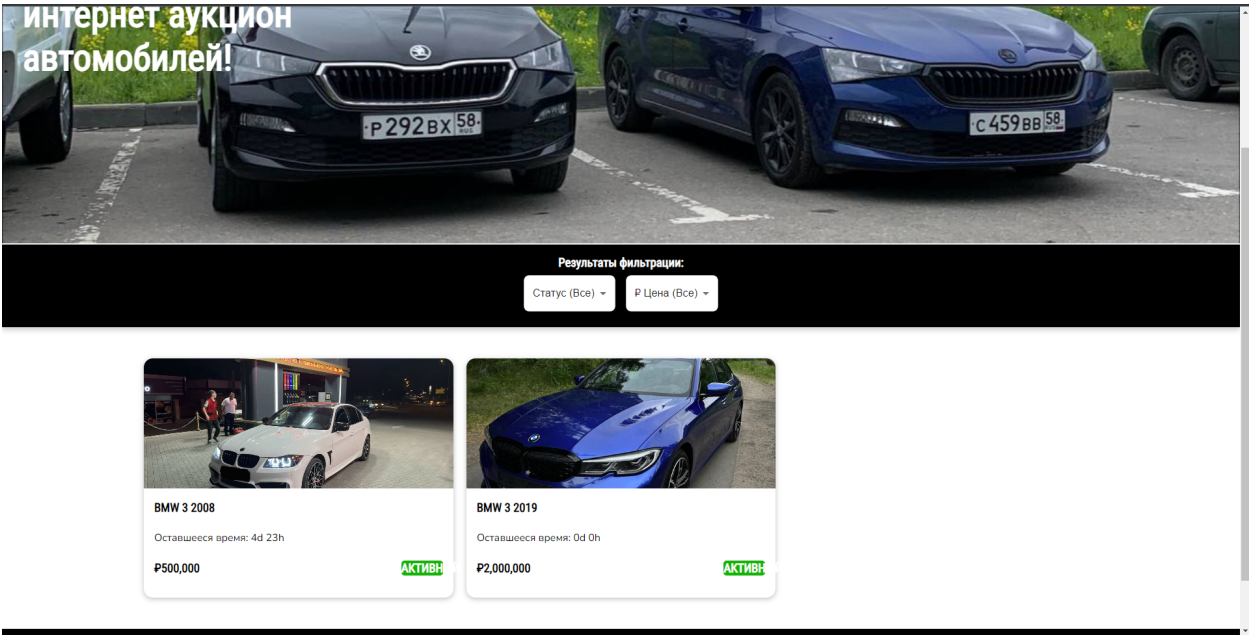


Рисунок Г.1 – Результат теста №1

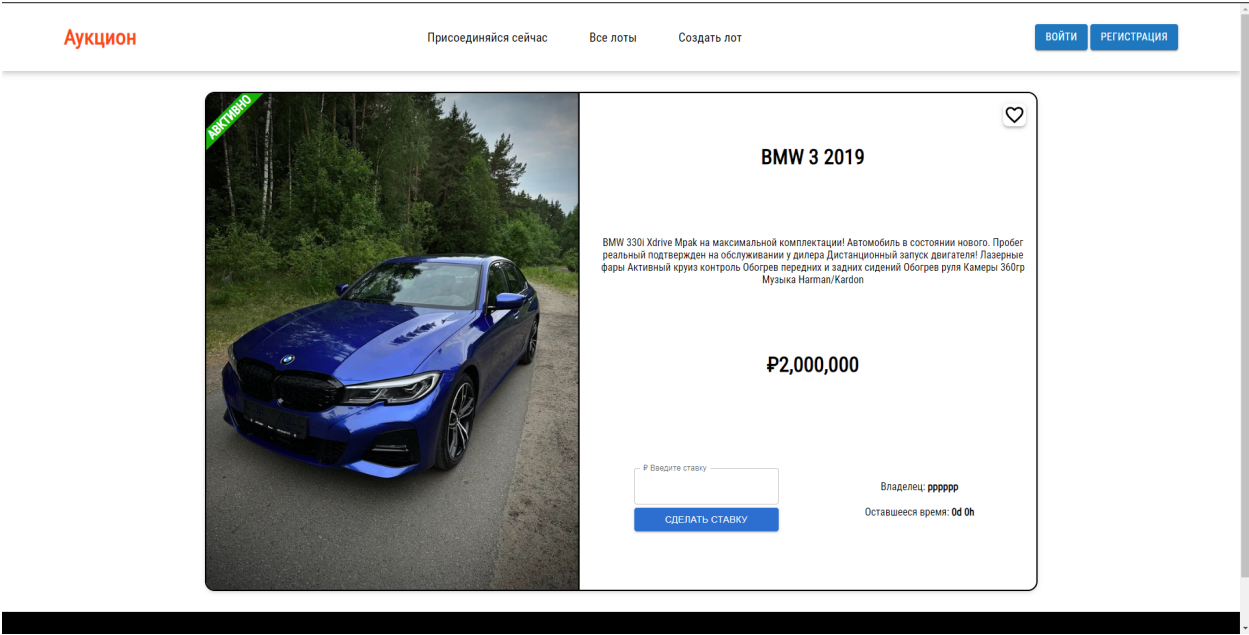


Рисунок Г.2 – Результат теста №2

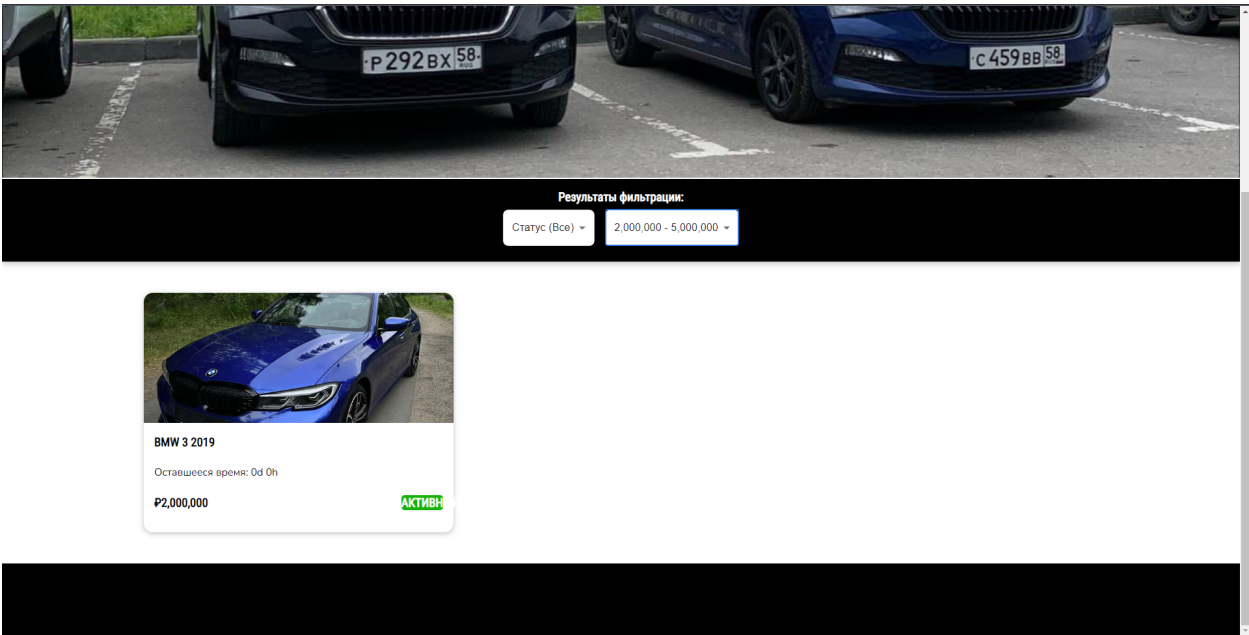


Рисунок Г.3 – Результат теста №3

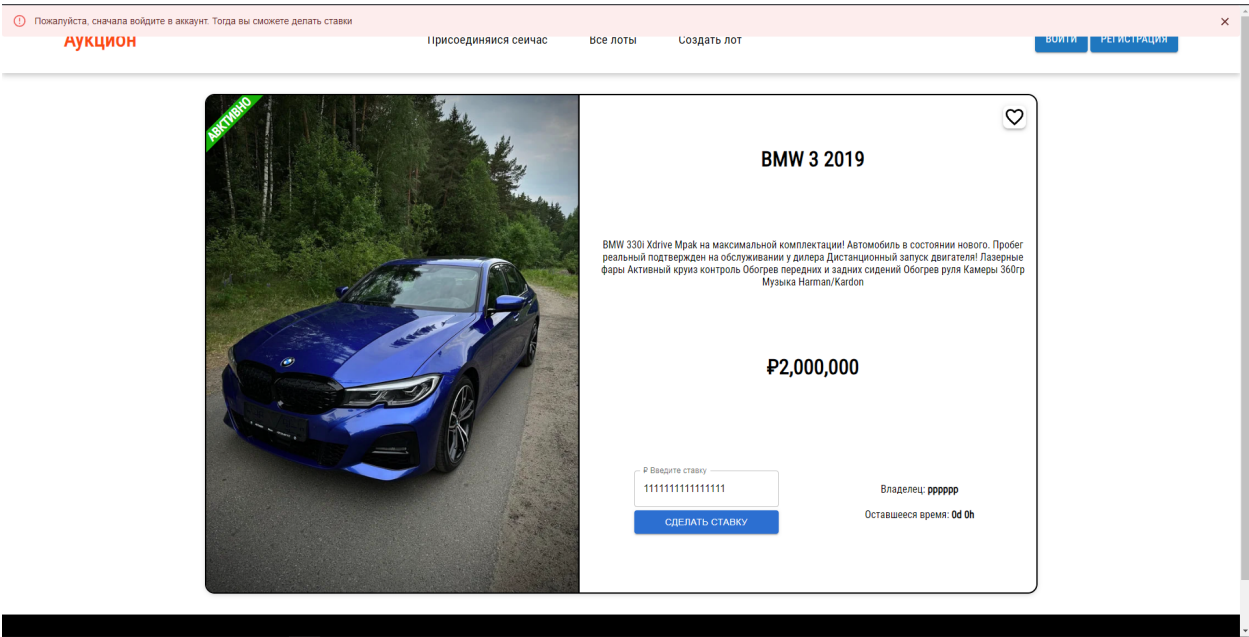


Рисунок Г.4 – Результат теста №4

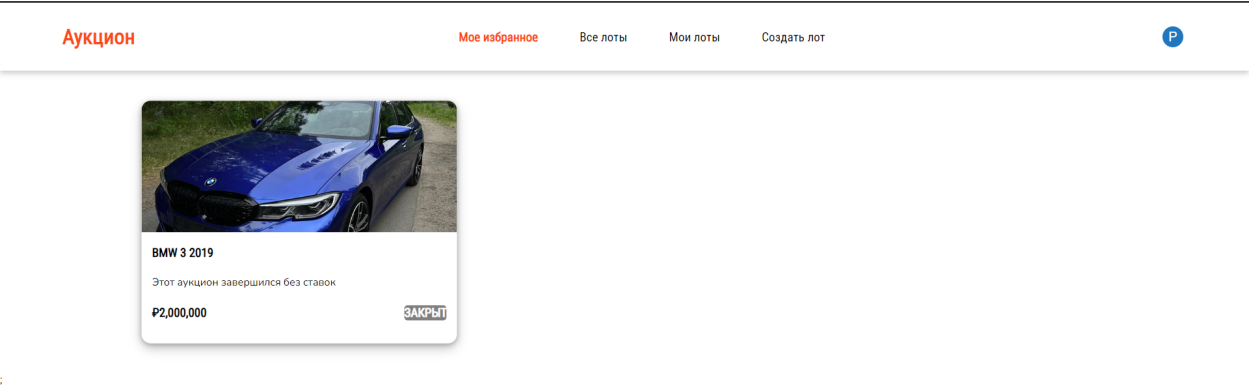


Рисунок Г.5 – Результат теста №5

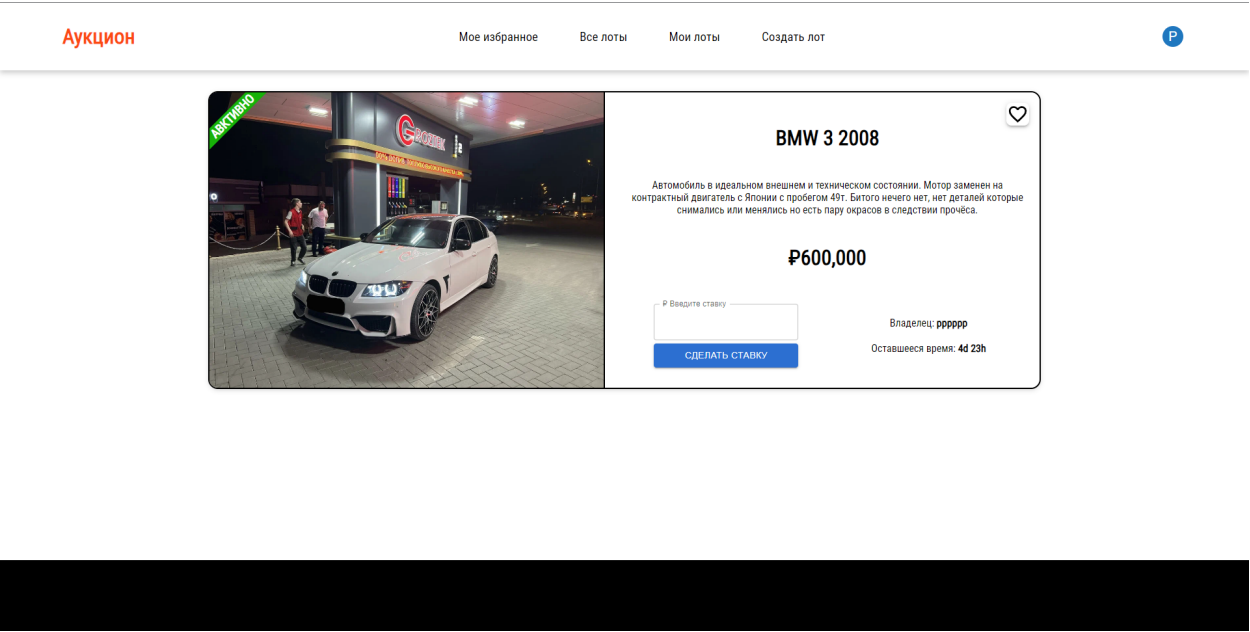


Рисунок Г.6 – Результат теста №6

Аукцион

Мое избранное

Все лоты

Мои лоты

Создать лот

Р

Активно



Skoda Rapid 2020

в хорошем состоянии

₽1,500,000

Введите ставку

СДЕЛАТЬ СТАВКУ

Владелец: rrrrrr

Оставшееся время: 21d 23h

Рисунок Г.7 – Результат теста №7

Аукцион

Присоединяйся сейчас

Все лоты

Создать лот

ВОЙТИ

РЕГИСТРАЦИЯ

Если вы хотите продать автомобиль, вам сначала нужно создать аккаунт

ЗАРЕГИСТРИРОВАТЬСЯ

Уже есть аккаунт?

ВОЙТИ

Рисунок Г.8 – Результат теста №8