Аннотация

В выпускной квалификационной работе был рассмотрен процесс создания Информационной системы по определения местонахождения звукового оборудования для сетевой арендной организации. В ходу создания были использованы редактор кода кода Visual Studio Code и язык JavaScript с использованием технологии Node.js и React. В процессе разработки были пройдены все этапы от выявления требований до реализации и тестирования приложения.

Ключевые слова: программное приложение, React, NodeJS, ПО, интерфейс, CSS, PostgreSQL.

Работа состоит из 53 страниц и 4 разделов.

Работа содержит 12 рисунков, 8 формул, 8 таблиц, 18 литературных источников.

Оглавление

Введение 6
1. Исследовательский раздел 8
1.1. Анализ предметной области 8
1.2. Модель организации процесса аренды 10
1.3. Анализ информационных процессов 11
1.4. Анализ и сравнение с аналогами 13
1.5. Вывод к главе 17
2. Аналитический раздел 18
2.1. Разработка диаграмм и структур для разрабатываемого программного продукта 18
2.2. Проектирование бизнес-процессов и алгоритмов 19
2.3. Вывод к главе 22
3. Экономический раздел 23
3.1. Организация и планирование работ по теме 23
3.2. Расчёт стоимости проведения работ 25
4. Технологический раздел 29
4.1. Обоснование выбора средств для реализации. 29
4.2. Описание разработанного программного средства. 30
4.3. Тестирование программного продукта. 33
4.4. Вывод к главе 34
Заключение 36
Список использованных источников 37
Приложение 1 39
Приложение 2 43

Введение

В настоящий момент фирмы аренды и проката разного рода предметов, от квартир, машин и заканчивая музыкальной аппаратурой и банальными дисками от игр, начинают свой подъем. Для небольших и средних компаний в условиях, когда оборудование начинает стоить крайне дорого, целесообразнее его взять в аренду, чем тратиться на его приобретение.

Прокатное оборудование пользуется большим спросом. Это происходит по многим причинам. Вот некоторые из них:

– нужно сделать одноразовую работу и для нее требуется дорогостоящее профессиональное оборудование;

– прокат позволит не задумываться о возможных поломках оборудования, которые приводят к обращению в ремонтные мастерские;

– при выполнении какой-либо работы прокат даст возможность спланировать свои расходы;

– собственное оборудование было сдано на длительный ремонт, а оно постоянно необходимо для работы.

Основной задачей разрабатываемого веб-сервиса является упростить процесс учета и контроля предметов, которые сдают в аренду арендодатели. Кроме того, сервис должен помочь арендодателю следить за тем, в каком месте находится его оборудование в данный момент или его предыдущие места прибывания.

Актуальность обусловлена тем, что в текущее нестабильное время программного обеспечения в данной сфере не много, а те, что есть, могут в любой момент уйти с рынка. Чтобы обезопасить рынок и создать конкуренцию и разрабатывается данное веб-приложение.

Целью данного исследования является разработка Веб-сервис определения местонахождения звукового оборудования для сетевой организации

Для достижения поставленной цели необходимо решить следующие задачи:

– провести анализ бизнеса по аренде движимого оборудования с точки зрения учета его местонахождения;

– провести анализ и сравнение имеющихся на рынке средств автоматизации системы хранения;

– выявить основные требования к функционалу разрабатываемого решения;

– разработать Веб-сервис определения местонахождения звукового оборудования для сетевой организации

– провести экономический анализ решения.

1. Исследовательский раздел

1.1. Анализ предметной области

Арендный бизнес, на текущий момент, одно из самых перспективных направлений инвестиций. С одной стороны, это инвестиции в недвижимость, сдача территории в аренду может приносить большой доход и при этом не требующий особых временных и ресурсных затрат.

Кроме того, есть инвестиции в технику и оборудование. Такие инвестиции требуют больших временных и ресурсных затрат, так как появляется необходимость содержать склад, следить за состоянием товара, который в дальнейшем будет сдаваться в аренду, проводить взаимодействие с клиентом, например доставку арендуемого товара клиенту, помощь с использованием и привоз товара обратно.

Использование современных информационных технологий может значительно облегчить работу организаций, занимающихся прокатным оборудованием. Особенно, если это специализированное программное обеспечение для определенной области прокатных инструментов.

Для последнего типа аренды и будет разрабатываться программное средство.

Основными в рассматриваемой предметной области являются следующие составляющие:

– система управления складом;

– система взаимодействия с клиентом;

– контроль арендуемого товара;

Система управления складом – информационная система, обеспечивающая автоматизацию управления бизнес-процессами складской работы профильного предприятия.

Архитектура автоматизированной информационной системы управления складом построена по трехуровневому принципу [3].

Первый компонент представляет собой видимую для пользователя часть – интерфейс типа «человек-машина» – «клиентское приложение», с помощью

которого пользователь осуществляет ввод, изменение и удаление данных, дает запросы на выполнение операций и запросы на выборку данных (получение отчетов); этот компонент может быть доступен на компьютере, ТСД, планшете, смартфоне [3].

Второй компонент (скрытая от пользователей часть системы) – сервер базы данных, осуществляет хранение данных. Пользователь через клиентское приложение запрашивает, вводит, изменяет или удаляет данные в базе данных (БД) [3].

Третий компонент – бизнес-логика осуществляет инициированную пользователем обработку данных, и возвращает обработанные данные из БД, сообщая пользователю через экран клиентского приложения о завершении запрошенной обработки [3].

Так же поскольку одной основных особенностей фирм проката является доставка оборудования к месту аренды, необходимо учитывать компонент логистики. В случае фирмы проката наиболее интересны лишь несколько аспектов логистики это распределительная логистика, складская логистика и транспортная логистика.

Таблица 1.1 – Сферы логистики

Сферы логистики	Описание
1	2
Распределительная логистика	Включает в себя оптимизацию сбыта и физического распределения доступных запасов материалов. Также распределительная логистика управляет перевозками, складированием и другими операциями разного характера, совершаемая в процессе поступления готовой продукции до потребителя. К процессам распределения относятся прогнозирование спроса, обработка заказов, управление запасами, хранение на складе и обслуживание запасов, транспортировка [5, с. 295-297; 6, с. 154-156].
Транспортная логистика	Отвечает за управление транспортировкой грузов, конкретнее за операции перемещения и промежуточного хранения товаров из мест хранения в места потребления с использованием транспортных средств. Транспортная логистика включает в себя задачи создания транспортных систем, выбора вида транспортного средства и способа транспортировки, рационализации транспортного процесса и составления оптимальных маршрутов доставки [6, с. 225-228].

Продолжение таблицы 1.1

Складская логистика	Включает в себя деятельность, связанную с выполнением комплекса мероприятий по совершенствованию процессов грузопереработки и перемещения грузов внутри склада и при доставке клиентам, включая организацию труда работников склада. Складскую логистику трактуют, как деятельность по планированию, организации и осуществлению приемки и хранения различных материальных ценностей, подготовки их к производственному потреблению и распределению между потребителями при наличии информационной системы управления складскими потоками [5, с. 266-267; 5, с. 242].

1.2. Модель организации процесса аренды

На рисунке 1 представлена типовая структурная схема для организации аренды в общем случае с учетом того, что арендуемый товар является движимым (аппаратура, автомобиль и т.д.).

Исходя из структурной схемы организации аренды можно выделить следующие типы взаимодействий:

– **взаимодействие с клиентом.** Здесь происходит общение с клиентом и выяснение необходимых деталей об аренде, например на какой срок собираются провести аренду, сколько и какое оборудование необходимо и расчет итоговой стоимости;

– **взаимодействие арендодателя с складской командой.** В данном этапе Арендодатель связывается с складом и передает информацию о заявке. Склад в свою очередь собирает команду монтажников (если они необходимы) и готовит все необходимое для перевозки. Собирает оборудование к необходимому сроку и передает клиенту. По истечению срока аренды Склад собирает оборудование и перевозит его обратно на склад, после чего докладывает арендодателю о состоянии арендуемого оборудования.

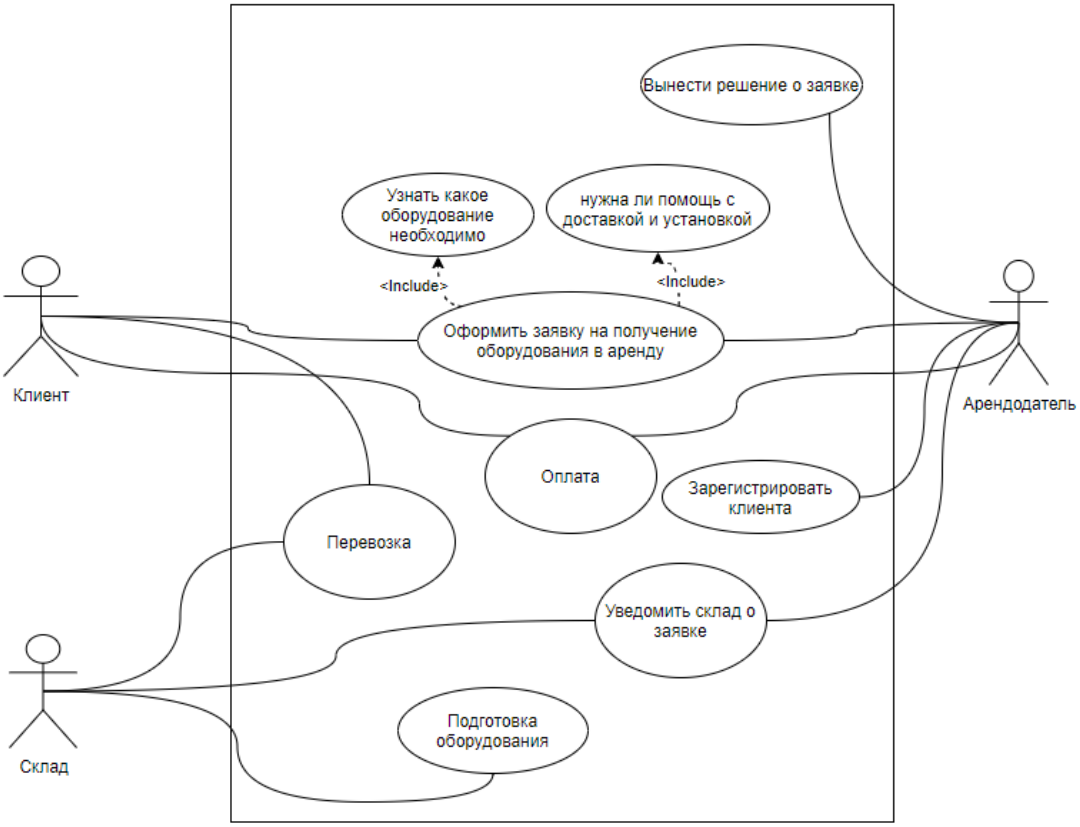

Рисунок 1.1 – Структурная схема организации аренды

Кроме этого, происходит еще одно взаимодействие между складской командой и клиентом. Здесь происходит неформальное взаимодействие, касающиеся организационных моментов, например вопрос о том, куда поставить оборудования, как и где его лучше подключить. Здесь же команда узнает примерное время окончания мероприятия, и возможность времени начала сборки оборудования для дальнейшей перевозки его обратно на склад или на следующую площадку.

1.3. Анализ информационных процессов

В ходе анализа предметной области было выявлено, что фирма по аренде оборудования зачастую самостоятельно собирает, вывозит и обслуживает оборудование на месте аренды.

На основе этой информации была составлена схема бизнес-процессов, связанная с процессом аренды. Схема представлена на рисунке 1.2.

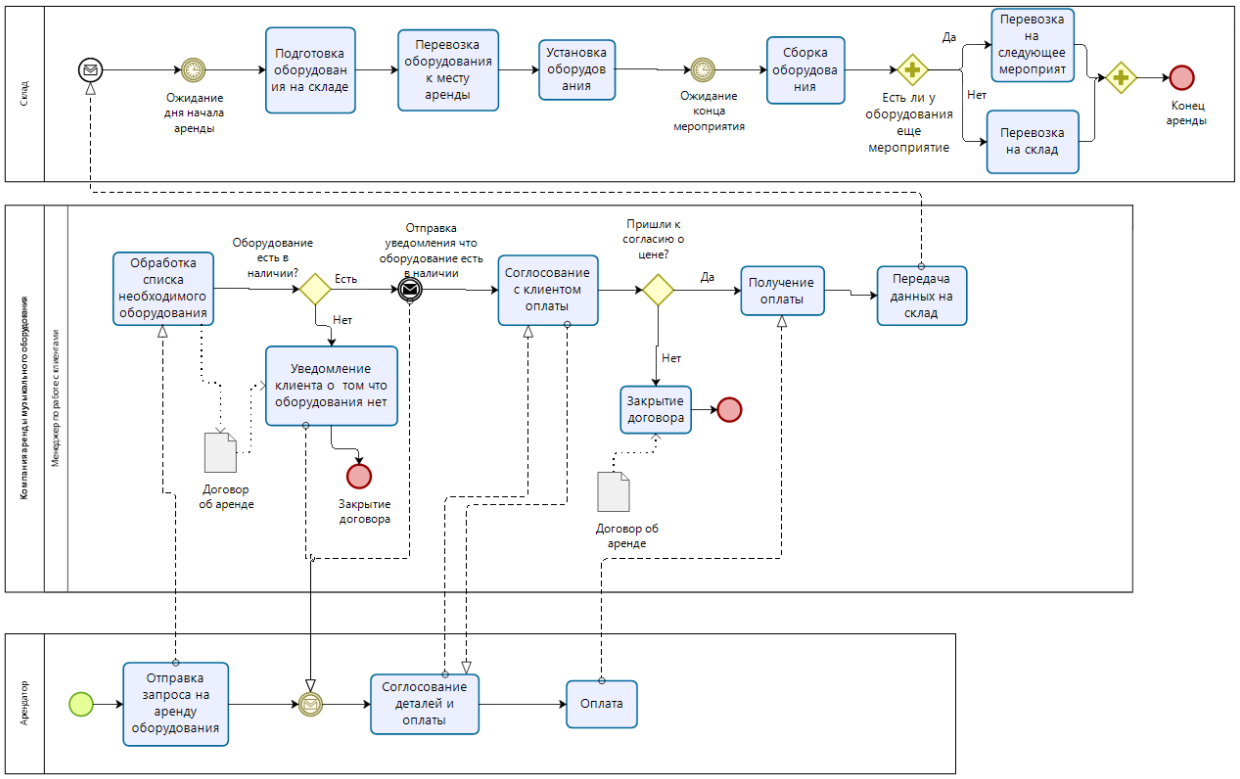

Рисунок 1.2 – Схема бизнес-процессов компании аренды оборудования

Как можно видеть на схеме, процесс регистрации заявки начинается с инициативы клиента. Создается профиль клиента в базе данных и открывается заявка. Затем проверяется смотрит есть ли оборудование, которое запрашивает клиент, у компании и доступно ли оно в день, выбранный клиентом.

Если оборудования нет, то клиенту отправляется соответствующее уведомление, а если есть, то с клиентом происходит диалог о согласовании цены. Если в цене сойтись не вышло, то заявка клиента закрывается.

Если все прошло успешно и в цене с клиентом сошлись, то ожидаем оплаты и передаем заявку на склад.

Когда заявка получена складом, она откладывается до дня аренды. В день аренды происходит сборка оборудования на складе и его проверка на работоспособность. Затем оно грузится в машину и отвозится к месту аренды.

Там оборудование разгружают и подготавливают к использованию. После того как оборудование успешно подготавливается, специалисты жду окончания мероприятия и собирают оборудование для дальнейшей перевозки его к другой точке аренды или на склад.

1.4. Анализ и сравнение с аналогами

1.4.1. Оформление сравнительной таблицы характеристик программных продуктов, по функциональным возможностям схожих с разрабатываемым программным продуктом

Основная система, которая будет разрабатываться – это складская система. Основным отличительным элементом этой системы планируется сделать возможность отслеживать место, куда было отправлено оборудование, с сохранением истории.

Возьмем аналогичные системы и рассмотрим их функционал. Для сравнения было выбрано 3 программных средства: «МойСклад», «RENT IN HAND» и «WS.Автопрокат».

Таблица 1.2 – Исследование аналогов разрабатываемой ИС

Критерии\ Аналоги	Описание	«Мой Склад»	«RENT IN HAND»	«WS.Автопрокат»
Наглядность отображения информации	Программное обеспечение должно иметь интуитивно понятный интерфейс, на котором должна находиться информация, которую легко интерпретировать.	-	+	+
Систематизированный документооборот	Документы могут создаваться автоматически или заполняться в соответствии с образцом. Доступ к документам зависит от уровня доступа персонала.	+	+	+
Гибкие функциональные возможности	Программное обеспечение должно обладать функционалом, которые можно не использовать и/или использоваться не на постоянной основе.	+	+	-
Возможность посмотреть местоположение прокатного оборудования	Программное средство должно предоставлять возможность отследить местоположение оборудования на карте.	-	+	+
Автоматический расчет цен на прокатный товар	Цены на прокатное оборудование должно рассчитываться в зависимости от изначальной цены товара и учитывая сложность установки его на точке аренды.	-	+	+

«Мой Склад» для поддержки склада предлагает следующее:

– товароучетная система – управление товарами и ценами;

– составление отчетов;

– инвентаризации;

– создание заказов и их редактирование.

В целом система обладает обширным функционалом для фирм, которые хранят товар на складах, а затем реализовывает их, продавая на полках в магазинах.

«RENT IN HAND» – программное обеспечение, которое создано для разных типов прокатного бизнеса. В этом программном продукте предлагается следующее:

– контроль и учет операций с товарами и клиентами;

– отчетность;

– расчет цен на аренду и штрафы.

Система сбалансирована под то, чтобы можно было грамотно взаимодействовать с клиентом в сфере аренды оборудования.

«WS.Автопрокат» – это специально спроектированная система под аренду автомобилей. Несмотря на специфичность области применения, аренда происходит движимого объекта, а значит включение в список сравнения корректен.

Из основных функций выделяются:

– карточки пользователи;

– интеграция с картами;

– финансовый учет.

Самостоятельно проведенное исследование информационных систем организаций в сети Интернет, показало наличие нескольких сайтов с похожим функционалом, но точных аналогов не найдено. И та, и другая система в плане взаимодействия с складом имеет свои положительные черты. Система, которая должна получится в ходе выполнения данной дипломной работы, должна объединить положительные стороны вышеуказанных систем.

Исходя из проведенных исследований конечный предполагаемый функционал должен выглядеть следующим образом.

Функции незарегистрированных пользователей:

– просмотр спецоборудования;

– поиск спецоборудования;

– формирование заявки;

– отправка заявки;

– возможность регистрации;

Функционал зарегистрированных пользователей:

– просмотр спецоборудования;

– поиск спецоборудования;

– формирование заявки;

– отправка заявки;

– авторизация;

Функции администратора:

– просмотр заявок;

– добавление нового спецоборудования;

– добавление технических характеристик спецоборудования;

– составление отчетов о прибыли и количестве продаж;

В целом все функции можно разделить на 4 категории:

– **товароучетная система**. Система должна учесть содержать в себе большую часть функций складского программного средства, для того чтобы арендодатель знал обо всем, что происходит на складе;

– **отчетность**. Арендодатель должен иметь возможность создавать отчеты о доходах с продаж и о товаре, который находится на складе. Отчеты должны быть максимально простыми, но наиболее понятными и содержательными;

– **сохранение истории проката оборудования**. Должна вестись запись истории проката оборудования. Арендодатель должен иметь возможность знать историю сдачи оборудования в течении не менее месяца. Это нужно для того,

чтобы можно было вернуться на место аренды в случае потери, какой-то части оборудования или в случае непредвиденной ситуации;

– **контроль и учет операций, связанных с клиентами**. Информация о клиенте должна сохраняться в системе, чтобы было видно историю аренды клиента, для расчета коэффициента доверия клиенту.

1.4.2. Выбор технологии разработки реализуемого программного продукта исходя из данных анализа аналогов

На сегодняшний день существует большое количество разнообразных методологий разработки информационных систем. Выбор методологии зависит от специфики проекта, например, от таких показателей как размер проекта, количество разработчиков, присутствие четких требований к системе и тд.

Реализуемая в рамках исследования система отслеживания местонахождения звукового оборудования, состоит из каркаса и дополнительных функциональных модулей. Каркас представляет собой набор методов для отслеживания состояния склада и графический интерфейс программы. Дополнительные модули состоят из интеграции картографической системы и системы создавания автоматических отчетов и графиков.

При разработке было принято решение пользоваться методологией «Инкрементная модель». Этому способствовало то, что задача была поставлена ясно и понятно, но некоторые моменты могли уточняться и меняться, например, формат вывода данных в отчетах, формат предоставление графиков.

Основная идея, являющаяся базовой в инкрементной модели, состоит в том, что программное средство разрабатывается по принципу приращений, так, чтобы можно было использовать данные, полученные при разработке более ранних версий программного средства. Таким образом разработчик получает новые данные, как и в процессе разработки, так и в процессе использования. Ключевые этапы этого процесса – простая реализация подмножества требований к программе и совершенствование модели в серии последовательных релизов до тех пор, пока не будет реализовано ПО во всей полноте [9].

В ходе каждой итерации организация модели изменяется, и к ней добавляются новые функциональные возможности.

1.5. Вывод к главе.

В ходе исследований произведен анализ текущей предметной области, позволивший выявить основные характеристики, которые должны быть у разрабатываемого программного продукта.

Также было произведен сравнительный анализ конкурирующих продукций, в ходе которого были добавлены некоторые пункты в особенности разрабатываемого программного продукта.

2. Аналитический раздел

Поскольку разрабатываемое программное средство является веб-приложением, архитектура у него будет клиент-серверная, при этом клиент будет «тонким», то есть все действия по обработке данных будет происходить на сервере. Это позволяет снизить требования к клиентским устройствам и сделать программное средство более распространённым.

2.1. Разработка диаграмм и структур для разрабатываемого программного продукта

2.1.1. Разработка структуры базы данных

Для простоты взаимодействия сервера и СУБД, было принято решение, под каждый элемент данных выделить свою таблицу. В итоге получилось 7 таблиц.

На рисунке 2.1 изображена ERD диаграмма базы данных.

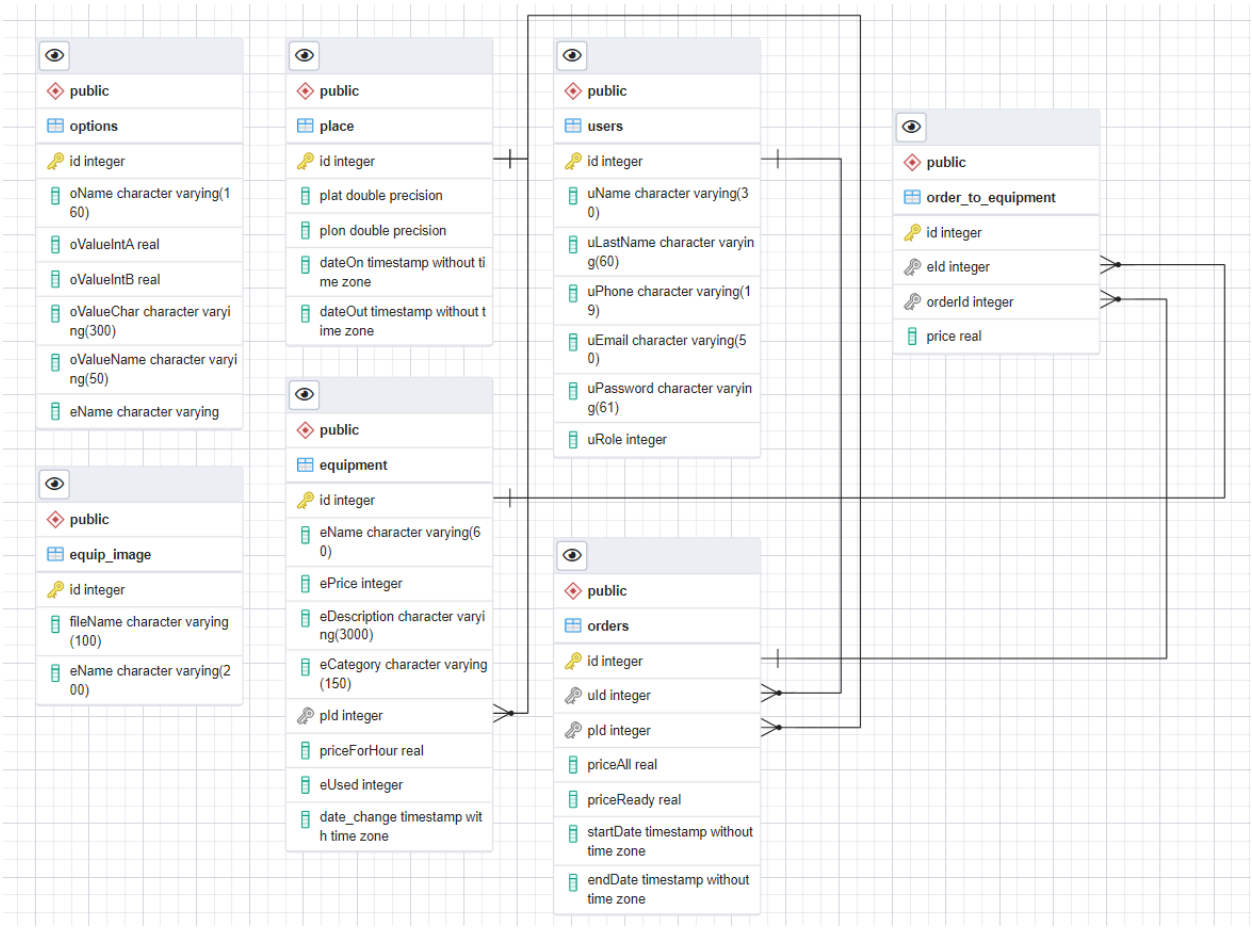

Рисунок 2.1 – ERD диаграмма базы данных

2.1.2. Разработка диаграммы развертывания

На рисунке 2.2 представлена диаграмма развертывания разрабатываемого программного средства.

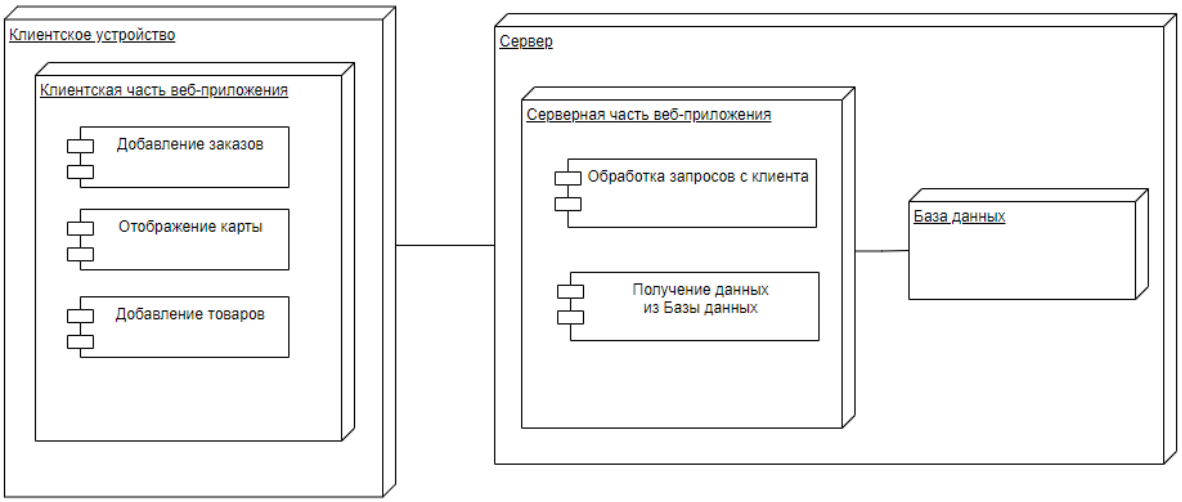

Рисунок 2.2 – Диаграмма развертывания программного средства

Поскольку архитектура программного средства является клиент-серверной, то серверная часть может находится на разных устройствах с клиентской частью средства. Сервер в данном случае отвечает за обработку, хранение и выдачу данных, а клиентское устройство, являющееся в «тонким» клиентом, отправляет, принимает и выводит данные в понятном пользователю формате.

2.2. Проектирование бизнес-процессов и алгоритмов

2.2.1. Проектирование Бизнес-процессов в нотации ARIS

В качестве основной нотации для изображения бизнес-процесс в нотации ARIS. На рисунках 2.3, 2.4 и 2.5 представлены бизнес-процессы, нарисованные в данной нотации.

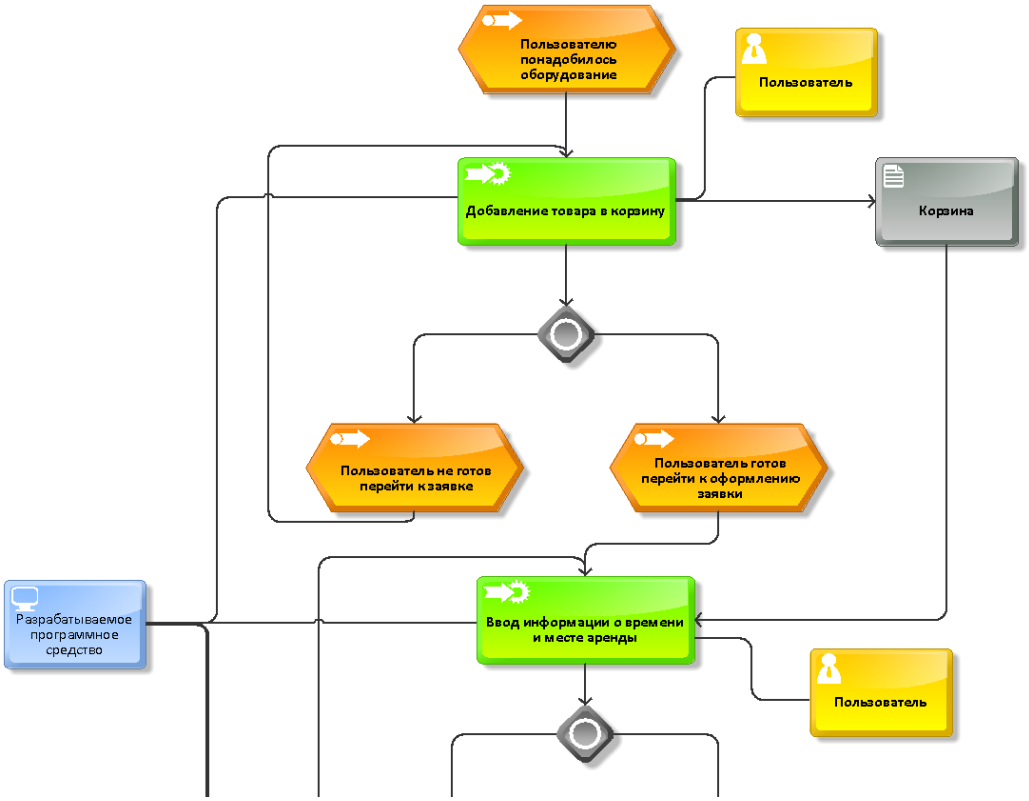

Рисунок 2.3 – Бизнес-процессы в нотации ARIS. Часть 1

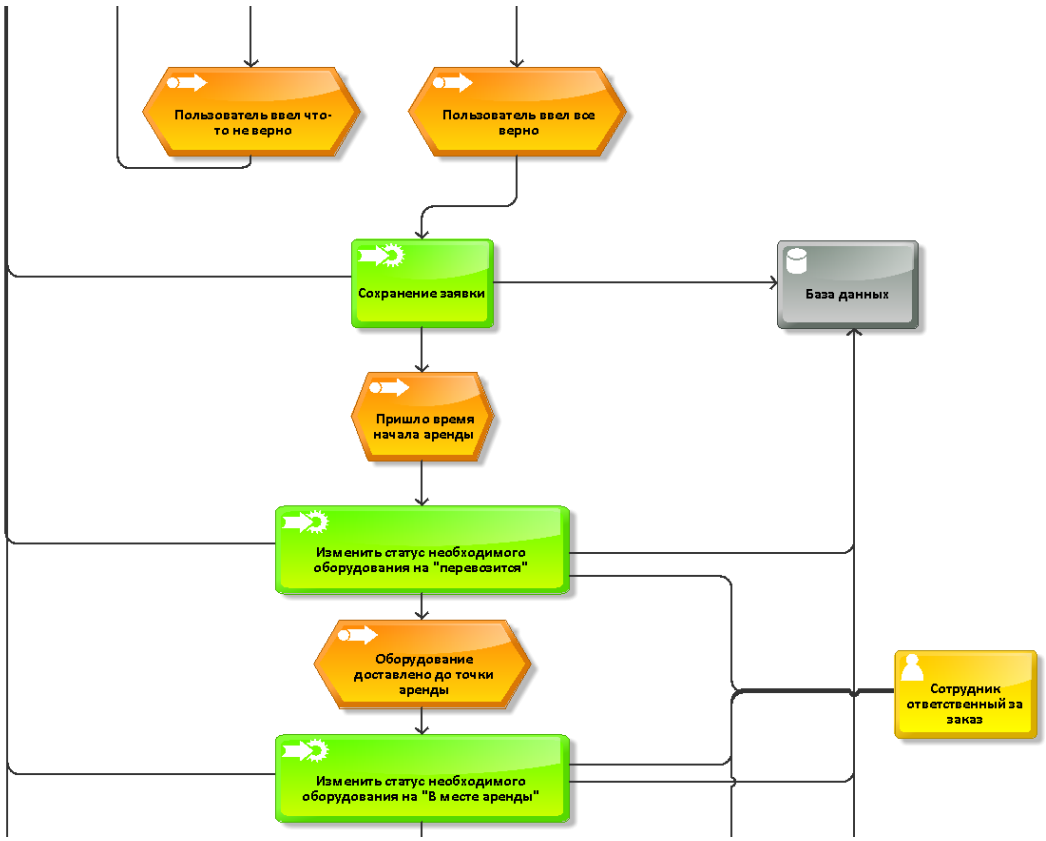

Рисунок 2.4 – Бизнес-процессы в нотации ARIS. Часть 2

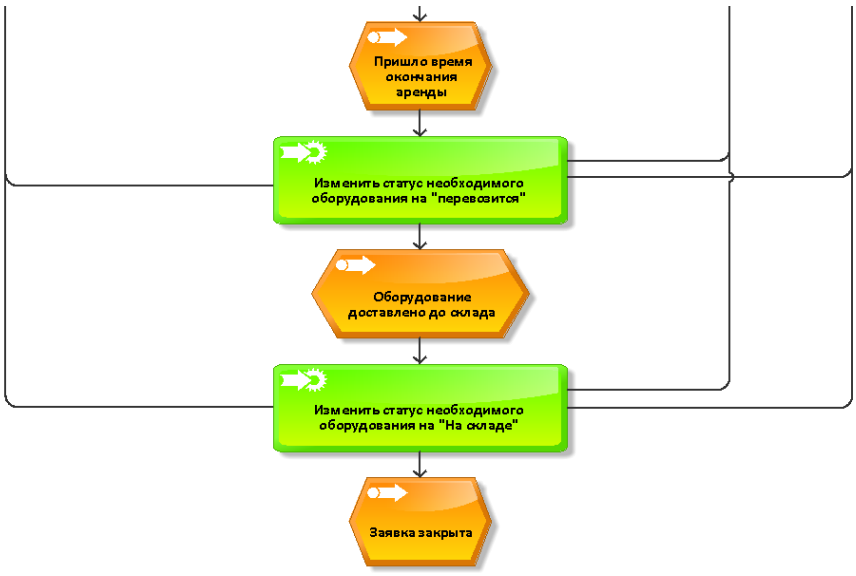

Рисунок 2.5 – Бизнес-процессы в нотации ARIS. Часть 3

Схема начинается с желания пользователя взять оборудование в аренду. Пользователь выбирает оборудование и добавляет его в корзину. Пользователь может добавить еще оборудования в корзину если он хочет.

Далее пользователь вводит необходимую информацию об аренде и если он ввел все правильно, то заявка сохраняется в базе данных.

Затем ожидается дата начала заказа, сотрудник компании, за которым закреплен заказ, изменяет статусы товаров внутри заявки на «перевозится» и оборудование доставляется на точку аренды, где статусы меняются на «в месте аренды». Когда товар доставляется обратно, то статусы изменятся на «перевозится», а затем «на складе».

2.2.2. Разработка алгоритмов функционирования программного продукта.

В программном средстве используется множество алгоритмов, как простых, получение данных из базы данных, так и посложнее. Множество алгоритмов имеет одну и ту же структуру, поэтому расписывать все не имеет практического смысла. На рисунке 2.6 представлен пример алгоритма по получению информации из базы данных для функции получения оборудования с фильтрами.

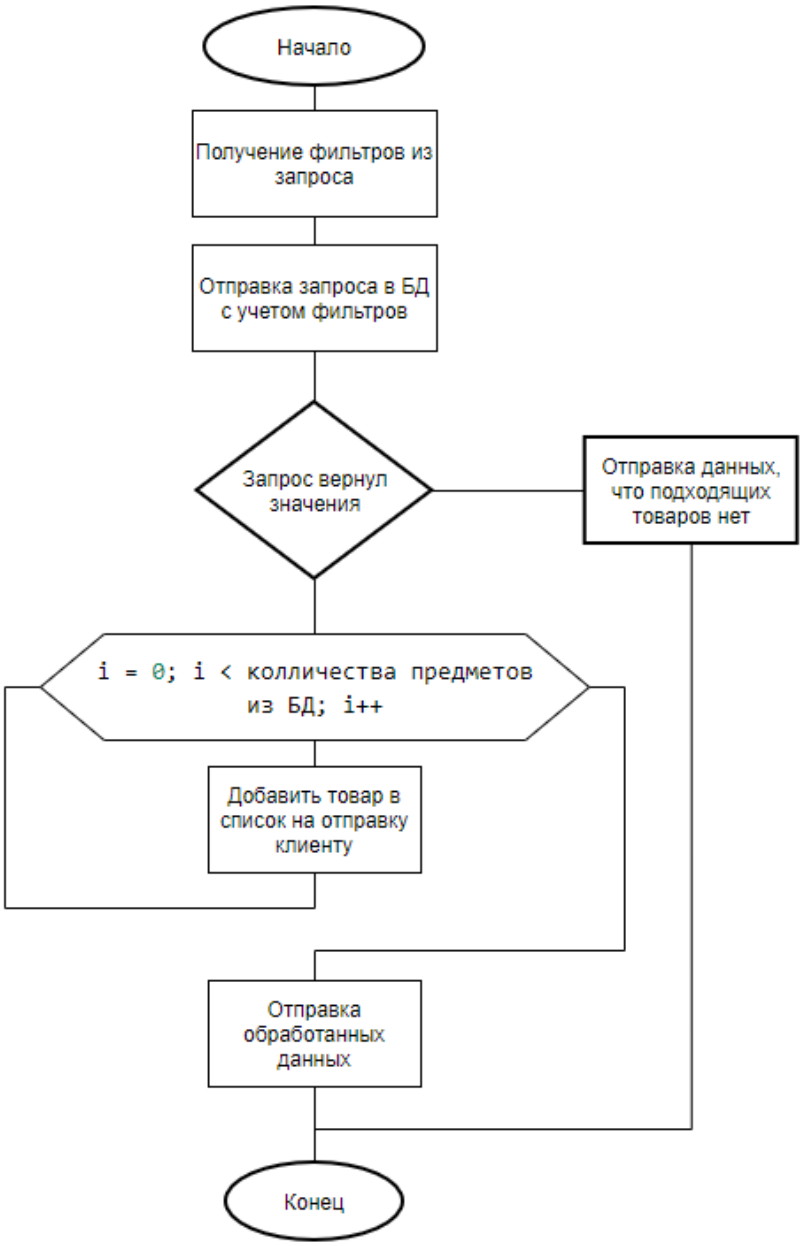

Рисунок 2.6 – Алгоритм получения оборудования с фильтрами

2.3. Вывод к главе

В ходе работы над данной главой, была выбрана и описана архитектура реализуемого программного средства. Выбрана система управления базами данных, в которой создана структура хранения данных.

Так же для понимания того, как должна функционировать программное средство были разработаны и описаны бизнес-процесс в нотации Aris и типовой алгоритм получения данных из базы данных.

3. Экономический раздел

3.1. Организация и планирование работ по теме

В составе работы задействовано 3 человека:

– руководитель ВКР (Жигалов К.Ю., доцент, кафедра КИС) – отвечает за грамотную постановку задачи, контролирует отдельные этапы работы, вносит необходимые коррективы и оценивает выполненную работу в целом;

– консультант (Чижанькова И.В., доцент, кафедра экономики) – отвечает за консультирование в области экономической части проекта;

– разработчик (Лебедев О.А., ИКБО-09-18) – реализация всех поставленных задач, в том числе проведение тестирования готового продукта и подготовка проектной документации.

Состав задействованных в работе участников представлен на рисунке 3.1.

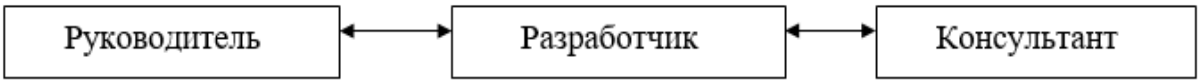

Рисунок 3.1 – Состав участников

3.1.1. Организация работ

На разработку отводится 90 рабочих дней.

Этапы разработки представлены в таблице 3.1.

Таблица 3.1 – Этапы работы

№	Название этапа	Исполнитель	Трудоемкость, чел/дни	Продолжительность работ, дни
1	2	3	4	5
1	**Разработка и утверждение технического задания**	Руководитель	5	**5**
		Разработчик	5	
2	**Технические предложения**	Руководитель	7	**7**
		Консультант	1	
		Разработчик	7	

Продолжение таблицы 3.1

3	**Эскизный проект:**			**16**
3.1	Анализ исходных данных и требований	Разработчик	9	
3.2	Постановка задачи	Консультант	5	
3.3	Разработка общего описания алгоритма функционирования	Руководитель	2	
		Разработчик	7	
4	**Технический проект:**			**15**
4.1	Определение формы представления входных и выходных данных	Руководитель	2	
		Разработчик	5	
4.2	Разработка структуры программы и логической структуры базы данных	Руководитель	2	
		Консультант	2	
		Разработчик	10	
5	**Рабочий проект:**			**47**
5.1	Программирование и отладка программы	Разработчик	24	
5.2	Испытание программы	Разработчик	4	
5.3	Корректировка программы по результатам испытаний	Разработчик	5	
5.4	Подготовка технической документации на программный продукт	Консультант	3	
		Разработчик	7	
5.5	Сдача готового продукта и внедрение	Руководитель	2	
		Консультант	2	
		Разработчик	7	
Итого				**90**

3.1.2. График проведения работ

Календарный график исполнения работы представлен на рисунке 1.2. Из рисунка 3.2 так же видно, что общий срок разработки составит 90 дней.

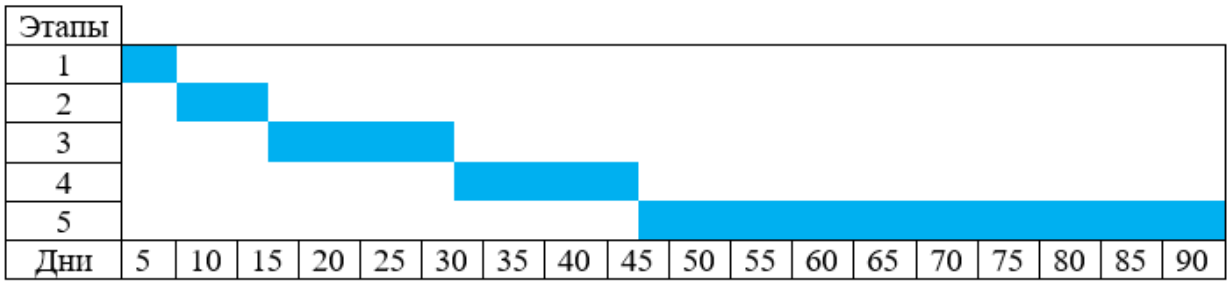

Рисунок 3.2 – График исполнения работы

3.2. Расчёт стоимости проведения работ

Себестоимость состоит из 9 статей:

– 1 статья «Материалы, покупные изделия и полуфабрикаты + ТЗР (20%) от суммы итого по материалам;

– 2 статья «Специальное оборудование» - как правило, затрат нет;

– 3 статья «Основная заработная плата»;

– 4 статья «Дополнительная заработная плата» 20-30% от основной заработной;

– 5 статья «Страховые отчисления» - 30% от ФОТ;

– 6 статья «Командировочные расходы» - как правило, затрат нет;

– 7 статья «Контрагентские услуги» - как правило, затрат нет;

– 8 статья «Накладные расходы» - 250% от основной заработной платы;

– 9 статья «Прочие расходы» - затрат нет.

В выпускной квалификационной работе объем затрат на НИР и ОКР был проведен методом калькуляции.

1 статья «Материалы, покупные изделия и полуфабрикаты»

В таблице 3.2 представлены данные о стоимости материалов и покупных изделий.

Таблица 3.2 – «Материалы, покупные изделия и полуфабрикаты»

№ пп	**Наименование материалов**	**Единицы измерения**	**Количество**	**Цена за единицу (руб)**	**Стоимость (руб)**
1	**2**	**3**	**4**	**5**	**6**
1	USB-флеш-накопитель 32Гб	шт	1	800	800
2	Ручка	шт	4	70	280
4	Бумага А4	пачка	1	1200	1200
5	Карандаш	шт	2	25	50
6	Чернила для струйного принтера Epson 101 (черные)	шт	1	1449	1449
7	Чернила для принтера Epson 101 (цветные)	шт	3	1199	3597
Итого материалов					**7 376**
Транспортно-заготовительные расходы					**1 465,2**
Итого					**8 841,2**

2 статья «Специальное оборудование»

Затраты на специальное оборудование отсутствуют.

3 статья «Основная заработная плата»

Расчет основной заработанной платы показана в таблице 3.3.

Таблица 3.3 – Расчет основной заработанной платы

№ пп	Наименование этапа	Исполнитель (должность)	Мес. оклад (руб)	Трудоемкость (чел/дни)	Оплата за день (руб)	Оплата за этап (руб)
1	2	3	4	5	6	7
1	ТЗ	Руководитель	85 000	5	3 863,63	19 318,15
		Разработчик	48 000	5	2 181,81	10 909,05
2	ТП	Руководитель	85 000	7	3 863,63	27 045,41
		Консультант	55 000	1	2500	7500
		Разработчик	48 000	7	2 181,81	15 272,67
3	Эскизный проект	Руководитель	85 000	2	3 863,63	7 727,26
		Консультант	55 000	5	2500	12500
		Разработчик	48 000	16	2 181,81	34 908,96
4	Технический проект	Руководитель	85 000	4	3 863,63	15 454,52
		Консультант	55 000	2	2500	5000
		Разработчик	48 000	15	2 181,81	32 727,15
5	Рабочий проект	Руководитель	85 000	2	3 863,63	7 727,26
		Консультант	55 000	5	2500	12500
		Разработчик	48 000	47	2 181,81	102 545,07
Итого						**311 135,5**

4 статья «Дополнительная заработная плата»

Дополнительная заработная плата рассчитывается по формуле (3.1).

$$ДЗП = ОЗП \times 0,25 = 311\ 135,5 \times 0,2 = 62\ 227,1 \text{ руб.} \quad (3.1)$$

Дополнительная заработная плата научного и производственного персонала составляет по проекту 62 227,1руб.

5 статья «Страховые отчисления»

Отчисления на социальные нужды (3.3) составляют 30% от фонда оплаты труда (ФОТ), который состоит из основной и дополнительной заработной платы (3.2).

$$ФОТ = ОЗП + ДЗП = 311\ 135,5 + 62\ 227,1 = 373\ 362,6 \text{ руб}, \quad (3.2)$$

$$СВ = ФОТ \times 30\% = 373\,362{,}6 \times 0{,}30 = 112\,008{,}78 \text{ руб.} \quad (3.3)$$

6 статья «Командировочные расходы»

Расходы по данному разделу отсутствуют.

7 статья «Контрагентские услуги»

В процессе разработки данного проекта услуги сторонних организаций не использовались.

8 статья «Накладные расходы»

Накладные расходы высчитываются по формуле (3.4).

$$НР = ОЗП \times 250\% = 311\,135{,}5 \times 2{,}5 = 777\,838{,}75 \text{ руб.} \quad (3.4)$$

9 статья «Прочие расходы»

Для выполнения проекта расходы по этой статье не требуются.

Полная себестоимость проекта представлена в таблице 3.4.

Таблица 3.4 – Полная себестоимость проекта

№ пп	**Номенклатура статей расходов**	**Затраты (руб)**
1	**2**	**3**
1	Материалы, покупные изделия и полуфабрикаты (за вычетом отходов)	8841,2
2	Специальное оборудование для научных (экспериментальных) работ	-
3	Основная заработная плата научного и производственного персонала	311135,5
4	Дополнительная заработная плата научного и производственного персонала	62227,1
5	Страховые взносы в социальные фонды	112008,78
6	Расходы на научные и производственные командировки	-
7	Оплата работ, выполненных сторонними организациями и предприятиями	-
8	Накладные расходы	777838,75
9	Прямые расходы	-
Итого		**1272051,33**

Договорная цена продукта рассчитывается по формуле (3.5).

$$ДЦ = С + П + НДС, \quad (3.5)$$

где С – себестоимость,

П – прибыль.

Норма прибыли составляет 20-30% от стоимости разработки. В рамках данного проекта прибыль составляет 20% от себестоимости. Норма прибыли высчитывается по формуле (3.6).

$$П = С \times 0,2 = 1\ 272\ 051,33 \times 0,2 = 254\ 410,27 \text{ руб.} \quad (3.6)$$

Данный вид работы облагается налогом на добавочную стоимость (НДС) в размере 20%. НДС высчитывается по формуле (3.7).

$$НДС = (С + П) \times 0,2 = (1\ 272\ 051,33 + 254\ 410,27) \times 0,2 = \quad (3.7)$$
$$= 305\ 292,32 \text{ руб.}$$

Таким образом, договорная цена высчитывается по формуле (3.8).

$$ДЦ = С + П + НДС = 1\ 272\ 051,33 + 254\ 410,27 + \quad (3.8)$$
$$+ 305\ 292,32 = 1\ 831\ 753,92 \text{ руб.}$$

4. Технологический раздел

4.1. Обоснование выбора средств для реализации.

Программный продукт решено реализовывать в формате web-приложения, так как это наиболее распространенный и простой способ взаимодействия пользователя и системы. Один из самых распространенных языков для web-программирования является JavaScript. Кроме того, с его помощью можно реализовать и front, и back части приложения. Именно поэтому он был выбран в качестве основного языка программирования.

JavaScript — мультипарадигменный язык программирования. Поддерживает объектно-ориентированный, императивный и функциональный стили. Является реализацией спецификации ECMAScript [16].

Язык JavaScript имеет множество написанных библиотек для разработки пользовательского интерфейса. Самыми популярными фреймворками за историю развития веб-технологий являются Ext JS, Backbone, Ember, Meteor, Angular, React и Vue. Некоторые из перечисленных фреймворков уже устарели и их все реже используют в разработке для коммерческих организаций. После тщательного изучения всех плюсов и минусов приведенных фреймворков для разработки была выбрана библиотека React [16].

ReactJS — Библиотека JavaScript использующаяся для разработки пользовательских интерфейсов и веб-приложений. Преимуществами данной библиотеки можно назвать простоту в изучении и использовании, а также наличие виртуальной DOM, которая упорядочивает документы HTML, что делает загрузку веб-элементов на странице быстрее [10].

Виртуальный DOM (VDOM) — это концепция программирования, в которой идеальное или «виртуальное» представление пользовательского интерфейса хранится в памяти и синхронизируется с «настоящим» DOM при помощи библиотеки, такой как ReactDOM [10].

Backend часть приложения должна наладить взаимодействие между базой данных и клиентской частями приложения. Кроме удобного получения и сохранения данных, платформа должна обладать обширными возможностями по

обработке этих данных. Для нужд разработки web-приложения было решено выбрать серверное решение Node.js.

Node.js — это кроссплатформенная среда выполнения JavaScript с открытым исходным кодом и библиотека для запуска веб-приложений вне браузера клиента. Райан Даль разработал его в 2009 году, а его последняя версия v13.8.0 была выпущена 30 января. Node.js используется для создания серверных веб-приложений и идеально подходит для приложений, интенсивно использующих данные, поскольку он использует асинхронное событие.

Поскольку Node.js является однопоточным, то приходится использовать кластеры – модуль, создающий дочерние процессы, использующие один и тот же порт, что позволяет обеспечивать многопоточность.

4.2. Описание разработанного программного средства.

Разрабатываемое приложение состоит из двух частей: frontend (клиентская часть) и backend (серверная часть). Затем прописывается связь между этими частями и получается рабочее приложение.

4.2.3. Клиентская часть приложения

В ходе разработки клиентской части приложения создаются страницы, на которых в дальнейшем будет отображаться информация, взятая из серверной части.

После создания приложения общее количество страниц стало равно 6. Каждая страница содержит в себе функциональные модули для получения, обработки и обменом данными с серверной частью. Например, на рисунке 4.1 изображена страница, где для пользователей выводятся доступные для заказа товары.

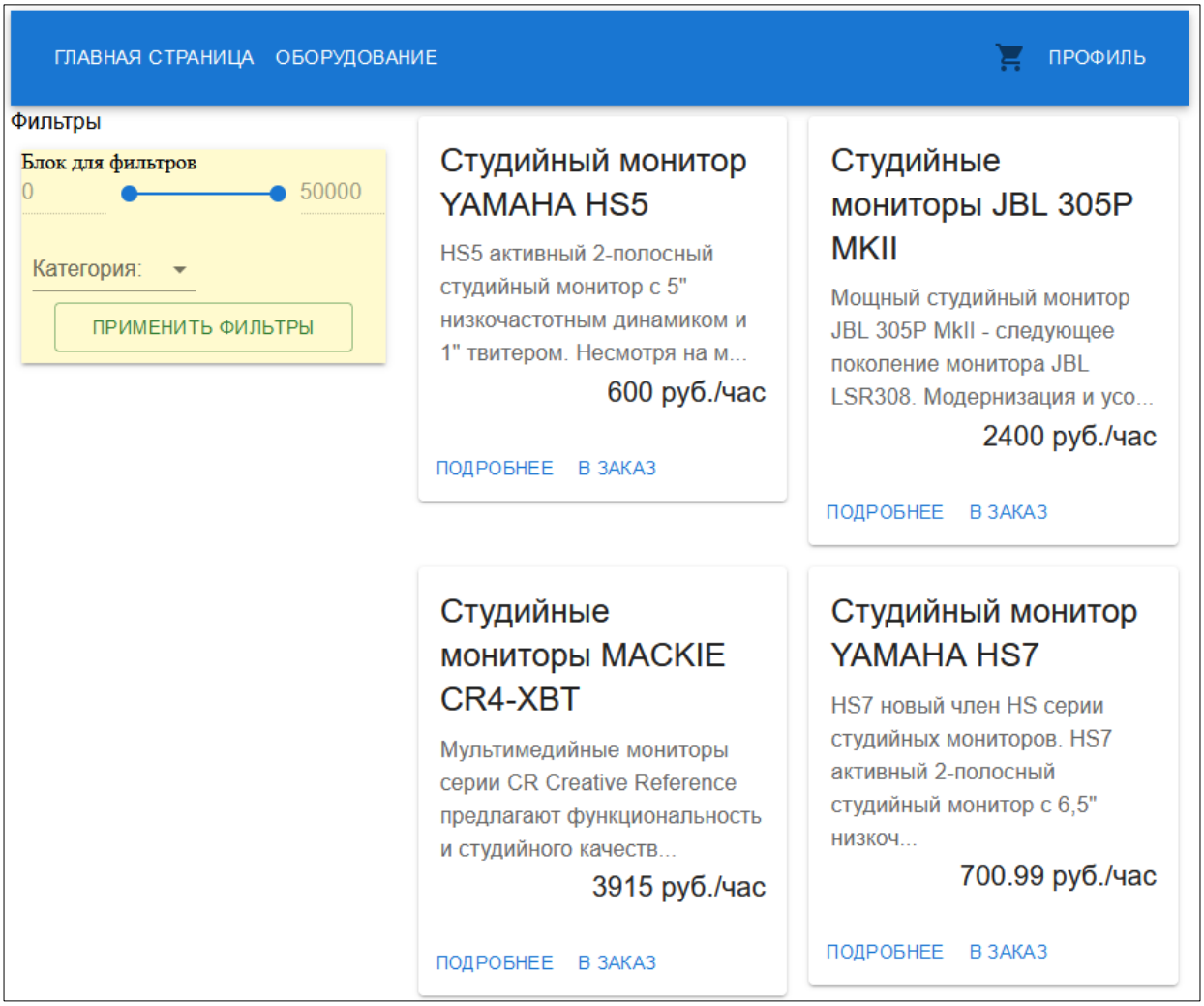

Рисунок 4.1 – Страница для заказа товаров

Рассмотрим функционал этой страницы подробнее. При загрузке страницы происходит несколько запросов на сервер для получения оборудования, чтобы отобразить его на странице, и для получения всех категорий оборудования, для использования в фильтрах.

После того как были получены данные об оборудовании эта информация передается в модуль, который отвечает за отрисовку блоков с конкретным товаром. В этом блоке выводятся наименование товара, краткое описание, цену аренды за час времени, и две кнопки: одна отвечает за переход на страницу с описанием конкретного оборудования, а вторая за добавления товара в корзину.

Так же на странице есть модуль, отвечающий за фильтры. При передвижении ползунка меняется диапазон цен, в котором должны находится товары и выпадающий список с категориями оборудования. При нажатии на

кнопку «Применить фильтр» происходит запрос к серверу для получения нового списка оборудования, который подходит под выбранные фильтры.

Полный код страницы представлен в приложении 1.

4.2.1. Серверная часть приложения

При разработке серверной части приложения пишутся обработчики для запросов с клиентской части приложения и запросы к базе данных с первичной обработкой полученных данных.

Возьмем примером запросы, которые приходят со страницы на рисунке 4.1.

Запрос на получение всех оборудования при загрузке страницы представлен в листинге 4.1

Листинг 4.1 – Фрагмент кода получения оборудования

```
router.get("/getItems", async (req, res) => {
  try {
    const itemsArray = await db.query("SELECT * FROM equipment");
    const items = new Map();
    let minPrice = itemsArray.rows[0].priceForHour;
    let maxPrice = itemsArray.rows[0].priceForHour;
    for (let i = 0; i < itemsArray.rowCount; i++) {
      if (minPrice > itemsArray.rows[i].priceForHour) {
        minPrice = itemsArray.rows[i].priceForHour;
      }
      if (maxPrice < itemsArray.rows[i].priceForHour) {
        maxPrice = itemsArray.rows[i].priceForHour;
      }
      if (items.has(itemsArray.rows[i].eName)) {
        if (
          items.get(itemsArray.rows[i].eName).eUsed >= itemsArray.rows[i].eUsed
        ) {
          items.set(itemsArray.rows[i].eName, itemsArray.rows[i]);
        }
      } else {
        items.set(itemsArray.rows[i].eName, itemsArray.rows[i]);
      }
    }
    let array = Array.from(items, ([name, value]) => ({ name, value }));
    res.status(201).json({ items: array, maxPrice, minPrice });
  } catch (e) {
    res.status(400).json({ message: e.message });
  }
});
```

Когда приходит данный запрос на сервер, происходит получение данных из базы данных хранящиеся в таблице «equipment», после получения оборудование записывается в словарь, для избегания множественного вывода одного и того же товара на странице, также параллельно высчитываются максимальные и минимальные ценники оборудования, для того чтобы в дальнейшем использовать их в фильтрах.

Полный код обработчиков запросов для рассматриваемой страницы представлен в приложении 2.

4.3. Тестирование программного продукта.

Для тестирования было выбрано функциональное тестирование.

Функциональное тестирование – это вид тестирования, которые проверяет программное средство на соответствие функциональным требованиям и спецификациям. Цель такого тестирования в том, чтобы проверить все функции программного средства, подавая на вход различные данные, и сравнивать полученные выходные данные с функциональными требованиями.

Функциональное тестирование в основном включает тестирование черного ящика и не касается исходного кода приложения. Это тестирование проверяет качество базы данных, пользовательского интерфейса, безопасность и другие особенности, и функциональные возможности тестируемого средства

Данные тестирования представлены в таблицах 4.1 и 4.2.

Таблица 4.1 – Тестирование основных функций клиентской части приложения

Полномочия	Описание проверки	Ожидаемый результат	Статус проверки
Не зарегистрированный	Регистрация без введенных данных	Ошибка регистрации введите данные.	Пройдено
Не зарегистрированный	Регистрация с введенными данными	Пользователь успешно создан	Пройдено
Пользователь/ администратор/не зарегистрированный	Применение всех фильтров	Успешное применение и изменение списка товаров	Пройдено
Пользователь/ администратор/не зарегистрированный	Добавление товаров в корзину	Успешное добавление товара в корзину пока он есть в наличии	Пройдено

Продолжение таблицы 4.1

Не зарегистрированный	Попытка создать заказ	Предупреждение о том, что пользователь не зарегистрирован	Провалено
Пользователь/ администратор	Попытка создать заказ без введенных дат или адреса	Заказ не создан пользователь уведомлен	Пройдено
Пользователь	Попытка зайти на страницу для администратора	Не возможность зайти на страницу из-за низкой роли	Пройдено
Пользователь/ администратор	Изменения личных данных и пароля	Корректное изменение данных	Пройдено
Администратор	Добавление нового оборудования	Корректное добавление в БД	Пройдено

Таблица 4.2 – Тестирование основных функций серверной части приложения

Полномочия	Описание проверки	Ожидаемый результат	Статус проверки
Не зарегистрированный	Возможность зарегистрироваться с пустыми полями	Ошибка регистрации введите данные.	Пройдено
Не зарегистрированный	Регистрация с введенными данными	Пользователь успешно добавлен в базу данных и на клиент выслан токен доступа	Пройдено
Пользователь/ администратор/не зарегистрированный	Получение товаров с фильтрами и без	Успешное получение данных из Базы данных, запись в словарь и отправка клиенту	Пройдено
Пользователь/ администратор	Добавление оборудования	Успешное добавление в базу данных оборудование и его характеристик	Пройдено
Пользователь/ администратор	Получение геокоординат из полученного адреса	Успешное получение геокоординат и запись их в базу данных	Пройдено

4.4. Вывод к главе.

В ходе выполнения 4 главы было разработано программное средство, которое способно выводить отметку на карте с текущим местоположением арендуемого оборудования. Так же средство обладает функционалом создания заказов на аренду и расчета его стоимости.

Для создания системы была составлена характеристика предметной области, Выбраны и проанализированы технологии, с помощью которых можно создавать данное средство. Была составлена структура базы данных и реализовано веб-приложение, состоящее из клиентской и серверной части.

Заключение

В заключении стоит еще раз отметить актуальность данной разработки. Аналогичные решения представляют обрезанный функционал за большие деньги, при этом не все функции будут нужны в специфичной области аренды звукового оборудования.

В ходе разработке был выявлен необходимый функционал и продуманы методы и алгоритмы реализации этого функционала. Разработанное веб-приложение готово к использованию в реальных задачах и обладает выявленными функциями:

– просмотр спецоборудования;

– поиск спецоборудования;

– формирование заявки на аренду;

– отправка заявки на аренду;

– возможность регистрации;

– просмотр спецоборудования;

– поиск спецоборудования;

– формирование заявки;

– отправка заявки;

– авторизация;

– просмотр заявок;

– добавление нового спецоборудования;

– добавление технических характеристик спецоборудования;

– составление отчетов о прибыли;

Проведенный экономический анализ решения показал, что на разработку такого программного средства не требует больших бюджетных вложений и в перспективе возможно будет более выгоден чем решения, которые можно приобрести на рынке.

Список использованных источников

1. Баулина К.В. Имитационное моделирование процесса сдачи в аренду объектов технопарка //Форум молодых ученых. – 2018. – №. 11-1. – С. 184-188.

2. Семёнов Е.С. Особенности проектирования веб-ориентированных систем для предприятий прокатного бизнеса // Образование. Наука. Производство. – 2018. – С. 1857-1860.

3. Дюженкова, Н. В. Использование WMS для решения актуальных задач складской логистики / Н. В. Дюженкова, К. Г. Стерлигова // Вестник научных конференций. – 2016. – № 5-5(9). – С. 132-136. – EDN WFIXTR.

4. Потовиченко М. А., Шатилов Ю. Ю. Разработка клиентской части одностраничного web-приложения с использованием библиотеки React //Научное обозрение. Технические науки. – 2020. – №. 1. – С. 39-43.

5. Ивуть Р. Б. Логистика : учебное пособие для студентов специальностей 1-27 01 01 «Экономика и организация производства (по направлениям)», 1-27 02 01 «Транспортная логистика (по направлениям)» / Р. Б. Ивуть. – Минск : БНТУ, 2021. -462 с. / ISBN 978-985-583-617-0

6. Левкин, Г. Г. Основы логистики : учебник / Г. Г. Левкин, А. М. Попович – М.-Берлин: Директ-Медиа, 2015. – 387 с. / ISBN 978-5-4475-5187-2

7. Чехарин, Е. Е. Инкрементное моделирование / Е. Е. Чехарин // Славянский форум. – 2018. – № 4(22). – С. 78-84. – EDN YVQFLV.

8. Хэррон Д. Node. js Разработка серверных веб-приложений на JavaScript. – Litres, 2022. – С. 12-19.

9. Савенко, И. И. Технология разработки программного обеспечения: конспект лекции / И. И. Савенко. — Томск : Издательство Томского политехнического университета, 2013. — 66 с.

10. Мурзин Д. Г., Ярова А. В., Мурзин В. М. Библиотека для создания пользовательских интерфейсов react. Js.

11. Выходцев Н.А. Разработка информационной системы аренды спецтехники коммерческой организации. – 2016.

12. Пивень А. А., Скорин Ю. И. Тестирование программного обеспечения //Системи обробки інформації. – 2012. – №. 4 (1). – С. 56-58.

13. Drake J. D., Worsley J. C. Practical PostgreSQL. – " O'Reilly Media, Inc.", 2002.

14. DaData, API сервиса [Электронный ресурс]: URL: https://dadata.ru/api/

15. Документация к PostgreSQL 14.3 [Электронный ресурс]: URL: https://postgrespro.ru/docs/postgresql/14/index

16. JavaScript [Электронный ресурс]: URL: https://ru.wikipedia.org/wiki/JavaScript

17. Node.js v17.9.0 documentation [Электронный ресурс]: URL: https://nodejs.org/docs/latest-v17.x/api/

18. ReactJS документация [Электронный ресурс]: URL: https://ru.reactjs.org/docs/getting-started.html

Приложение 1

Листинг страницы «витрины» товаров

В листинге 1.1 приведен фрагмент кода из файла ShopPage.js, в котором описан функционал и отрисовка страницы с оборудованием

Листинг 1.1 – фрагмент кода из файла ShopPage.js

```
const ShopPage = () => {
  const [itemArray, setItemArray] = useState([]);
  const [price, setPrice] = useState([0.0, 100.0]);
  const [category, setCategory] = useState("");
  const [filterForm, setFilterForm] = useState({
    price: [0.0, 50000.0],
    category: "all",
  });

  const handleGetItems = useCallback(async () => {
    await axios.get("/api/item/getItems").then((res) => {
      setItemArray(res.data.items);
      //setPrice([res.data.minPrice, res.data.maxPrice]);
    });
  }, []);

  //#region filter price
  const minDistance = 10;
  const handleChangePrice = (event, newValue, activeThumb) => {
    if (!Array.isArray(newValue)) {
      return;
    }
    if (activeThumb === 0) {
      setPrice([Math.min(newValue[0], price[1] - minDistance), price[1]]);
    } else {
      setPrice([price[0], Math.max(newValue[1], price[0] + minDistance)]);
    }
    setFilterForm({ ...filterForm, price: [price[0] * 500, price[1] * 500] });
  };
  const calculatePriceFilter = (value) => {
    return value * 500;
  };
  //#endregion

  //#region category
  const [categoryList, setCategoryList] = useState([]);
  const handleChangeCategory = (event) => {
    setCategory(event.target.value);
    setFilterForm({ ...filterForm, category: event.target.value.toString() });
  };
  const handleGetCategorys = useCallback(async () => {
    await axios.get("/api/item/getAllCategory").then((res) => {
```

```
    setCategoryList(res.data.category);
  });
}, []);
//#endregion

const handleGetFilterItems = async () => {
  console.log(filterForm);
  await axios
    .post("/api/item/getFilterItems", { ...filterForm })
    .then((res) => {
      setItemArray(res.data.items);
    });
};

useEffect(() => {
  handleGetCategorys();
  handleGetItems();
}, [handleGetItems, handleGetCategorys]);

return (
  <Box sx={{ flexGrow: 1, flexWrap: "wrap" }}>
    <div>
      <Toaster />
    </div>
    <Grid container spacing={1}>
      <Grid item xs={4}>
        <Typography> Фильтры </Typography>
        <Box
          sx={{
            boxShadow: 2,
            backgroundColor: "#fffacf",
            m: 1,
            xs: 1,
            display: "flex",
            flexDirection: "column",
          }}
        >
          Блок для фильтров
          <Box sx={{ display: "flex", flexDirection: "row" }}>
            <TextField
              size="small"
              variant="standard"
              value={price[0] * 500}
              disabled
            ></TextField>
            <Slider
              value={price}
              sx={{ width: "300px", ml: 2, mr: 2 }}
              valueLabelDisplay="auto"
              size="small"
              disableSwap
```

```
          scale={calculatePriceFilter}
          onChange={handleChangePrice}
          getAriaValueText={(value) => {
            return `${value}руб`;
          }}
        />
        <TextField
          variant="standard"
          size="small"
          value={price[1] * 500}
          disabled
        ></TextField>
      </Box>
      <Box sx={{ display: "flex", flexDirection: "row" }}>
        <FormControl variant="standard" sx={{ m: 1, minWidth: 120 }}>
          <InputLabel>Категория:</InputLabel>
          <Select
            value={category}
            onChange={handleChangeCategory}
            label="Категория:"
            autoWidth
          >
            <MenuItem value="all">
              <em>Все Категории</em>
            </MenuItem>
            {categoryList.map(
              (item) =>
                item !== null && (
                  <MenuItem value={item.value}>
                    <em>{item.value}</em>
                  </MenuItem>
                )
            )}
          </Select>
        </FormControl>
      </Box>
      <Button
        color="success"
        variant="outlined"
        sx={{ mb: 1, ml: 3, mr: 3 }}
        onClick={handleGetFilterItems}
      >
        Применить фильтры
      </Button>
    </Box>
  </Grid>

  <Grid
    item
    xs={8}
    columns={{ xs: 4, sm: 12, md: 12 }}
```

```
          sx={{ display: "flex", flexWrap: "wrap" }}
        >
          {itemArray.map((item) => (
            <Grid item xs={2} sm={4} md={4} key={item.id}>
              <ItemCard item={item.value} />
            </Grid>
          ))}
        </Grid>
      </Grid>
    </Box>
  );
};

export default ShopPage;
```

Приложение 2

Листинг страницы обработки запросов связанных с товарами

В листинге 2.1 представлен код из файла item.route.js. В нем описан функционал обработки запросов приходящие по адресу «*/api/item/*». В частности, присутствует функционал обработки запросов с страницы с товарами.

Листинг 2.1 – код из файла ItemRoute.js

```
const Router = require("express");
const config = require("config");
const db = require("../db");
const jwt = require("jsonwebtoken");
const { check, validationResult } = require("express-validator");
const bcrypt = require("bcryptjs");
const auth = require("../middleware/auth.middleware");

const router = new Router();

// */api/item/getItem
router.post(
  "/getItem",
  [
    check("id", "Пароль не может быть пустым").exists(),
    check("id", "Поле id не может быть пустым").notEmpty(),
  ],
  async (req, res) => {
    try {
      const errors = validationResult(req);
      if (!errors.isEmpty()) {
        console.log(errors);
        return res.status(400).json({ message: errors });
      }
      const { id } = req.body;
      const item = await db.query('SELECT * FROM equipment WHERE "id"=$1', [
        id,
      ]);
      if (item.rowCount === 0) {
        return res.status(201).json({ message: "Ошибка в получении данных" });
      }
      res.status(201).json({ item: item.rows[0] });
    } catch (e) {
      res.status(400).json({ message: e.message });
    }
  }
);

// */api/item/getItems
router.get("/getItems", async (req, res) => {
```

```
  try {
    const itemsArray = await db.query("SELECT * FROM equipment");
    const items = new Map();
    let minPrice = itemsArray.rows[0].priceForHour;
    let maxPrice = itemsArray.rows[0].priceForHour;
    for (let i = 0; i < itemsArray.rowCount; i++) {
      if (minPrice > itemsArray.rows[i].priceForHour) {
        minPrice = itemsArray.rows[i].priceForHour;
      }
      if (maxPrice < itemsArray.rows[i].priceForHour) {
        maxPrice = itemsArray.rows[i].priceForHour;
      }
      if (items.has(itemsArray.rows[i].eName)) {
        if (
          items.get(itemsArray.rows[i].eName).eUsed >= itemsArray.rows[i].eUsed
        ) {
          items.set(itemsArray.rows[i].eName, itemsArray.rows[i]);
        }
      } else {
        items.set(itemsArray.rows[i].eName, itemsArray.rows[i]);
      }
    }
    let array = Array.from(items, ([name, value]) => ({ name, value }));
    res.status(201).json({ items: array, maxPrice, minPrice });
  } catch (e) {
    res.status(400).json({ message: e.message });
  }
});

// */api/item/getFilterItems
router.post(
  "/getFilterItems",
  [
    check("price", "Пароль не может быть пустым").exists(),
    check("category", "Поле id не может быть пустым").exists(),
  ],
  async (req, res) => {
    try {
      const errors = validationResult(req);
      if (!errors.isEmpty()) {
        console.log(errors);
        return res.status(400).json({ message: errors });
      }
      const { price, category } = req.body;
      console.log(price, category);
      let itemsArray;
      if (category === "all") {
        itemsArray = await db.query(
          'SELECT * FROM equipment WHERE "priceForHour" <$2 and "priceForHour" >$1',
          [price[0], price[1]]
        );
```

```
      } else {
        itemsArray = await db.query(
          'SELECT * FROM equipment WHERE "priceForHour" <$2 and "priceForHour" >$1
and "eCategory" = $3',
          [price[0], price[1], category]
        );
      }
      console.log(itemsArray);
      const items = new Map();
      for (let i = 0; i < itemsArray.rowCount; i++) {
        if (items.has(itemsArray.rows[i].eName)) {
          if (
            items.get(itemsArray.rows[i].eName).eUsed >=
            itemsArray.rows[i].eUsed
          ) {
            items.set(itemsArray.rows[i].eName, itemsArray.rows[i]);
          }
        } else {
          items.set(itemsArray.rows[i].eName, itemsArray.rows[i]);
        }
      }
      let array = Array.from(items, ([name, value]) => ({ name, value }));
      res.status(201).json({ items: array });
    } catch (e) {
      res.status(400).json({ message: e.message });
    }
  }
);

// */api/item/getCategory
router.post(
  "/getCategory",
  [check("id", "Пароль не может быть пустым").exists()],
  async (req, res) => {
    try {
      const errors = validationResult(req);
      if (!errors.isEmpty()) {
        console.log(errors);
        return res.status(400).json({ message: errors });
      }
      const { id } = req.body;
      const category = await db.query(
        'SELECT "eCategory" FROM equipment WHERE "id" =$1',
        [id]
      );
      res.status(201).json({ category: category.rows[0].eCategory });
    } catch (e) {
      res.status(400).json({ message: e.message });
    }
  }
);
```

```
// */api/item/getAllCategory
router.get("/getAllCategory", async (req, res) => {
  try {
    const categoryArr = await db.query('SELECT "eCategory" FROM equipment');
    const categorys = new Map();
    for (let i = 0; i < categoryArr.rowCount; i++) {
      if (!categorys.has(categoryArr.rows[i].eCategory)) {
        categorys.set(
          categoryArr.rows[i].eCategory,
          categoryArr.rows[i].eCategory
        );
      }
    }
    let array = Array.from(categorys, ([name, value]) => ({ name, value }));
    res.status(201).json({ category: array });
  } catch (e) {
    res.status(400).json({ message: e.message });
  }
});

// */api/item/getOptions
router.post(
  "/getOptions",
  [check("id", "Пароль не может быть пустым").exists()],
  async (req, res) => {
    try {
      const errors = validationResult(req);
      if (!errors.isEmpty()) {
        console.log(errors);
        return res.status(400).json({ message: errors });
      }
      const { id } = req.body;
      const itemName = await db.query(
        'SELECT "eName" FROM equipment WHERE id = $1',
        [id]
      );
      const options = await db.query(
        'SELECT DISTINCT ON ("oName") "oName", options.id,  "oValueIntA",
"oValueIntB", "oValueChar",options."eName", "oValueName"  FROM options INNER JOIN
equipment USING ("eName") where options."eName" = $1 ORDER BY "oName" DESC',
        [itemName.rows[0].eName]
      );
      if (options.rowCount === 0) {
        return res.status(201).json({
          result:
            "Характеристике объекта еще не внесены, просим прощения за
предоставленные неудобства",
        });
      }
      res.status(201).json({ options: options.rows });
```

```
    } catch (e) {
      res.status(400).json({ message: e.message });
    }
  }
);
// */api/item/getOptionsSort
router.post(
  "/getOptionsSort",
  [check("id", "Пароль не может быть пустым").exists()],
  async (req, res) => {
    try {
      const errors = validationResult(req);
      if (!errors.isEmpty()) {
        console.log(errors);
        return res.status(400).json({ message: errors });
      }
      const { id } = req.body;
      const itemName = await db.query(
        'SELECT "eName" FROM equipment WHERE id = $1',
        [id]
      );
      const options = await db.query(
        'SELECT DISTINCT ON ("oName") "oName", options.id,  "oValueIntA",
"oValueIntB", "oValueChar",options."eName", "oValueName"  FROM options INNER JOIN
equipment USING ("eName") where options."eName" = $1 ORDER BY "oName" ASC',
        [itemName.rows[0].eName]
      );
      if (options.rowCount === 0) {
        return res.status(201).json({
          result:
            "Характеристике объекта еще не внесены, просим прощения за
предоставленные неудобства",
        });
      }
      res.status(201).json({ options: options.rows });
    } catch (e) {
      res.status(400).json({ message: e.message });
    }
  }
);
// */api/item/updateOptions
router.post(
  "/updateOptions",
  [
    check("options", "Вы не обновили ни один параметр").exists(),
    check("id", "Поле email не может быть пустым").exists(),
  ],
  async (req, res) => {
    const { options, id } = req.body;
    const errors = validationResult(req);
    if (!errors.isEmpty()) {
```

```
      console.log(errors);
      return res.status(400).json({ message: errors });
    }
    const itemName = await db.query(
      'SELECT "eName", "eCategory" FROM equipment WHERE id = $1',
      [id]
    );
    options.map(async (option) => {
      //!option[0] - имя option, [1] - параметры
      const check = await db.query(
        'SELECT id, "oValueIntA", "oValueIntB" FROM options WHERE "oName" = $1 and
"eName" =$2',
        [option[0], itemName.rows[0].eName]
      );
      console.log(check.rowCount, check.rows);
      if (check.rowCount === 0) {
        await db.query(
          'INSERT INTO options( "oName", "oValueChar", "eName", "oValueName",
"oValueIntA", "oValueIntB") VALUES ( $1, $2, $3, $4, $5, $6)',
          [
            option[1].oName,
            option[1].oValueChar,
            itemName.rows[0].eName,
            option[1].oValueName,
            Number(option[1].oValueIntA),
            Number(option[1].oValueIntB),
          ]
        );
      } else {
        if (option[1].oValueIntA === null && option[1].oValueIntB === null) {
          await db.query(
            'UPDATE options SET "oName"=$1, "oValueChar"=$2, "eName"=$3,
"oValueName"=$4, "oValueIntA"=$5, "oValueIntB"=$6 WHERE "oName"=$1 and "eName"=$3',
            [
              option[1].oName,
              option[1].oValueChar,
              itemName.rows[0].eName,
              option[1].oValueName,
              Number(check.rows[0].oValueIntA),
              Number(check.rows[0].oValueIntB),
            ]
          );
        } else if (
          option[1].oValueIntA === null &&
          option[1].oValueIntB !== null
        ) {
          await db.query(
            'UPDATE options SET "oName"=$1, "oValueChar"=$2, "eName"=$3,
"oValueName"=$4, "oValueIntA"=$5, "oValueIntB"=$6 WHERE "oName"=$1 and "eName"=$3',
            [
              option[1].oName,
```

```
              option[1].oValueChar,
              itemName.rows[0].eName,
              option[1].oValueName,
              Number(check.rows[0].oValueIntA),
              Number(option[1].oValueIntB),
            ]
          );
        } else if (
          option[1].oValueIntA !== null &&
          option[1].oValueIntB === null
        ) {
          await db.query(
            'UPDATE options SET "oName"=$1, "oValueChar"=$2, "eName"=$3,
"oValueName"=$4, "oValueIntA"=$5, "oValueIntB"=$6 WHERE "oName"=$1 and "eName"=$3',
            [
              option[1].oName,
              option[1].oValueChar,
              itemName.rows[0].eName,
              option[1].oValueName,
              Number(option[1].oValueIntA),
              Number(check.rows[0].oValueIntB),
            ]
          );
        } else {
          await db.query(
            'UPDATE options SET "oName"=$1, "oValueChar"=$2, "eName"=$3,
"oValueName"=$4, "oValueIntA"=$5, "oValueIntB"=$6 WHERE "oName"=$1 and "eName"=$3',
            [
              option[1].oName,
              option[1].oValueChar,
              itemName.rows[0].eName,
              option[1].oValueName,
              Number(option[1].oValueIntA),
              Number(option[1].oValueIntB),
            ]
          );
        }
      }
    });
    res.status(201);
  }
);

// */api/item/addItem
router.post(
  "/addItem",
  [
    check("name", "Пароль не может быть пустым").exists(),
    check("price", "Цена не может быть пустым").exists(),
    check("description", "Описание не может быть пустым").exists(),
    check("priceForHour", "Арендная цена не может быть пустым").exists(),
```

```
    check("number", "Колличество не может быть пустым").exists(),
    check("options", "Опции не могут быть пустыми").exists(),
  ],
  async (req, res) => {
    try {
      const errors = validationResult(req);
      if (!errors.isEmpty()) {
        console.log(errors);
        return res.status(400).json({ message: errors });
      }
      const {
        category,
        name,
        price,
        description,
        priceForHour,
        number,
        options,
      } = req.body;
      for (let i = 0; i < number; i++) {
        await db.query(
          'INSERT INTO equipment( "eName", "ePrice", "eDescription", "eCategory",
"priceForHour") VALUES ( $1, $2, $3, $4, $5)',
          [category + " " + name, price, description, category, priceForHour]
        );
      }
      options.map(async (option) => {
        await db.query(
          'INSERT INTO options( "oName", "oValueChar", "eName", "oValueName",
"oValueIntA", "oValueIntB") VALUES ( $1, $2, $3, $4, $5, $6)',
          [
            option[1].oName,
            option[1].oValueChar,
            category + " " + name,
            option[1].oValueName,
            Number(option[1].oValueIntA),
            Number(option[1].oValueIntB),
          ]
        );
      });
      res.status(201).json({ status: "OK" });
    } catch (e) {
      res.status(400).json({ message: e.message });
    }
  }
);

// */api/item/getImage
router.post("/getImage", async (req, res) => {
  const { id } = req.body;
  const itemName = await db.query(
```

```
    'SELECT "eName", "eCategory" FROM equipment WHERE id = $1',
    [id]
  );
  const image = await db.query(
    'SELECT "fileName" FROM equip_image WHERE "eName" = $1',
    [itemName.rows[0].eName.slice(itemName.rows[0].eCategory.length + 1)]
  );
  if (image.rowCount === 0) {
    return res.status(201).json({
      status: "not",
      image:
        "https://www.google.com/url?sa=i&url=https%3A%2F%2Ficons8.cn%2Ficon%2FujQ2TKd
Wp5vZ%2Fempty-
box&psig=AOvVaw2b_J1G_Tg8yVymwM5Kc4CQ&ust=1652359985085000&source=images&cd=vfe&ved=0
CAwQjRxqFwoTCKCHoeq-1_cCFQAAAAAdAAAAABAJ",
    });
  }
  res.status(201).json({ status: "ok", image: image.rows });
});

// */api/item/getAllItems
router.get("/getAllItems", async (req, res) => {
  const items = await db.query(
    'SELECT equipment.id, "eName", "ePrice", "eDescription", "eCategory", "pId",
date_change, "priceForHour", "eUsed", place."pIat", place."pIon" FROM equipment left
join place on "pId" = place.id order by equipment.id asc'
  );
  res.status(201).json({ items: items.rows });
});

// */api/item/findFreeItem
router.post("/findFreeItem", async (req, res) => {
  const { ids } = req.body;
  let freeID = new Map();
  const itemName = await db.query(
    'SELECT "eName" FROM equipment WHERE id = $1',
    [Number(ids[0])]
  );
  const AllID = await db.query('SELECT id FROM equipment where "eName" =$1', [
    itemName.rows[0].eName,
  ]);
  for (let i = 0; i < AllID.rowCount; i++) {
    if (!ids.includes(AllID.rows[i].id.toString())) {
      console.log(!ids.includes(AllID.rows[i].id.toString()), AllID.rows[i].id);
      freeID.set("next", AllID.rows[i].id);
    }
  }
  if (freeID.has("next")) {
    res.status(201).json({ nextID: [...freeID] });
  } else {
    res.status(201).json({ nextID: 0 });
```

```
  }
});

// */api/item/ChangeStatus
router.post("/ChangeStatus", async (req, res) => {
  const { status, id } = req.body;
  const lastStatus = await db.query(
    'SELECT "pId" FROM equipment where id = $1 ',
    [id]
  );
  if (status === 0 && lastStatus.rows[0].pId === null) {
    await db.query('UPDATE equipment  SET "pId"=$1, date_change=$2  WHERE id =$3', [
      0,
      new Date().toISOString(),
      id,
    ]);
  } else if (status === null) {
    if (lastStatus.rows[0].pId !== 0 && lastStatus.rows[0].pId !== null) {
      const eqOrder = await db.query(
        'SELECT "startDate", "endDate", "pId" FROM order_to_equipment inner join
orders on "orderId" = orders.id where "eId" = $1 order by "startDate"',
        [id]
      );
      for (let i = 0; i < eqOrder.rowCount; i++) {
        if (new Date() < new Date(eqOrder.rows[i].endDate)) {
          return res.status(201).json({ message: "alert" });
        } else {
          await db.query(
            'UPDATE equipment SET "pId"=$1, date_change=$2  WHERE id =$3',
            [, new Date().toISOString(), id]
          );
          await db.query('UPDATE place SET "dateOut"=$1 WHERE id=$2', [
            new Date().toISOString(),
            eqOrder.rows[i].pId,
          ]);
        }
      }
    } else {
      await db.query(
        'UPDATE equipment SET "pId"=$1, date_change=$2  WHERE id =$3',
        [, new Date().toISOString(), id]
      );
    }
  } else if (status === 1 && lastStatus.rows[0].pId === null) {
    const eqOrder = await db.query(
      'SELECT "startDate", "endDate", "pId" FROM order_to_equipment inner join orders
on "orderId" = orders.id where "eId" = $1 order by "startDate"',
      [id]
    );
    if (eqOrder.rowCount !== 0) {
      for (let i = 0; i < eqOrder.rowCount; i++) {
```

```
        if (new Date() > new Date(eqOrder.rows[i].startDate)) {
          await db.query(
            'UPDATE equipment SET "pId"=$1, date_change=$2  WHERE id =$3',
            [eqOrder.rows[i].pId, new Date().toISOString(), id]
          );
          await db.query('UPDATE place SET "dateOn"=$1 WHERE id=$2', [
            new Date().toISOString(),
            eqOrder.rows[i].pId,
          ]);
          break;
        }
      }
    }
  }
  res.status(201).json({ message: "OK" });
});

module.exports = router;
```