МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
федеральное государственное бюджетное образовательное учреждение
высшего образования
«Тольяттинский государственный университет»

Институт математики, физики и информационных технологий
(наименование института полностью)

Кафедра «Прикладная математика и информатика»
(наименование)

09.03.03 Прикладная информатика
(код и наименование направления подготовки, специальности)

«Бизнес-информатика»
(направленность (профиль) / специализация)

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА (БАКАЛАВРСКАЯ РАБОТА)

на тему Разработка веб–приложения для обработки заказов предприятия проката оборудования (на примере ООО «Айсберг»)

Студент М.Ю. Родькина
(И.О. Фамилия) (личная подпись)

Руководитель Н.Н. Казаченок
(ученая степень, звание, И.О. Фамилия)

Тольятти 2020

Аннотация

Тема выпускной квалификационной работы: «Разработка веб–приложения для обработки заказов предприятия проката оборудования (на примере ООО «Айсберг»)».

Цель данной выпускной квалификационной работы – разработка веб-приложения для учета заказов на прокат оборудования в пункте проката ООО «Айсберг».

Структура выпускной квалификационной работы представлена введением, тремя главами, заключением, списком используемой литературы и приложением.

В первой главе проведен анализ предметной области – бизнес процессов учета заказов проката оборудования, дана характеристика предприятия, проведено концептуальное моделирование, осуществлён сравнительный анализ существующих решений в области и сформулирована цель разработки и требования к веб–приложению.

Во второй главе произведено логическое моделирование, проведен сравнительный анализ архитектур для реализации веб–приложения и технологий разработки программного обеспечения, обоснован выбор архитектуры и технологии разработки, описаны программные модули, взаимодействие между ними, функционал разработанного веб–приложения и результаты тестирования.

В третье главе произведены расчеты экономической эффективности от внедрения веб-приложения для учета заказов в работу пункта проката ООО «Айсберг».

В заключении представлены выводы, которые были получены в ходе написания бакалаврской работы.

Работа представлена на 65 страницах, содержит 23 рисунка и 6 таблиц. При написании выпускной квалификационной работы использовалось 36 библиографических источников, в том числе 5 иностранных.

Оглавление

Введение

В последние годы пункты проката приобретают все большую популярность, они позволяют потребителю воспользоваться оборудованием, инструментами, техникой, предметами обихода в краткосрочном периоде за небольшую цену. Важную роль в работе любого пункта проката играет учет заказов. Многие пункты используют ручной учет или используют простые инструменты такие как, например, таблицы Microsoft Excel.

Очевидно, что автоматизация процесса учетов заказов повысит эффективность работы пункта проката, исходя из этого для пункта проката оборудования для ремонта и строительства ООО «Айсберг» г. Тольятти было принято решение о разработке автоматизированной системы учета заказов, а именно веб–приложения.

Актуальность данной выпускной квалификационной работы обусловлена потребностью в автоматизации бизнес-процесса управления заказами на прокат оборудования.

Объектом исследования является процесс учета заказов на прокат оборудования.

Предметом исследования – автоматизация процесса учета заказов на прокат оборудования.

Цель выпускной квалификационной работы – разработка веб-приложения для учета заказов на прокат оборудования в пункте проката ООО «Айсберг».

Для того чтобы достигнуть поставленной цели необходимо решить следующие задачи:

- осуществить поиск и анализ учебной и учебно-методической литературе информации;
- исследовать и проанализировать существующие цифровые решения в области проката;

- проанализировать бизнес–процесс учета заказов и выявить аспекты, которые нуждаются в модернизации;
- сформулировать требования к веб–приложению;
- выполнить концептуальное моделирование предметной области;
- изучить возможные средства реализации, выбрать подходящие для данного веб–приложения и обосновать выбор;
- выполнить логическое моделирование предметной области;
- спроектировать интерфейс веб–приложения;
- реализовать веб–приложение;
- описать функционал разработанного веб–приложения;
- оценить экономическую эффективность.

При написании выпускной квалификационной работы были использованы методы структурного и объектно-ориентированного анализа и проектирования.

При выполнении данной выпускной квалификационной работы использовались методы исследования: анализ, системный подход, методы моделирования бизнес–процессов, методы моделирования информационной системы и ее баз данных при помощи CASE средств. Кроме того, для данной работы использовались учебные пособия, стандарты по моделированию, проектированию и разработке информационных систем.

Выпускная квалификационная работа состоит из введения, трех глав, заключения, списка используемой литературы и приложения.

В первой главе проведен анализ предметной области – бизнес процессов учета заказов проката оборудования, дана характеристика предприятия, для которого осуществляется разработка, проведено концептуальное моделирование, осуществлён сравнительный анализ существующих решений в области, сформулирована цель разработки и требования к веб–приложению

Во второй главе произведено логическое моделирование, построены диаграмма вариантов использования и диаграмма классов, в рамках проведен

сравнительный анализ архитектур для реализации веб–приложения и технологий разработки программного обеспечения, а также обоснован выбор, описаны программные модули, взаимодействие между ними, описан функционал разработанного веб–приложения и результаты тестирования.

В третье главе произведены расчеты экономической эффективности от внедрения веб-приложения для учета заказов в работу пункта проката ООО «Айсберг».

В заключении выпускной квалификационной работы представлены результаты и выводы о проделанной работе. Итог выпускной квалификационной работы – разработанное веб–приложение для обработки заказов на прокат оборудования в пункте проката ООО «Айсберг».

Глава 1 Анализ деятельности пункта проката

1.1 Технико-экономическая характеристика деятельности пункта проката ООО «Айсберг»

1.1.1 Характеристика сферы услуг – проката инструмента для строительства и ремонта

По мнению экспертов, рынок проката в России сейчас переживает небывалый рост, а такая область как прокат строительного инструмента набирает с каждым днем всё большую популярность. Как известно, для ремонта, строительства, как и других работ оптимально применять наиболее качественное оборудование и инструменты. Это помогает значительно сэкономить время и силы.

Очевидно, что приобрести для себя профессиональный инструмент может далеко не каждый, это, как правило требует значительных вложений и при этом, в подавляющем большинстве случаев, оборудование или инструмент требуются на короткий промежуток времени. Таким образом, приобретать его для себя совершенно экономически не выгодно.

Именно поэтому пункты проката различного оборудования для стройки, ремонта, дачных работ пользуются значительной популярностью, именно прокат может значительно уменьшить стоимость ремонтных, строительных или дачных работ. По оценкам экспертов товарного рынка, услугами подобных пунктов проката готово пользоваться около 10% жителей современных городов.

1.1.2 Характеристика предприятия пункта проката ООО «Айсберг»

Предприятие ООО «Айсберг» работает в сфере проката, продажи и ремонта оборудования для строительства и ремонта. Организация представляет собой подразделение компании «Впрок», которая существует на рынке с 2013 года, и имеет филиалы в 13 городах России, в том числе в

Липецке, Воронеже, Курске, Самаре, Тольятти, Азове, Твери и других. История создания компании началась в 2013 году с небольшой фирмы, сдающей в аренду несколько единиц техники, со временем, благодаря удачно выбранной нише и выгодному предложение для потребителя предприятие разрослось и в последующие годы были открыты филиалы в других городах страны. Компания помогает потребителю не только приобретать необходимое оборудование, но и осуществляет гарантийный и пост гарантийный ремонт.

Миссия предприятия «Айсберг»: «Развивать успешный и эффективный бизнес, предлагая Клиентам оптимальный ассортимент оборудования для строительства и ремонта при оптимальном уровне сервиса».

Основная целевая аудитория предприятия – это строительные бригады различной направленности работ, частники, использующие инструмент на даче и в саду, и для строительства индивидуальных жилых домов, дач, осуществления ремонтных работ в собственных квартирах.

Таким образом, принцип работы фирмы — закупать и сдавать в краткосрочную аренду оборудование для ремонта и строительства. Также фирма занимается продажей расходных материалов, некоторого оборудования и осуществляет ремонт строительной техники.

В предприятии ООО «Айсберг» четыре основных подразделения — отдел проката оборудования, отдел продаж, отдел ремонта и финансовый отдел. Организационную структуру можно увидеть на схеме, представленной на рисунке 1.

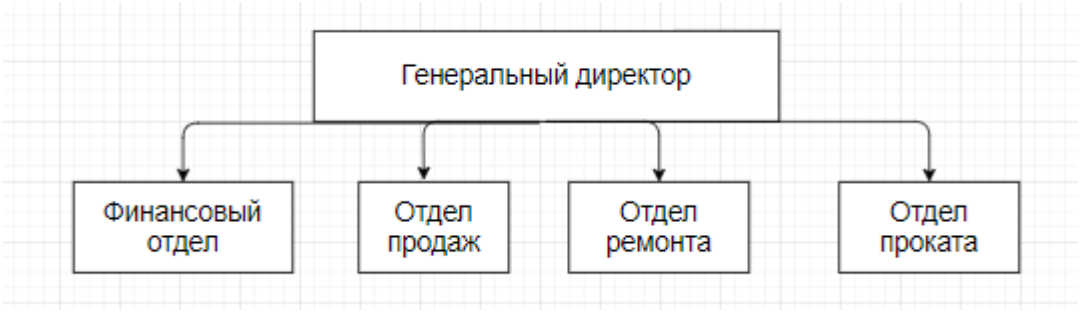

Рисунок 1 – Организационная структура ООО «Айсберг»

Как следует из схемы, все сотрудники напрямую подчинены генеральному директору.

Генеральный директор осуществляет общее руководство процессами и принимает решения по всем вопросам, обеспечивающим работу пункта проката: регулирование и осуществление контроля за всеми подразделениями, разрешение конфликтных проблем внутри предприятия, проведение анализа работы пункта проката, анализ возможностей развития, создание оптимальных условий для повышения продуктивности деятельности фирмы, проведение работы для расширению деятельности предприятия и повышению конкурентоспособности, налаживание взаимодействия и сотрудничества со стороны фирм, способных оказать помощь в реализации деятельности фирмы.

Финансовый отдел занимается регулированием финансовой деятельности организации и ведением бухгалтерского учета: ведение первичной документации, начисление и удержания с заработной платы, начисление и отчетность в фонды социального обеспечения, свод баланса, анализ финансовой деятельности предприятия, отчетность в налоговые органы.

Отдел продаж выполняет следующие функции: подбор оборудованя под потребности клиентов, заказ оборудование для продажи, ведение внутренней отчетность и документации.

Отдел ремонта осуществляет ремонт собственного оборудования, принимает заказы и осуществляет ремонт оборудования клиентов, ведет внутреннюю отчетность.

1.1.3 Краткая характеристика подразделения проката и его видов деятельности

Основным подразделением фирмы является отдел проката оборудования.

Подразделение проката выполняет функции по обеспечению деятельности проката оборудования для строительства и ремонта, а именно:

- личные и телефонные консультации клиентов по необходимому оборудованию;
- подбор оптимального оборудования под потребности клиента,
- информирование потребителя о правилах проката, прейскуранте цен, устройстве и правилах эксплуатации инвентаря и оборудования, имеющегося в прокатном фонде;
- ведение учета и отчетности, оформление договоров;
- прием от клиентов возвращаемых предметов проката, проверка их исправности и комплектности;
- получение денег за прокат, оформление актов на предметы, вышедшие из строя по вине клиентов;
- ведение документации по движению выданных предметов проката и другой документации;
- сдача неисправных предметов проката в ремонт;
- оформление актов на списание износившегося инвентаря, заявок на закупку дополнительных единиц оборудования для проката, как на замену вышедших из стоя, так и выявленных в процессе анализа потребностей клиентов.

Деятельность подразделения проката будет проанализирована в данной главе.

1.2 Концептуальное моделирование деятельности подразделения проката

1.2.1 Методология и технология проектирования информационной системы

Наиболее распространенной технологией проектирования информационной системы предприятий является технология проектирования системы и ее компонентов на базе процессного подхода, а также спиральной

модели жизненного цикла автоматизированной информационной системы [25]. Процесс проектирования состоит из следующих этапов:

- анализ проблемы;
- проектирование автоматизированной информационной системы;
- реализация автоматизированной информационной системы;
- внедрение автоматизированной информационной системы;
- сопровождение автоматизированной информационной системы.

Основа проекта – понятие бизнес–модели автоматизированной информационной системы, опирающееся на следующие уровни описания системы:

- концептуальный уровень (содержательное описание автоматизированной информационной системы на основе структурного подхода);
- логический уровень (формализованное/модельное описание автоматизированной информационной системы на основе объектно-ориентированного подхода);
- физический уровень (программное–аппаратная реализация автоматизированной информационной системы) [24].

Чтобы провести концептуальное проектирование веб-приложения была использована методология реинжиниринга бизнес–процессов, которая подразумевает следующие этапы:

- построение модели бизнес–процесса «Как есть», для описания существующего положения;
- создание модели «Как должно быть» для описания улучшенных процессов предприятия;
- реализация на предприятии желаемых способов протекания бизнес–процессов с помощью современных информационных технологий.

Перейдем к выбору нотации для концептуального моделирования

1.2.2 Обоснование выбора нотации для концептуального моделирования

Моделирование предметной области - это очень важный этап на стадии проектирования любой информационной системы. Существует несколько нотаций для концептуального моделирования, среди которых наиболее известными являются UML, IDEF0 и ARIS.

UML (Unified Modeling Language) – это язык для графического описания, который предназначен для объектно-ориентированного моделирования бизнес–процессов. Он используется для создания абстрактной модели системы, которая называется UML моделью.

IDEF0 (Integration Definition for Function Modeling) – это методология для функционального моделирования бизнес–процессов. IDEF0 отображает структуру и функции системы, а также потоки информации, которые связывают эти функции.

ARIS (Architecture of Integrated Information Systems) – методология для моделирования бизнес–процессов организаций и программный продукт компании Software AG.

Наиболее подходящей нотацией для целей моделирования небольшого веб–приложения можно назвать IDEF0, данная нотация проста для изучения и очень удобна для создания моделей. Основа методологии IDEF0 заключается в графическом описании взаимосвязанных бизнес-процессов.

Модель IDEF0 представляет собой блок, который выполняет некоторую бизнес–функцию. Четыре стороны этого блока имеют различное предназначение [27].

Для наглядности бизнес–процессы, которые представляют собой совокупность взаимосвязанных действий, направленных на создание определенного продукта, визуализируются в виде блок–схем.

Модель бизнес–процесса «КАК ЕСТЬ» это визуальная модель состояния предприятия, существующая в данный момент. Подобная модель систематизирует существующие процессы и используемые информационные

объекты. Если провести анализ данной модели, то можно обнаружить так называемые «узкие места» в организации взаимодействия бизнес–процессов, и уже после этого можно говорить о необходимости изменений в той структуре, которая существует на данный момент [30].

Данную модель определяют как функциональную и проектируют при помощи различных CASE–средств с использованием различных графический нотаций.

1.2.3 Разработка и анализ модели бизнес–процесса «КАК ЕСТЬ» деятельности подразделения проката

На первом этапе проанализируем деятельность подразделения проката. Диаграмма IDEF0 верхнего представлена на рисунке 2. Диаграмма показывает, что на входе процесс получает информацию о клиента, запрос клиента на оборудование, список оборудования, управляющие данные - это прайс-лист и шаблоны документов, механизмы – сотрудник проката, таблицы Excel, принтер, на выходе – акт выдачи, отчеты, договор проката и история заказов

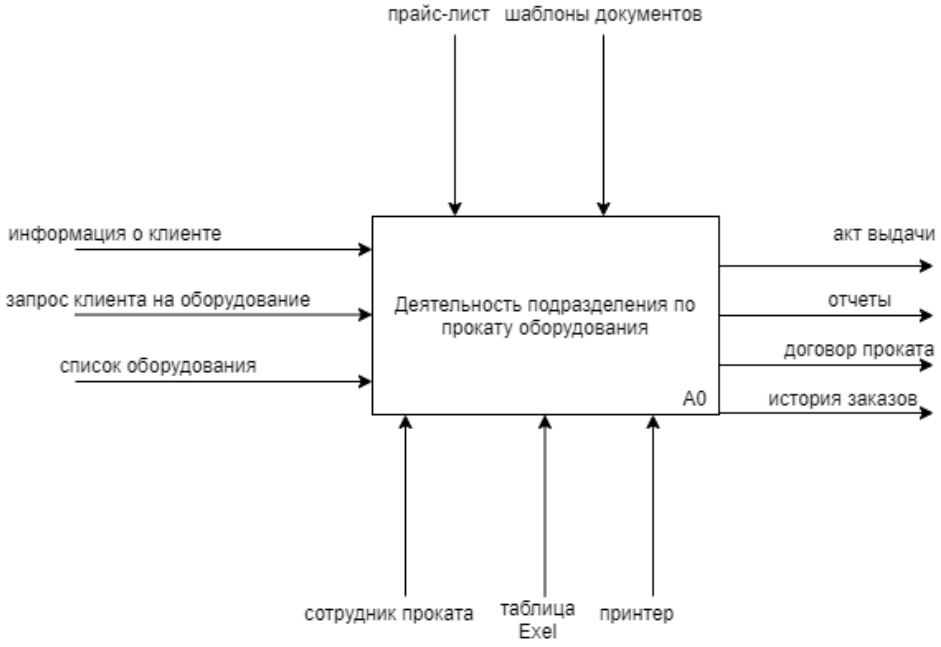

Рисунок 2 – Диаграмма IDEFO верхнего уровня

Проведем декомпозицию процесса. Декомпозиция представляет собой разделение целого процесса на части. Модель проведенной декомпозиции приведена на рисунке 3.

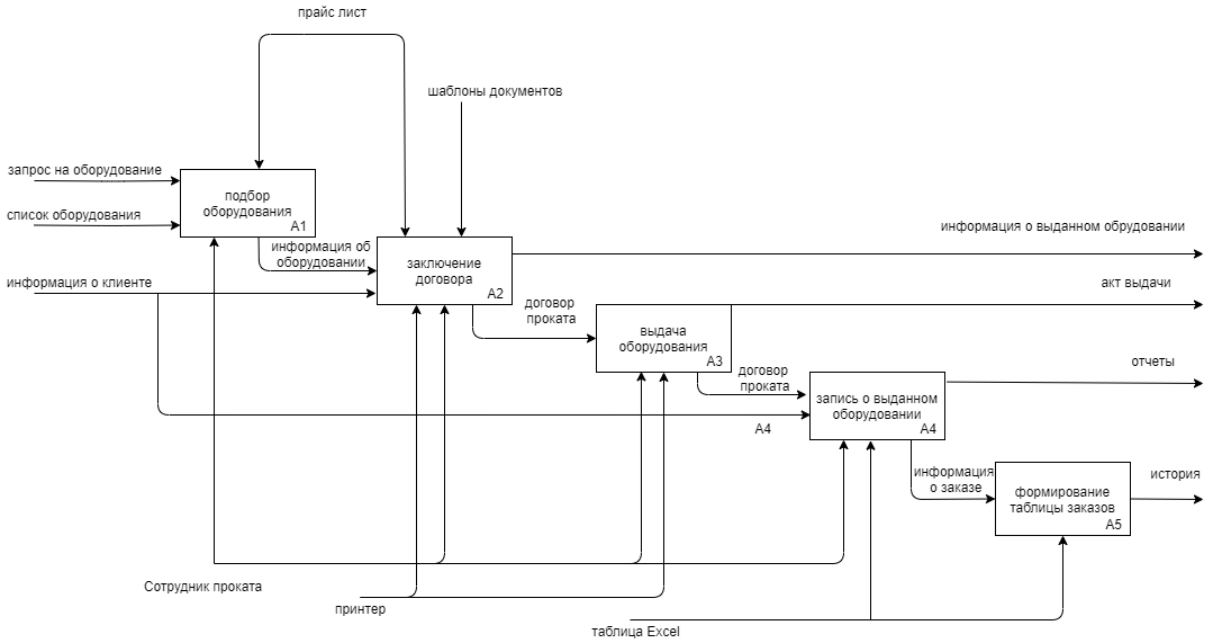

Рисунок 3 – Декомпозиция контекстной модели «КАК ЕСТЬ» второго уровня

Существующий бизнес–процесс деятельности отдела проката может быть описан следующим образом:

- подбор оборудования на основании запроса об оборудовании и списка оборудования, производится сотрудником проката (процесс A1);
- заключение договора на выбранное оборудование (процесс A2);
- выдача оборудования, опираясь на данные договора (процесс A3);
- запись о выданном оборудовании (процесс A4);
- формирование таблицы заказов (процесс A5);

Для более глубокого анализа проведем декомпозицию процесса A4 - запись о выданном оборудовании. Диаграмма декомпозиции представлена на рисунке 4.

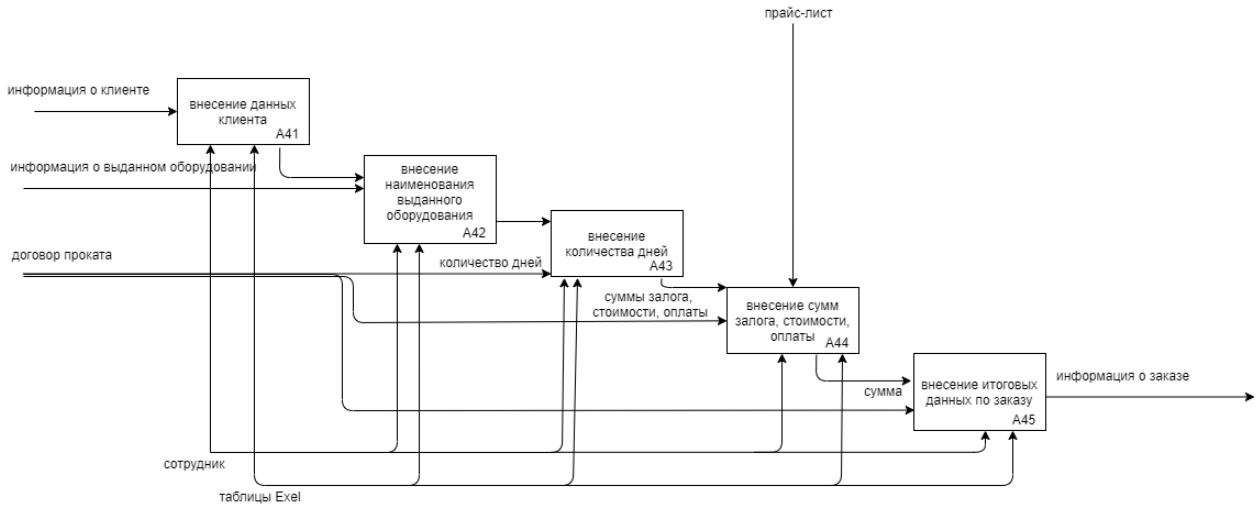

Рисунок 4 – Декомпозиция контекстной модели «КАК ЕСТЬ» третьего уровня

Как видно из схемы, процесс «запись о выданном оборудовании» состоит из следующих подпроцессов:

- внесение данных клиента;
- внесение наименования выданного оборудования;
- внесение количества дней, на которые оборудование берется в прокат;
- внесение сумм залоги, стоимости, оплаты;
- внесение итоговых данных по заказу.

Рассмотрим более подробно каждый процесс подробнее.

Процесс A41 «внесение данных клиента» подразумевает, что сотрудник получает необходимые данные от клиента и вносит их базу

Процесс A42 «внесение наименования выданного оборудования» подразумевает выбор сотрудником оборудования из имеющегося в наличии, подходящим под требования клиента.

Процесс A43 «внесение количества дней» это занесение в таблицу количества дней.

Процесс A44 «внесение сумм залоги, стоимости, оплаты» это занесение в таблицу информации о стоимости.

Процесс А45 «внесение итоговых данных по заказу» это внесение данных об итоговой сумме.

Анализ процессов показывает следующие недостатки:

- Данные о клиенте вносятся либо вручную, либо путем копирования из уже существующей ячейки (при этом сотрудник должен помнить, что клиент, действующий).
- Данные об оборудовании вносятся путем копирования из списка, затрачивается время на поиск. А так как список оборудования постоянно расширяется поиск требует времени.
- Все данные о заказах представляют собой таблицу с текущими и завершенными заказами, нет возможности посмотреть отдельно текущие заказы или историю заказов.
- Для доступа к информации других сотрудников в директора файл пересылается по почте, на это требуется время, это не оперативно.
- Нет общей клиентской базы в удобном виде, наличие которой было бы для бизнеса преимуществом.

После определения существующих недостатков перейдем к разработке анализу бизнес-процесса «КАК ДОЛЖНО БЫТЬ».

1.3.3 Разработка и анализ модели бизнес–процесса «КАК ДОЛЖНО БЫТЬ» деятельности подразделения проката

Для того чтобы отобразить как будет выглядеть предполагаемое состояние предметной области после внедрения решения, создается модель «КАК ДОЛЖНО БЫТЬ» на основе контекстной модели «КАК ЕСТЬ» с устранением ранее обнаруженных недостатков в текущей организации бизнес–процессов, а также добавляются бизнес–процессы для усовершенствования и оптимизации.

Структурно–функциональная модель «КАК ДОЛЖНО БЫТЬ» это также и техническое задание на создание или модернизацию информационной системы и представляет собой концептуальную модель усовершенствованного бизнес–процесса [14].

На рисунке 5 представлена модель учета «Как должно быть».

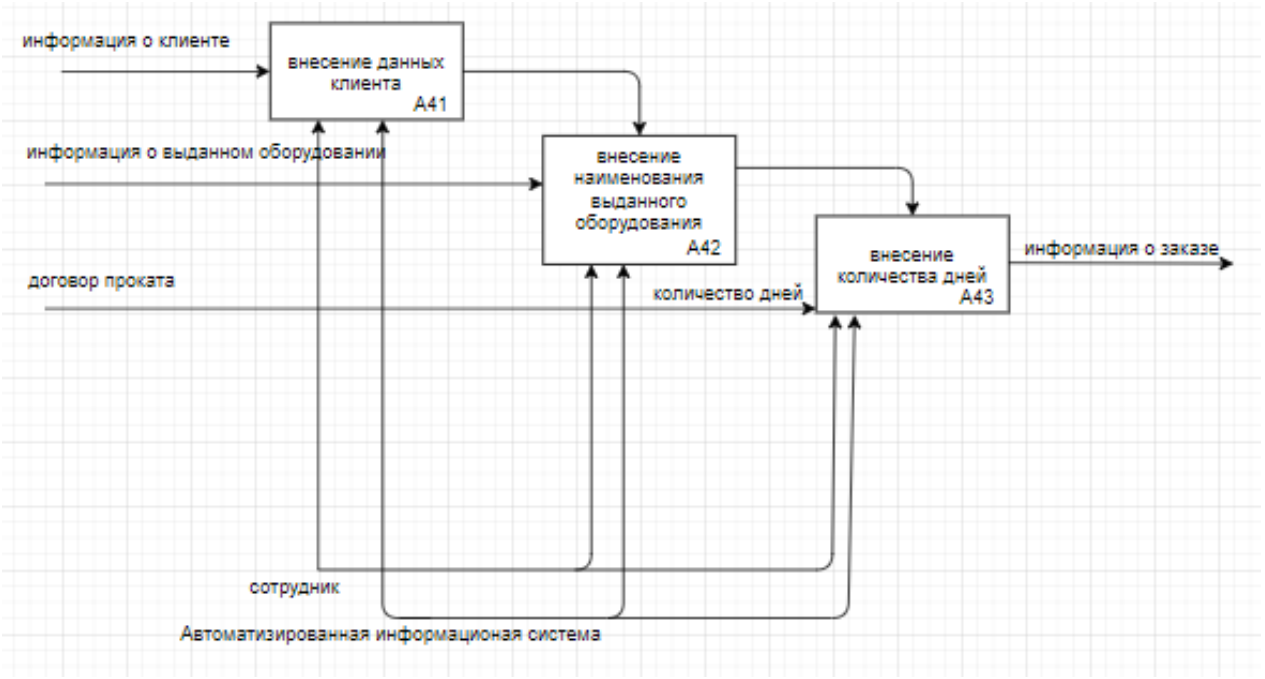

Рисунок 5 – Модель «КАК ДОЛЖНО БЫТЬ» для процесса учета заказов

Вместо таблицы Excel используется Автоматизированная информационную система, что сокращает процесс внесения данных по заказу. Если в модели «Как ЕСТЬ» процесс внесения данных о заказе состоял из пяти процессов, то в модели «КАК ДОЛЖНО БЫТЬ» только из трех. Процессы «внесение сумм залоги, стоимости, оплаты» и ««внесение итоговых данных по заказу» в модели «КАК ДОЖНО БЫТЬ» не требуются, так как они автоматизированы.

1.2.4 Сущность задачи автоматизации

Опираясь на вышеприведенный анализ, можно заключить что сущность задачи автоматизации состоит в создании автоматизированной информационной системы для учета заказов на прокат оборудования. Именно использование автоматизированной информационной системы позволяет улучшить бизнес–процессы и избавиться от описанных выше недостатков. Автоматизированная информационная система должна обеспечивать доступ к базе данных инструментов и оборудования, базе постоянных клиентов,

обеспечивать автоматическое заполнение данных о стоимости услуг проката, вычисление итоговой суммы, формировать отчеты для анализа и статистики.

1.3 Анализ существующих решений в области учета заказов пунктах проката

Проведем сравнительный анализ существующих решений для учета заказов в пунктах проката. Это нужно, чтобы точнее определить направления для дальнейшего проектирования веб–приложения, а также чтобы не избежать уже существующих ошибок при разработке.

Прокат того или иного оборудования, предметов быта, одежды, транспорта в целом довольно большая сфера, но логика процесса учетов заказов во все предприятия сектора проката сходная. Проанализируем решения, которые используются в сфере проката.

1.3.1 Ручной учет и учет в электронных таблицах

Вести учет аренды и проката непросто, без вспомогательных средств сотрудник вынужден был бы удерживать в голове большой объем информации: кто арендовал, когда срок возврата, какой внесен залог, какая оплата и прочее.

Оперировать таким количеством данных без вспомогательных средств было бы крайне затруднительно. Вспомогательные средства на относящиеся к автоматизированным информационным системам – это ручной учет и учет в таблицах Microsoft Excel.

«Ручной» учет или отказ от любой автоматизации – это такой учет, при котором используется ручка и тетрадь. Характерен для совсем небольших фирм и удобен только на первоначальном этапе. Основное достоинство – отсутствие материальных затрат (кроме минимальных затрат на покупку канцтоваров), отсутствие необходимости в оборудовании (компьютера), не нужно обучать сотрудника, ведущего учет навыкам работы с компьютером.

Недостатки очевидны: невозможность вести статистику, сложность ориентироваться в записях, выполненных другим человеком, ручной труд, ручные вычисления, затраты времени, путаница и прочее.

Учет в таблицах Exel также используется в небольших фирмах. Достоинства такого подходы:

- позволяет автоматизировать часть процессов (использовать функции копировать – вставить, использование формул для подсчета, возможность сортировки, автопоиска, автосуммы);
- незначительные финансовые затраты;
- гибкость;
- быстрое обучение.

Это несомненно лучше ручного способа, но обладает рядом уже упомянутых недостатков.

1.3.2 Использование CRM систем с настройкой под нужды под пункты проката

Для нужд проката можно настроить популярную систему управления отношениями с клиентами или CRM-систему. CRM-система — это прикладное программное обеспечение для предприятий, которое призвано автоматизировать стратегии взаимодействия с клиентами. Крупные корпорации в сфере проката, например, сети проката автомобилей, часто используют в работе «1С CRM», «Битрикс24» и подобные решения. Не крупные игроки могут использовать менее дорогие CRM, которые можно настроить под нужды пунктов проката, например, «Yclients», «Salesforce CRM», «Vtiger CRM», «SugarCRM», «Highrise» и прочие. Некоторые из них имеют уже готовые решения, подходящие для бизнеса проката, например, «Yclients» предлагает программу для проката инвентаря. На рисунке 6 представлен скриншот предложения «Yclients», на котором программа проката инвентаря охарактеризована как «мощная и интуитивно понятная платформа автоматизации».

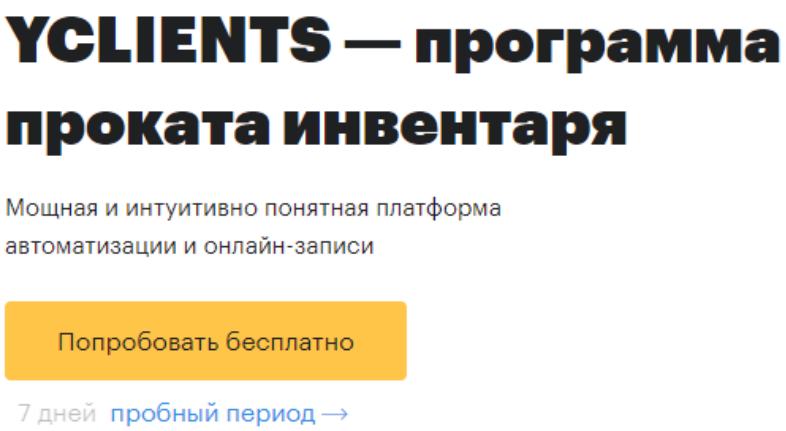

Рисунок 6 – Скриншот предложения YCLIENTS – программы проката инвентаря

Достоинства такого подхода:

- гибкость настроек;
- множество функций;
- поддержка пользователей;

Недостатки, часто упоминаемые в отзывах пользователями:

- финансовые затраты на постоянной основе на оплату подписки или поддержки;
- необходимость специального обучения сотрудников работе с программой;
- перегруженный интерфейс.

Для многих небольших пунктов проката перечисленные недостатки критичны, поэтому такой подход используется не часто.

1.3.3 Использование систем/приложений, специально разработанных для бизнеса проката

Как уже упоминалось, рынок проката достаточно обширный и учет заказов в пунктах проката имеет сходную логику независимо от того, что сдается в аренду. Поэтому на рынке приложений возникли предложения, которые разработаны именно для пунктов проката.

Популярные решения в данной области – это, например, приложения «АрендаSoft», «Rent in Hand», «Прокат Эксперт», «Easy Pro» и некоторые другие.

В качестве примера рассмотрим приложение «Прокат Эксперт».

Программа «Прокат–Эксперт» создана для автоматизации пунктов проката любых предметов и оборудования. Программа производит учет выдаваемых предметов, платежей и взаиморасчетов с клиентами и подготавливает к печати документы и отчеты. Интерфейс программы представлен на рисунке 7.

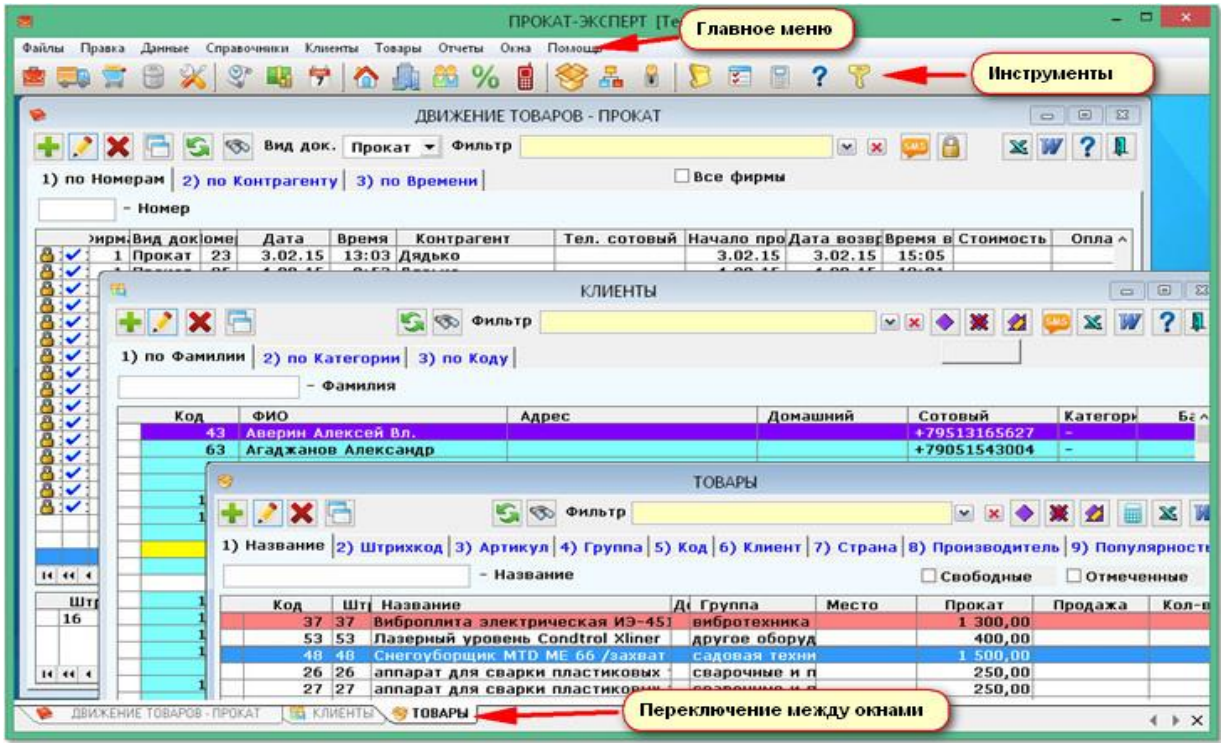

Рисунок 7 – Скриншот Общий вид программы проката «Прокат Эксперт»

Как видно из скриншота, программа имеет обширное главное меню и меню инструментов. Работа в программе может проводиться с применением технологий штрихового кодирования и членских карточек. В программе существует система разделения доступа разных сотрудников к различным

функциям и отчетам. Существует возможность печатать этикетки товаров, а также карточки для постоянных клиентов.

Для работы с разными базами данных используются разные окна, например, движение товара можно посмотреть в разделе, скриншот которого представлен на рисунке 8.

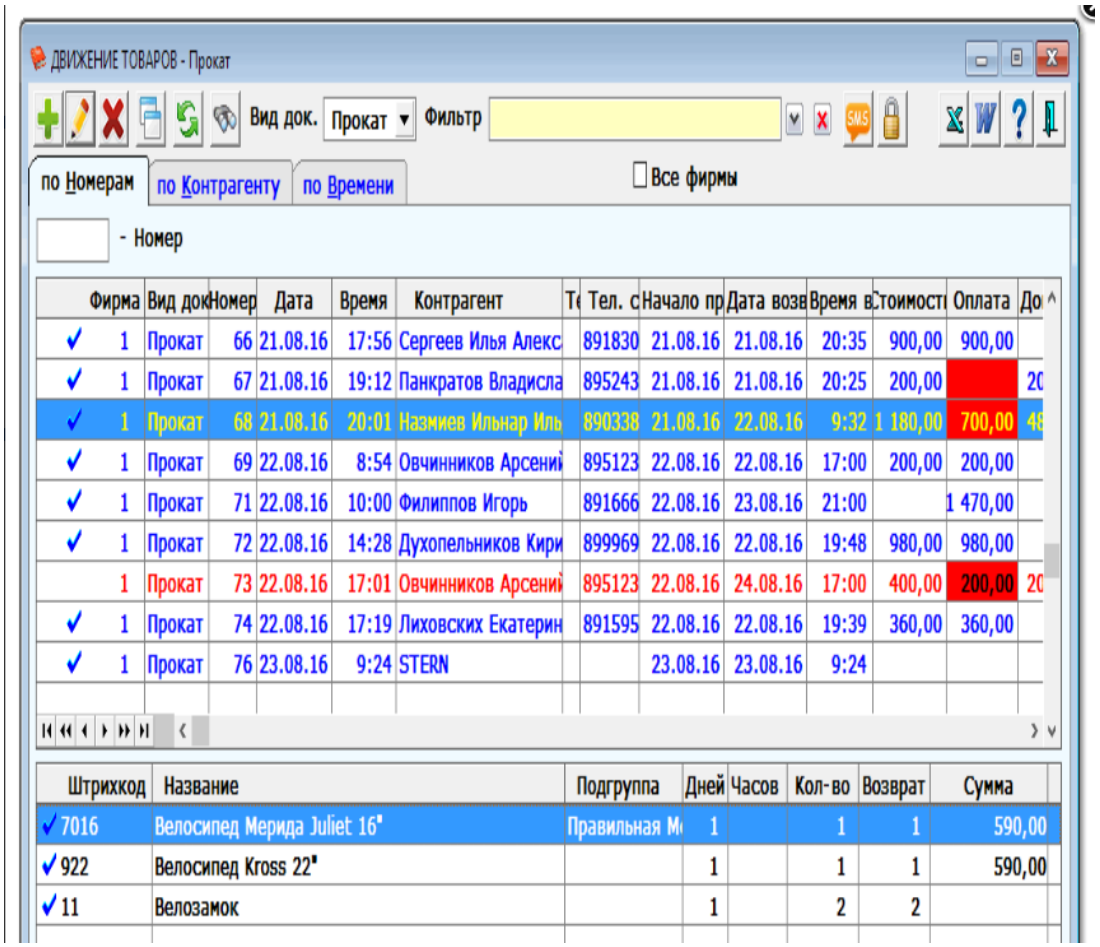

Рисунок 8 – Скриншот программы Прокат Эксперт

Скриншот представленный на рисунке 8 демонстрирует что пользователь может совершать различные операции с базой клиентов, например, изменять, редактировать, фильтровать.

Достоинства этого и подобных решений:

- настроено под нужды бизнеса проката;
- цена меньше, чем CRM широкого профиля;

- широкие возможности для анализа и статистики.

Недостатки:

- значительные финансовые затраты;
- обилие функций, которые не будут использоваться в небольшом бизнесе, поэтому перегруженный интерфейс только затрудняет работу;
- необходимость обучения сотрудников;
- больше вероятность ошибок.

Сравнение вышеперечисленных решений приведено в таблице 2.

Таблица 2 – Сравнение существующих решений для учета заказов в пунктах проката

Критерии	Ручной учет	Таблицы Exel	1С-CRM	Yclients	Прокат-эксперт
Низкие финансовые затраты	да	да	нет	нет	нет
Простой интерфейс	да	нет	нет	нет	нет
Быстрое обучение сотрудников	да	да	нет	нет	нет
Гибкость настроек	нет	нет	да	да	да

Как видно из сравнительной таблицы, каждый из вышеописанных подходов имеет свои недостатки, наиболее критичные из которых для небольшого пункта проката, каким является ООО «Айсберг», это в первую очередь:

- значительные финансовые затраты;
- слишком много функций, сложный интерфейс;
- необходимость дополнительно обучать сотрудников.

Информацию о данных недостатках необходимо учитывать при разработке автоматизированного решения задачи.

1.4 Цель и назначение автоматизированного варианта решения задачи

Опираясь на проведенный анализ, учитывая обнаруженные в других решениях недостатки, перейдем в формулировке цели и требований к будущему веб–приложению.

Цель создания информационной системы – это предоставление удобного инструмента для учета заказов в виде веб–приложения.

Для достижения поставленной цели были сформулированы основные требования к будущему веб–приложению:

- интуитивно–понятный и простой интерфейс без лишнего функционала;
- возможность добавлять новые заказы;
- создание и редактирование реестра клиентов;
- создание и редактирование реестра инструментов;
- возможность просмотра истории заказов.

Таким образом, с помощью создаваемого веб–приложения сотрудники проката смогут управлять заказами, а именно добавлять, удалять и редактировать заказы, создавать новые заказы, используя реестр клиентов и реестр оборудования, просматривать историю заказов.

Выводы по главе 1

В первой главе был проведен анализ предметной области, охарактеризовано предприятие, для которого будет разрабатываться АИС, выбрана технология концептуального моделирования и осуществлена разработка и анализ модели бизнес–процесса «КАК ЕСТЬ», после чего был сделан вывод о том какие процессы нуждаются в автоматизации.

Глава 2 Разработка и реализация проектных решений

2.1 Логическое моделирование предметной области

2.1.1 Выбор средств описания логической модели

Логическое моделирование предметной области можно охарактеризовать как анализ логики развития прогнозируемого объекта и создание на этой основе моделей–образов. Цель логического моделирования – сделать проектирование программы отдельным процессом, не связанным непосредственно с написанием кода. При использовании логического моделирования, повышается эффективность: уменьшаются сроки разработки, снижается число программных ошибок, программные модули подходят для повторного использования.

При описании логической модели разработанного веб-приложения будут использованы диаграммы вариантов использования и диаграммы классов.

2.1.2 Описание логической модели

При создании логической модели информационной системы перейдем от созданной контекстной модели «КАК ДОЛЖНО БЫТЬ» к диаграмме вариантов использования.

Диаграмма вариантов использования нужна для того чтобы выделить основные процессы, происходящие в системе, определить их взаимосвязь, также она способствует выделению функциональной структуры информационной системы. Диаграмма вариантов может описать функциональные возможности рассматриваемой информационной системы «КАК ДОЛЖНО БЫТЬ» и дает возможность для анализа информации об отношениях между различными вариантами использования и внешними пользователями–актерами, помогает описать типичные взаимодействия между пользователями системы и самой системой и предоставить описание процесса её функционирования. Разработанная диаграмма вариантов

использования для веб-приложения для обработки заказов в пункте проката представлена на рисунке 9.

Рисунок 9 – Диаграмма вариантов использования

На диаграмме вариантов использования, отображенной на рисунке 9 представлены следующие действующие лица (актеры):

- сотрудник отдела проката – пользователь веб–приложения, который управляет информацией об оборудовании, клиентах, заказах;
- директор предприятия – пользователь веб–приложения, который просматривает информацию о созданных заказах, истории заказов, клиентах с целью контроля, анализа, статистики, разработки маркетинговых и иных мероприятий.

В таблице 2 отражена характеристика прецедентов (или вариантов использования) диаграммы. Прецеденты описывают типичное взаимодействие между пользователями системы и самой системой и процессы ее функционирования.

Таблица 2 – Краткая характеристика прецедентов

Прецедент	Характеристика
Регистрация	Регистрация пользователя в системе
Добавление клиента	Пользователь добавляет информацию по клиенту: ФИО, адрес, телефон
Редактирование клиента	Пользователь редактирует информацию по клиенту
Удаление клиента	Пользователь удаляет информацию по клиенту
Добавление оборудования	Пользователь добавляет информацию по оборудованию: название, цена, залог, оплата
Редактирование оборудования	Пользователь редактирует информацию по оборудования
Удаление оборудования	Пользователь удаляет информацию по клиенту
Добавление заказа	Пользователь добавляет информацию по заказу: клиент, оборудование, заказ
Редактирование оборудования	Пользователь редактирует информацию по заказу
Удаление оборудования	Пользователь удаляет информацию по заказу
Завершение заказа	Пользователь завершает заказ и переносит его из текущих заказов в историю
Просмотр текущих заказов	Пользователь просматривает текущие заказы
Просмотр истории	Пользователь просматривает историю заказов

Для того чтобы отобразить структуру информации в контексте классов и их взаимосвязей разрабатывается диаграмма классов, которая является дальнейшим развитием концептуальной модели проектируемой системы, она предназначена для отображения классов, а также их атрибутов и операций, и связей между классами.

Класс в языке UML предназначен для обозначения множества объектов, с одинаковой структурой, поведением и отношениями с объектами из других классов. Графически класс изображается в виде прямоугольника, разделенного на 3 блока горизонтальными линиями:

- имя класса,
- атрибуты (свойства) класса,
- операции (методы) класса.

Для атрибутов и операций может быть указан один из трех типов видимости:

- частный,

- защищенный,
- общий.

Рассмотрим диаграмму классов, представленную на рисунке 10.

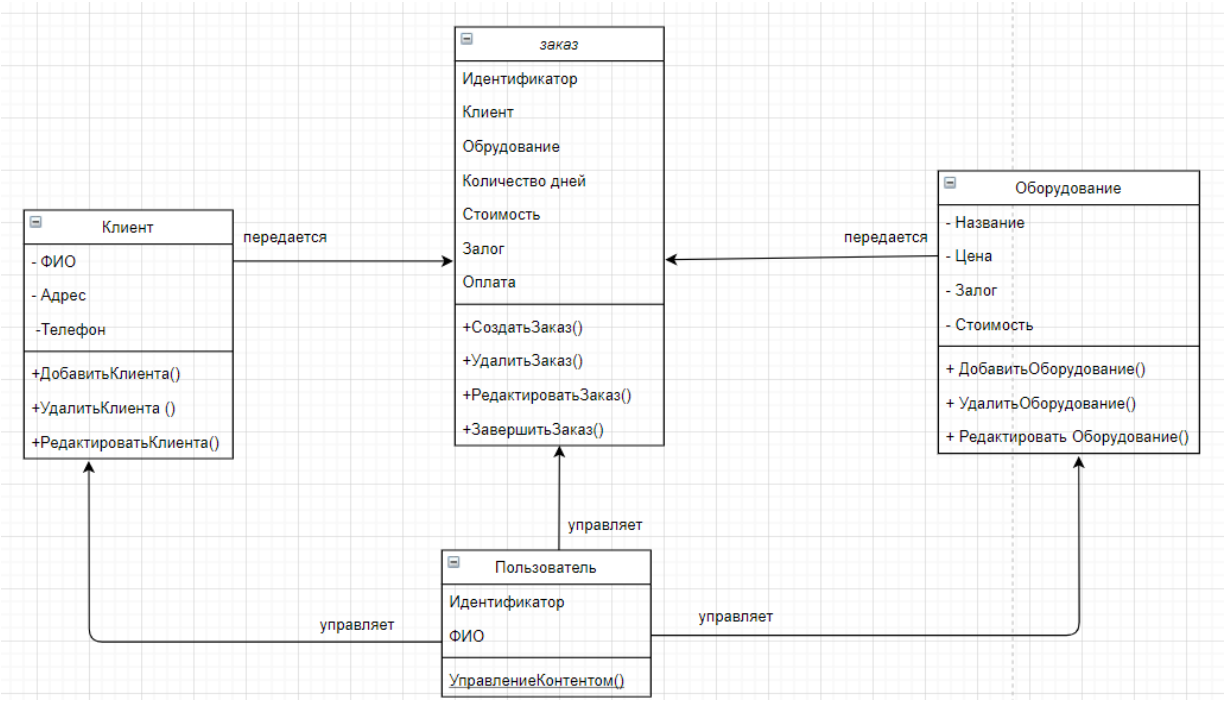

Рисунок 10 – Диаграмма классов

На диаграмме представлены следующие классы:

Класс «Клиент» содержит атрибуты: ФИО, адрес, телефон и методы для управления: добавить, удалить, редактировать.

Класс «Оборудование» содержит атрибуты название, цена, залог, стоимость проката методы для управления: добавить, удалить, редактировать

Класс «Заказ» содержит атрибуты – клиент, оборудование, количество дней, стоимость, залог, оплата.

Класс «Пользователь» содержит атрибуты идентификатор, ФИО и метод управления контентом.

Отношения между классами:

- «пользователь» управляет классом «клиент»,
- «пользователь» управляет классом «оборудование»,

- «пользователь» управляет классом «заказ»,
- класс «клиент» передается в класс «заказ»,
- класс «оборудование» передается в класс «заказ».

Таким образом, были определены основные классы, их атрибуты взаимосвязи.

2.1.3 Характеристика нормативно-справочной и входной информации.

Для функционирования большинства автоматизированных систем используется нормативно-справочная информация, это условно-постоянная часть которая существенно не изменяется в процессе повседневной деятельности предприятия. К нормативно-справочной информации относят, например, словари, справочники, классификаторы.

Справочники - это особенная группа наборов данных, которая систематизируют данные разных типов. При внесении данных в справочник пользователь имеет возможность добавить только те значение, которые присутствуют в этом справочнике.

Для разработанного веб-приложения справочной информацией являются справочник «Инструменты», который содержит информацию об инструментах и оборудовании для проката: наименование инструмента, стоимость, залог, оплата одного дня проката, а также справочник «Клиенты», содержащий данные клиентов – фамилию, имя, отчество, адрес, телефон.

Справочник «Инструменты» заполняется один раз при внедрении в работу веб-приложения, в него заносится информация об инструментах и оборудовании, которые сдаются в прокат. В дальнейшем справочник пополняется только в случае закупки нового оборудования.

Справочник «Клиенты» также заполняется при внедрении в работу веб-приложения, в него вносятся данные о всех клиентах проката. В дальнейшем справочник пополнятся в том случае, если клиента еще нет в базе.

Под входной информацией понимается вся информация, необходимая для решения задачи веб-приложением.

Входная информация для веб-приложения поступает из договора на прокат оборудования, заключённого на данный заказ. Входной информацией относительно веб-приложения для обработки заказов в пункте проката являются фамилия клиента, наименование оборудования, количество дней, на которые оборудование берется в прокат. Для сбора входной информации используется диалоговая форма, содержащая поля: «Клиент», «Инструмент», «Количество дней».

2.1.4. Характеристика результатной информации

В ходе работы веб-приложения формируется результатная информация в виде таблиц: «Заказы», «История заказов».

Таблица «Заказы» отражает информацию по текущим заказам и содержит в себе следующие данные:

- номер заказа,
- фамилию, имя, отчество клиента,
- название инструмента,
- сумма залога,
- количество дней проката,
- стоимость одного дня проката,
- итоговую сумму оплаты.

Данная информация содержит оперативную информацию, необходимую в работе сотрудникам проката – какое оборудование сдано в прокат, у каких клиентов оно находится.

Таблица «История Заказов и содержит в себе данные аналогичные таблице «Заказы» и отражает информацию по всем заказам, как открытым, так и закрытым. Данная информация предназначена для руководства (директора пункта проката) и позволяет контролировать работу, анализировать статистику спроса и финансовые результаты деятельности пункта проката.

Таким образом, в рамках параграфа была разработана логическая модель веб-приложения для учета заказов в пункте проката и

охарактеризована справочная, входная и результатная информация, используемая при работе веб-приложения.

2.2 Физическое моделирование веб–приложения для обработки заказов предприятия проката оборудования

2.2.1 Выбор архитектуры веб-приложения учета заказов

Архитектура информационной системы – это распределение функций по подсистемам и компонентам, она определяет границы подсистем и взаимодействие подсистем между собой.

Проанализируем известные архитектуры.

При файл–серверной архитектуре для хранения программы и данных используются сетевой ресурс. Данные и код программы хранит сервер, а обработка данных осуществляется на стороне клиента. Хранение и обработка данных при этом осуществляется в разных местах, данные приходится передавать по сети.

Недостатки такой архитектуры – она имеет низкую производительность и низкую надежность, а также слабые возможности расширения.

Другой вариант архитектуры «Клиент-серверная архитектура» Это вычислительная или сетевая архитектура, в которой задания, и сетевая нагрузка распределены между серверами и клиентами Чаще всего клиенты и серверы взаимодействуют через компьютерную сеть и могут быть как различными физическими устройствами, так и программным обеспечением. Как пример, удаленная база данных может быть расположена на компьютере–сервере сети, а приложение, работающее с этой базой данной, расположено на компьютере пользователя.

В архитектуре «клиент–сервер» клиент отправляет запрос на предоставление данных и получает только запрашиваемые данные,

обработка запросов при этом осуществляется на удаленном сервере. Такая архитектура обладает следующими достоинствами:

- меньше нагрузка на сеть из–за меньших объемов запрашиваемой информации;
- повышение безопасности информации, связанное с общими для всех пользователей правила использования базы данных;
- менее сложные клиентские приложения;

Недостатками такой архитектуры являются:

- если не работает сервер, то не будет работать вся вычислительная сеть
- более сложное администрирование;
- более высокая стоимость оборудования.

Еще один вариант - это Трехуровневая клиент–серверная архитектура, то есть архитектурная модель программного комплекса, предполагающая наличие в нём трёх компонентов: клиента, сервера приложений (к которому подключено клиентское приложение) и сервера баз данных (с которым работает сервер приложений).

Клиент является интерфейсным компонентом, с которым оперирует конечный пользователь. На сервере приложений сосредоточена логика работы с данными. Сервер баз данных нужен для хранения данных.

В простых конфигурациях все компоненты могут быть совмещены на одном вычислительном узле. В сложных конфигурациях используют выделенный вычислительный узел для сервера баз данных или кластер серверов баз данных, для серверов приложений.

Данная архитектура обладает хорошей масштабируемостью, гибкостью, конфигурируемостью, а также имеет высокую надежность и безопасность. Все функции распределяются между сервером приложений и сервером баз данных, что обеспечивает высокую скорость работы. Данная архитектура имеет не высокие требования к скорости сети между клиентом и сервером приложений, низкие требования к техническим характеристикам и

производительности клиента. Очевидно, что именно трехзвенная клиент–серверная архитектура лучше всего подходит для разработки веб-приложения для учета заказов в пункте проката.

2.2.2 Описание архитектуры веб–приложения для обработки заказов предприятия проката оборудования

Итак, для разрабатываемого веб–приложения была выбрана трехзвенная клиент–серверная архитектура. Она состоит из трех звеньев: клиент – сервер приложений – сервер баз данных.

Первое звено – клиент, это веб–браузер (например, «Google Chrome», «Opera», «Mozilla Firefox», «Edge» и др.). Клиентская часть реализована с помощью Javascript фреймворка и представляет пользовательский интерфейс.

Второе звено – это серверная часть, представляет собой веб–сервер на платформе Node.js с использованием веб–фреймворка «Express» для создания программного интерфейса приложения.

Третье звено – сервер баз данных (слой данных). Для реализации сервера базы данных используется система управления базами данных MySQL.

Клиент взаимодействует с сервером приложений при помощи программного интерфейса приложения. Сервер приложений взаимодействует с сервером баз данных при помощи SQL–запросов.

После того как была выбрана архитектура информационной системы, перейдем к выбору подхода к разработке программного обеспечения информационной системы.

2.2.3 Выбор подхода к разработке веб–приложения для обработки заказов предприятия проката оборудования

Веб-приложения в отличие от десктопных приложений позволяют интернет–пользователям получить доступ к функционалу предоставляемого сервиса или инструмента, используя только браузер. Это в значительной мере экономит время пользователя, так как программу не нужно скачивать и устанавливать.

Существуют два основных подхода к разработке веб приложения, можно создавать многостраничные приложения (Multi Page Application, MPA) или одностраничные веб–приложения (Single Page Application, SPA).

Многостраничные приложения работают более традиционным способом. При изменении данных или выгрузке информации на сервер происходит рендеринг новой страницы в браузере. Многостраничные приложения, очевидно, требуют больше памяти чем одностраничные, и обычно нацелены на отображение большего количества контента.

Преимущества многостраничных приложений:

- возможность создать многоуровневое меню и привычные средства навигации;
- более простая SEO (Search Engine Optimization) оптимизация.

Недостатки многостраничных приложений:

- разработка MPA довольно сложна, так как она требует использования фреймворков как на клиентской, так и на серверной стороне;
- многостраничные приложения требуют больше ресурсов.

Одностраничные приложения – это приложения, которые функционируют в рамках браузера и не нуждаются в перезагрузке страницы или загрузки дополнительных страниц во время использования. Одностраничное приложение похоже на веб-страницу, оно подгружает и обновляет контент без перезагрузки, с помощью JavaScript. SPA отправляет запрос на разметку страницы и её контент, а после этого производит рендеринг конечного вида страницы непосредственно в самом браузере. Такого эффекта можно достигнуть благодаря фреймворкам JavaScript, таким как AngularJS, React.js, Vue.js, Ember.js и другие.

Преимущества одностраничных приложений:

- SPA характеризуются быстродействием, так как ресурсы, которые они используют (HTML, CSS и Javascript), загружаются один раз в

течение сессии использования приложения. После совершения действий изменяются только данные;

- гибкость и отзывчивость пользовательского интерфейса — так как веб-страница всего одна, проще построить насыщенный интерфейс;
- разработка SPA обычно быстрее и эффективнее. Не нужно писать отдельный код для рендера страницы на стороне сервера.
- если есть SPA, есть возможность быстро создать мобильное приложение.

Недостатки одностраничных приложений:

- сложная SEO–оптимизация одностраничных приложений, так как контент приложений загружается, в то время как SEO–оптимизация базируется на устойчивости контента в каждой отдельно взятой странице;
- могут довольно долго загружаться, если требуется загрузка тяжелых фреймворков;
- SPA требуют, чтобы в браузере был активирован Javascript. Если кто–то из пользователей вручную отключит использование Javascript, то приложение не будет работать.

Однако, вышеуказанные недостатки одностраничных приложений не критичны при разработке веб-приложения для учета заказов в пункте проката. Для целей данного приложения не требуется SEO-оптимизация, ручное отключение Javascript исключено. Также относительно долгая загрузка не существенна, так как программа будет запускаться один раз утром рабочего дня. Исходя из этого в качестве подхода к созданию веб–приложения была выбрана разработка одностраничного приложения.

2.2.4 Выбор технологии разработки программного обеспечения веб–приложения для обработки заказов предприятия проката

2.2.4.1 Обзор преимуществ использования Javascript–фреймворков

В последние годы технологическое обеспечение в сфере веб-разработки сильно изменилась. Технологии, которые еще недавно были

передовыми как, например, JQuery, были заменены мощными Javascript–фреймворками и библиотеками, позволяющим разработчикам создавать интерактивные и удобные приложения.

Цель фреймворка или библиотеки – ускорить и облегчить процесс проектирования и сопровождения приложения [23].

Основные преимущества использования фреймворков;

- эффективность – проекты, которые раньше требовали много времени и большое количество кода, сейчас могут быть реализованы значительно быстрее с хорошо структурированными готовыми шаблонами и функциями;
- безопасность – Javascript фреймворки имеют систему безопасности и хорошо поддерживаются;
- расходы – большинство фреймворков с открытым кодом и бесплатны, они помогают программистам быстрее разрабатывать пользовательские решения и итоговая стоимость веб приложения ниже.

Есть несколько JS–библиотек, которые в больше степени подходят для создания одностраничного веб-приложения, кратко рассмотрим их.

2.2.4.12 Фреймворк Angular

Angular поставляется с большим списком функций, которые позволят разработать все, начиная от веб до десктопных и мобильных приложений. Фреймворк построен на Typescript от Microsoft. Структура этого фреймворка это комплекс сложных отношений между многими объектами, которые представляют собой отдельные уровни архитектуры кода. Код логически подразделяется на различные категории – инжекторы, компоненты, шаблоны, директивы и другие. Подобные функциональные блоки помогают разработчикам создавать легко масштабируемые приложения.

Angular имеет расширенный шаблонный синтаксис с множеством логических операторов, встроенных в макет. Эти операторы позволяют разработчикам создавать интерактивные макеты

В структуре фреймворка компоненты управляются из разных модулей, которые и образуют SPA. Связь между компонентами и модулями реализуется с помощью специальных объектов, называемых метаданными. В них определяется, какой шаблон, селектор, модуль и поставщики присутствуют у каждого компонента.

В Angular привязка данных двусторонняя, она автоматически изменяет данные в шаблоне, демонстрируемому пользователю, если изменяются данные в модуле. Также она функционирует и в обратном направлении – Angular изменяет данные в модуле, если пользователь взаимодействует с интерактивными элементами управления.

Angular отличает масштабируемость и архитектурной строгостью. Его часто называют лучшим вариантом для корпоративных приложений или для сред программирования с высокими стандартами к читаемости кода. При этом он считается одним из самых сложных веб–фреймворков.

2.2.4.3 Фреймворк React.JS

React это Javascript–фреймворк разработанный компанией Facebook. Цель фреймворка сделать манипулирование данными на веб-странице как можно более быстрым. Подход React можно описать как разделение всей веб–страницы на компоненты. Компоненты могут быть вложены друг друга, но каждый из них будет иметь свой собственный контекст данных. Это облегчает обновление информации на веб–странице путем рендеринга компонента в зависимости от динамически изменяемых данных [20].

В React используется синтаксис разметки, называемый JSX, что является его преимуществом. JSX очень похож на HTML, и при этом он имеет некоторые дополнительные возможности. Фреймворк позволяет создавать динамическую компоновку данных в разметке страницы и встраивать компоненты в разметку HTML. С помощью JSX можно встраивать функции на языке Javascript в макет для вывода динамических данных.

Каждый компонент имеет свое собственное состояние, состояние нужно для хранения различных параметров компонента, в зависимости от которых он рендерится веб–браузером.

В React встроена специальная логика для управления деревом компонентов, которую называют «Virtual DOM». Данные, как и обычный DOM, хранятся в виде дерева. Но каждый раз, если состояние компонента изменяется, React вызывает его функцию рендеринга для перерисовки содержимого компонента. При этом он определяет, где данные должны быть изменены, и производит только выборочные изменения в «Virtual DOM» и в браузером DOM–дереве. Таким образом React делает наименьшее возможное количество манипуляций с DOM, что значительно ускоряет взаимодействие с веб-приложением [12].

Функциональность этого фреймворка также легко расширяется с использованием сторонних библиотек.

2.2.4.4 Фреймворк Vue.js

Vue.js – это JavaScript библиотека для создания веб-интерфейсов с использованием шаблона архитектуры «Model – View – ViewModel».

Фреймворк Vue.js использует виртуальный DOM, а также поддерживает серверный рендеринг.

Vue.js работает только на «уровне представления» и не используется для промежуточного программного обеспечения и бэкэнда, поэтому он может легко интегрироваться с другими проектами и библиотеками. В Vue.js представлена широкая функциональность для уровня представлений, и он хорошо подходит как для небольших, так и объемных проектов.

2.2.4.5 Обоснование выбора Javascript фреймворка для разработки веб–приложения для обработки заказов предприятия проката

Вышеописанные фреймворки являются наиболее эффективными для одностраничных приложений. Все они имеют свои преимущества, но для реализации этого проекта решено было использовать React.JS, так как он имеет несколько особых преимуществ.

Во–первых, React.JS обеспечивает удобную архитектуру, управляемую компонентами, которая упрощает рендеринг.

Во–вторых, эта структура повышает производительность веб–сайта с использованием JSX и Virtual DOM. Для данного проекта более удобно использовать встраивание компонентов JSX, предлагаемое React, поскольку модульная система, используемая в Angular, слишком сложна для такого проекта, как веб-приложение для обработки заказов в пункте проката.

В–третьих, React – это легкая библиотека, и это хорошее решение для нужд этого проекта, в то время как функциональность, например, Angular была бы чрезмерной.

2.2.4.6 Обоснование использования фреймворка Bootstrap для разработки веб-приложения для обработки заказов предприятия проката

Bootstrap – один из самых популярных инструментов, который используется при создании сайтов и веб–приложений, это открытый и бесплатный HTML, CSS и JS фреймворк, который используется для быстрой разработки адаптивных дизайнов сайтов и веб–приложений. Bootstrap это инструмент, позволяющий быстро создать сайт или приложение из стандартных блоков [18].

Bootstrap включает в себя множество разных готовых компонентов для веб–сайтов: типографику, веб–формы, кнопки, блоки навигации и другое. Разработчику на нужно создавать верстку и стили для традиционных элементов веб–страницы или веб–приложения [30].

Преимущества использования Bootstrap:

- использование Bootstrap значительно ускоряет и упрощает процесс разработки. Bootstrap дает готовые решения и их применение позволяет сократить время, затрачиваемое на верстку;
- Bootstrap позволяет создавать адаптивные сайты. Это значит, что дизайн сайта будет правильно отображаться на экранах устройств разных размеров;

- страницы, сделанные с использованием Bootstrap, будут правильно отображаться во всех современных браузерах;
- Bootstrap легко использовать в разработке, в нем легко разобраться.

В сочетании с React удобно использовать библиотеку «react bootstrap», это популярная библиотека, которая позволяет использовать элементы bootstrap в стиле React. В «классическом» Bootstrap для стилизации используются классы, а в этой библиотеке можно использовать сразу компоненты. На рисунке 11 представлен фрагмент кода, разработанного веб–приложения для управления заказами в пункте проката, который демонстрирует использование компонента <Modal> для создания модального окна и компонента <Form> для создания формы.

```
<Modal show={this.props.customers.customerModalShow} onHide={this.props.actCustomerModalHide}>
    <Modal.Header>
        <Modal.Title>Add New Customer</Modal.Title>
    </Modal.Header>
    <Modal.Body>
    </Modal.Body>
    <Form className="add-form">
        <FormGroup>
            <ControlLabel className="">Customer Name:</ControlLabel>
            <FormControl type="text" placeholder={this.props.customers.editingCustomer === 0 ? "Input Customer Name": thi
                         onBlur={this.props.actChangeInputCustomerValue}
                         name="customerName"/>
        </FormGroup>
```

Рисунок 11 – Пример кода с использованием библиотеки react–bootstrap

Данный фрагмент кода наглядно демонстрирует, что с использованием библиотеки «react bootstrap» сложные элементы могут быть реализованы небольшим количеством кода.

2.2.4.7 Обоснование использования сборщика проекта Webpack

В современной разработке используется модульный подход, это подразумевает, что код делится на отдельные модули. Для того чтобы объединить их в один файл и загрузить на сайт используется инструменты сборки или бандлеры. В последние годы это неотъемлемая часть веб-разработки, в основном из–за все возрастающей сложности приложений.

Бандлеры позволяют упаковывать, компилировать, организовывать множество ресурсов и библиотек, необходимых для современного веб–проекта.

Одним из самых мощных и гибких инструментов для сборки frontend части проекта является сборщик Webpack. Функционирование Webpack происходит следующим образом. При работе Webpack статически перемещается по всем модулям для построения графа и использует его для генерации одного бандла.

Бандл – это файл JavaScript, содержащий код из всех модулей проекта и объединенных в правильном порядке. Когда Webpack создает граф зависимостей, он не выполняет исходный код, а объединяет модули и их зависимости в бандл.

При выполнении задач Webpack опирается на конфигурацию, в которой указано, как файлы и ресурсы следует трансформировать.

В соответствии с предоставленной конфигурацией Webpack запускается с точек входа и обрабатывает каждый модуль, с которым сталкивается, при построении графа зависимостей.

Если модуль содержит зависимости, процесс выполняется рекурсивно для каждой зависимости.

Webpack лучше всего подходит для сборки одностраничных приложений, разработанных с помощью Javascript-фреймворков и поэтому используется в описываемом проекте.

2.2.4.8 Обоснование использования библиотеки Redux

Redux – это библиотека с открытым исходным кодом для улучшения предсказуемости состояния в приложениях, написанных с использованием JavaScript и JavaScript-фреймворков. Это независимая библиотека, обычно она используется с другими библиотеками, такими как React и Angular, для лучшего управления состоянием приложения.

Когда приложение становится больше, становится сложнее управлять состоянием и отлаживать проблемы. Это становится проблемой, чтобы

отследить, когда и где состояние изменяется и где изменения должны быть отражены.

Redux решает эту проблему, предоставив несколько простых правил для обновления состояния, чтобы оно было предсказуемым.

Другими словами, Redux является менеджером состояний. Чаще всего данную библиотеку используют с фреймворком React, но он может быть использован в сочетании и с другими фреймворками.

В Redux общее состояние приложения представлено одним объектом JavaScript - state (состояние) или state tree (дерево состояний). Неизменяемое дерево состояний доступно только для чтения, оно не подлежит прямым изменениям. Изменения возможны только при отправке action (действия).

Redux это удобная библиотека и использование фреймворка React в сочетании с библиотекой Redux одно из самых распространённых решений при выборе технологий разработки. Поэтому данная библиотека была также использована для разработки веб-приложения для обработки заказов в пункте проката.

2.2.4 Функциональная схема проекта

Рассмотрим функциональную схему проекта, представленную на рисунке 12.

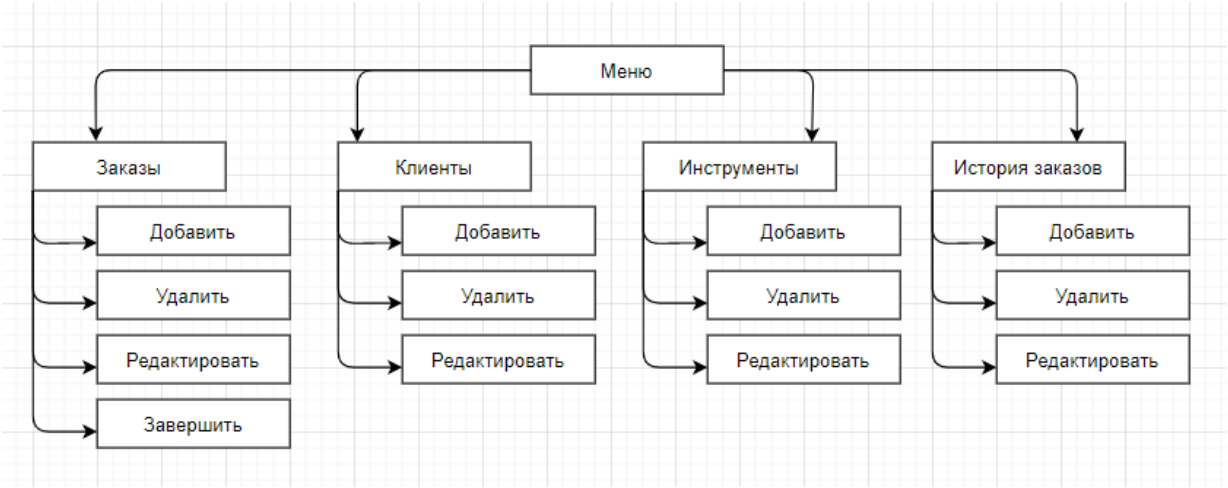

Рисунок 12 – Функциональная схема проекта

Как видно из схемы, пользователь из меню переходит в один из подразделов: «Заказы», «Клиенты», «Инструменты», «История заказов», в каждом из которых есть функции «Добавить», «Удалить», «Редактировать», а в подразделе «Заказы» также функция «Завершить» которая переводит заказ из текущих заказов в «Историю заказов»

2.2.5 Описание программных модулей и связи между ними

Перейдем к описанию программных модулей и связи между ними. Схема взаимосвязи программных модулей представлена на рисунке 13.

Разработанное приложение можно разделить на компоненты React и Редьюсер, который реализован с помощью библиотеки Redux. «Точка входа» в приложение это файл APP.js который содержит в себе модуль «Меню» и компоненты для отрисовки модальных окон – окно добавления нового клиента, окно добавления нового инструмента, окно добавления нового заказа. Модальные окна по умолчанию скрыты и могут быть показаны вызовом соответствующей функции.

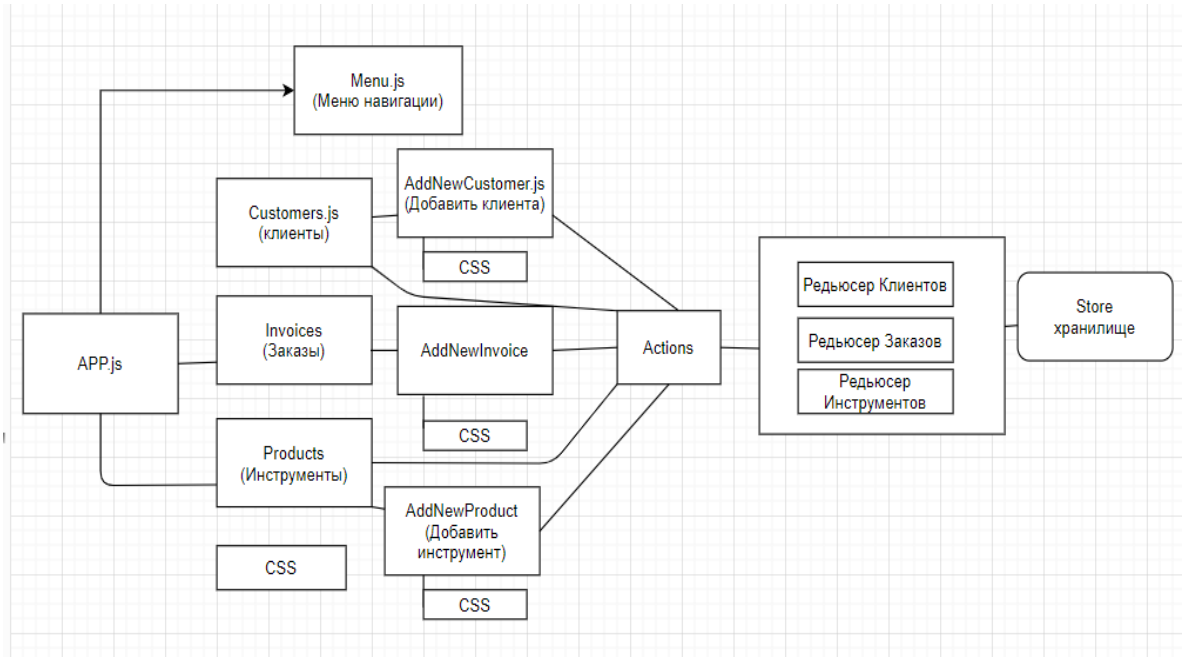

Рисунок 13 – Схема взаимосвязи программных модулей

Внутри компонента «Меню» вложены компоненты, которые отвечают за генерацию подразделов приложения: списка клиентов, списка инструментов, списка заказов. Переключение между ними реализовано с помощью библиотеки «react–router», которая отображает один из необходимых в данный момент компонентов и создает у пользователя впечатление переключения между страницами с помощью изменения URL в адресной строке.

Состояние приложения хранится в едином хранилище, обеспечиваемой библиотекой Redux и все изменения происходят через функции этой библиотеки.

Запросы к серверу – получение данных, удаление данных, редактирование данных реализованы с помощью библиотеки «axios», которая предоставляет удобный подход к работе с запросами.

Стилизация компонентов по большей части осуществляется внутри фреймворка Bootstrap, также разработаны 4 файла с кодом CSS для кастомизации стилей.

2.3. Технологическое обеспечение задачи

2.3.1. Организация технологии сбора, передачи, обработки и выдачи информации

Технологический процесс обработки информации это упорядоченная последовательность действий по обработке входных данных и информации, которая приводит пользователя к получению необходимого ему результата. Технологический процесс можно охарактеризовать как комплекс взаимосвязанных операций по преобразованию информации в соответствии с поставленной целью с момента входа в информационную систему до момента предоставлению информации пользователям.

Состав операций и последовательность их выполнения зависят от характера задач, которые должны быть решены и комплекса технических средств на каждом уровне обработки [17].

Технологический процесс обработки может включать следующие действия:

- сбор данных;
- обработка данных;
- генерация данных;
- хранение данных;
- передача данных.

Рассмотрим технологический процесс обработки информации о заказе. В рассматриваемом веб-приложении сбор данных о заказе производится в форме добавления нового заказа. Источником данных является раннее заключенный договор, который содержит информацию: данные клиента, наименование оборудования, которое берется в прокат, количество дней на оплату.

На следующем этапе данные обрабатываются: во-первых, проверяется полнота заполнения данных. Если данные заполнены не полностью, подтверждение отправки формы не активно. Во-вторых, на основании внесенных данных об оборудовании и количестве дней, данные дополняются сведениями о сумме залога, стоимости оплаты одного дня проката. Данная информация берется из справочника «Инструменты» путем запроса к базе данных «Инструменты» на сервере.

Далее рассчитываются данные об итоговой оплате следующим образом: суммируется произведение количества дней и стоимости одного дня с величиной залога. После чего происходит передача обработанной информации на сервер с последующим сохранением в базу данных «Заказы». Для генерации обновленных данных на странице производится запрос к базе данных «Заказы», данный запрос возвращает информацию о всех текущих заказах.

2.3.2. Схема технологического процесса сбора, передачи, обработки и выдачи информации

Рассмотрим схему технологического процесса сбора, обработки передачи и выдачи информации веб-приложения для обработки заказов, представленную на рисунке 14.

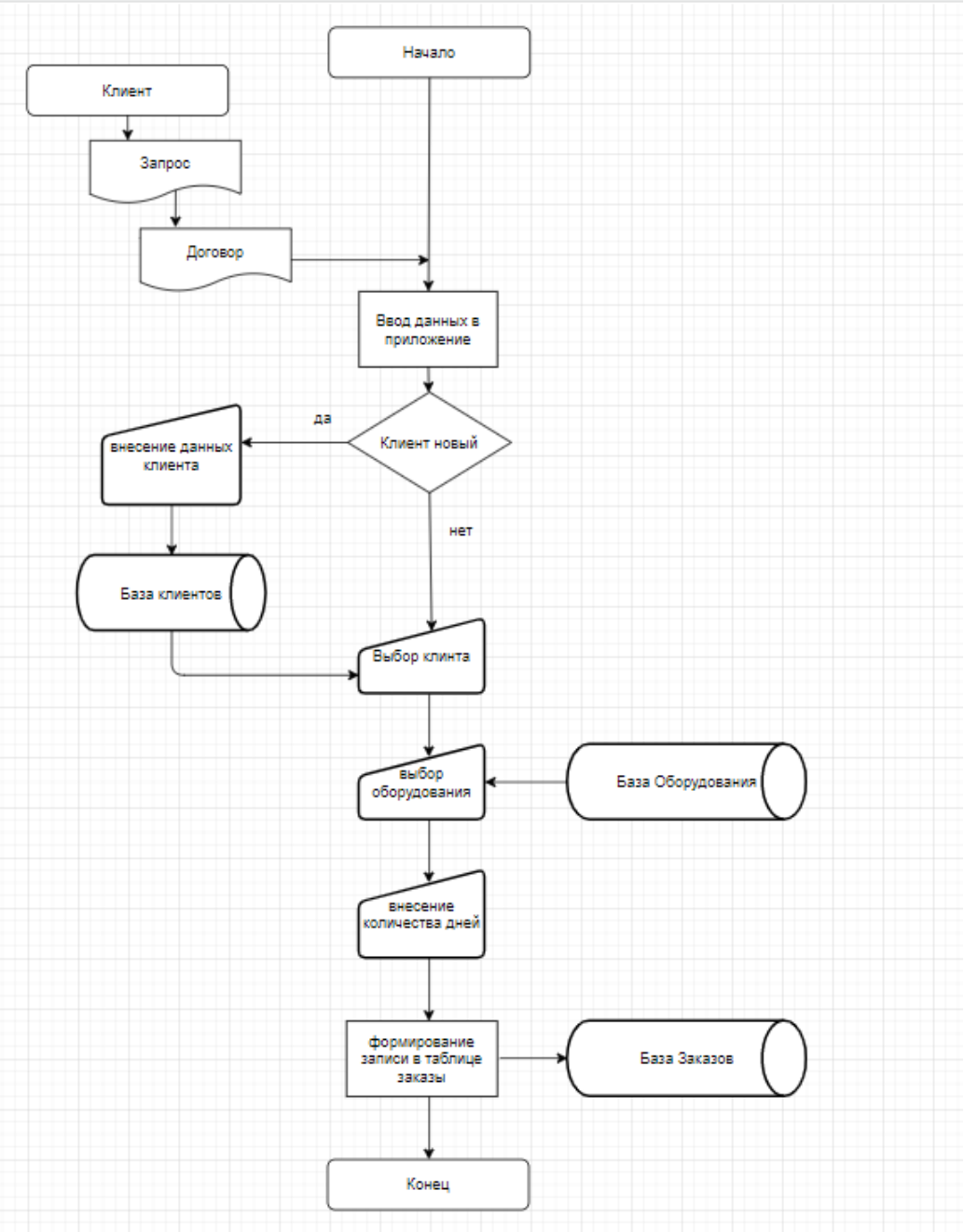

Рисунок 14 - Схема технологического процесса сбора, обработки передачи и выдачи информации

В начале работы пользователь вводит в приложение данные из договора, ранее заключенного с клиентом, на основе запроса клиента. Если клиент новый, пользователь переходит в базу данных клиентов и добавляет сведения о клиенте в форму. Если клиент на момент заказа уже есть в базе предприятия, пользователь выбирает клиента из базы клиентов, затем выбирает оборудование из базы оборудования, затем вносит количество дней на прокат оборудования, после чего информация о новом заказе передается в базу заказов.

После сохранения пользователем на странице происходит генерация данных из обновленной базы заказов.

Таким образом, в данном параграфе рассмотрено технологическое обеспечение задачи, а именно организация технологии сбора, передачи, обработки и выдачи информации и схема технологического процесса сбора, обработки передачи и выдачи информации

2.4. Контрольный пример реализации проекта и его описание

2.4.1 Описание функциональности веб приложения для обработки заказов предприятия проката оборудования

Перейдем к описанию функций разработанного веб-приложения для обработки заказов.

На рисунке 15 отображено меню разработанного приложения.

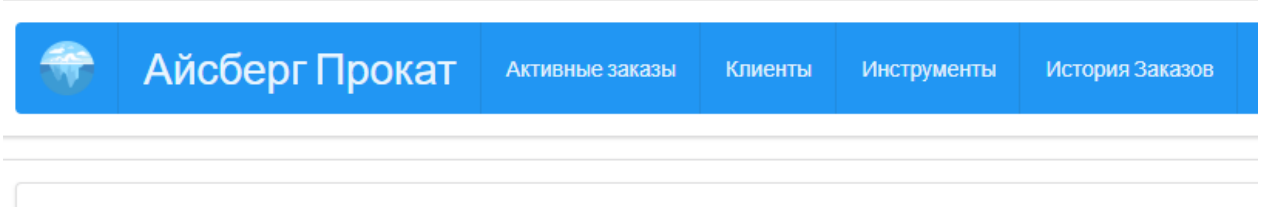

Рисунок 15 – Скриншот меню веб-приложения

Веб-приложение для обработки заказов состоит из 4 разделов: «Активные заказы», «Клиенты», «Инструменты», «История Заказов», переключение между которыми осуществляется в «Меню».

Первый раздел, скриншот которого представлен на рисунке 16 представляет собой реестр активных заказов, для перехода в который пользователь веб-приложения должен выбрать первый раздел «Активные заказы».

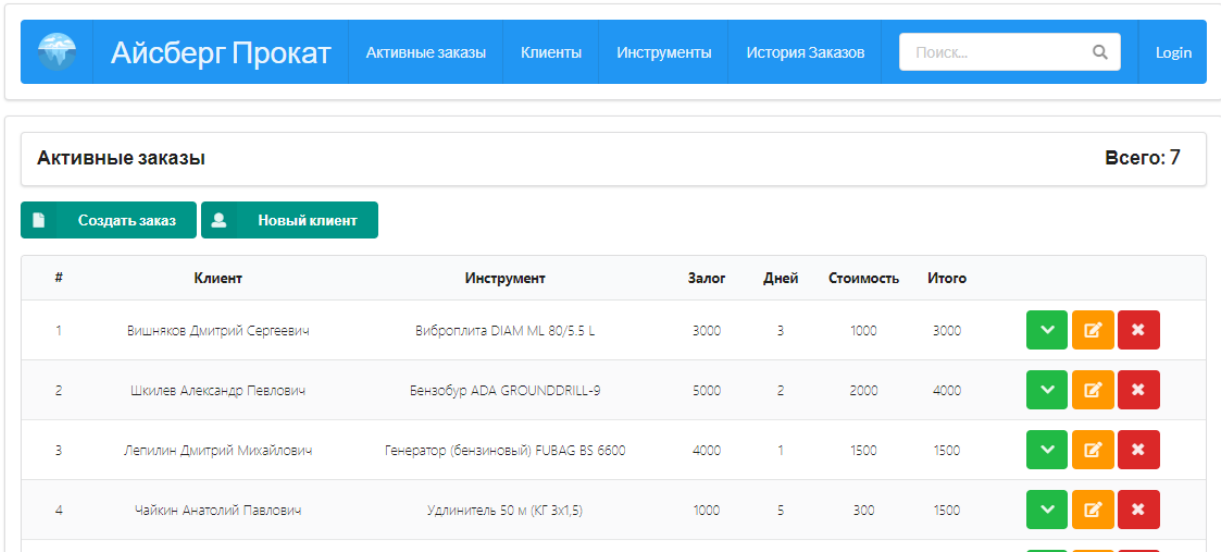

Рисунок 16 – Реестр активных заказов

В данном разделе пользователь веб-приложения может добавить новый заказ путем нажатия на кнопку «Создать заказ», в таком случае на экране всплывает модальное окно «Добавить новый заказ», что продемонстрировано на рисунке 17.

В меню данного модального окна пользователь выбирает из выпадающего списка фамилию клиента, название инструмента или оборудования и вносит количество дней, на которое инструмент выдан клиенту.

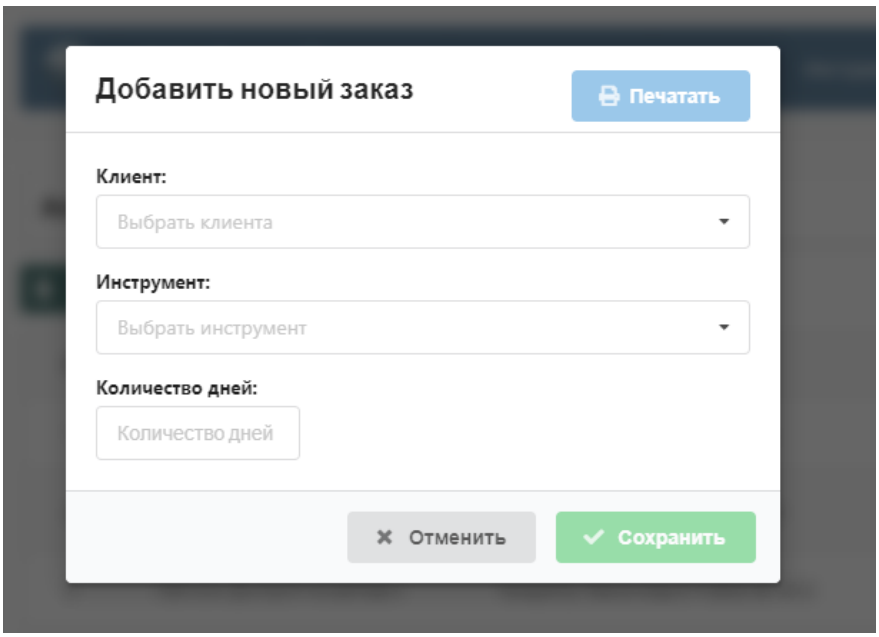

Рисунок 17 – Модальное окно добавления нового заказа

На рисунке 18 продемонстрировано модальное окно с заполненным полем «клиент» и показано, как пользователь может выбрать инструмент из списка.

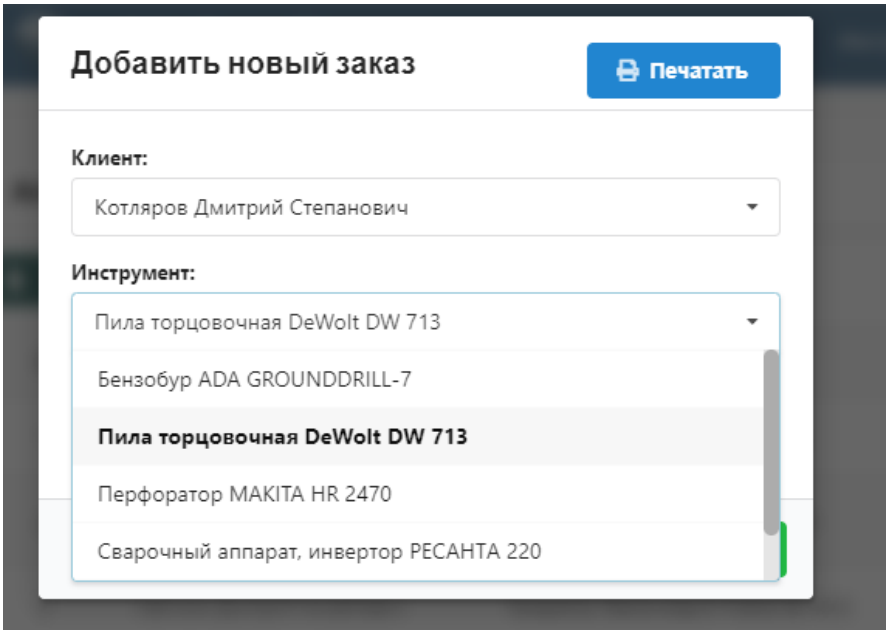

Рисунок 18 – Внесение данных в форму нового заказа

Нажатие кнопки «Отменить» отменяет действие и закрывает модальное окно, нажатие кнопки «Сохранить» отправляет внесенные данные на сервер и также закрывает модальное окно. После чего страница «Заказы» обновляется и добавленный заказ появляется в списке заказов.

#	Клиент	Инструмент	Залог	Дней	Стоимость	Итого	
1	Вишняков Дмитрий Сергеевич	Виброплита DIAM ML 80/5.5 L	3000	3	1000	3000	
2	Шкилев Александр Павлович	Бензобур ADA GROUNDDRILL-9	5000	2	2000	4000	
3	Лепилин Дмитрий Михайлович	Генератор (бензиновый) FUBAG BS 6600	4000	1	1500	1500	

Рисунок 19 – Кнопки обработки заказов

Как демонстрирует рисунок 19 возле каждого заказа есть кнопки обработки:

- кнопка «редактировать» – открывает существующий заказ в модальном окне и позволяет вносить изменения в существующие данные;
- кнопка «удалить» удаляет заказ;
- кнопка «завершить заказ» используется при возврате клиентом взятого в прокат оборудования, в таком случае заказ удаляется из раздела активных заказов, но сохраняется в базе заказов и отображается в разделе «История заказов».

Раздел «История заказов» имеет сходный интерфейс с разделом «Заказы» за исключением кнопки «Добавить заказ» и кнопок «Завершить заказ». Как упомянуто выше, раздел «История заказов» хранит все заказы для анализа и статистики.

Раздел «Клиенты» представляет собой справочник клиентов предприятия проката и отображен на рисунке 20.

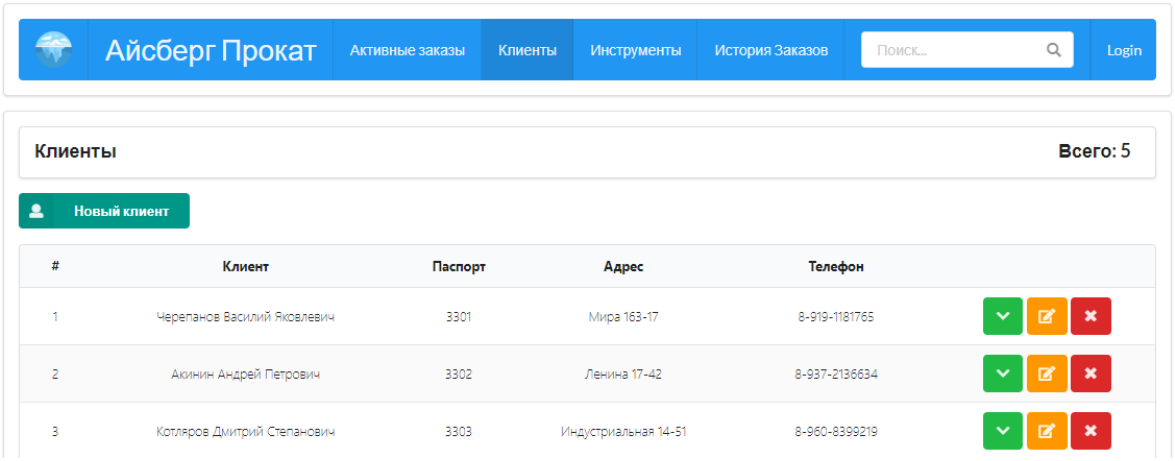

Рисунок 20 – Справочник клиентов

В данном разделе представлена таблица с именами, адресами и телефонами клиентов. Для каждой строки реализована возможность редактирования и удаления.

Кнопка «Новый клиент» вызывает модальное окно, показанное на рисунке 21.

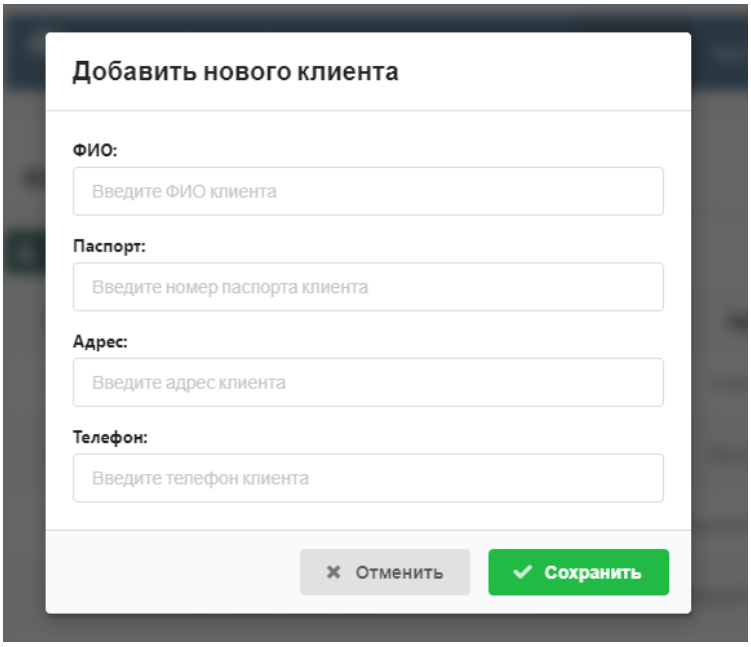

Рисунок 21 – Модальное окно добавления клиента

В данном модальном окне пользователь веб-приложения имеет возможность добавить нового клиента, путем заполнения полей «Имя», «Адрес», «Телефон».

Из меню веб-приложения пользователь также может перейти в раздел «Инструменты», скриншот которого представлен на рисунке 22. Данный раздел отображает справочник инструментов, которые могут быть выданы в прокат.

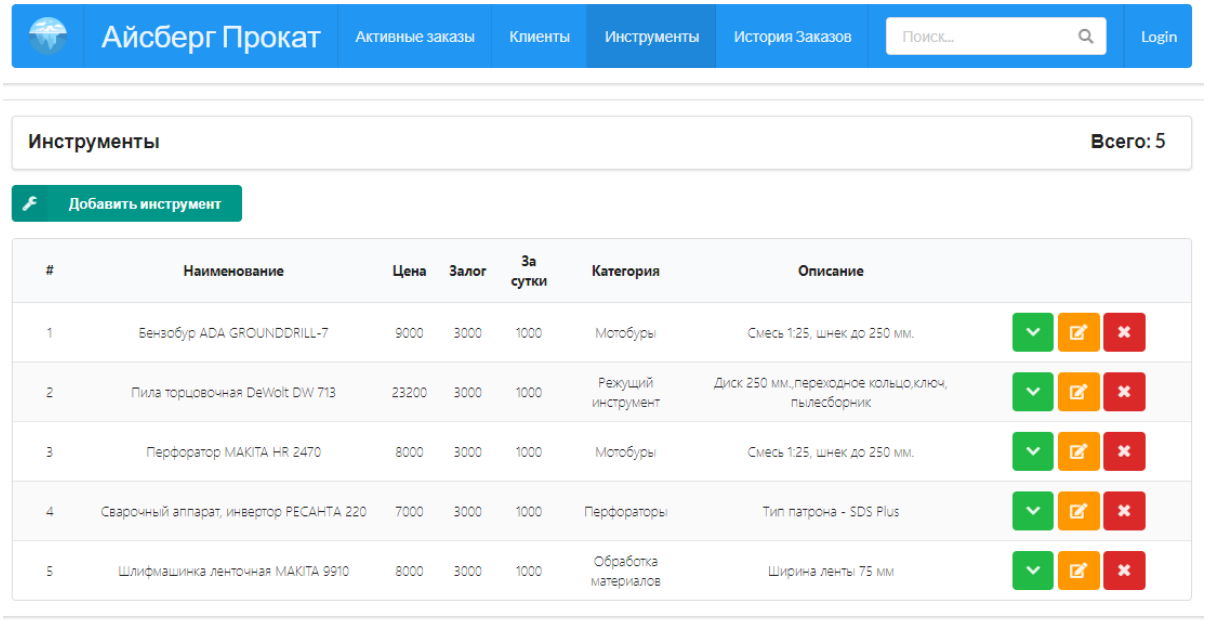

Рисунок 22 – Справочник «Инструменты»

Для каждого инструмента в данной таблице отображаются столбцы «Наименование», «Цена», «Залог», «За сутки», «Категория», «Описание», а также кнопки «Редактировать» и «Удалить».

Кнопка «Добавить инструмент», расположенная над таблицей открывает для пользователя модальное окно, скриншот которого представлен на рисунке 23.

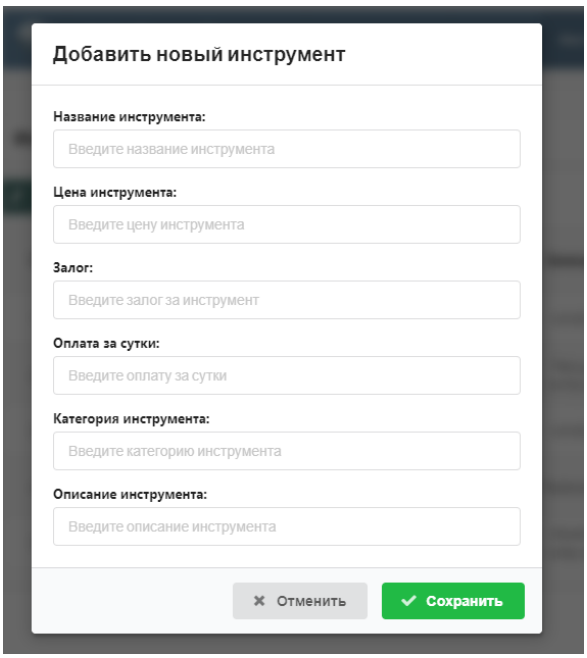

Рисунок 23 – Модальное окно добавления инструмента

В данном модальном окне можно добавить новый инструмент, путем заполнения полей. Кнопка «Отменить» закрывает модальное окно без сохранения данных. Кнопка «Сохранить» активна только после заполнения всех полей. При нажатии на кнопку «Сохранить» данные отправляются на сервер в базу данных «Инструменты», модальное окно закрывается, страница «Инструменты» перерисовывается с добавлением внесенного заказа.

2.4.2 Тестирование

2.4.2.1 Постановка задачи для тестирования

Для проверки функционала веб-приложения протестируем функционал.

Чтобы провести тестирование, определим задачи, которые должны быть успешно выполнены:

- зайти в приложение;
- проверить визуальную корректность таблиц;
- создать нового клиента;
- создать новый инструмент;
- создать новый заказ;

- выполнить во всех разделах операцию удаления;
- выполнить во всех разделах операцию редактирования;
- завершить заказ.

Результаты выполнения данных задач позволят оценить успешность реализации веб-приложения для обработки заказов.

2.4.2.1 Результаты тестирования

При выполнении тестирования получены следующие результаты:

- успешный вход в приложение;
- таблицы отображаются корректно, ошибок нет;
- при нажатии кнопок «Добавить ...» открывается модальное окно с формой;
- кнопка «добавить» активна только после заполнения всех полей в модальном окне;
- запись о новом клиенте успешно создается и добавляется в список заказов;
- запись о новом инструменте успешно создается и добавляется в список;
- новый заказ успешно создается и добавляется в список;
- операция удаления корректно функционирует во всех подразделах;
- операция редактирования успешно функционирует во всех подразделах.

Таким образом, проведенное тестирование свидетельствует об успешной реализации веб-приложения приложения.

Выводы по главе 2

Во второй главе осуществлено логическое проектирование предметной области. С помощью выбранного средства моделирования был построена диаграмма вариантов использования и диаграмма классов. Затем была описана нормативно-справочная, входная и результатная информация.

После чего были проанализированы популярные архитектуры информационных систем и была выбрана трехзвенная клиент–серверная архитектура. В качестве подхода была выбрана концепция одностраничного приложения. Затем был проведен анализ и подобраны инструменты для технической реализации: фреймворки React.js, библиотека Redux, фреймворк Bootstrap, и сборщик Webpack. После чего были описаны программные модули и схема взаимодействия между ними, а также технологическое обеспечение задачи. В завершающей части главы описан функционал разработанного приложения и результаты тестирования, которые позволяют говорить об успешном решении поставленной задачи.

Глава 3 Оценка и обоснование экономической эффективности проекта разработки и внедрения приложения по учету заказов в пункте проката

3.1. Выбор и обоснование методики расчета экономической эффективности

При разработке программного обеспечения важным этапом процесса является расчёт экономической эффективности проекта. Данный расчет дает возможность оценить влияние от внедрения в работу проекта на показатели деятельности организации.

Экономическая эффективность проекта - это мера, определяющая соотношения затрат и результатов функционирования разрабатываемого программного продукта [3]. Обоснование экономической эффективности от внедрения веб приложения дает возможность рассчитать срок окупаемости затрат на разработку веб-приложения.

Существует несколько методик для обоснования экономической эффективности, одна из наиболее популярных это сопоставление показателей деятельности до внедрения разрабатываемого продукта и тех показателей, которые будут достигнуты в результате использования разработанного программного обеспечения. Именно она и была использована для расчета эффективности описываемого проекта.

При оценке экономической эффективности веб-приложения по учету заказов на прокат оборудования целесообразно оценить трудовые и стоимостные показатели. Данные показатели дают возможность измерить экономию от внедрения разработанного веб-приложения для обработки заказов относительно прежнего варианта, при котором использовались таблицы Exel.

Экономическая эффективность разработки веб-приложения для обработки заказов в предприятии проката оборудования включает две составляющие:

- косвенный эффект, который характеризуется уменьшением ошибок сотрудников, удобством работы, доступом информации;
- прямой эффект, который характеризуется снижением трудовых и стоимостных показателей за счет сокращения времени и стоимостных затрат на внесения данных для учета заказов на прокат и сокращением трудоемкости работы.

Чтобы определить прямой эффект были рассчитаны показатели трудовых и стоимостных затрат.

К трудовым показателям относятся следующие:

- Абсолютное снижение трудовых затрат (ΔТ) (час), рассчитываемое по формуле:

$$\Delta T = T_0 - T_1, \quad (1)$$

где T_0 - это трудовые затраты на внесение информации по при использовании базового варианта (час),

T_1- трудовые затраты на внесение информации по заказу по проектному варианту(час);

- Коэффициент относительного снижения трудовых затрат ($К_т$) (%), чтобы его рассчитать используется формула:

$$K_T = (\Delta T / T_O) * 100\%, \quad (2)$$

- Индекс снижения трудовых затрат или повышение производительности труда (Y_T), который можно рассчитать по формуле:

$$Y_T = T_0 / T_1, \quad (3)$$

Перечислим стоимостные показатели, к которым относят:

- Абсолютное снижение стоимостных затрат (ΔС) (руб.), которое рассчитывается по формуле:

$$\Delta C = C_0 - C_1, \quad (4)$$

где C_0 - стоимостные затраты (в рублях) на внесение информации по заказам в базовом варианте,

C_1 - стоимостные затраты (в рублях) на внесение информации по заказу с использованием проектного варианта;

- Коэффициент относительного снижения стоимостных затрат ($К_С$) (%), который рассчитывают по формуле:

$$K_C = (\Delta C / C_O) * 100\%, \quad (5)$$

- Индекс снижения стоимостных затрат или повышение производительности труда ($Y_С$), для которого используют формулу:

$$Y_C = C_0 / C_1, \quad (6)$$

Кроме данных показателей также будет рассчитан срок окупаемости затрат на внедрение проекта ($Т_{ок}$), для чего используем формулу:

$$T_{OK} = K_П / \Delta C \quad (7)$$

После расчетов данные будут сведены в таблицу.

3.2. Расчет показателей экономической эффективности проекта

Для расчёта показателей экономической эффективности были проведены замеры времени, необходимого сотруднику для внесения информации о заказе. В базовом варианте – при внесении информацию в электронную таблицу Excel сотрудник затрачивает в среднем 4,8 минуты.

В проектном варианте, при внесении данных в разработанное веб-приложение, сотрудник затрачивает 2,8 минут, если клиент новый и необходимо добавить клиента в базу клиентов. Если клиент уже есть в базе клиентов, то сотрудник затрачивает 2 минуты. Учитывая, что примерно 52% клиентов пункта проката это постоянные клиенты, среднее время, необходимое на внесение данных в веб-приложение будем считать равным 2,4 минуты. В день сотрудник проката вносит в среднем информацию о 30 заказах, что в базовом варианте составляет 30 * 4,8 минут = 144 минуты или 2,4 часа. В проектном варианте это 30 * 2,4 = 72 минуты или 1,2 часа

Рассчитаем абсолютное снижение трудозатрат на одну операцию по формуле (1)

$$\Delta T = T_0 - T_1 = 2{,}4 - 1{,}2 = 1{,}2$$

Затем рассчитаем коэффициент относительного снижения трудовых затрат ($К_т$) по формуле (2)

$$K_T = (\Delta T / T_0) * 100\ \% = 1{,}2/2{,}4 = 50\%$$

И рассчитаем индекс снижения трудовых затрат или повышение производительности труда ($Y_Т$).

$$Y_T = T_0 / T_1 = 2{,}4/1{,}2 = 2$$

Рассчитаем стоимостные показатели

Зарплата сотрудника проката в пересчете на часовую ставку составляет 205 руб./час, пункт проката работает без выходных дней, таким образом в месяц по базовому варианту затраты времени составляли 70,3 часов в месяц (2,4 часа в день при среднем количестве 29,3 дней в месяце), а показатель в денежном выражении 14411 руб./мес. При внесении информации по проектному варианту затраты составляют 7208 руб./мес. (1,2 часа в день при среднем количестве 29,3 дней в месяце). Рассчитаем абсолютное снижение стоимостных затрат по формуле (4):

$$\Delta C = C_0 - C_1 = 14411 - 7208 = 7203 \text{ руб.}$$

Коэффициент относительного снижения стоимостных затрат был рассчитан по формуле (5).

$$K_C = (\Delta C / C_0) * 100\ \% = 7203/14411 = 50\ \%$$

Индекс снижения стоимостных затрат рассчитан по формуле (6)

$$Y_C = C_0/C_1 = 14411/7208 = 2$$

Для наглядного представления отразим все полученные показатели в таблице 5.

Таблица 5 – Трудовые и стоимостные показатели проекта разработки веб-приложения для учета заказов

	Затраты		Абсолютное изменение затрат	Коэффициент изменения затрат	Индекс изменения затрат
	Базовый вариант	Проектный вариант			
Трудовые	2,4	1,2	1,2	50%	2
Стоимостные	144411	7208	7203	50%	2

Далее перейдём к расчет срока окупаемости затрат на внедрение проекта. Для этого определим затраты на внедрение проекта. Затраты на внедрение проекта отражены в таблице 6, они складываются из оплаты труда разработчика, а проектирование, разработку и тестирование веб-приложения, а также оплаты труда сотрудников пункта проката на внесение в веб-приложение существующей базы клиентов и базы оборудования

Таблица 6 – Затраты на внедрение проекта

	Трудоемкость (дней)	Трудоемкость (часов)	Часовая ставка (руб.)	Итого (руб.)
Проектирование, разработка, тестирование веб-приложения	20	160	105	16800
Внесение баз данных	1	8	205	1640
Итого				18440

Рассчитаем окупаемость проекта по формуле (8)

$$T_{ОК} = K_{\text{п}}/\Delta\text{С} = 18440/7203 = 2{,}56$$

Итого окупаемость проекта составляет 2,56 месяца

Кроме количественного экономического эффекта, важно учитывать и качественные изменения в учете заказов. Этот эффект выражен в появлении нового, более удобного способа внесения информации и снижение количества ошибок в данных.

Таким образом, проведенный расчет показателей экономической эффективности свидетельствует об экономической выгодности проекта и целесообразности его внедрения.

Заключение

В процессе выполнения данной выпускной квалификационной работы бакалавра был проведен анализ бизнес–процессов в пункте проката оборудования ООО «Айсберг». В результате анализа деятельности пункта проката было определено, что бизнес–процесс учета заказов на прокат оборудования нуждается в автоматизации, что может быть осуществлено путем разработки веб–приложения для учета заказов, затем были определены требования к разработке будущего веб-приложения. После этого был проведен сравнительный анализ существующих решений для управления заказами в пунктах проката, определены недостатки и сформулированы цель разработки и требования к разрабатываемому веб-приложению. На следующем этапе было осуществлено логическое проектирование предметной области, была построена логическая модель, диаграмма вариантов использования и диаграмма классов. Опираясь на поставленные задачи, были выбраны наиболее подходящие средства разработки веб-приложения и разработано веб-приложение для учета заказов. Далее был описан функционал веб–приложения и результаты проведенного тестирования. Также был проведен расчет показателей экономической эффективности, которые показали целесообразности внедрения веб-приложения в работу.

Разработанное веб-приложение имеет удобный пользовательский интерфейс и позволяет управлять заказами на прокат оборудования, базой клиентов, базой оборудования, а именно создавать, редактировать, удалять записи, а также просматривать историю. Внедрение разработанного в рамках выпускной квалификационной работы веб–приложения в работу пункта прока ООО «Айсберг» можно считать выгодным решением для учета заказов на прокат оборудования.

Список используемой литературы и используемых источников

1. Адигеев, М.Г. Жизненный цикл программного обеспечения / М.Г. Адигеев. – Ростов-на-Дону: Изд-во ЮФУ, 2013. – 41 с.

2. Аквино К., Ганди Т. A38 Front-end. Клиентская разработка для профессионалов. Node.js, ES6, REST./К.Аквино - СПб.: Питер, 2017. — 512 с.

3. Бабанов, А.М. Технология разработки программного обеспечения: структурный подход / А.М. Бабанов. – Томск: ТГУ, 2016. – 157 с.

3. Балдин, К.В. Информационные системы в экономике / К.В. Балдин, В.Б. Уткин. – М.: Дашков и К, 2015. – 395 с.

4. Блюмин А.М. Информационные ресурсы: Учебное пособие для бакалавров / А.М. Блюмин, Н.А. Феоктистов. – 3-е изд., перераб. и доп. – М.: Издательско-торговая корпорация «Дашков и Ко», 2015. – 384 с.

5. Бэнкс Алекс, Порселло Ева GraphQL: язык запросов для современных веб-приложений./ Бэнкс А — СПб.: Питер, 2019. — 240 с.

6. Варзунов А. В. Анализ и управление бизнес-процессами : учеб. пособие / А. В. Варзунов, Е. К. Торосян, Л. П. Сажнева. – Санкт-Петербург: Университет ИТМО, 2016. – 114 с.

7. Варфоломеева Е.В. Информационные системы в экономике: Учебное пособие / Е.В. Варфоломеева, Т.В. Воропаева и др.; Под ред. Д.В. Чистова - М.: НИЦ ИНФРА-М, 2015. – 234 с.

8. Гагарина, Л. Г. Технология разработки программного обеспечения: учеб. пособие / Л. Г . Гагарина, Е. В. Кокорева, Б. Д. Сидорова-Виснадул ; под ред. Л. Г. Гагариной. – Москва : Форум : ИНФРАМ, 2017. 400 с.

9. Гвоздева В.А. Информатика, автоматизированные информационные технологии и системы: Учебник / В.А. Гвоздева. - М.: ИД ФОРУМ: НИЦ ИНФРА-М, 2015. – 544 с.

10. Грошев А. С., Закляков П. В. Информатика: учеб. для вузов – 3-е изд., перераб. и доп. – М.: ДМК Пресс, 2015. – 588 с.

11. Долженко, А.И. Управление информационными системами / А.И. Долженеко – Ростов-на-Дону: Изд-во РГУ, 2017. – 191 с.

12. Изучение React — для чего, откуда, как? / [Электронный ресурс] – URL: https://habrahabr.ru/post/269303/ (дата обращения: 27.02.2020).

13. Котляров В. П. Основы тестирования программного обеспечения: учеб. пособие / В. П. Котляров. - 2-е изд., испр. - Москва : ИНТУИТ, 2016. – 335с.

14. Кумагина, Е.А. Модели жизненного цикла и технологии проектирования программного обеспечения / Е.А. Кумагина, Е.А. Неймарк. – Нижний Новгород: Изд-во ННГУ, 2016. – 41 с.

15. Основные методологии обследования организаций. Стандарт IDEF0. / [Электронный ресурс] – URL: http://www.cfin.ru/vernikov/idef/idef0.shtml (дата обращения: 13.04.2020).

16. Реинжиниринг бизнес-процессов: учеб. пособие / А. О. Блинов - Москва : ЮНИТИ-ДАНА, 2012. - 341 с.

17. Рудинский И. Д. Технология проектирования автоматизированных систем обработки информации и управления : учеб. пособие / И. Д. Рудинский. – М. : Горячая линия - Телеком, 2016. - 30 4 с.

18. Сильвио Морето Bootstrap в примерах. / Пер. с англ. Рагимов Р. Н. / Науч. ред. Киселев А. Н. – М.: ДМК Пресс, 2017. – 314 с.

19. Тестирование информационных систем / [Электронный ресурс] – http://aplana.ru/services/testing/testirovanie-sistem, (дата обращения: 12.01.2020).

20.Томас, М.Т. React в действии/ М.Т. Томас. – Санкт-Петербург: Издательский Дом Питер. – 368 с.

21. Тюгашев, А.А. Основы программирования / А.А. Тюгашев. – СПб.: Университет ИТМО, 2016. – 160 с.

22. Фуфаев Э. В. Базы данных. Учебное пособие / Э.В. Фуфаев, Д.Э. Фуфаев. - М.: Академия, 2014. - 320 с.

23. Хортон А., Вайс Р. Х82 Разработка веб-приложений в ReactJS: пер. с англ. Рагимова Р. Н. - М.: ДМК Пресс, 2016. - 254 с.

24. Чистякова В. И. Проектирование информационных систем. Учебник для студентов учреждений высшего профессионального образования / В.И. Чистякова, В.В.Белов – М.: Академия, 2015. – 36 2 с.

25.Чистов, Д. В. Проектирование информационных систем. Учебник и практикум / Д. В. Чистов, П. П. Мельников, А. В. Золотарюк, Н. Б. Ничепорук. – М.: Юрайт, 2016. – 260 с.

26. Шелухин О. И. Моделирование ин формационных систем: учеб. пособие. 00 4 / О. И. Шелухин. - 2- е изд., перераб. и доп. – М. : Горячая линия - Телеком, 2016. - 518 с.

27. IDEF0: функциональное моделирование деловых процессов / [Электронный ресурс] – URL: http://www.infosystem.ru/designing/methodology/sadt/sadt_for_bp.html, (дата обращения: 18.01.2020).

28. Alan Dennis, Barbara Haley Wixom, David Tegarden: Systems Analysis and Design with UML - 4th Edition, Wiley, 2012.

29. Alan Mark Davis. Just Enough Requirements Management: Where Software Development Meets Marketing. — Dorset House, 2015.

30. Bootstrap : на англ. яз. : [Электронный ресурс] / Bootstrap – URL: https://getbootstrap.com (дата обращения 11.03.2020)

31. Bootstrap 3 Tutorial : на англ. яз. : [Электронный ресурс] www.w3schools.com : THE WORLD'S LARGEST WEB DEVELOPER SITE – URL: https://www.w3schools.com/bootstrap/default.asp (дата обращения 21.04.2020).

32. Eric A. Meyer: Transforms in CSS: Revamp the Way You Design – O`Reilly Media, 2015.

33. Wallace Jackson, HTML5 Quick Markup Reference – учебное пособие, издательство: Apress – 2016 – 80с.

Приложение А

Фрагмент программного кода

Модуль App.js

```
class App
extends
Component {
                render() {
                return (
                <div className="App app-container">


                <PageHeader className= "page-header">Invoices App</PageHeader>
                <Menu/>
                <AddNewCustomer/>
                <AddNewProduct/>
                </div>
                );
                }
               }
               const mapStateToProps = store => {
                return {
                invoices: store.invoices,
                }
               }


               const mapDispatchToProps = dispatch => {
                return {


                }
               }


               export default connect(
               mapStateToProps,
                mapDispatchToProps
               )(App)
```

Модуль Menu.js

```
import React,
{ Component,
Fragment }
from 'react';


import { connect } from 'react-redux'
import {actSetAddNewActive} from "./../reducers/actions_creators.js"
import { BrowserRouter as Router, Route, Link } from "react-router-dom";
import Invoices from "./Invoices";
import Customers from "./Customers";
import Products from "./Products";


class Menu extends Component {
 render() {
 return (
 <Router>
 <Fragment>
 <nav className="navbar navbar-default">
 <div className="container-fluid">


 <div className="collapse navbar-collapse" id="navbar-main">


 <ul className="nav navbar-nav">
 <li><Link to='/'>Invoices</Link></li>
 <li><Link to='/customers'>Customers</Link></li>
 <li><Link to='/products'>Products</Link></li>


 </ul>


 </div>
 </div>
 </nav>


 <Route exact path='/' component={Invoices}/>
 <Route path='/customers' component={Customers}/>
 <Route path='/products' component={Products}/>
 </Fragment>
 </Router>
```

```
 );
 }
}


const mapStateToProps = store => {
 return {
 invoices: store.invoices,
 }
}


const mapDispatchToProps = dispatch => {
 return {
 actSetAddNewActive: payload => dispatch(actSetAddNewActive(payload)),


 }
}


export default connect(
mapStateToProps,
 mapDispatchToProps
)(Menu)
```

Модуль Invoices.js

```
import React,
{ Component
} from
'react';
 import { Button, Table } from 'react-bootstrap';
 import { connect } from 'react-redux'
 import {
  actSetAddNewActive,
  actDeleteInvoice,
  actStartEditing
 } from './../reducers/actions_creators';
 import AddNew from './AddNew/AddNew';


 class Invoices extends Component {
  deleteInvoice = id => event => {
  event.preventDefault(event);
  this.props.actDeleteInvoice({ id });
  };
```

Продолжение Приложения А

```
startEditInvoice = id => event => {
event.preventDefault(event);
this.props.actStartEditing({ id });
};


render() {
const { invoices, isAddingInvoice } = this.props.invoices;
const { actSetAddNewActive } = this.props;
return (
<div className="" style={{ marginTop: '20px' }}>
<div className="top-line top-line-inv">
<div className=" title">Invoices </div>
{isAddingInvoice ? null : <Button className="col-xs-2" bsStyle="info"
onClick={actSetAddNewActive} >Add New</Button> }
</div>
{isAddingInvoice ? <AddNew /> : null}
<Table striped bordered condensed hover>
<thead>
<tr>
<th className="col-xs-2 text-center">#</th>
<th className="col-xs-6 text-center">Customer</th>
<th className="col-xs-1 text-center">Discount</th>
<th className="col-xs-1 text-center">Total</th>
<th className="col-xs-1 text-center" />
<th className="col-xs-1 text-center" />
</tr>
</thead>
<tbody>
{invoices.map(item => (
<tr key={item.id}>
<td className="text-center">{item.id}</td>
<td className="text-center">{item.customer}</td>
<td className="text-center">{item.discount}%</td>
<td className="text-center">${item.total}</td>
<td className="text-center">
<Button className="" bsStyle="info"
onClick={this.startEditInvoice(item.id)}>
Edit
</Button>
</td>
<td className="text-center">
<Button className="" bsStyle="info"
onClick={this.deleteInvoice(item.id)}>
Delete
</Button>
</td>
</tr>
))}
```

Продолжение Приложения А

```
 </tbody>
 </Table>
 </div>
 );
 }
}


const mapStateToProps = store => {
 return {
 invoices: store.invoices,
 };
}


const mapDispatchToProps = dispatch => {
 return {
 actSetAddNewActive: payload => dispatch(actSetAddNewActive(payload)),
 actDeleteInvoice: payload => dispatch(actDeleteInvoice(payload)),
 actStartEditing: payload => dispatch(actStartEditing(payload))
 };
}
export default connect(
 mapStateToProps,
mapDispatchToProps
)(Invoices)
```