

**Частное образовательное учреждение высшего образования  
«Русско-Британский Институт Управления»  
(ЧОУВО РБИУ)**

Высшая школа менеджмента  
Кафедра математики и информатики

ДОПУСТИТЬ К ЗАЩИТЕ  
Заведующий кафедрой,  
к.ф.-м.н. С.С. Чеботарев

---

«\_\_» \_\_\_\_\_ 2017

**ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА**

На тему: «Разработка облачной CRM системы «Мой Сервис» на платформе  
.NET Core»

Направление подготовки: 38.03.05 Бизнес информатика

Профиль подготовки: Электронный бизнес

Научный руководитель  
Кандидат физико-  
математических наук  
*С.С. Чеботарев*

Автор работы  
студент группы БИ-402  
*Абрамов А.А.*

## Содержание

<b>Введение .....</b>	<b>3</b>
<b>Глава 1. Анализ предметной области.....</b>	<b>4</b>
1.1 Описание предметной области .....	4
1.2 Бизнес-процессы предметной области.....	5
1.3 Анализ существующих аналогичных программных решений.....	10
Выводы по первой главе.....	12
<b>Глава 2. Проектирование информационной системы.....</b>	<b>14</b>
2.1 Техническое задание на разработку АИС .....	14
2.3 Проектирование базы данных.....	17
2.4 Выбор технологий для разработки проекта .....	35
Выводы по второй главе.....	37
<b>Глава 3. Разработка программного решения .....</b>	<b>40</b>
3.1 Главная страница.....	40
3.2 Модуль «Заказы».....	43
3.3 Модуль «Кассы».....	49
3.4 Модуль «Склады» .....	51
3.5 Модуль «Отчеты».....	53
3.6 Модуль «Продажи».....	54
3.7 Модуль «Клиенты» .....	55
3.8 Модуль «Снабжение» .....	56
3.9 Модуль «Настройки» .....	57
Выводы по третьей главе.....	60
Обоснование экономической эффективности проекта .....	61
<b>Заключение.....</b>	<b>65</b>
<b>Библиографический список.....</b>	<b>67</b>
Приложение 1 .....	69
Приложение 2 .....	94

## **Введение**

Цель работы – разработать информационную систему, в задачи которой входит автоматизация бизнес-процессов сервисных центров. После введения системы предприятие получит следующие эффекты:

- 1) ускорение выполнения бизнес-процессов;
- 2) систематизированное и защищенное хранение информации;
- 3) удобство доступа к необходимой информации.

Процесс подготовки к непосредственной реализации данного программного продукта можно разделить на следующие задачи:

- 4) анализ предметной области;
- 5) составление автоматизируемых бизнес-процессов на основе CJM и USM;
- 6) разработка технического задания;
- 7) проектирование базы данных и архитектуры самого приложения;
- 8) выбор технологий для клиентской и серверной частей;
- 9) создание макетов веб-страниц.

В процессе разработке веб-приложения был использован индуктивный подход, а так же такие методы, как: абстракция и классификация.

1. Метод научной абстракции – отвлечение от несущественных сторон анализируемой предметной области и проектируемой системы, выделение общих существенных признаков и закономерностей.

2. Метод классификации – объединение различных объектов в группы на основе общих признаков.

# **Глава 1. Анализ предметной области**

## **1.1 Описание предметной области**

В общем понимании предметную область можно определить, как часть реального мира, рассматриваемую в рамках определенного контекста. Под контекстом можно понимать область исследования или область, которая является объектом определенной деятельности.

В качестве контекста данной информационной системы выступает деятельность сервисных центров, располагающихся на территории России.

Важно заметить, что данный программный продукт рассчитан на два уровня бизнеса.

1. Микробизнес – компания, численность сотрудников которой составляет до 15 человек. Преимущественно ИП.

2. Малый бизнес – компания, численность сотрудников которой составляет от 15 и до 50 человек. Преимущественно ООО.

Основные различия уровней бизнесов, которые были учтены при разработке программного продукта: микробизнес преимущественно состоит из 1-2 сотрудников, для которых простота и удобство системы играют большую роль. Более крупные компании, которые уже имеют свою сеть сервисных центров, нуждаются в соответствующем их уровне функционале, который не будет усложнять работу владельцам микробизнесов.

У сервисного центра есть два продукта деятельности: совершение продаж и выполнение ремонта техники. Основным видом деятельности является ремонт.

В качестве субъектов предметной области выступают:

1) сервисный центр – юридическое лицо, обязывающееся выполнить ремонт техники или совершить продажу имеющегося в наличии товара, в котором заинтересован клиент;

2) клиент – физическое или юридическое лицо, желающее за плату получить услугу по ремонту техники или или купить товар;

3) менеджер – физическое лицо, в задачи которого входит (или может входить) прием и выдача техники, регистрация и оплата заказов, прием товара;

4) подрядчик – юридическое лицо, обязующееся выполнить поставленную сервисным центром задачу по ремонту техники, если сервисный центр по тем или иным причинам не способен выполнить ремонт, а так же если клиент не против осуществления ремонта через подрядчика.

Клиент и подрядчик относятся к субъектам внешней среды предметной области. Обращение клиента в сервисный центр является зарождением соответствующего бизнес-процесса.

Работа сервисного центра с подрядчиками позволяет получать прибыль, отдавая работу по ремонту другой компании и оплачивая ее услуги на

## **1.2 Бизнес-процессы предметной области**

1. Создание заказа. Инициализация данного бизнес-процесса происходит после обращения клиента в сервисный центр. Процесс обращения клиента в сервисный центр подразумевает собой общение клиента менеджером для уточнения конкретной информации и наличие у клиента желания и возможности воспользоваться услугами сервисного центра.

Проблема бизнес-процесса заключается в том, что во многих организациях учет заказов ведется несистематизированно. Некоторые фирмы ведут учет в Microsoft Excel, 1С, других облачных CRM, на бумаге или вовсе совмещают некоторые из вариантов.

2. Закрытие заказа. Инициализация данного бизнес-процесса происходит при выполнении сервисным центром всех оговоренных задач по ремонту техники клиента и при посещении клиента сервисного центра с целью закрытия заказа.

Проблема схожа с открытием заказа: для полной систематизации информации с возможностью дальнейшего анализа необходимо вести учет в одной системе. Помимо этого в сервисных центрах в данном бизнес-процессе присутствует печать чеков, и для ускорения работы менеджера было бы эффективно делать это внутри одной системы.

3. Совершение продажи. Аналогично процессу «Создание заказа». Различие лишь в том, что процесс начинается и заканчивается в один момент, а в качестве услуги выступает продажа определенного сервисным центром товара.

4. Оприходование товара. Под этим термином понимается поступление товара на склад предприятия, зарегистрированное в системе учета. Многие из компаний делают это в Excel, и для различных манипуляций с товарами (проверка наличия, создание отчетности) требуется потратить относительно большое количество времени, если бы это было в одной системе.

5. Заказ запчастей. Сервисные центры при необходимости в какой-либо запчасти для ремонта пользуются услугами поставщиков, заказывая у них недостающие детали. Для этого нанимают отдельного человека. При наличии удобного интерфейса и возможности работы поставщикам внутри системы можно было бы избежать затрат на выделенный только для этих целей персонал.

На рис. 1 и рис. 2 продемонстрирован основной бизнес-процесс сервисного центра, включающий себя вышеупомянутые, а также его декомпозиция.

На рис. 3 показана диаграмма верхнего уровня, отражающая поведение программного продукта в целом. Цель создания данной модели – подготовка к автоматизации бизнес процессов, формирование технического задания на автоматизацию. Точка зрения для моделирования процессов была выбрана – разработчик.

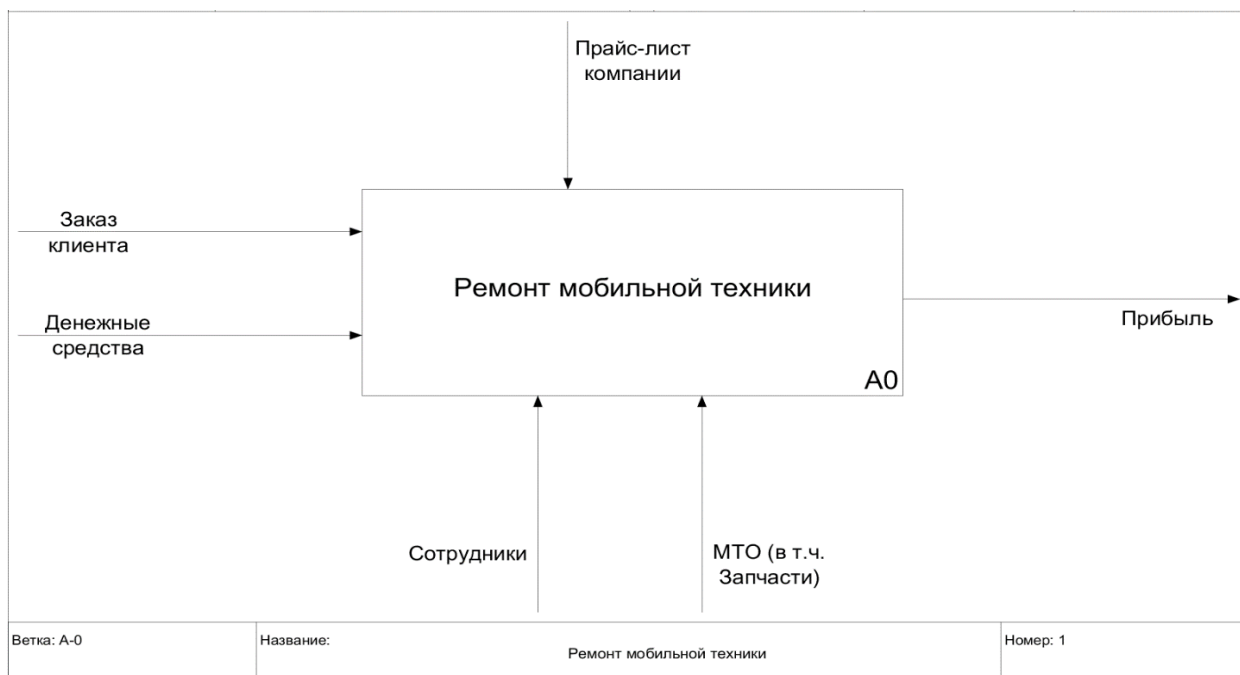


Рисунок 1 – Бизнес-процесс “Ремонт мобильной техники” в IDEF0

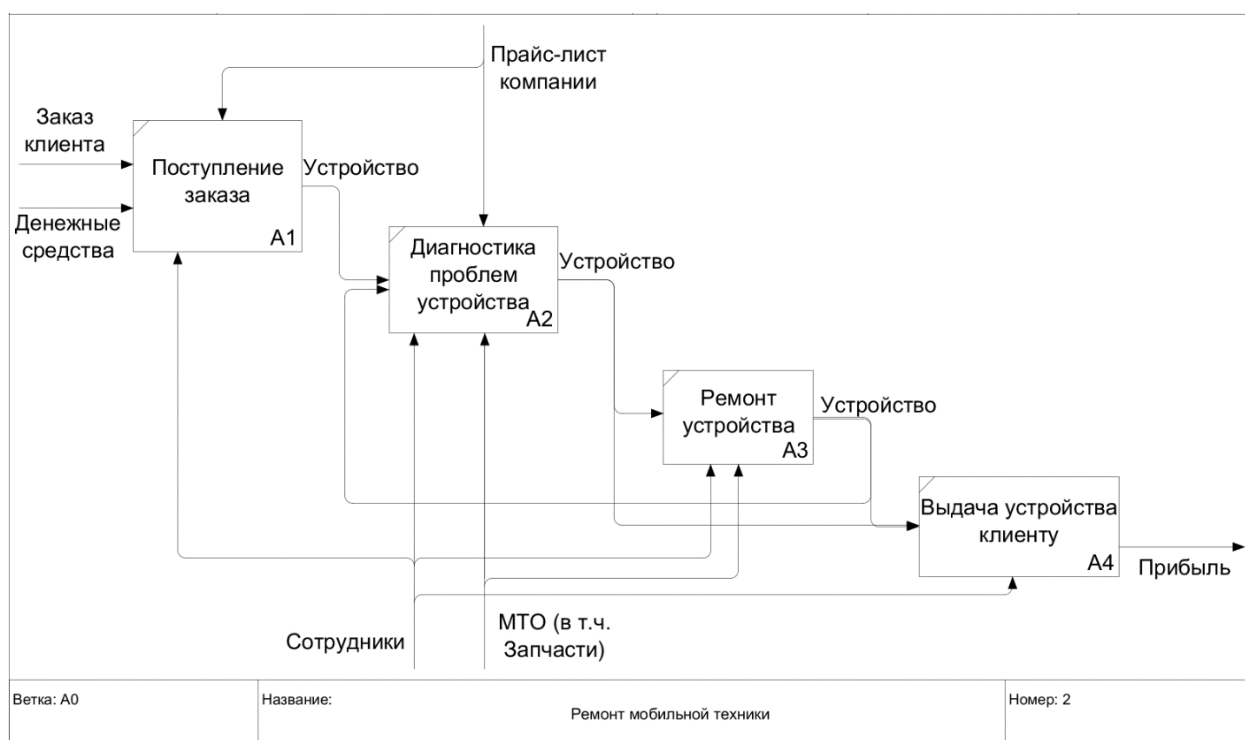


Рисунок. 2 – Бизнес-процесс «Ремонт мобильной техники» в IDEF0  
(декомпозиция)

На рис. 4 представлена диаграмма декомпозиции модели «Оптимизировать бизнес-процесс клиента». В результате декомпозиции были выделены следующие функциональные блоки: регистрация пользователя, оплата услуг, настройка и эксплуатация веб-сервиса клиентом, разработка и поддержка системы.

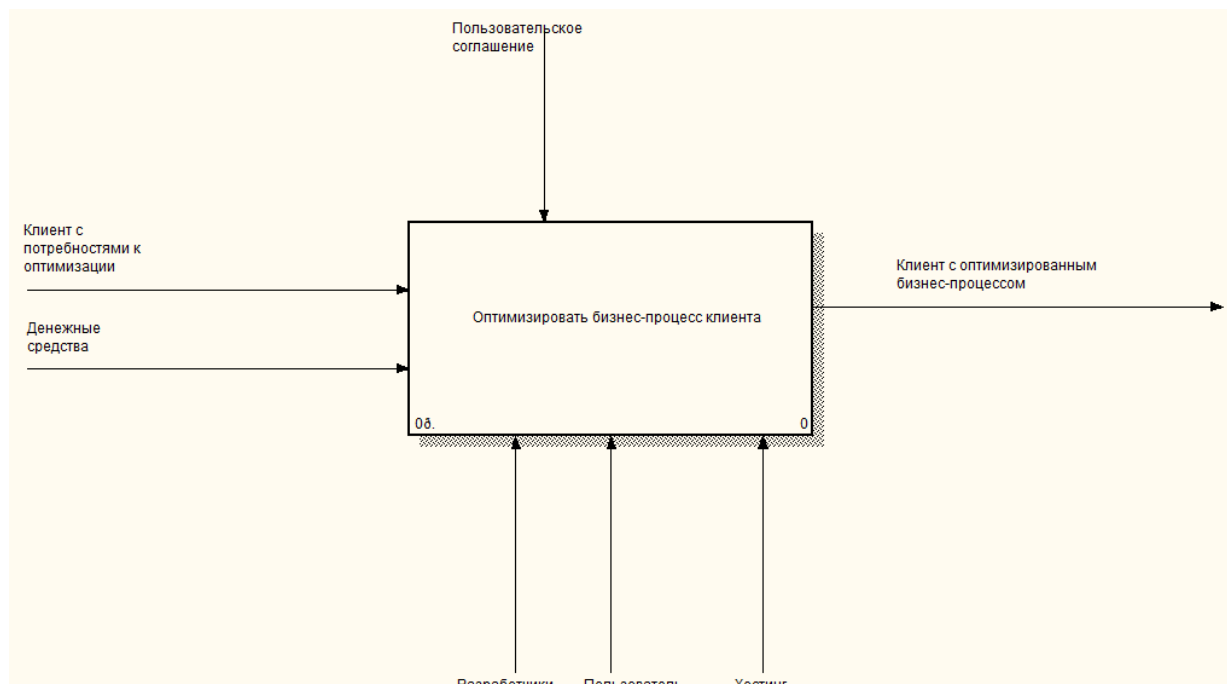


Рисунок 3 – Диаграмма верхнего уровня

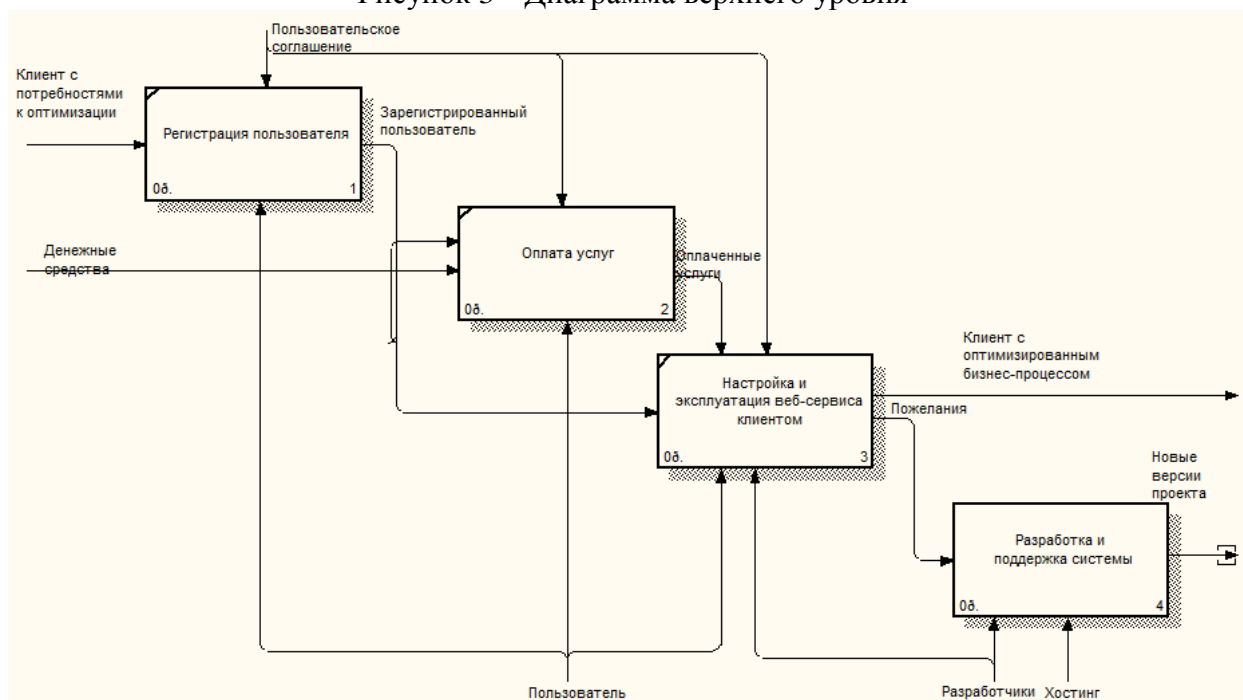


Рисунок 4 – Декомпозиция «Оптимизировать бизнес-процесс клиента»

На вход в блок «Регистрация пользователя» приходит клиент с потребности в автоматизации его бизнес-процессов, проходящий регистрацию в системе. При этом он ознакомляется с пользовательским соглашением.

Для полноценной работы с системой пользователю необходимо провести оплату услуг, которая и является следующим блоком декомпозиции



описываемого блока. Помимо пользователя на вход поступают денежные средства пользователя. На выходе идут оплаченные услуги.

Следующим важным шагом является настройка и эксплуатация веб-сервиса клиентов. Это основной шаг при работе с системой. На вход поступает зарегистрированный пользователь и оплаченные услуги. Из этого блока выходит клиент с оптимизированным бизнес-процессом, что и является конечным результатом.

Следующим шагом является скрытая от пользователя деятельность – разработка и поддержка системы (рис. 5). Этот шаг выполняют разработчики во все время жизни системы.

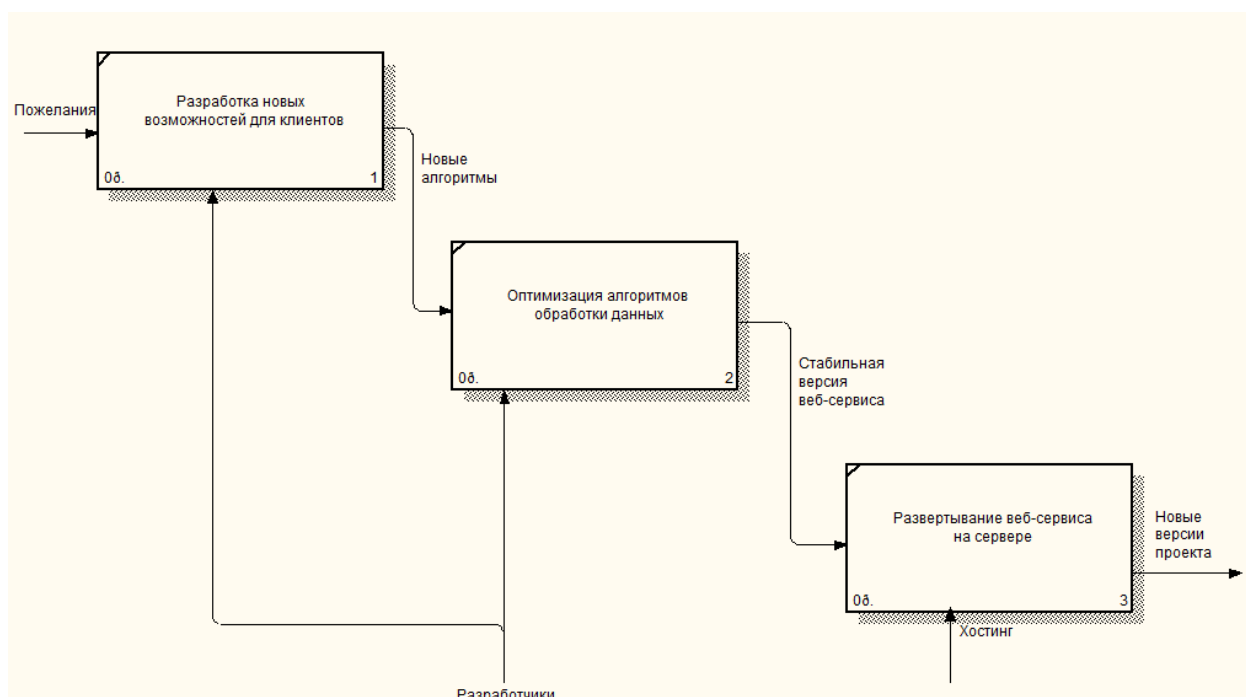


Рисунок 5 – Разработка и поддержка системы

Было решено декомпозировать блок «Разработка и поддержка системы», так как он является основным с точки зрения разработчика.

Он состоит из трех основных подблоков. Первый из них – разработка новых возможностей для клиентов. На вход в него получают пожелания от клиентов системы, а на выход – новые алгоритмы.

Далее идет блок «Оптимизация алгоритмов обработки данных». Он представляет собой исправление ошибок и доработку продукта до стабильного состояния. Соответственно, на выходе получается стабильная версия веб-сервиса.

Последним блоком является развертывание веб-сервиса на сервере. В него входит настройка веб-сервиса на production-сервере. Для этого используется хостинг, и на выходе получаются новые версии веб-сервиса.

### **1.3 Анализ существующих аналогичных программных решений**

После анализа потенциальных конкурентов на рынке было выведено несколько главных конкурентов.

#### **1. «Ремонт Онлайн».**

Это программное решение для сервисных центров, делающее ведение базы ремонтов очень простым и допустим в любое время в режиме онлайн. Данный программный продукт является облачным и доступен в любое время.

Можно выделить несколько основных возможностей данного программного продукта:

- 1) ускоренная и упрощенная обработка заказов, занимающая около минуты;
- 2) подробная информация о заказе и взаимодействиях с ним со стороны сотрудников компании;
- 3) просмотр детальной информации по изменениям в складах компании;
- 4) ведение клиентской базы;
- 5) печать документов внутри системы;
- 6) SMS-рассылка и email-уведомления.

В результате использования пробного периода и изучения фактического функционала данного программного продукта были выведены следующие плюсы:

- 1) программное решение простое и практически не требует материальных и временных затрат на обучение персонала;
- 2) реализован простой и понятный графический интерфейс;
- 3) не требует покупки и установки дополнительного оборудования, работа выполняется посредством тонкого клиента.

Недостатки:

1) реализовано множество ненужного многим сервисным центрам функционала, но при этом отсутствует то, что требуют многие пользователи (исходя из просмотра отзывов об этом продукте), в связи с чем пользователям приходится либо отказываться от этого продукта, либо использовать совместно с тем, где реализован отсутствующий в «Ремонт Онлайн» функционал;

2) имеются недоработки в программной части.

2. «Gincore».

Gincore это специализированное ПО, рассчитанное для использования сервисными центрами, включающее в себя функции ERP, CRM и WMS систем. Данное ПО создавалось для учета в ремонтных мастерских, автоматизации бизнес-процессов в сетях сервисных центров и их управлением.

В результате использования пробного периода и изучения фактического функционала данного программного продукта были выведены следующие плюсы:

1) в системе реализован функционал, необходимый любому сервисному центру;

2) система облачная и не требует установки на компьютер пользователя;

3) быстрая и отзывчивая администрация, помогающая по любому вопросу.

Недостатки:

1) сложный для понимания пользовательский интерфейс;

2) высокая стоимость;

3) после беседы с разработчиками было выявлено, что систему в дальнейшем будет сложно поддерживать, учитывая факт, что она разрабатывается на бесплатном шаблоне, сильно ограничивающем разработчиков.

В табл. 1 показан итоговый анализ существующих аналогов проекта.

Таблица 1 – Итоговый анализ существующих аналогов ПО

	Функционал	Удобство	Интерфейс	Закрытость проекта	Стоимость
Ремонт Онлайн	-	+	+	-	-
Gincore	+	-	-	+	-

### **Выводы по первой главе**

В процессе анализа похожих программных решений было выбрано и проанализировано 2 программных продукта. Каждый из программных продуктов обладает своими достоинствами и недостатками.

Под процессом анализа предметной области понимается следующий спектр выполненных задач:

- 1) сформировано понятие предметной области, а так же ее описание для конкретного программного решения;
- 2) списаны основные бизнес-процессы, решаемые программным решением;
- 3) описаны определения основных терминов в данной предметной области;
- 4) проведён анализ существующих аналогичных программных решений: РемонтОнлайн и Gincore;
- 5) проведён анализ целесообразности создания программного продукта, учитывая текущую ситуацию на рынке.

В результате внедрения данного программного решения в деятельность сервисного центра должно быть достигнуто следующее:

- 1) сокращено время на поиск информации о клиенте за счет сформированной единой клиентской базы;
- 2) сокращено время на создание и обработку заказов за счет единой базы заказов;

3) сокращено время на денежный расчёт за счет автоматизации расчёта стоимости и системе получения информации о стоимости конкретных запчастей/товаров;

4) автоматизирован процесс печати документов.

Учитывая недостатки каждого из проанализированных продуктов, было принято решение разработать собственное программное решение.

## Глава 2. Проектирование информационной системы

### 2.1 Техническое задание на разработку АИС

Изначально проект разрабатывался для организации ООО «АйРазбил». Организация по осуществлению ремонта устройств на базе iOS, а также разработке веб-решений в сфере IT, осуществляет свою деятельность в Челябинской области в г. Челябинск.

Компания имеет офис, оборудованный всей необходимой офисной техникой, предоставляет все удобства сотрудникам. Основной целью создания ООО «АйРазбил» является осуществление предпринимательской деятельности для получения прибыли. Организационная структура компании показана на рис. 6.



Рисунок 6 – Структура организации

Техническое задание представляет собой документ, в котором содержатся требования к программному решению, порядок развития или модернизации системы.

Техническое задание как официальный документ в процессе проектирования ИС составлено не было. Вместо него были использованы такие подходы к проектированию, как Impact Mapping, User Story Mapping и Customer Journey Mapping [4].

Impact Mapping это принцип проектирования ИС, который позволяет создать логическую цепочку от бизнес-целей до изменений в продукте, необходимых для достижения определенных целей.

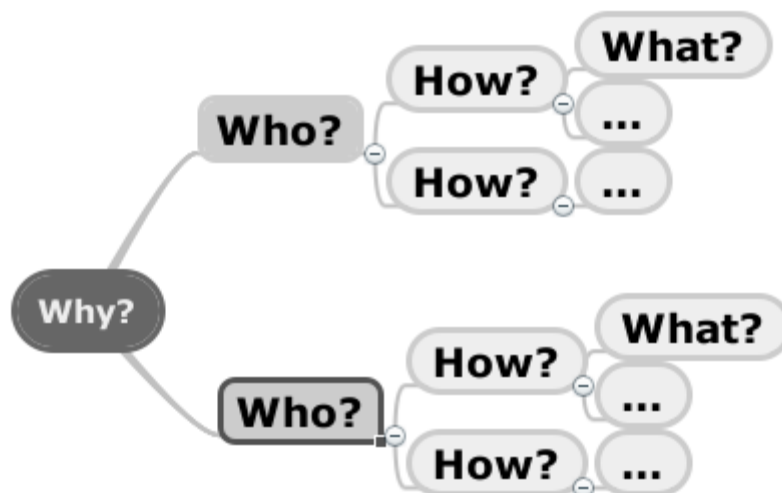


Рисунок 7 – Пример Impact Mapping

На рис. 7 продемонстрирована структура Impact Map.

1. «Why?» – это главный элемент продемонстрированной выше карты, отвечающий на главный вопрос: «Зачем мы это делаем?». Это является основной целью поставленной задачи.

2. «Who?» – На следующем уровне отображаются все заинтересованные в проекте стороны, способные повлиять на цели бизнеса.

3. «How?» – Затем для каждой заинтересованной стороны описываются способы достижения основной цели бизнеса.

4. «What?» – Это является последним уровнем, в котором описываются конкретные действия команды разработки, которые способствуют достижению основной цели бизнеса.

User Story Mapping – это техника визуального представления некоторой последовательности действий, которая в дальнейшем будет реализована в программном решении. Строится данная последовательность на двумерной сетке, чтобы показать последовательность и группировку ключевых

особенностей программного решения по горизонтали, а определенные детали и приоритет – по вертикали (рис. 8).

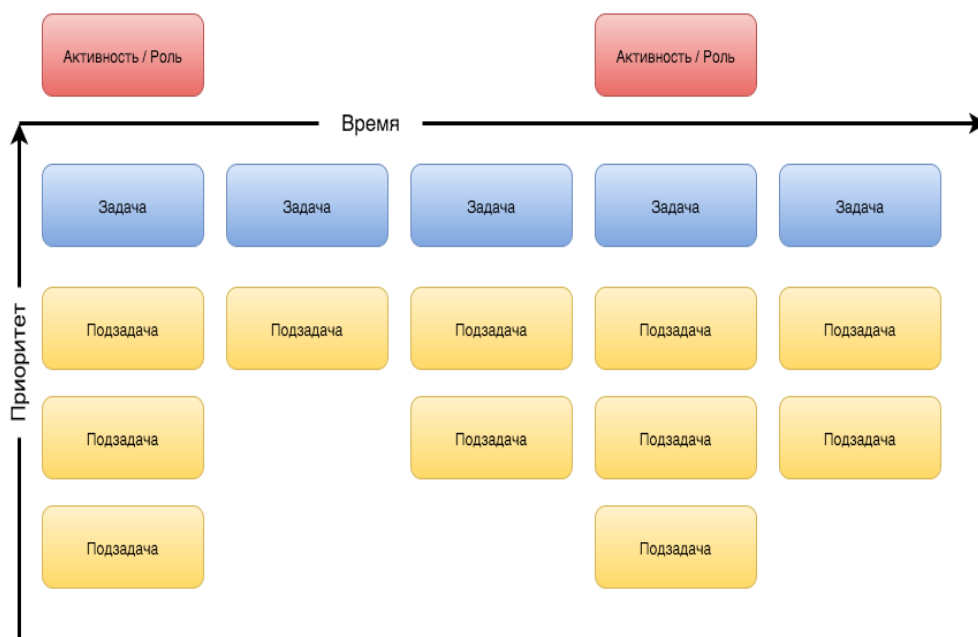


Рисунок 8 – Пример User Story Mapping

Пользовательские истории являются быстрым способом документации желаний клиента к системе без необходимости в разработке формализованных документов (технического задания), которые требуют дополнительных ресурсов на их поддержку.

Основная цель пользовательских историй заключается в том, чтобы без каких-либо временных и прочих накладных затрат вносить изменения в уже существующие поставленные условия к системе.

Они являются неофициальным определением требований к системе во время отсутствия процедуры приемочного тестирования. Перед созданием пользовательской истории клиенту необходимо определить необходимую приемную процедуру для гарантии того, что цель, поставленная данной историей, может быть достигнута.

Customer Journey Mapping – это инструмент визуализации взаимодействия пользователя и программного решения, отображающий данное взаимодействие в хронологическом порядке и раскладывая его на атомарные составляющие. Составляющие такого взаимодействия относятся как к процессу, в который могут входить цели или задачи пользователей, их



ожидаемый результат или же проблемы, которые препятствуют переходу на следующий шаг, так и к эмоциональному состоянию клиента: его ощущения, мысли и чувства в определенный момент времени (рис. 9).

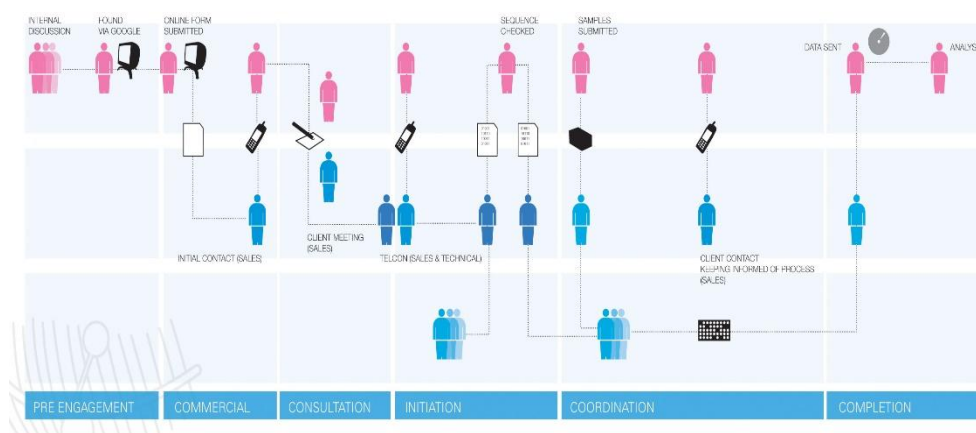


Рисунок 9 – Пример Customer Journey Mapping

Основной и немаловажный плюс данных инструментов проектирования по сравнению с обычным техническим заданием заключается в том, что в связи с участием в проектировании не только одного заказчика, но еще и группы разработчиков требования к проекту меняются путем выявления истинной необходимости в данном программном решении. Это достигается путем выявления такого программного решения, которое окажет максимально возможный положительный эффект на бизнес клиента.

На рис. 10 показана UML-диаграмма работы программного решения.

## 2.3 Проектирование базы данных

Перед непосредственной реализацией базы данных была построена ER-диаграмма, изображенная на рис. 11. Стоит заметить, что это первая и упрощенная версия базы данных. В дальнейшем с каждой итерацией цикла разработки структура базы данных будет изменяться.

На данный момент описаны основные сущности базы данных: пользователь, заказ, мастерская, товар, устройства, склад и фильтры.

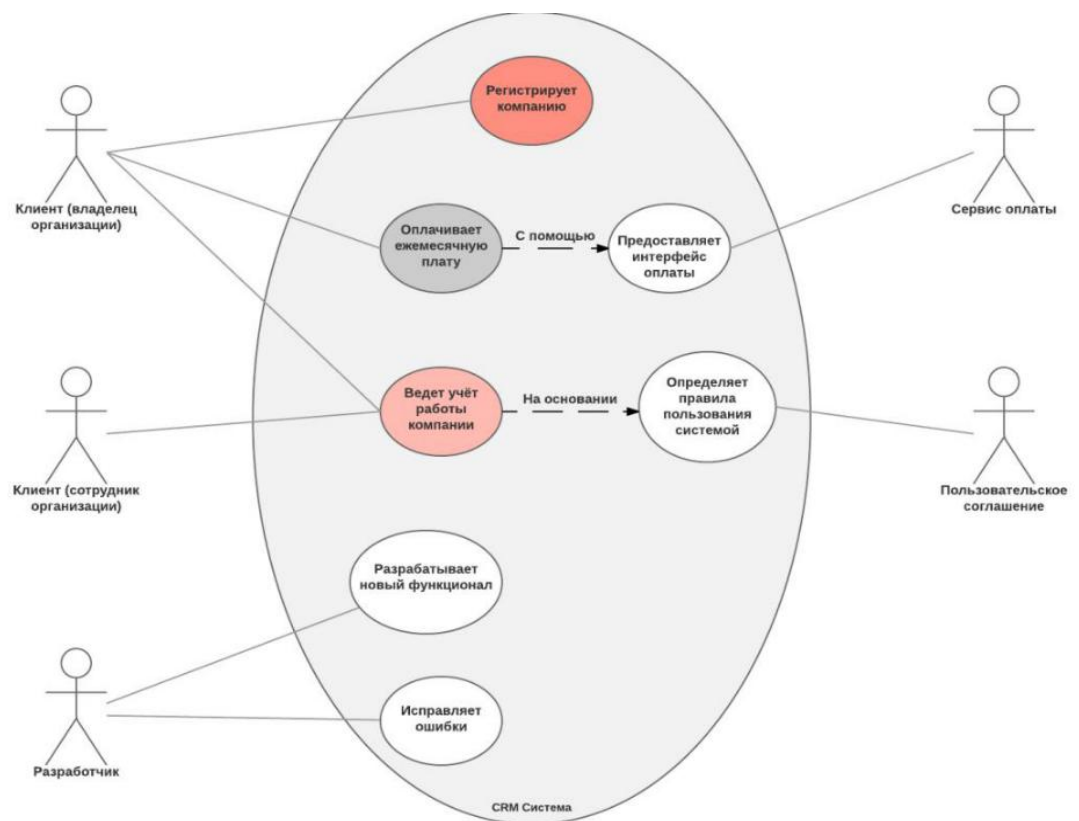


Рисунок 10 – UML диаграмма прецендентов CRM-системы

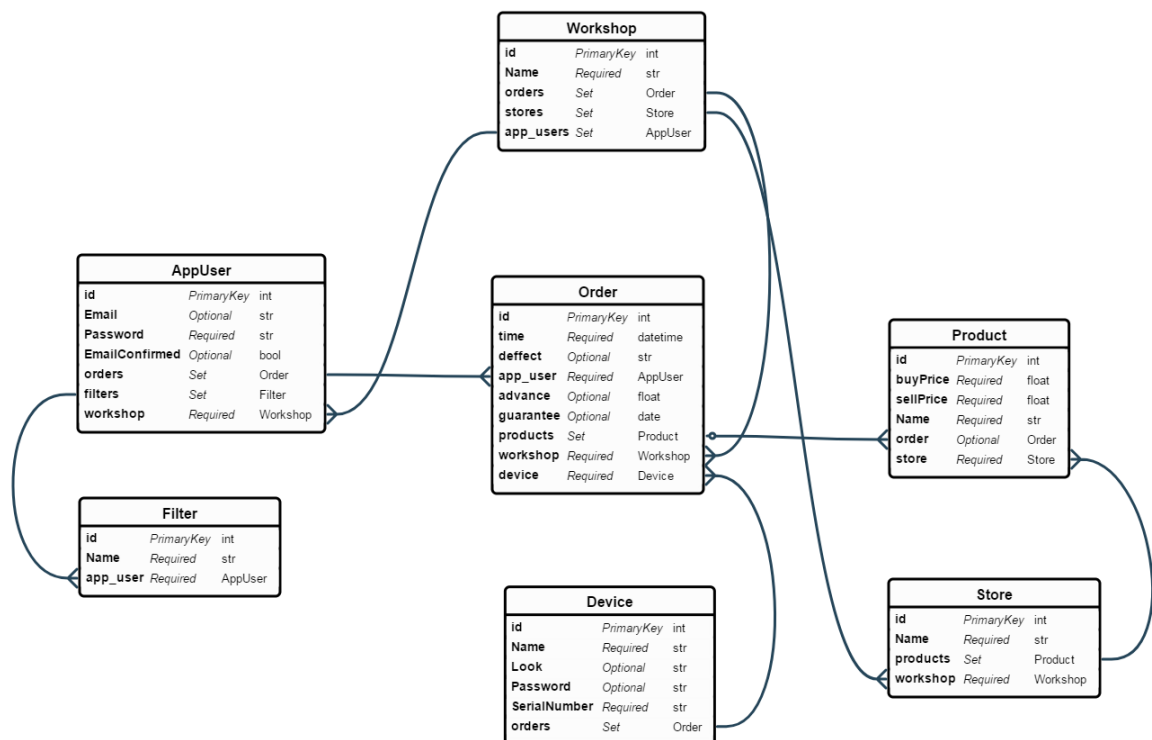


Рисунок 11 – ER диаграмма CRM-системы

В качестве базы данных для CRM системы была использована PostgreSQL [9].

PostgreSQL – это свободная объектно-реляционная система управления базами данных, которая существует в реализациях для множества UNIX-подобных платформ, включая AIX, различные BSD-системы, HP-UX, IRIX, Linux, Mac OS X, а также для Microsoft Windows.

Разработаны следующие таблицы:

1. Адреса – хранит информацию об адресах клиентов (табл. 2).

Таблица 2 – Таблица БД «Адреса»

ID	Город	Страна	Улица

Типы данных:

- 1) ID – int;
- 2) все остальные столбцы – text.

Для столбцов «Город», «Страна» и «Улица» разрешено значение NULL, так как клиент имеет права не сообщать свой адрес по личным соображениям.

2. Утверждения о ролях (системная таблица) – хранит дополнительную информацию о ролях пользователей (табл. 3).

Таблица 3 – Таблица БД «Утверждения»

ID	Тип	Значение	ID роли

Типы данных:

- 1) ID – int;
- 2) тип – text;
- 3) значение – text;
- 4) ID роли int.

3. Роли – хранит информацию о ролях пользователей (табл. 4).

Таблица 4 – Таблица БД «Роли»

ID	ID компании	Штамп параллелизма	Описание	Название	Системное	Системное значение

Типы данных:

- 1) ID – int;
- 2) ID компании – int;
- 3) системное bool;
- 4) остальные – text.

4. Пользователи – хранит сведения о зарегистрированных пользователях в системе (табл. 5).

Таблица 5 – Таблица БД «Пользователи»

ID	ID компании	Штамп параллелизма	Email	Подтвержден ли email	Удален	Адаптированный Email

Продолжение таблицы 5

Адаптированный логин	Хэш пароля	Телефон	Телефон подтвержден	Штамп безопасности	Логин

Типы данных:

- 1) ID – int;
- 2) ID компании – int;
- 3) удален – bool;
- 4) телефон подтвержден – bool;
- 5) все остальные столбцы – text;

5. Кассы – хранит информацию о кассах в компаниях и их баланс (табл. 6).

Таблица 6 – Таблица БД «Кассы»

ID	Баланс	ID компании	Удалена	Название	Тип	ID отделения

Типы данных:

- 1) ID – int;
- 2) ID компании – int;
- 3) ID отделения – int;
- 4) тип – int;
- 5) удалена – bool;
- 6) баланс –real;
- 7) название – text.

6. Кассовые движения – хранит информацию о проведенных денежных операциях (табл. 7).

Таблица 7 – Таблица БД «Кассовые движения»

ID	Баланс	ID кассы	Комментарий	Дата	Удалена	Сумма

Продолжение таблицы 7

ID автора	ID заказа

Типы данных:

- 1) ID – int;
- 2) баланс –real;
- 3) ID кассы –int;
- 4) дата –timestamp;
- 5) удалена –bool;
- 6) сумма –real;
- 7) ID автора –int;
- 8) ID заказа –int;
- 9) комментарий – text.

7. Категории – хранит список категорий материальных ресурсов предприятия, которые предназначены для продажи (табл. 8).

Таблица 8 – Таблица БД «Категории»

ID	ID компании	Удалена	Название

Типы данных:

- 1) ID – int;
- 2) ID компании – int;
- 3) Удалена – bool;
- 4) Название – text.

8. Источники клиентов – хранит информацию рекламных кампаний, доступных для присваивания к конкретному клиенту при оформлении заказа (табл. 9).

Таблица 9. – Таблица БД «Источники клиентов»

ID	ID компании	Название

Типы данных:

- 1) ID – int;
- 2) ID компании – int;
- 3) Название – text.

9. Клиенты – хранит информацию о клиентах компании (табл. 10).

Таблица 10 – Таблица БД «Клиенты»

ID	Email	Удален	ФИО	ID источника	Адрес	ID компании

Продолжение таблицы 10

Дата	Это поставщик	Внутренний ID

Типы данных:

- 1) ID – int;
- 2) email – text;

- 3) удален – bool;
- 4) ФИО – text;
- 5) ID источника – int;
- 6) адрес – text;
- 7) ID компании – int;
- 8) дата – timestamp;
- 9) это поставщик – bool;
- 10) внутренний ID – int.

10. Компании – хранит информацию о компаниях, зарегистрированных в системе (табл. 11).

Таблица 11 – Таблица БД «Компании»

ID	ID адреса	Название	ID владельца	ID настроек	Дата	Email

Типы данных:

- 1) ID – int;
- 2) ID адреса – int;
- 3) название – text;
- 4) ID владельца – int;
- 5) ID настроек – int;
- 6) дата – timestamp;
- 7) email – text.

11. Подрядчики – хранит информацию о подрядчиках как об отдельных пользователях (табл. 12).

Таблица 12 – Таблица БД «Подрядчики»

ID	Дата	Email	Удален	Логин	Пароль	Телефон

Продолжение таблицы 12

ID роли	ID компании	Наименование

Типы данных:

- 1) ID – int;
- 2) дата – timestamp;
- 3) email – text;
- 4) удален – bool;
- 5) логин – text;
- 6) пароль – text;
- 7) телефон – text;
- 8) наименование – text;
- 9) все остальные столбцы – int.

12. Кредиты – хранит информацию о списаниях материальных ресурсов со склада компании (табл. 13).

Таблица 13 – Таблица БД «Кредиты»

ID	Комментарий	Внутренний ID	Дата	ID заказа	ID склада	ID пользователя

Типы данных:

- 1) ID – int;
- 2) дата – timestamp;
- 3) комментарий – text;
- 4) все остальные столбцы – int.

13. Дебеты – хранит информацию о поступлениях материальных ресурсов на склад компании (табл. 14).

Таблица 14 – Таблица БД «Дебеты»

ID	Комментарий	Внутренний ID	Дата	Номер счета	ID склада	ID пользователя

Продолжение таблицы 14

ID поставщика	ID платежа



Типы данных:

- 1) ID – int;
- 2) комментарий – text;
- 3) дата – timestamp;
- 4) номер счета – text;
- 5) все остальные столбцы – int.

14. Типы устройств – хранит информацию о типах устройств, зарегистрированных в компании (табл. 15).

Таблица 15 – Таблица БД «Типы устройств»

ID	ID компании	Удален	Наименование

Типы данных:

- 1) ID – int;
- 2) ID компании – int;
- 3) удален – bool;
- 4) наименование – text.

15. Устройства – хранит информацию об устройствах, ремоентируемых компанией (табл. 16).

Таблица 16 – Таблица БД «Устройства»

ID	Внешний вид	Модель	Пароль	Серийный номер	ID типа	Комплектация

Типы данных:

- 1) ID – int;
- 2) ID типа – int;
- 3) все остальные столбцы – text.

16. Теги документов – хранит список тегов документов, используемых при редактировании документов в качестве переменных для подстановки значения (табл. 17).

Таблица 17 – Таблица БД «Теги документов»

ID	Описание	Группа	Наименование	Системный	Тип

Типы данных:

- 1) ID – int;
- 2) описание – text;
- 3) группа – int;
- 4) наименование – text;
- 5) системный – bool;
- 6) тип – int.

17. Документы – хранит информацию о документах, используемых в компании (табл. 18).

Таблица 18 – Таблица БД «Документы»

ID	ID компании	Содержимое	Название	Тип

Типы данных:

- 1) ID – int;
- 2) ID компании – int;
- 3) содержимое – text;
- 4) название – text;
- 5) тип – int.

18. Файлы – хранит информацию о системных и пользовательских файлах (табл. 19).

Таблица 19 – Таблица БД «Файлы»

ID	Описание	Наименование	Путь	Системный	Тип

Типы данных:

- 1) ID – int;
- 2) описание – text;
- 3) наименование – text;
- 4) путь – text;
- 5) системный – bool;
- 6) тип – int.

19. Фильтры – хранит информацию о созданных пользователями фильтрами заказов (табл. 20).

Таблица 20 – Таблица БД «Фильтры»

ID	Бренд	ID клиента	ID компании	ID типа устройства	Инженеры	Гарантия

Продолжение таблицы 20

Дата от	Дата До	Менеджеры	Наименование	Статусы	ID пользователя

Типы данных:

- 1) ID – int;
- 2) ID клиента – int;
- 3) ID компании – int;
- 4) ID типа устройства – int;
- 5) гарантия – bool;
- 6) дата от – timestamp;
- 7) дата до – timestamp;
- 8) ID пользователя – int;
- 9) все остальные столбцы – text.

20. Перемещения – хранит информацию о перемещениях материальных ресурсов компании между ее складами (табл. 21).

Таблица 21 – Таблица БД «Перемещения»

ID	Комментарий	Внутренний ID	Дата	ID Склада Из	ID Склада В	ID пользователя

Типы данных:

- 1) ID – int;
- 2) комментарий – text;
- 3) дата – timestamp;
- 4) все остальные столбцы – int.

21. События заказов – хранит информацию о событиях заказов в мастерских, которые генерируются после любой манипуляции, проведенной с заказом (табл. 22).

Таблица 22 – Таблица БД «События заказов»

ID	ID автора	Дата	ID заказа	ID статуса	Текст	Тип

Типы данных:

- 1) ID – int;
- 2) текст – text;
- 3) дата – timestamp;
- 4) все остальные столбцы – int.

22. Заказы – хранит информацию о созданных заказах в компании, которые имеют уникальный и внутренний идентификатор (табл. 23).

Таблица 23 – Таблица БД «Заказы»

ID	Первый платеж	ID автора	ID клиента	Дата	Внутренний ID	Дефект

Продолжение таблицы 23

ID устройства	ID инженера	ID приемщика	Гарантия	Удален	Прочее

Типы данных:

- 1) ID – int;
- 2) первый платеж – real;
- 3) дата – timestamp;
- 4) дефект – text;
- 5) гарантия – int;
- 6) удалён – bool;
- 7) все остальные столбцы – int.

23. Группы разрешений – хранит информацию о существующих в системе группах разрешений (табл. 24).

Таблица 24 – Таблица БД «Группы разрешений»

ID	Наименование

Типы данных:

- 1) ID – int;
- 2) наименование – text.

24. Разрешения – хранит информацию о разрешениях пользователей (табл. 25).

Таблица 25 – Таблица БД «Разрешения»

ID	Описание	ID группы	ID состояния	Значение

Типы данных:

- 1) ID – int;
- 2) описание – text;
- 3) значение – text;
- 4) все остальные столбцы – int.

25. Телефоны – хранит информацию о номерах телефонов (табл. 26).

Таблица 26 – Таблица БД «Телефоны»

ID	Номер	ID владельца

Типы данных:

- 1) ID – int;
- 2) значение – text;
- 3) все остальные столбцы – int.

26. Описания товаров – хранит информацию о когда-либо оприходованных материальных ресурсов в компании (табл. 27).

Таблица 27 – Таблица БД «Описания товаров»

ID	ID категории	Описание	Удалено	Наименование	ID компании

Типы данных:

- 1) ID – int;
- 2) ID категории – int;
- 3) описание – text;
- 4) удалено – bool;
- 5) наименование – text;
- 6) ID компании – int.

27. Цены на ресурсы – хранит информацию о ценах на материальных ресурсы компании с учётом времени (табл. 28).

Таблица 28 – Таблица БД «Цены на ресурсы»

ID	Дата	ID определения	Значение

Типы данных:

- 1) ID – int;
- 2) дата – timestamp;
- 3) ID определения – int;
- 4) значение – real.

28. Товары – хранит информацию о товарах компании (табл. 29).

Таблица 29 – Таблица БД «Товары»

ID	Статья	Цена покупки	ID кредита	ID внутренний	ID дебета	Удален

Продолжение таблицы 29

ID перемещения	ID заказа	ID склада	ID продажи	Цена продажи	ID определения

Типы данных:

- 1) ID – int;
- 2) статья – text;
- 3) удален – bool;
- 4) все остальные столбцы – int.

29. Профили – хранит дополнительную информацию о пользователях (табл. 30).

Таблица 30 – Таблица БД «Профили»

ID	ID аватара	Оплата за работу	Имя	Оплата за обработку заказа	Оплата за продажу	Фамилия

Продолжение таблицы 30

Оплата за создание заказа	Должность

Типы данных:

- 1) ID – int;
- 2) ID аватара – int;
- 3) оплата за работу – real;
- 4) оплата за обработку заказа – real;
- 5) оплата за продажу – real;
- 6) все остальные столбцы – text.

30. Запросы – хранит информацию о запросах запчастей для заказа (табл. 31).

Таблица 31 – Таблица БД «Запросы»

ID	Количество	ID автора	Дата	ID заказа	Цена	Статус

Продолжение таблицы 31

ID курьера	ID отделения	ID дебета

Типы данных:

- 1) ID – int;
- 2) дата – timestamp;
- 3) цена – real;
- 4) все остальные столбцы – int.

31. Продажи – хранит информацию о созданных продажах в компании (табл. 32).

Таблица 32 – Таблица БД «Продажи»

ID	ID автора	ID клиента	Комментарий	ID внутренний	Дата	Оплачено

Продолжение таблицы 32

ID склада	Удален	ID кредита	ID кассового движения	ID отделения

Типы данных:

- 1) ID – int;
- 2) дата – timestamp;
- 3) комментарий – text;
- 4) оплачено – real;
- 5) удален – bool;
- 6) все остальные столбцы – int.



32. Услуги – хранит информацию о созданных услугах в заказах мастерских (табл. 33).

Таблица 33 – Таблица БД «Услуги»

ID	Наименование	ID заказа	Цена

Типы данных:

- 1) ID – int;
- 2) наименование – text;
- 3) ID заказа – int;
- 4) цена – real.

33. Настройки – хранит информацию о настройках компании в системе (табл. 34).

Таблица 34 – Таблица БД «Настройки»

ID	Внешний вид	Гарантия

Типы данных:

- 1) ID – int;
- 2) внешний вид – text;
- 3) гарантия – int.

34. Статусы – хранит информацию о статусах заказа (табл. 35).

Таблица 35 – Таблица БД «Статусы»

ID	ID компании	Удален	ID категории	Системный	Наименование

Типы данных:

- 1) ID – int;
- 2) наименование – text;
- 3) удален – bool;
- 4) системный – bool;
- 5) все остальные – int.

35. Состояния – хранит информацию о состояниях системы, необходимых для сокрытия интерфейса на клиентской части (табл. 36).

Таблица 36 – Таблица БД «Состояния»

ID	Наименование

Типы данных:

- 1) ID – int;
- 2) Наименование – text.

36. Склады – хранит информацию о складах компании (табл. 37).

Таблица 37 – Таблица БД «Склады»

ID	ID компании	Удален	Наименование	Тип	ID отделения

Типы данных:

- 1) ID – int;
- 2) ID компании – int;
- 3) Удален – bool;
- 4) Наименование – text;
- 5) Тип – int;
- 6) ID отделения – int.

37. Теги – хранит информацию о тегах для фильтрации кассовых движений (табл. 38).

Таблица 38 – Таблица БД «Теги»

ID	ID компании	Значение

Типы данных:

- 1) ID – int;
- 2) ID компании – int;
- 3) Значение – text.

38. Отделения – хранит информацию о филиалах компании (табл. 39).

Таблица 39 – Таблица БД «Отделения»

ID	ID адреса	ID компании	Удалено	Наименование	ID иконки

Типы данных:

- 1) ID – int;
- 2) ID адреса – int;
- 3) ID компании – int;
- 4) Удалено – bool;
- 5) Наименование – text;
- 6) ID иконки – int.

Помимо этого в базе реализовано еще 12 дополнительных таблиц для реализации связей между этими таблицами и работы с .NET Identity [2].

В общей сложности проект использует 50 таблиц базы данных PostgreSQL, включая таблицы-помощники для хранения значений справочников и системные таблицы для корректной работы выбранного фреймворка (рис. 12).

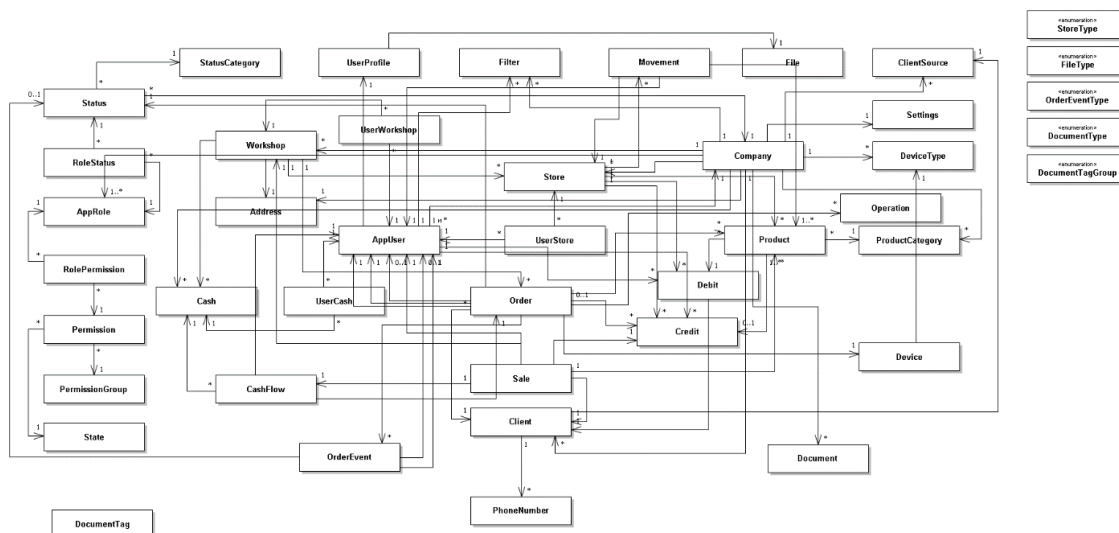


Рисунок 12 – Физическая модель базы данных

## 2.4 Выбор технологий для разработки проекта

Разработка программных решений такого уровня подразумевает под собой разбиение логического функционала программного решения на слабовзависимые друг от друга компоненты, в итоге соединенные между

собой. Удобство такого подхода заключается в том, что компоненты можно с легкостью заменить в случае необходимости. Например, если потребуется перейти на другую СУБД, не нужно будет затрагивать программный код в других модулях, а изменить только тот модуль, который отвечает за работу непосредственно с самой базой.

Для разработки CRM системы «МойСервис» был выбран фреймворк .NET Core.

.NET Core – аналог .NET Framework с открытым исходным кодом. Он является кроссплатформенным, то есть работает как на Windows, так и на Mac и Linux.

Как показала практика, .NET Core показал себя с хорошей стороны. Скорость выполнения одних и тех же запросов API увеличилась вдвое, и в добавок появилась возможность сократить стоимость на хостинг благодаря кроссплатформенности.

Систему было решено в два этапа: реализация одностраничного приложения и внешней оболочки. Обе эти системы представляют собой веб-приложения.

Внешняя оболочка – это индексируемое поисковыми системами веб-содержимое (посадочная страница, информация о системе и т.д.). Для того, чтобы данный контент был индексируемым, он должен быть сгенерирован на серверной части, и для этого была использована технология обработки представлений в ASP.NET – «Razor».

Одностраничное приложение – это веб-приложение, которое использует лишь один HTML-документ как каркас для остальных веб-страниц. Особенность данного типа веб-приложений в том, что маршрутизация обрабатывается исключительно на клиентской части (с помощью JavaScript), а от серверной части требуется отдавать лишь index.html и обрабатывать API-запросы для получения данных [21].

Учитывая то, маршрутизация обрабатывается на клиентской части, страницу веб-приложения обновлять не нужно, и заметно улучшает

производительность веб-приложения, а так же добавляет новые возможности для создания графических эффектов (плавное переключение между активными окнами и прочее).

Учитывая то, что структура базы данных будет постоянно меняться, и ее содержимое нужно будет переносить между таблицами, было решено использовать Entity Framework Migrations для решения этой проблемы. Эта технология позволяет автоматизировать генерацию SQL-команд при изменении моделей базы данных при подходе Code First. Перед их непосредственным выполнением можно проверить команды на валидность и поменять их код, если это необходимо [15].

Для реализации данного веб-приложения было выбрано два фреймворка: AngularJS и Bootstrap 3.

AngularJS – это JS-фреймворк с открытым исходным кодом, предназначенный для разработки Single Page Application. Основан на паттерне MVC [1].

Bootstrap 3 представляет собой крупный набор шаблонов элементов на HTML и CSS для быстрого создания разметки сайтов. Помимо этого имеются готовые компоненты, использующие JS [3].

### **Выводы по второй главе**

Цели проекта можно считать достигнутыми в том случае, если были выполнены следующие требования, сформированные на этапе проектирования системы:

- 1) система может интегрироваться в любой из сервисных центров уровня малого и микро-бизнеса;
- 2) система ведет учет клиентов, заказов, касс, кассовых движений и продаж;
- 3) система имеет возможность создавать, изменять и удалять печатные формы любых документов, а так же формировать по ним готовые к печати документы при работе с заказом или продажей;

4) владелец сервисного центра больше не использует 2 и более сервисов для ведения учета своей деятельности.

На этом этапе было совершено проектирование базы данных, в ходе которой были разработаны следующие таблицы:

1. AppUser – хранит информацию о пользователях.
2. UserProfile – хранит дополнительную информацию о пользователях.
3. Status – хранит статусы.
4. Role – хранит роли.
5. Permission – хранит модификаторы доступа для пользователей.
6. State – хранит список состояний веб-сервиса.
7. Workshop – хранит список мастерских.
8. Address – хранит список адресов для мастерских.
9. Cash – хранит денежные средства.
10. Cashflow – хранит движения денежных средств.
11. OrderEvent – хранит список действий с заказами.
12. Filter – хранит список фильтров заказов.
13. Store – хранит список складов.
14. Order – хранит список заказов.
15. Sale – хранит список продаж.
16. Client – хранит список клиентов.
17. Movement – хранит список передвижений товаров.
18. Company – хранит информацию о компаниях.
19. Product – хранит информацию о товарах.
20. Debit – хранит поступления денежных средств в кассу.
21. Credit – хранит отчисления денежных средств из кассы.
22. File – хранит информацию о файлах.
23. Document – хранит информацию о документах.
24. ClientSource – хранит информацию о рекламных кампаниях.
25. Settings – хранит информацию о настройках веб-сервиса для клиента.
26. DeviceType – хранит информацию о типах устройств.

- 27. Operation – хранит информацию об операциях пользователя.
- 28. ProductCategory – хранит информацию о типах товаров.
- 29. Device – хранит информацию об устройствах.
- 30. PhoneNumber – хранит список телефонов.

Помимо вышеперечисленных в системе еще используются системные таблицы, описывающие связи между ними и хранящие метаданные.

## Глава 3. Разработка программного решения

### 3.1 Главная страница

Под главной страницей подразумевается посадочная страница программного продукта, содержащая в себе краткую информацию о продукте в таком виде, чтобы только что посетивший пользователь был заинтересован в дальнейшем использовании программного решения (рис. 13).

Для реализации эффективных посадочных страниц существует множество информации, среди которой можно выделить такие основные идеи, как:

- 1) наличие шапки сайта, в которой содержится идентифицирующая проект информация;
- 2) наличие информации об особенностях проекта;
- 3) наличие отзывов о проекте. Важно заметить, что отзывы наиболее эффективны, если к ним прикреплены реальные фотографии;
- 4) контакты;
- 5) «побудители» действия, то есть элементы интерфейса, наличие которых побуждает пользователя совершить то или иное действие. В частности, это кнопки регистрации в системе.

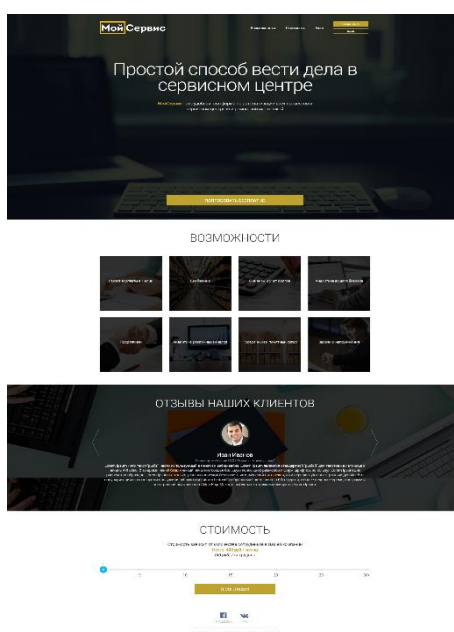


Рисунок 13 – Посадочная страница «МойСервис»



Посадочная страница использует такие технологии, как HTML5, CSS3 и JQuery. В процессе вёрстки был использован сборщик файлов клиентской части Gulp [18].

Gulp представляет собой инструмент для веб-разработчиков, позволяющий автоматизировать рутинные задачи в веб-разработке. С его помощью были реализовано следующее:

- 1) был запущен веб-сервер с помощью BrowserSync;
- 2) был использован CSS препроцессор Sass;
- 3) все файлы стилей (включая библиотеки) были сжаты и сконкатинированы в один файл;
- 4) все файлы JS (включая библиотеки) были сжаты и сконкатинированы в один файл;
- 5) были сжаты изображения;
- 6) был использован плагин live-reload, обновляющий представление страницы после обновления файлов.

Если пользователь нажмет на любую кнопку, побуждающую на действие, у него откроется модальное окно регистрации (рис. 14 и рис. 15).

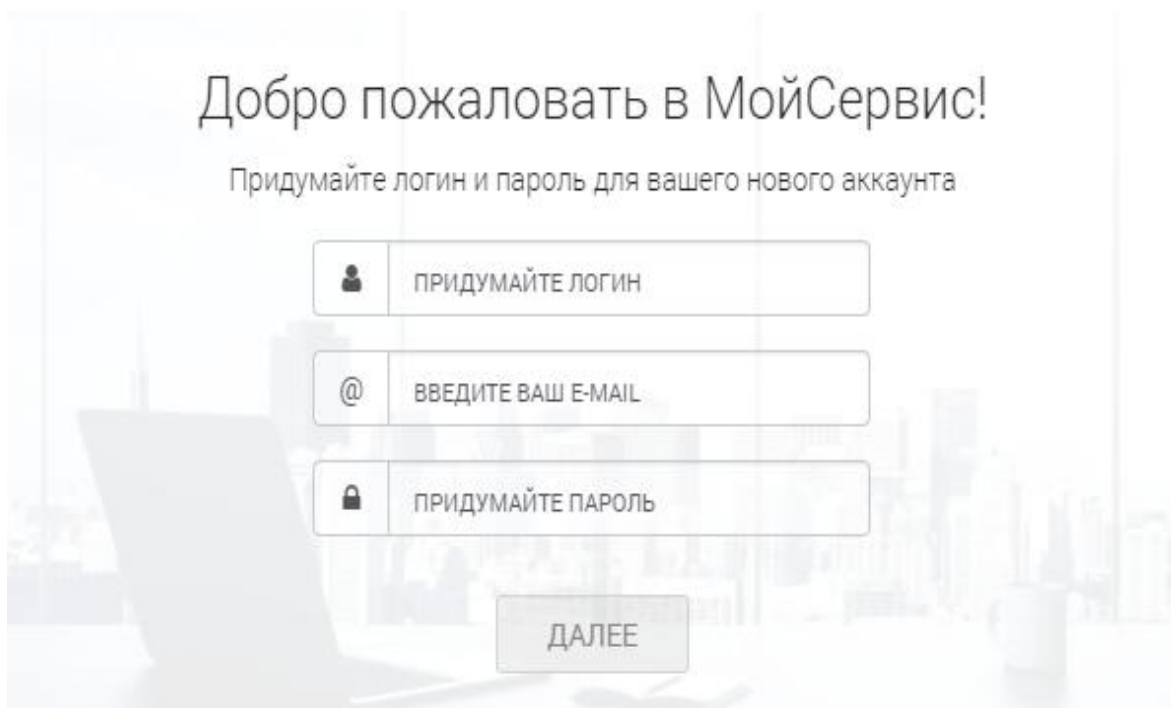


Рисунок 14 – Модальное окно регистрации компании

Создайте первого сотрудника вашей компании!

НАЗВАНИЕ ВАШЕЙ КОМПАНИИ

ИМЯ

ФАМИЛИЯ

ЗАРЕГИСТРИРОВАТЬСЯ

Рисунок 15 – Модальное окно регистрации компании. Этап 2

После прохождения двух этапов регистрации пользователь перенаправляется из посадочной страницы на одностраничное приложение (рис. 16).

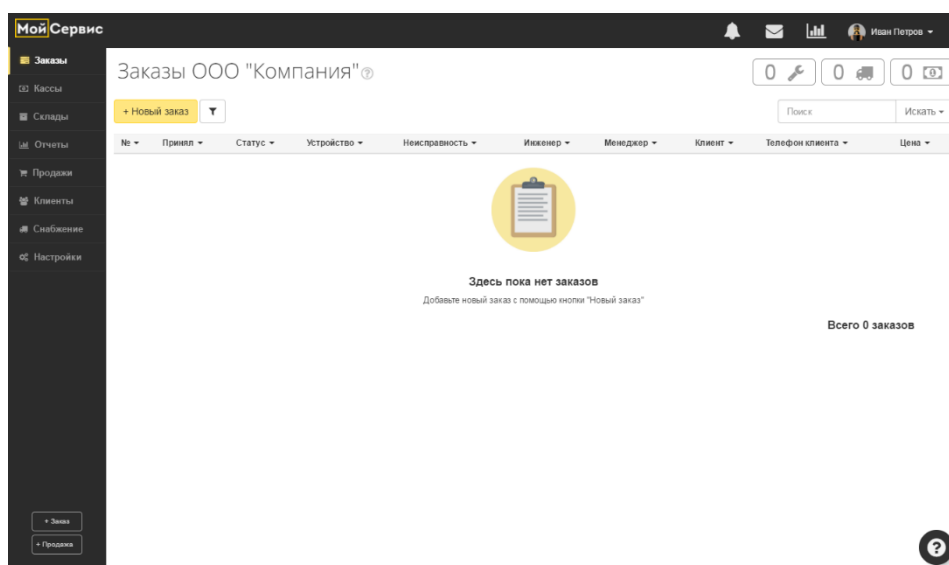


Рисунок 16 – Модальное окно регистрации компании. Этап 2

При полной загрузке однострачного приложения пользователь может полноценно работать с ним. У него становится доступна работа со следующими модулями:

- 1) заказы;
- 2) кассы;
- 3) склады;
- 4) отчеты;

- 5) продажи;
- 6) клиенты;
- 7) снабжение;
- 8) настройки.

## 3.2 Модуль «Заказы»

Это основной модуль в системе, через который осуществляется работа с заказами. При отсутствии заказов пользователь может лишь создать новый заказ. Для этого нужно нажать на соответствующую кнопку под названием модуля (рис. 17).

Создание нового заказа №1

---

**Клиент**

Имя \*

Телефон +7

Откуда узнал Выберите рекламный канал ▼ ➕

Подменное устройство ☐

**Устройство**

Тип устройства \* Выберите тип устройства ▼ ➕

Наименование \*

Серийный номер \*


Неисправность \*

Внешний вид Царапины, потертости

Пароль

Комплектация

Заметки

Менеджер  Иван Петров ▼

Инженер Не выбран ▼

Аванс 0  Р

Ориентировочная стоимость  Р

---

Рисунок 17 — Модальное окно создания заказа

При открытии модального окна происходит взаимодействие с серверной частью. Происходит следующий ряд AJAX-запросов:

- 1) загрузка предварительного номера заказа;
- 2) загрузка рекламных кампаний;
- 3) загрузка типов устройств;
- 4) загрузка списка менеджеров;

- 5) загрузка списка инженеров;
- 6) загрузка документов для печати.

Как только пользователь заполняет все обязательные поля в корректной форме, он пройдет валидацию, и ему станет доступна кнопка создания заказа.

При создании заказа объект заказа передается в серверную часть в соответствующий action. В action данные перепроверяются на валидность, и если проверка была пройдена, создается заказ и сохраняется в базу. В случае, если вместе с заказом была введена информация о новом пользователе, на сервере так же создается сущность нового пользователя и привязывается к заказу.

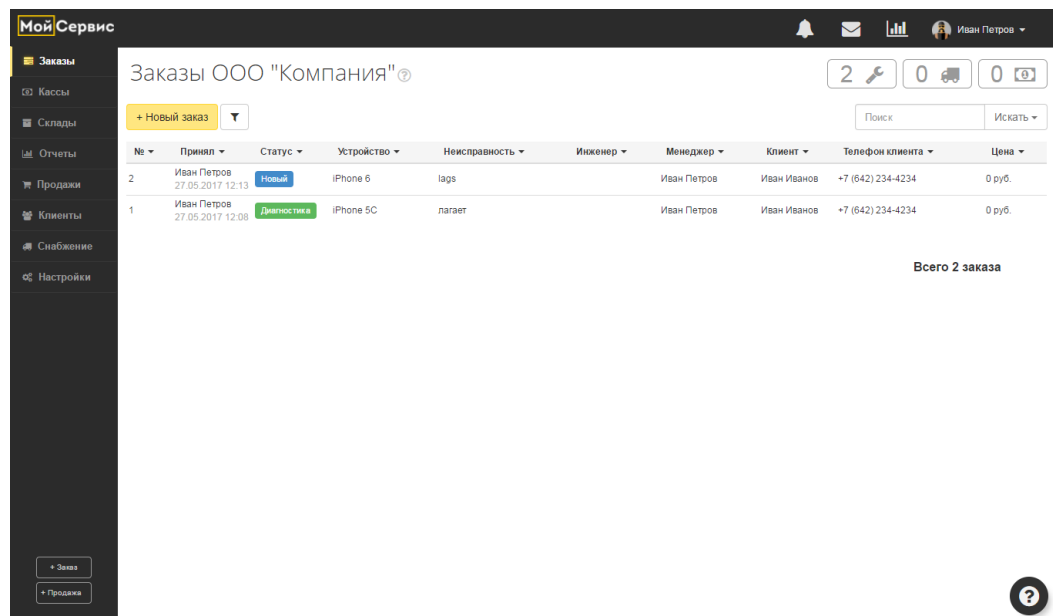


Рисунок 18 — Модуль заказов после создания заказов

На рис. 18 видно, что после создания 2 заказов интерфейс несколько обновился. В таблице можно наблюдать краткую информацию о заказе, над таблицей есть элементы управления для фильтрации существующих заказов. Наиболее простым в использовании является поиск.

Элемент поиска представляет собой текстовое поле с выпадающим списком параметров. Пользователю достаточно ввести какой-либо текст и выбрать критерий поиска, по которому этот текст будет обрабатываться на сервере. В случае, если пользователь не выберет критерий поиска, а просто

введет текст, система будет искать совпадения по всем доступным полям (рис. 19).

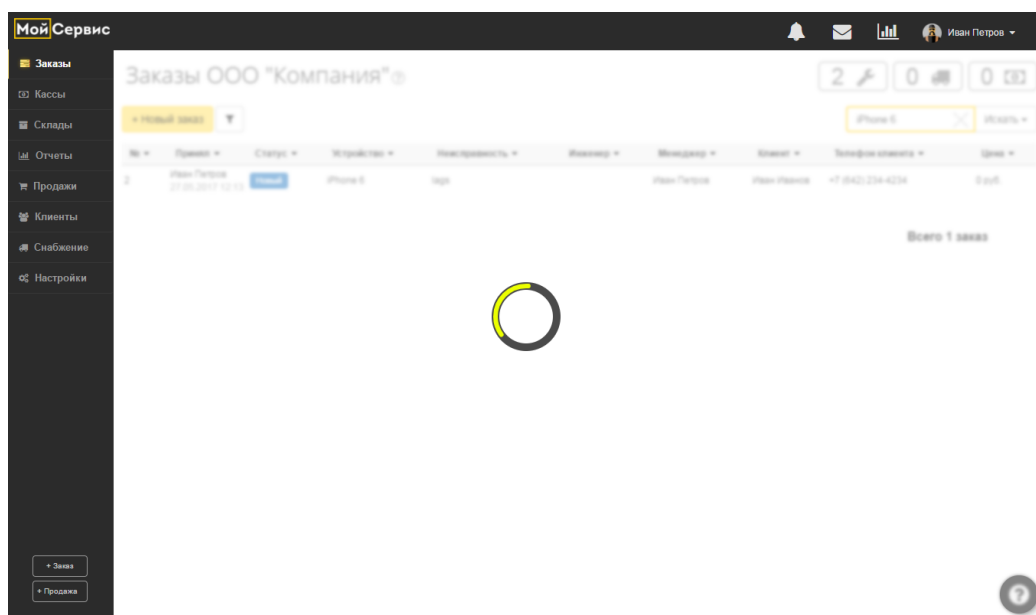


Рисунок 19 – Элемент поиска в модуле заказов

Следующим элементом фильтрации заказа являются предустановленные фильтры, располагающиеся над элементом поиска. Если навести курсор на любой из этих элементов, пользователю выводится краткое описание данного фильтра (рис. 20).

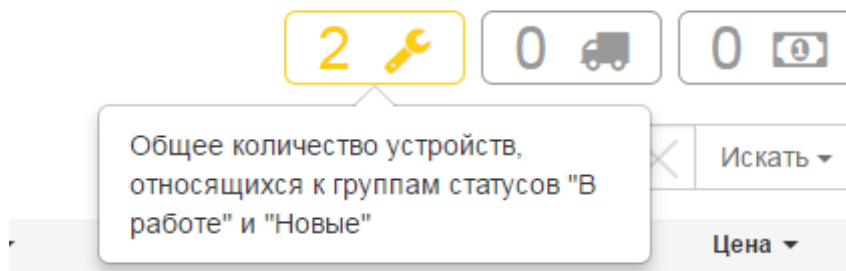


Рисунок 20 – Предустановленный фильтр

После нажатия на фильтр отправляется соответствующий запрос на сервер, после чего обновляется таблица так же, как это было с элементом поиска.

Третим элементом фильтрации заказа является фильтр заказов. Этот инструмент имеет наиболее расширенный функционал. Он позволяет фильтровать существующие заказы по заданным пользователям критериям.

Помимо фильтрации, пользователь может сохранить фильтр в случае, если ему будет необходимо использовать его не один раз, а так же позволять использовать этот фильтр другим сотрудникам компании (рис. 21).

МойСервис

Заказы ООО "Компания" ©

+ Новый заказ

Поиск Искать

Статус Любой Тип заказа Любой Дата заказа За все время

Тип устройства Бренд Модель

Клиент Менеджер Любой Исполнитель Любой

Применить Сохранить фильтр Сбросить фильтр

№	Принят	Статус	Устройство	Неисправность	Инженер	Менеджер	Клиент	Телефон клиента	Цена
2	Иван Петров 27.05.2017 12:13	Новый	iPhone 6	lags		Иван Петров	Иван Иванов	+7 (642) 234-4234	0 руб.
1	Иван Петров 27.05.2017 12:08	Диагностика	iPhone 5C	лагает		Иван Петров	Иван Иванов	+7 (642) 234-4234	0 руб.

Всего 2 заказа

+ Заказ + Продажа

?

Рисунок 21 – Фильтр заказов

После заполнения необходимых пользователю для его нужд полей он может сохранить или применить фильтр, нажав на соответствующие кнопки слева внизу блока фильтров.

В случае применения фильтра таблица будет обновлена. Если же сохранить фильтр, то будет открыто модальное окно, в котором пользователю предложат задать название и доступ к фильтру (рис. 22).

## Создание нового фильтра

Название \*

Придумайте название фильтра

☐ Общий фильтр

Создать Отмена

Рисунок 22 – Создание фильтра

Помимо фильтрации пользователю доступен просмотр существующего заказа с возможностью его изменения и удаления (рис. 23).

Заказ №1

Текущий статус: Диагностика

27 мая

Клиент

Имя \* Иван Иванов  
Телефон +7 (642) 234-4234  
Аванс 500 ☐ Внести  
Ор. цена 4500 ☐  
Поданное устройство ☒  
Заметки  
Менеджер Иван Петров

Устройство

Тип устройства \* Смартфон  
Наименование \* iPhone 5C  
Серийный № \* 41141442  
Неисправность \* лагает  
Внешний вид Царапины, потертости  
Пароль  
Комплектация

Работы и запчасти

Инженер Не выбран  
Гарантия ☒ 3 мес.  
Введите название работы руб. +  
Введите название запчасти +  
Наименование работы Цена  
Итого: 0 руб.  
Введите название запчасти  
Выбрать со склада Создать запрос  
№ Название запчасти Цена  
Итого: 0 руб.  
Добавить техническое заключение  
Сохранить Отмена Действие

Новый

Назначен менеджер: Иван Петров

Диагностика

+ Добавить комментарий

Оплачено: 500 руб. Сумма: 0 руб.

Рисунок 23 — Карточка заказа

В карточке заказа пользователю доступен просмотр и изменение информации о клиенте, устройстве и деталях конкретного заказа. Помимо этого реализована хронологическая история действий с заказом, каждое действие из которых закрепляется за конкретным пользователем системы.

В нижней части карточки заказа можно добавить конкретные работы и привязать запчасти к заказу. А так же можно выбрать инженера, выполняющего работу по заказу (рис. 24).

## Работы и запчасти

Инженер Иван Петров

Гарантия ☒ 3 мес.

Введите название работы руб. +

Введите название запчасти +

Наименование Цена

Установка дисплея 750 руб.

Итого: 0 руб.

Введите название запчасти

Выбрать со склада Создать запрос

№ Название запчасти Цена

1 экран iphone 5c 6 000 руб.

Итого: 0 руб.

Добавить техническое заключение

Сохранить Отмена Действие

Рисунок 24 — Карточка заказа. Работы и запчасти

У каждого заказа есть возможность печати документов. Печать осуществляется через стандартную функцию print() в окне браузера (рис. 25).

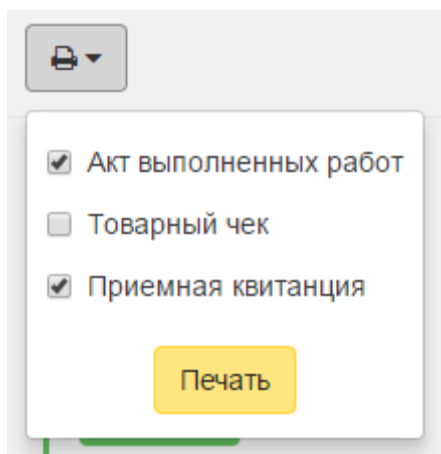


Рисунок 25 – Выбор документов для печати

После нажатия кнопки печать на сервер отправляется AJAX-запрос. В ответе возвращается сгенерированный на сервере html-разметку документа для печати. Эта разметка подставляется в скрытое окно, после чего начинается печать (рис. 26).

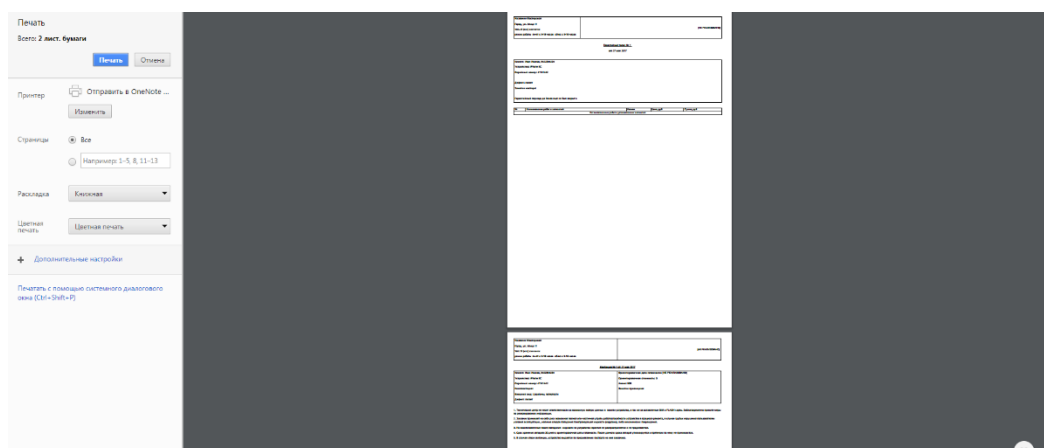


Рисунок 26 – Печать документов через Google Chrome

В заказах реализована система статусов, содержащая в себе системные и пользовательские виды статусов. Часть из них заполняется автоматически при создании компании, остальные пользователь может создать вручную на свое усмотрение (рис. 27).

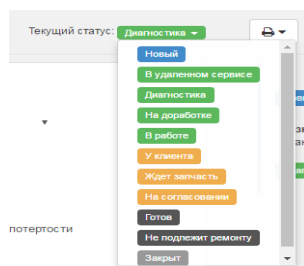


Рисунок 27 – Стандартные статусы заказа



### 3.3 Модуль «Кассы»

Этот модуль предназначен для учета денежных оборотов внутри компании пользователя системы.

Интерфейс данного модуля разделен на две составляющие: кассы и денежные потоки.

Кассы представляют собой цифровой аналог касс пользователя в его организации. При работе с заказами в процессе трудовой деятельности менеджер указывает в системе когда и сколько было выплачено клиентами, а так же по какому заказу. Все это реализуется с помощью удобного интерфейса (рис. 28).

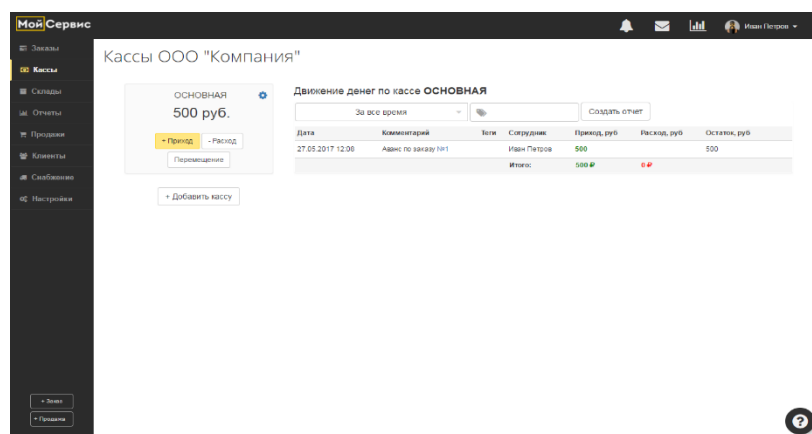


Рисунок 28 — Модуль «Кассы»

Если пользователю необходимо создать новую кассу, под блоком касс есть соответствующая кнопка, по нажатию на которую появится следующее модальное окно (рис. 29).

#### Создание новой кассы

Название кассы \*

☒ Касса отделения ?

☐ Глобальная касса ?

Рисунок 29 — Модальное окно создания кассы

Помимо создания, пользователю доступно ее изменение и удаление. Этот функционал доступен после нажатия на шестеренку в блоке нужной пользователю кассы. Интерфейс изменения полностью идентичен созданию новой кассы за исключением наименования модального окна и текста в кнопке действия.

С кассой можно проводить следующие операции:

- 1) приход денежных средств;
- 2) расход денежных средств;
- 3) перемещение денежных средств.

Данные операции можно совершить, нажав на соответствующие кнопки в блоке кассы. Все операции имеют идентичные модальные окна (рис. 30).

## Приход денежных средств

---

**Сумма \***

**Комментарий**

**Теги**

Можете ввести теги через запятую

Выполнить

Отмена

Рисунок 30 — Модальное окно прихода денежных средств

Теги в данных операциях представляют собой возможность их дальнейшей фильтрации (рис. 31).

## Движение денег по кассе **ОСНОВНАЯ**

За все время		до-ввода-системы		Создать отчет		
Дата	Комментарий	Теги	Сотрудник	Приход, руб	Расход, руб	Остаток, руб
27.05.2017 13:09	Начальный остаток май 2017 до-ввода-системы		Иван Петров	5 000		5 500
Итого:				5 000 Р	0 Р	

Рисунок 31 – Фильтрация денежных средств

### 3.4 Модуль «Склады»

Следующий модуль предназначен для учета материальных ресурсов компании. Он предназначен для работников, отвечающих за складской учет и имеющих привелегии для обработки запросов на определенные товары и запчасти. Как правило, в небольших компаниях для этого не выделяют отдельную должность, а отвечает за это инженер или директор компании, поэтому в настройках системы имеется возможность установки уровня доступа к модулю.

В данном модуле пользователь имеет следующий спектр возможностей:

- 1) просмотр и работа с остатками;
- 2) создание и просмотр истории оприходований;
- 3) создание и просмотр истории списаний;
- 4) создание и просмотр истории перемещений.

При работе с остатками пользователь может выбрать интересующий его склад, после чего выполнится AJAX-запрос, результат которого будет помещен в таблицу. В таблице будут отображены все товары, оставшиеся на складе на данный момент. У пользователя имеется возможность менять последнюю розничную цену выбранного товара в таблице остатков, а так же просмотреть полный список товаров с их сквозной нумерацией. Помимо этого товара выведена его розничная цена, остаток и наименование (рис. 32).

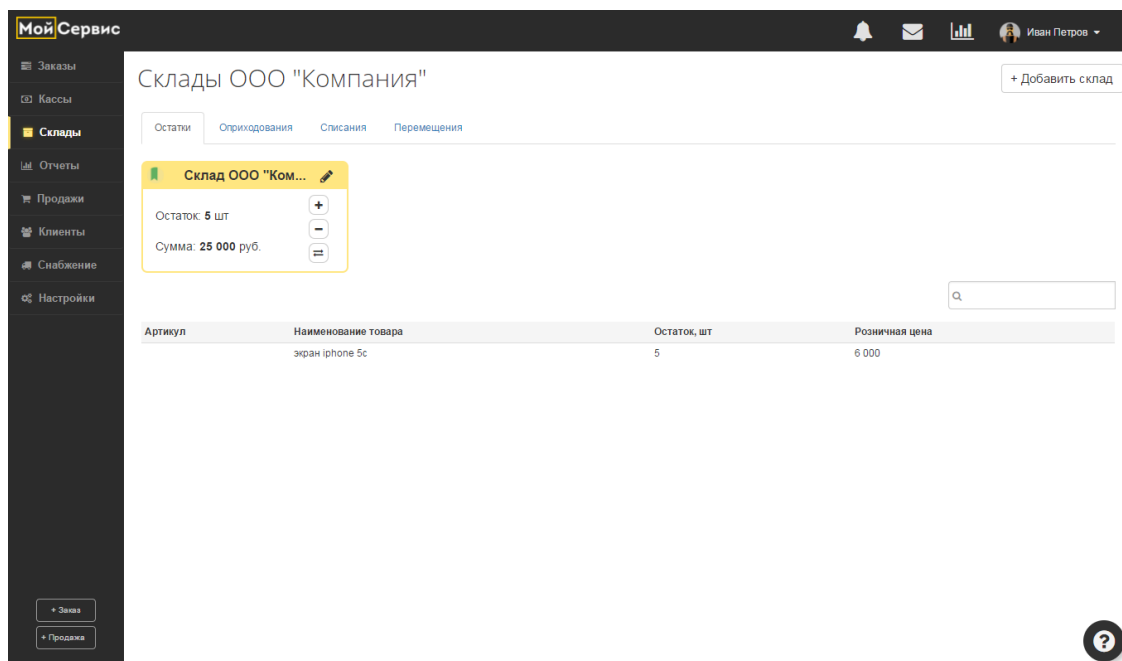


Рисунок 32 – Остатки товаров в складах

Для добавления товара на склад его нужно оприходовать. Это делается в соответствующей вкладке после нажатия на кнопку (рис. 33).

#### Оприходование товара на склад

Приходуем на склад \*  

Склад ООО "Компания" ▾

Поставщик \*

Накладная №  
 от 27.05.2017

Наименование \*  

Название запчасти

Количество \*  

1

Артикул

Закупочная цена \*  

0

Розничная цена \*  

0

Тип \*  

Товары ▾

Поля, отмеченные \* - обязательны к заполнению

☐ Оплатить из кассы

Комментарий

+ Добавить

Отмена

Рисунок 33 – Оприходование товаров

Для создания оприходования нужно выбрать поставщика, ввести информацию о товаре и нажать кнопку «Добавить». После данной череды действий оприходование пройдет валидацию, и его можно будет сохранить.

После этого можно работать с этими товарами в других модулях системы.

### 3.5 Модуль «Отчеты»

Данный модуль был создан специально для владельцев сервисных центров или директоров. Он показывает работу сервисного центра пользователя с помощью отчетов в виде графиков и таблиц.

В данном модуле есть возможность просматривать следующие виды отчетов:

- 1) статистика;
- 2) отчеты по заказам;
- 3) отчеты по зарплате;
- 4) финансовые отчеты;
- 5) рекламные каналы.

Статистика позволяет просмотреть итоговые показатели по выбранному периоду: выручку, прибыль, расходы на оприходования, принятые, выполненные заказы и продажи (рис. 34).

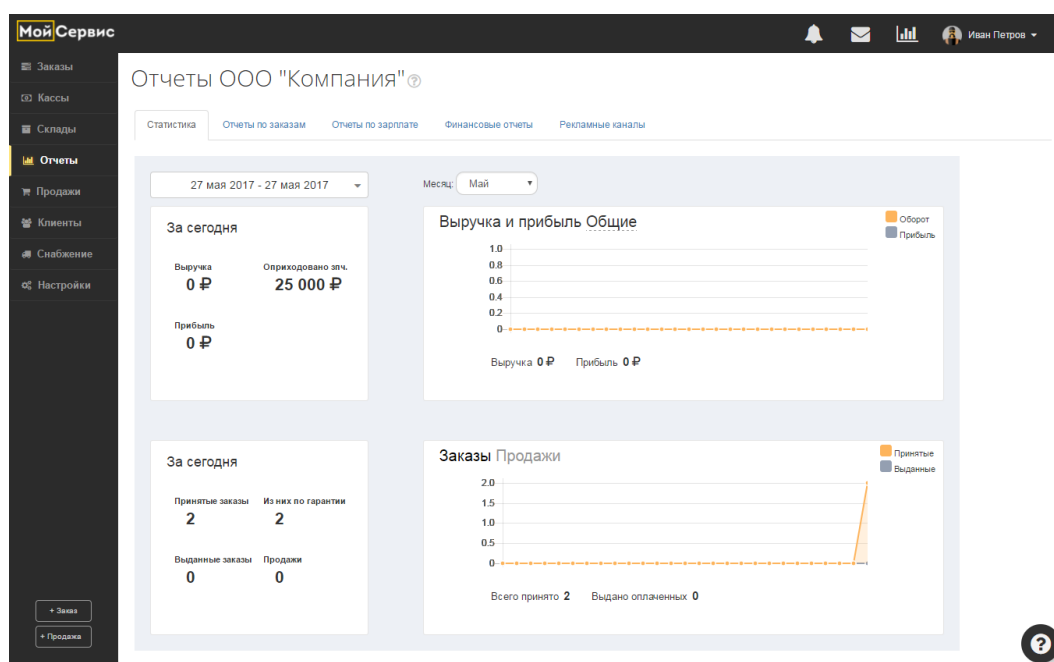


Рисунок 34 – Статистика

В отчетах по заказам показаны принятые или закрытые заказы по отчетному периоду (рис. 35).

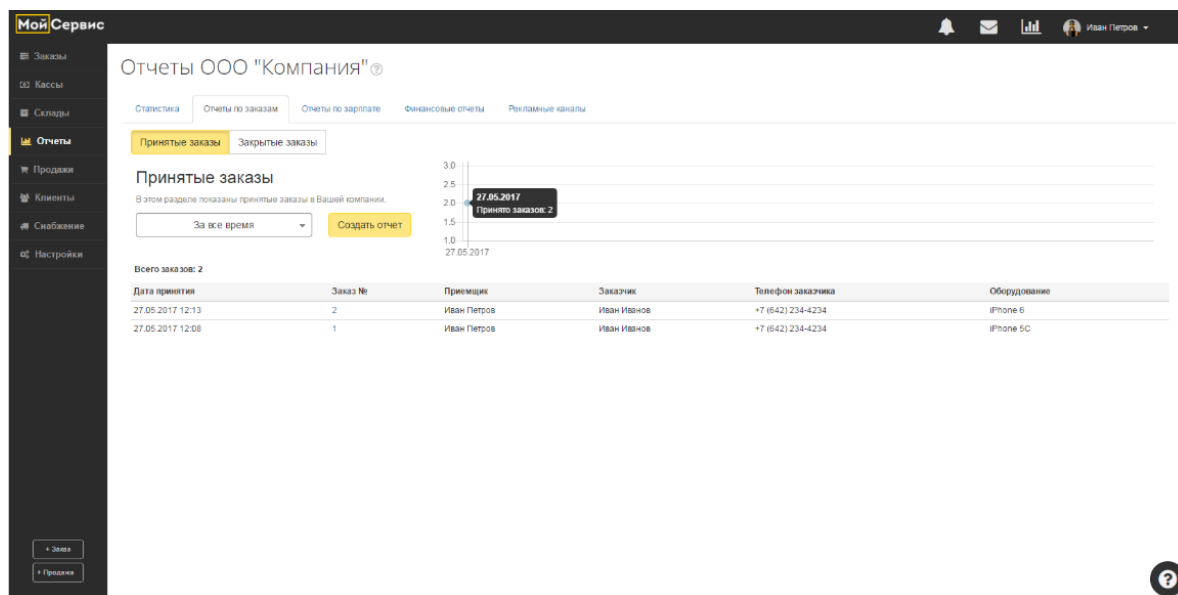


Рисунок 35 – Отчеты по заказам

В отчетах по зарплате пользователь может выбрать интересующего его работника и просмотреть выполненные им действия и расчет его зарплаты (рис. 36).

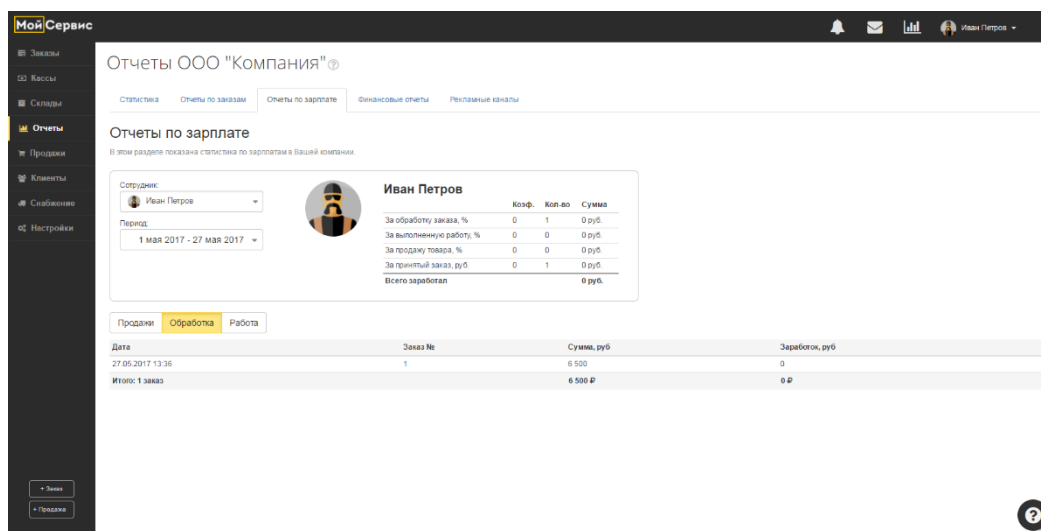


Рисунок 36 – Отчеты по зарплате

### 3.6 Модуль «Продажи»

Данный модуль представляет собой аналог модуля «Заказы» с более упрощенным интерфейсом. Под продажей подразумевается заказ, выполнение которого несет за собой приход денежных средств и списание товара со склада (рис. 37).

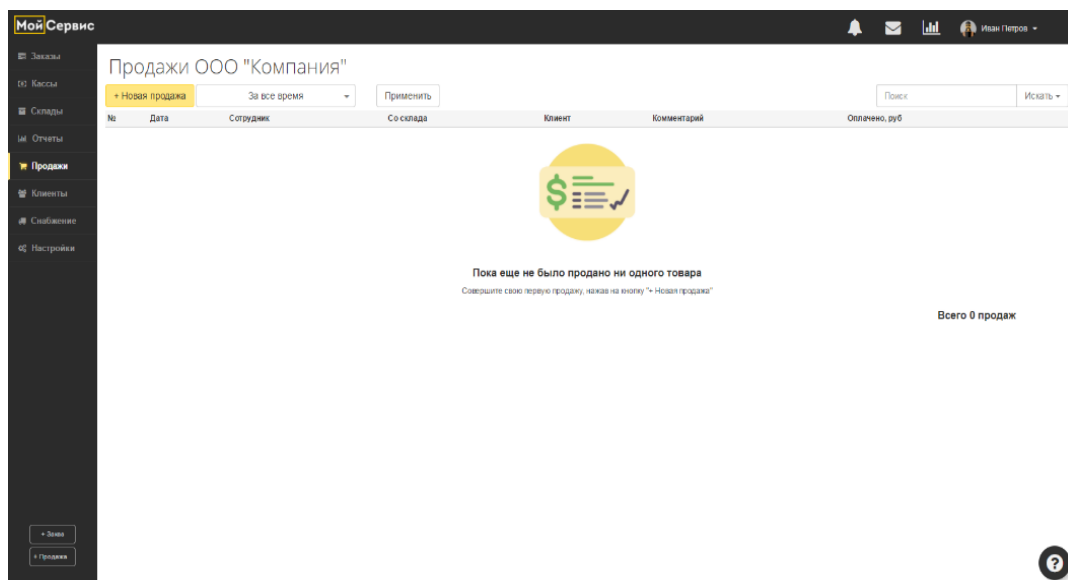


Рисунок 37 — Продажи

### 3.7 Модуль «Клиенты»

Данный модуль ведет учет клиентов, зарегистрированных в компании. Через этот модуль можно добавлять, изменять или удалять информацию о клиентах.

Стандартно клиенты создаются при создании продажи или заказа, но есть более расширенный способ создания или изменения информации о клиента — через модуль «Клиенты» (рис. 38).

#### Новый клиент

☐ Обычный
 ☐ Конфликтный
 ☐ VIP

Имя \*

Телефон

[+ Добавить телефон](#)

Email

Адрес

Откуда узнал

Примечание

Рисунок 38 — Модальное окно клиента

Интерфейс модуля так же имеет элемент управления поиска для удобной фильтрации клиентов (рис. 39).

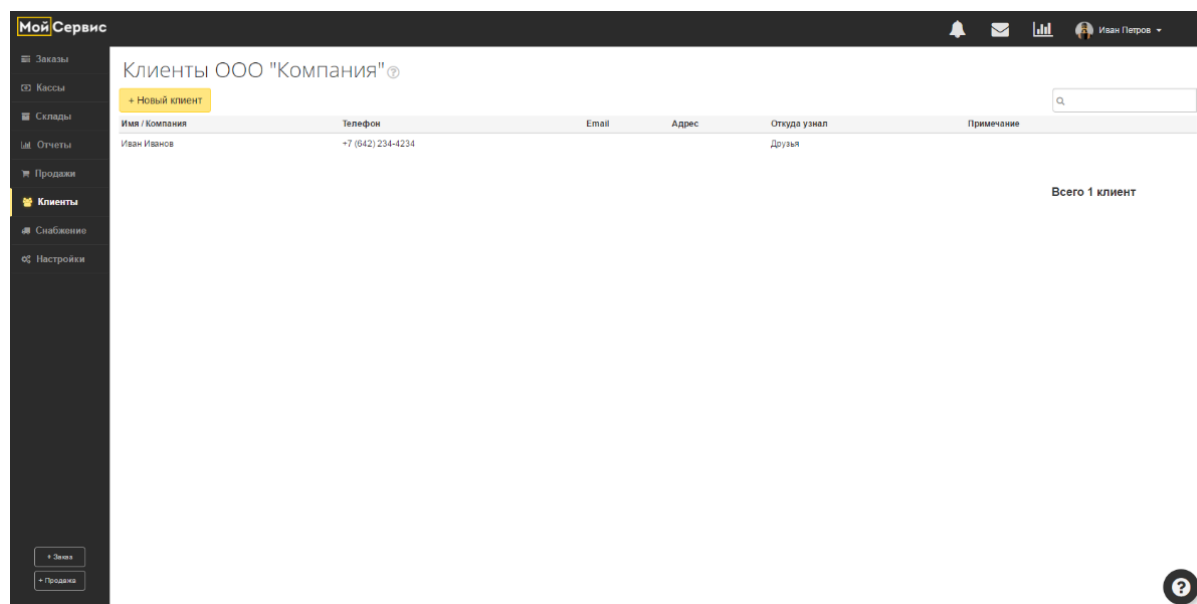


Рисунок 39 – Модуль «Клиенты»

### 3.8 Модуль «Снабжение»

Модуль «Снабжение» предназначен для предоприходования еще не существующих товаров в систему. Реализуется это с помощью создания запросов, которые можно создать в карточке заказа в блоке деталей (рис. 40).

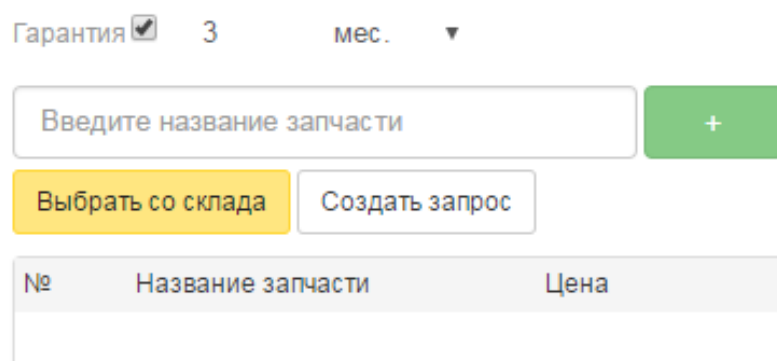


Рисунок 40 – Кнопка создания запроса в карточке заказа

После нажатия на данную кнопку появится следующее модальное окно, в котором нужно указать наименование необходимой детали. Это можно сделать путем ввода уникального наименования или выбора существующего из подсказок (рис. 41).



### Создание запроса на запчасть

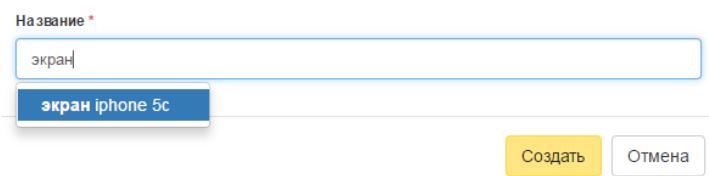
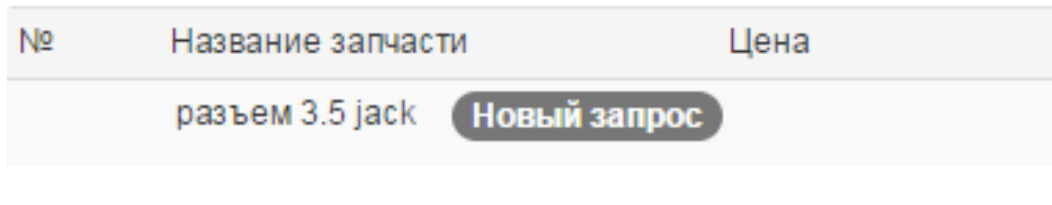


Рисунок 41 – Модальное окно создания запроса на запчасть

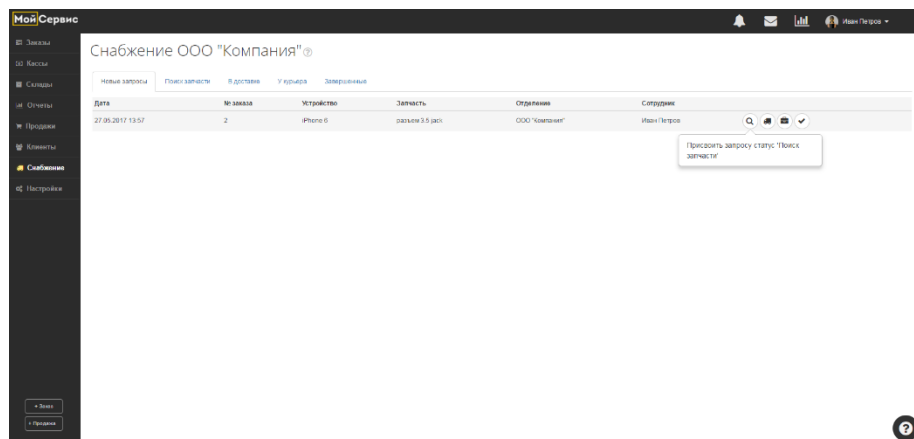
После создания запроса он будет отображаться в карточке заказа как деталь, но с селектором статуса запроса (рис. 42).



№	Название запчасти	Цена
	разъем 3.5 jack	Новый запрос

Рисунок 42 – Отображение запроса в карточке заказа

Также данный запрос будет доступен непосредственно в самом модуле «Снабжение» с возможностью работы с ним (рис. 43).



Дата	№ заказа	Устройство	Запчасть	Отделение	Сотрудник
27.05.2017 13:57	2	iPhone 6	разъем 3.5 jack	ООО "Компания"	Иван Петров

Рисунок 43 – Новые запросы в модуле «Снабжение»

## 3.9 Модуль «Настройки»

Данный модуль необходим для настройки системы в рамках компании. В нем устанавливаются predetermined значения, автоматически заносимые в какие-либо поля модальных окон при их инициализации или при печати документов (рис. 44).

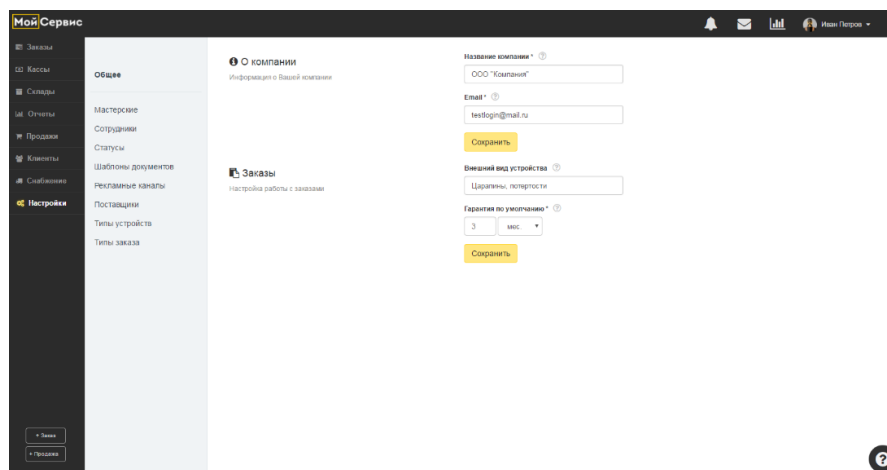


Рисунок 44 – Общие настройки

В данном модуле реализованы следующие справочники:

- 1) мастерские;
- 2) сотрудники;
- 3) статусы;
- 4) рекламные каналы;
- 5) поставщики;
- 6) типы устройств;
- 7) типы заказа.

Все типы справочников имеют единую форму отображения в виде таблицы (рис. 45).

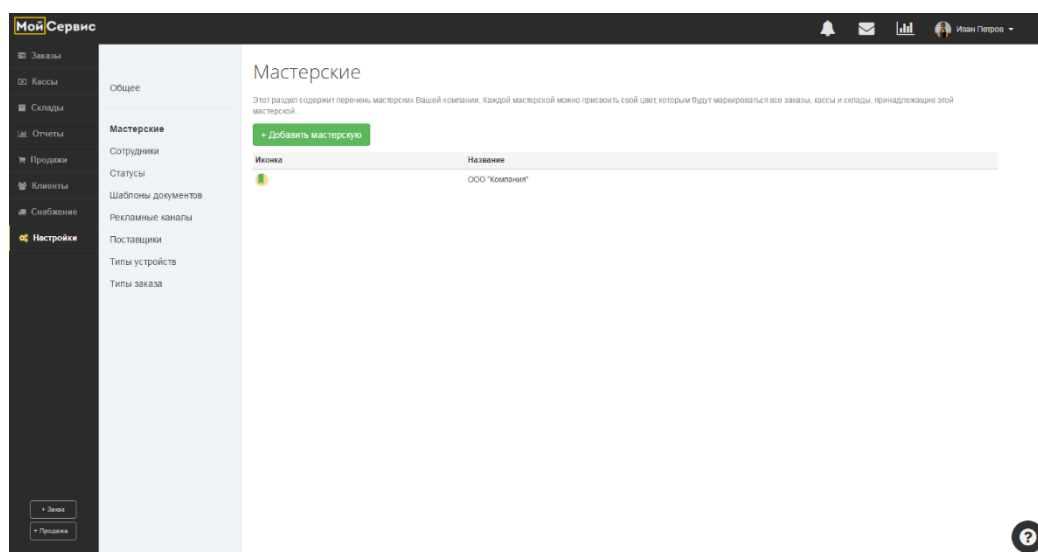


Рисунок 45 – Общий вид справочников

Справочник «Сотрудники» включает в себя так же справочник «Роли», так как эти справочники очень взаимосвязаны, с ними удобнее работать в одном окне (рис. 46).

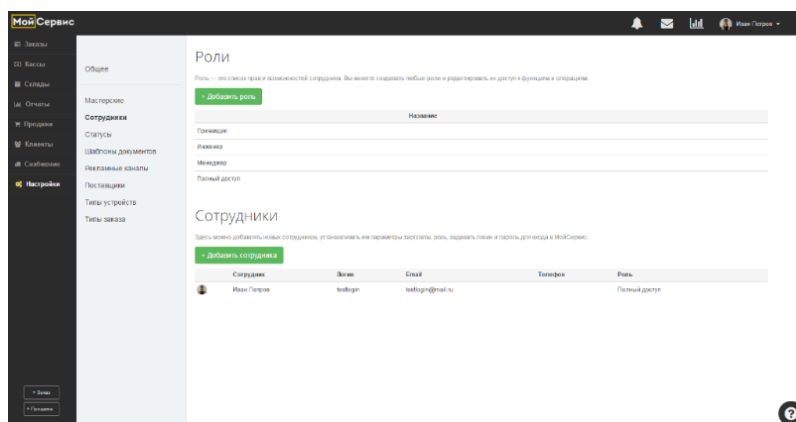


Рисунок 46 – Справочники «Сотрудники» и «Роли»

Справочник «Сотрудники» для работы с его сущностями имеет уникальное модальное окно, в котором помимо основной информации о сотруднике (имя, фамилия, телефон, email, логин, пароль) можно указать еще информацию для расчета его заработной плате, которая указывается в виде процентов от выполненной работы, продажи товара, обработки заказа или его принятия. Помимо этого ему можно указать доступ и выбрать аватар (рис. 47).

Изменить сотрудника

---

Имя \* Иван      Фамилия \* Петров

Логин \* testlogin

Пароль \*

Телефон +7

Email \* testlogin@mail.ru

Должность

Роль Полный доступ ▼

**Заработная плата**

За выполненную работу	0	% ⓘ
За продажу товара	0	% ⓘ
За обработку заказа	0	% ⓘ
За принятие заказа	0	% ⓘ

**Доступ сотрудника**

- ☒ ООО "Компания"
- ☒ Основная
- ☒ Склад ООО "Компания"

**Выбрать аватар**

Выбрать аватар

Принять
Отмена

Рисунок 47 – Модальное окно сотрудника

Помимо справочников в настройках реализован механизм настройки печати документов. Он доступен пользователю после перехода по ссылке «Шаблоны документов» в текущем модуле (рис. 48).

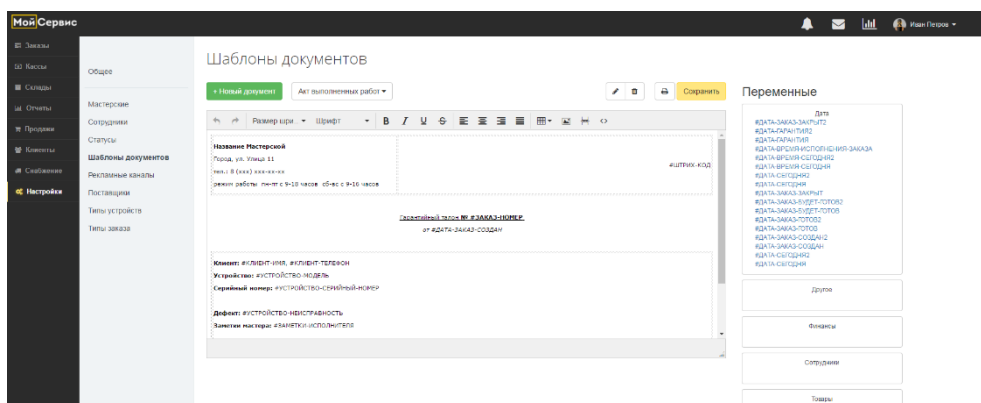


Рисунок 48 – Шаблоны документов

В данном модуле можно создать шаблон документа, который представляет из себя сущность, состоящую из названия и HTML-содержимого, которое редактируется с помощью визуального текстового редактора TinyMCE.

## Выводы по третьей главе

Была проделана следующая работа:

- 1) был разработан индексируемый поисковыми системами контент, обрабатываемый технологией «Razor»;
- 2) была реализована серверная архитектура с использованием .NET Core и PostgreSQL;
- 3) было реализовано одностраничное приложение с использованием AngularJS и Bootstrap 3;
- 4) разработан модуль «Заказы»;
- 5) разработан модуль «Кассы»;
- 6) разработан модуль «Склады»;
- 7) разработан модуль «Отчеты»;
- 8) разработан модуль «Продажи»;
- 9) разработан модуль «Клиенты»;
- 10) разработан модуль «Снабжение»;

- 11) разработан модуль «Настройки»;
- 12) реализована работа с почтой при регистрации клиента и восстановлении пароля;
- 13) реализована генерация HTML-контента для печатных форм.

### **Обоснование экономической эффективности проекта**

Расходы, необходимые для реализации проекта «МойСервис», могут быть определены с помощью метода калькуляций. В основе этого метода заключается принцип суммирования расходов по некоторым статьям расходов.

Расходы можно поделить на постоянные и переменные. Для начала будут рассмотрены постоянные издержки.

Для реализации проекта были приобретены два компьютера общей стоимостью 50 000 руб. Исходя из этого, можно подсчитать балансовую стоимость компьютера по формуле (1):

$$C_{бал} = C_{рын} * Z_{уст}, \quad (1)$$

где  $C_{бал}$  – балансовая стоимость компьютеров, руб.;

$C_{рын}$  – рыночная стоимость компьютера, руб./шт.;

$Z_{уст}$  – затраты на доставку и установку компьютера, %

Затрат на доставку и установку компьютера не было, так как это было произведено своими силами. Отсюда:

$$C_{бал} = 50000 * 1,0 = 50000 \text{ руб.}$$

Для разработки программного продукта был приобретен хостинг по цене 300 руб/мес. Проект был в разработке 90 дней. Отсюда следует, что общая стоимость хостинга составляет 900 рублей.

На компьютер и на хостинг должны быть рассчитаны амортизационные отчисления, которые и входят в группу постоянных издержек. Общую амортизацию можно рассчитать по формуле (2):

$$A_{\Pi} = A_K + A_X, \quad (2)$$

где  $A_K$  – амортизационные отчисления на приобретенные компьютеры за время их использования;

$A_X$  – амортизационные отчисления на хостинг за время его аренды.

Отсюда следует:

$$A_K = (50000 * 0,25) / 365 * 90 = 3082.19 \text{ (руб.)};$$

$$A_X = (900 * 0,25) / 365 * 90 = 55.48 \text{ (руб.)};$$

$$A_{II} = 3082.19 + 55.48 = 3137.67 \text{ (руб.)}.$$

Сведения о постоянных издержках отображены в табл. 40.

Таблица 40 – Постоянные издержки проекта

Вид постоянных издержек	Денежная оценка, руб.
Амортизационные отчисления	3137,67
Итого:	3137,67

Теперь можно приступить к рассмотрению переменных издержек. К переменным издержкам можно отнести такие виды затрат, как:

- 1) затраты на покупку материалов;
- 2) затраты на электроэнергию;
- 3) затраты на оплату труда разработчиков.

Расчет затрат на покупку материалов включает также транспортные расходы (5% от стоимости материала). Ниже представлена таблица, содержащая информацию о необходимых материалах для разработки и их стоимости (табл. 41).

Таблица 41 – Переменные издержки проекта

Наименование	Единица измерения	Количество	Цена за единицу, руб.	Стоимость, руб.
Бумага для принтера	пачка	1	50	50
Стикеры	пачка	2	50	100
Итого:				150

Теперь можно рассчитать затраты на материалы с учетом транспортировки:

$$Z_M = 150 * 1,05 = 157,5 \text{ (руб.)}.$$

Далее рассмотрим расчет затрат на электроэнергию. Учитывая, что рабочий день равняется восьми часам, получаем формулу (3):

$$Z_{ЭЛ.ПЕР} = P_{ЭВМ} * T_{ПЕР} * 8 * C_{ЭЛ}, \quad (3)$$

где  $T_{ПЕР}$  – количество дней использования приобретенного компьютера.

В документации, приложенной к приобретенным компьютерам, сказано, что  $P_{ЭВМ} = 0,2$  кВт, а стоимость одного кВт/ч электроэнергии  $C_{ЭЛ} = 2,5$  руб.

Исходя из этого получаем следующее:

$$Z_{ЭЛ.ПЕР} = 0,2 * 90 * 8 * 2,5 = 360 \text{ (руб.)}.$$

Учитывая, что с разработчиками проекта не был заключен договор, а разработка происходила на доверительной основе, получаем следующие данные по расходам заработной платы (табл. 42)

Таблица. 42. Заработная плата разработчикам

Вид заработной платы	Денежная оценка, руб.
Основная заработная плата двум разработчикам	30000
Итого	30000

Далее получаем общие данные по переменным издержкам (табл. 43).

Таблица 43 – Итоговые переменные издержки проекта

Вид переменных издержек	Величина, руб.
Затраты на приобретение материалов	157,5
Затраты на электроэнергию	360
Затраты на оплату труда	30000
Итого	30517,5

Учитывая, что переменные и постоянные издержки рассчитаны, можно получить полные издержки (табл. 44).

Таблица 44 – Полные издержки проекта

Вид издержек	Величина, руб.
Постоянные	3137,67
Переменные	30517,5
Итого	33655,17

Для расчета полной себестоимости разработки проекта нужно воспользоваться формулой (4):

$$Z_{об} = Z_{пос} + Z_{пер}, \quad (4)$$

где  $Z_{об}$  – себестоимость проекта

$Z_{пос}$  – постоянные издержки;

$Z_{пер}$  – переменные издержки.

После подстановки значений получаем следующую себестоимость:

$$Z_{об} = 3137,67 + 30517,5 = 33655,17 \text{ (руб.)}$$

Целесообразность разработки проекта можно определить, рассчитав окупаемость проекта.

В табл. 45 показаны итоговые расчеты.

Таблица 45 – Окупаемость проекта

Месяц	Клиентов	Доход	Расходы-доход
1	5	2500	-31155,17
2	6	3000	-28155,17
3	8	4000	-24155,17
4	10	5000	-19155,17
5	13	6500	-12655,17
6	13	7500	-6155,17
7	15	8500	1344,83

Из полученных данных видно, что окупаемость проекта равна семи месяцам, а при сохранении тенденции роста клиентов проект заработает миллион рублей через 3,75 года.

Исходя из полученных данных можно сказать, реализация данного проекта целесообразна, учитывая низкую стоимость затрат и возможности роста в связи с выбранной стратегией развития проекта.



## Заключение

В заключении проведенной работы можно сделать вывод, что системы для автоматизации бизнес-процессов каких-либо компаний всегда будут иметь спрос и тенденцию развития, как и проект «МойСервис». Учитывая постоянную конкуренцию и стремление удержаться на определенном уровне, проект будет постоянно дорабатываться, внося новый функционал или упраздняя старый.

На данный момент эра облачных технологий заметно развивается, что несомненно дает большой плюс в проект «МойСервис», и это позволяет иметь значительное преимущество над продуктом от компании «1С», которым пользуется множество компаний.

В процессе проектирования и разработки проекта были выполнены восемь основных шагов.

1. Формирование основной идеи и особенностей конкретного программного продукта.
2. Анализ существующих аналогичных программных продуктов. Выделение их преимуществ и недостатков.
3. Построение автоматизируемых бизнес-процессов для последующего анализа, в ходе которого выяснится, где именно можно применить программное решение, учитывая опыт аналогичных программных продуктов.
4. Разработка User Story Mapping и графических макетов в качестве аналога технического задания.
5. Выделение сущностей и модирование базы данных.
6. Выбор технологий для реализации.
7. Тестирование в кругу разработчиков.
8. Тестирование в виде внедрения в сеть сервисных центров «iRazbil».

Реализация минимального жизнеспособного продукта может считаться выполненной, так как:

1) данное веб-приложение может быть использовано как сервисными центрами уровня микро-бизнеса, так и малого;

2) веб-приложение позволяет вести учет заказов, касс, кассовых движений, клиентов, сотрудников и действий с заказами;

3) веб-приложение пригодно и рекомендуется к интеграции с существующим бизнесом, и это не требует серьезных временных затрат на обучение персонала;

4) веб-приложение позволяет создавать, изменять, удалять и просматривать печатные формы документов, а так же использовать их для печати при работе с заказами;

5) веб-приложение позволяет просматривать детальную историю заказов в хронологическом порядке;

6) веб-приложение формирует статистические отчеты по отчетному периоду, отчеты по заказам, отчеты по зарплате, финансовые отчеты и отчеты по рекламным каналам.

Внедрение программного решения в компанию осуществляется в несколько небольших этапов, относительно программных решений, которые требуют установки своего программного обеспечения на компьютеры компании. Основными этапами являются:

1) регистрация компании в веб-приложении, которая осуществляется руководителем компании или иным уполномоченным лицом;

2) регистрация сотрудников компании через модуль «Настройки»;

3) ознакомление сотрудников с интерфейсом веб-приложения.

Так как веб-приложение будет активно развиваться, можно выделить следующие тенденции роста:

1) ускорение работы веб-приложения за счет рефакторинга кода и разбиения архитектуры на микросервисную;

2) создание блога для размещения новостей;

3) наблюдение за рынком для выявления спроса на определенный функционал.

## Библиографический список

1. AngularJS: Miscellaneous: Getting Started // [Электронный ресурс]: <https://docs.angularjs.org/misc/started>.
2. Entity Framework // [Электронный ресурс]: <https://msdn.microsoft.com/ru-ru/data/ef.aspx>.
3. Getting Started // [Электронный ресурс]: <http://getbootstrap.com/getting-started/>.
4. Gojko Adzic. Impact Mapping. – Gojko Adzic 2014. – 140 с.
5. LINQ // [Электронный ресурс]: <https://msdn.microsoft.com>.
6. Албахари Джозеф. С# 6.0. Справочник. Полное описание языка. – М.: ИД «Вильямс», 2017. – 1040 с.
7. Введение в язык С# // [Электронный ресурс]: <https://msdn.microsoft.com>.
8. Грабер Мартин. SQL для простых смертных. – Перевод. ИД «ЛОРИ», 2014. – 383 с.
9. Грубер М. Понимание SQL. – М.: ИД «ЛОРИ», 2006. – 354 с.
10. Зиборов В.В. Visual С# на примерах. – СПб.:БХВ-Петербург, 2013. – 480с.
11. Макконнелл С. Совершенный код. Мастер-класс. – Перевод. ИД «Русская редакция», 2010. – 896 с.
12. Марков А.С., Лисовский К.Ю. Базы данных. Введение в теорию и методологию. – М.: Финансы и статистика, 2006. – 512 с.
13. Нейгел К.. С# 2005 для профессионалов – Диалектика, 2007. – 1790 с.
14. Особенности регулярных выражений в JavaScript // [Электронный ресурс]: <http://JavaScript.ru/tutorial/regexp-specials>.
15. Первое приложение с Entity Framework. Code First // [Электронный ресурс]: <http://metanit.com/sharp/entityframework/1.2.php>.

16. Программирование на языке C#, платформа .NET Framework // [Электронный ресурс]: professorweb.ru
17. Регулярные выражения // [Электронный ресурс]: <http://JavaScript.ru/basic/regular-expression>.
18. Русская документация по jQuery // [Электронный ресурс]: <http://jQuery-docs.ru>.
19. Современный учебник JavaScript // [Электронный ресурс]: <http://learn.JavaScript.ru>
20. Справочник по C# // [Электронный ресурс]: <http://msdn.microsoft.com/ru-ru/library.html>.
21. Структура JavaScript // [Электронный ресурс]: <http://JavaScript.ru/tutorial/foundation/structure>.
22. Структура языка JavaScript // [Электронный ресурс]: <http://laptev-alex.ru/index.php/struktura-yazyka-javascript>.
23. Троелсен Эндрю. Язык программирования C# 5.0 и платформа .NET 4.5, 6-е издание. – М.: Изд-во Вильямс, 2013. – 1312 с.
24. Фаронов, В.В. Программирование на языке C#. – СПб.: Питер, 2007. – 240 с.
25. Флэнаган Дэвид. JavaScript. Подробное руководство – O'Reilly, 2013. – 1080 с.

## Приложение 1

# JS код приложения

### 1. OrdersController.

```
controllers.controller('ordersController', ['$scope', 'accountService', '$uibModal', '$log', '$filter', 'orderService',
'statusService', '$timeout', '$localStorage', '$stateParams', 'employeeService', '$rootScope', '$state', 'Notification',
'deviceTypeService', 'filterService', 'cashflowService', 'documentService', '$window', function ($scope,
accountService, $uibModal, $log, $filter, orderService, statusService, $timeout, $localStorage, $stateParams,
employeeService, $rootScope, $state, Notification, deviceTypeService, filterService, cashflowService, documentService,
$window) {
    var openFilter = function (type) {
        if (type == 0 && $localStorage.activeFilterId != null && $scope.filters)
            for (var i = 0; i < $scope.filters.length; i++) {
                if ($scope.filters[i].id == $localStorage.activeFilterId) {
                    $scope.openFilter($scope.filters[i]);
                    console.log($scope.filters[i]);
                    return;
                }
            }
        if (type == 1 && $localStorage.activePanelId != null && $scope.panels)
            for (var i = 0; i < $scope.panels.length; i++) {
                if ($scope.panels[i].id == $localStorage.activePanelId) {
                    $scope.openFilter($scope.panels[i], true);
                    return;
                }
            }
    }
    $scope.filterVisibility = false;
    $scope.getOrdersCountString = function (number) {
        var titles = ['заказ', 'заказа', 'заказов'];
        cases = [2, 0, 1, 1, 1, 2];
        return titles[(number % 100 > 4 && number % 100 < 20) ? 2 : cases[(number % 10 < 5) ? number % 10 : 5]];
    }
    if ($stateParams.reload) {
        $state.transitionTo($state.current);
    }
    $scope.ordersWidths = $localStorage.ordersWidths;
    $scope.headers = [{
        name: '<i class="fa fa-home" aria-hidden="true"></i>',
        visible: true
    },
    {
        name: '№',
        visible: true
    },
    {
        name: 'Принял',
        visible: true
    },
    {
        name: 'Статус',
        visible: true
    },
    {
        name: 'Устройство',
        visible: true
    },
    {
        name: 'Неисправность',
        visible: true
    },
    {
        name: 'Инженер',
        visible: true
    }
    ]
}
```

```

    },
    {
      name: 'Менеджер',
      visible: true
    },
    {
      name: 'Клиент',
      visible: true
    },
    {
      name: 'Телефон клиента',
      visible: true
    },
    {
      name: 'Цена',
      visible: true
    },
  ],
]
$scope.$on('filials-loaded', function (event, args) {
  if (typeof $localStorage.headers == 'undefined')
    $localStorage.headers = $scope.headers;
  $localStorage.headers[0].visible = args.showIcon;
  $scope.loadVisibilities();
});
$scope.loadVisibilities = function () {
  if (typeof $localStorage.headers == 'undefined') return;
  for (var i = 0; i < $scope.headers.length; i++) {
    if (typeof $localStorage.headers[i].visible != 'undefined')
      $scope.headers[i].visible = $localStorage.headers[i].visible;
  }
}
$scope.saveCols = function () {
  var targets = document.getElementsByClassName('order-heading-setter');
  var widths = []
  for (var i = 0; i < targets.length; i++) {
    widths.push(targets[i].clientWidth);
  }
  $localStorage.ordersWidths = widths;
}

$scope.updateGrid = function () {
  $(function () {
    $("#tab").resizableColumns();
  });
}
$rootScope.$on('$stateChangeStart',
  function (event, toState, toParams, fromState, fromParams) {
    $("#tab").resizableColumns('syncHandleWidths');
    $("#tab").resizableColumns('destroy');
  }
);
$scope.loadVisibilities();

$scope.getVisibility = function (index) {
  if (typeof $scope.headers[index] == 'undefined') return true;
  return $scope.headers[index].visible;
}

$scope.changeVisibility = function (header, $event) {
  var visibleCount = 0;
  header.visible = !header.visible;
  for (var i = 0; i < $scope.headers.length; i++) {
    if ($scope.headers[i].visible)
      visibleCount++;
  }
  if (visibleCount == 0)

```

```

        header.visible = !header.visible;
    $timeout(function () {
        $('#tab').resizableColumns('refreshHeaders');
    });
    $timeout(function () {
        $('#tab').resizableColumns('syncHandleWidths');
    });
    $localStorage.headers = $scope.headers;
    $event.stopPropagation()
}

$scope.dropdownAppendTo = angular.element(document.querySelector('#appender'));

var settings = {
    sortField: 'Id',
    descending: true,
    isRemoved: false,
    takeCount: 30,
    skipCount: 0
}
var refreshOrders = function () {
    if ($state.current.name === 'orders')
        orderService.getOrders(settings).success(function (rsp) {
            if ($localStorage.currentWorkshop > 0) $scope.headers[0].visible = false;
            else
                $scope.headers[0].visible = true;
            $scope.orders = rsp.items;
            $scope.totalItems = rsp.count;
            $scope.updateGrid();
        }).error(function (rsp) {
            Notification.error('Ошибка при загрузке списка заказов')
        })
}

$scope.createOrder = function (size) {
    if ($localStorage.currentWorkshop === 0) {
        Notification.error('Для создания заказа слева сверху выберите отделение, к которому будет привязан новый заказ.');
```

return;

```

    }
    var modalInstance = $uibModal.open({
        animation: $scope.animationsEnabled,
        templateUrl: 'app/views/modals/createorder.html',
        controller: 'createOrderModalInstanceController',
        size: size
    });

    modalInstance.result.then(function (newOrder) {
        refreshOrders()
    }, function () {});
};

$scope.openFilterBlock = function () {
    $scope.filterVisibility = !$scope.filterVisibility;
}

$scope.calculatePrice = function (o) {
    var price = 0;
    if (typeof o.workList !== 'undefined') {
        for (var i = 0; i < o.workList.length; i++) {
            price += o.workList[i].price;
        }
    }
    if (typeof o.detailsList !== 'undefined') {
        for (var i = 0; i < o.detailsList.length; i++) {

```

```

        price += o.detailsList[i].price;
    }
}
return price;
} // to del

// STATUS

$scope.changeOrderStatus = function (order, status, oldStatus, $event, payNeeded) {
    $scope.checkStatus(order, status, oldStatus, payNeeded);
    $event.stopPropagation();
}

var showDeviceNotification = function () {
    var modalInstance = $uibModal.open({
        animation: $scope.animationsEnabled,
        templateUrl: 'app/views/modals/notification.html',
        controller: 'notificationModalInstanceController',
        size: 'md',
        windowClass: 'zindex',
        resolve: {
            data: {
                text: 'Не забудьте забрать у клиента подменное устройство!'
            }
        }
    })
    modalInstance.result.then(function () {

    }, function () {});
}

var updOrderStatus = function (order, status, data) {
    orderService.changeOrderStatus(order.id, status.id, data).success(function (rsp) {
        order.status = status;
        updGrid();
        //print
        if (data && typeof data.print != 'undefined') {
            var pages = null;
            documentService.createDocument(data.print.id, data.print.docs, 1).success(function (rsp) {
                pages = rsp;
                if (pages == null) {
                    Notification.error('Нет документов для печати')
                    return;
                }
                // get html data
                angular.element(document.querySelector('#printElement')).html(pages);
                $window.print();
            }).error(function (rsp) {
                Notification.error('Ошибка при создании документов. Пожалуйста, перезагрузите страницу')
            })
        }
    }).error(function (rsp) {
        Notification.error('При обновлении статуса произошла ошибка. Повторите попытку позже')
    })
}

$scope.checkStatus = function (order, newValue, old, pay) {
    var isOk = false;
    var p = order.price - order.paid;
    if (p == 0) {
        updOrderStatus(order, newValue)
        //return replacement device notification
        if (order.replacementDevice)
            showDeviceNotification()
        return;
    }
    if (newValue.name == 'Закрыт' && order.price > 0 && pay) {

```



```

//create new modal that creates flow
var modalInstance = $uibModal.open({
  animation: $scope.animationsEnabled,
  templateUrl: 'app/views/modals/orderpay.html',
  controller: 'orderPayModalInstanceController',
  size: 'md',
  windowClass: 'zindex',
  resolve: {
    order: function () {
      return {
        customId: order.customId,
        price: p,
        printCancel: true
      }
    }
  }
})
modalInstance.result.then(function (flow) {
  updOrderStatus(order, newValue, flow)
  //return replacement device notification
  if (order.replacementDevice)
    showDeviceNotification()
}, function () {});
} else {
  if (newValue.name === 'Закрыт' && order.replacementDevice)
    showDeviceNotification()
  updOrderStatus(order, newValue)
}
}

// CREATE ORDER FROM MENU
$scope.$on('refresh-orders', function (event, data) {
  updGrid();
});

// UPDATE WHEN FILIAL CHANGES
$scope.$on('update-page', function (event, data) {
  if ($state.current.name === 'orders')
    updGrid();
});

// UPDATE ORDER IF STATUS CHANGED BY REWORK
$scope.$on('change-order-status', function (event, obj) {
  var data = obj.data;
  for (var i = 0; i < $scope.orders.length; i++) {
    if ($scope.orders[i].id === data.id) {
      $scope.orders[i].status = data.status;
      return;
    }
  }
});

//SEARCH

$scope.searchOptions = {
  debounce: {
    default: 500,
    blur: 250
  }
};

$scope.searchBy = function (searchType) {
  if (typeof $scope.search === 'undefined') return;
  settings.searchType = searchType;
  settings.input = document.getElementById('orders-search-input').value;
  updGrid()
}

```

```

}

$scope.clearSearchText = function ($event) {
    var target = document.getElementById('orders-search-input');
    target.value = '';
    $scope.searchBy(-1)
    $scope.searchCrossVisible = false;
}

$scope.checkInput = function ($event) {
    $scope.searchCrossVisible = $event.currentTarget.value.length > 0;
}

// FILTERS BELOW
$scope.filters = [];
var loadFilters = function () {
    filterService.getFilters().success(function (rsp) {
        $scope.filters = rsp.items;
        openFilter(0);
    }).error(function (rsp) {
        Notification.error('Ошибка при получении фильтров. Пожалуйста, повторите снова.')
    })
}
var deselectArray = function (array) {
    angular.forEach(array, function (value, key) {
        value['ticked'] = false;
    });
}
var getIndexOfObj = function (array, id) {
    for (var i = 0; i < array.length; i++) {
        if (array[i].id == id)
            return i;
    }
    return -1;
}
var deactivateFilters = function () {
    angular.forEach($scope.filters, function (f) {
        f.active = false;
    })
    angular.forEach($scope.panels, function (p) {
        p.active = false;
    })
}
var completeFilter = function (local) {
    //getting statuses ids
    if (local) {
        $scope.filter.statuses = []
        for (var i = 0; i < $scope.selectedStatuses.length; i++) {
            $scope.filter.statuses.push($scope.selectedStatuses[i].id)
        }
        if ($scope.filter.statuses.length == 0)
            delete $scope.filter.statuses
    } else {
        if (typeof $scope.filter.statuses != 'undefined')
            for (var i = 0; i < $scope.filter.statuses.length; i++) {
                var index = getIndexOfObj($scope.allStatuses, $scope.filter.statuses[i]);
                if (index != -1) {
                    $scope.allStatuses[index].ticked = true;
                }
            }
    }
}

//get client

//get order Type

```

```

if (local) {
    $scope.filter.orderTypes = []
    for (var i = 0; i < $scope.selectedOrderTypes.length; i++) {
        $scope.filter.orderTypes.push($scope.selectedOrderTypes[i].id)
    }
    if ($scope.filter.orderTypes.length == 0)
        delete $scope.filter.orderTypes
} else {
    if (typeof $scope.filter.orderTypes != 'undefined')
        for (var i = 0; i < $scope.filter.orderTypes.length; i++) {
            var index = getIndexOfObj($scope.allOrderTypes, $scope.filter.orderTypes[i]);
            if (index != -1) {
                $scope.allOrderTypes[index].ticked = true;
            }
        }
}

//get managers
if (local) {
    $scope.filter.managers = []
    for (var i = 0; i < $scope.selectedManagers.length; i++) {
        $scope.filter.managers.push($scope.selectedManagers[i].id)
    }
    if ($scope.filter.managers.length == 0)
        delete $scope.filter.managers
} else {
    if (typeof $scope.filter.managers != 'undefined')
        for (var i = 0; i < $scope.filter.managers.length; i++) {
            var index = getIndexOfObj($scope.allManagers, $scope.filter.managers[i]);
            if (index != -1) {
                $scope.allManagers[index].ticked = true;
            }
        }
}

//get dates
$scope.filter.datePeriod = {
    periodId: $scope.period.id,
    leftDate: $scope.period.dates.firstDate,
    rightDate: $scope.period.dates.lastDate
}

//get engineers
if (local) {
    $scope.filter.engineers = []
    for (var i = 0; i < $scope.selectedEngineers.length; i++) {
        $scope.filter.engineers.push($scope.selectedEngineers[i].id)
    }
    if ($scope.filter.engineers.length == 0)
        delete $scope.filter.engineers
} else {
    if (typeof $scope.filter.engineers != 'undefined')
        for (var i = 0; i < $scope.filter.engineers.length; i++) {
            var index = getIndexOfObj($scope.allEngineers, $scope.filter.engineers[i]);
            if (index != -1) {
                $scope.allEngineers[index].ticked = true;
            }
        }
}
settings.filter = $scope.filter;
}
loadFilters();

$scope.acceptFilter = function (local, panel) {
    $timeout(function () {
        if (panel)
            settings.panelId = $localStorage.activePanelId;
    });
}

```

```

        else
            completeFilter(local)
            refreshOrders();
        }, 50);
    }

$scope.denyFilter = function (skipUpdate) {
    $scope.filter = {}
    $scope.selectedStatuses = []
    $scope.selectedOrderTypes = [];
    $scope.selectedManagers = []
    $scope.selectedEngineers = [];
    deselectArray($scope.allManagers)
    deselectArray($scope.allEngineers)
    deselectArray($scope.allOrderTypes)
    deselectArray($scope.allStatuses)
    $scope.period = null;
    settings.filter = null;
    settings.panelId = null;
    if (!skipUpdate)
        $scope.acceptFilter(true);
}

$scope.saveFilter = function () {
    completeFilter(true)
    var modalInstance = $uibModal.open({
        animation: $scope.animationsEnabled,
        templateUrl: 'app/views/modals/orders/filter.html',
        controller: 'filterModalInstanceController',
        size: 'md',
        resolve: {
            filter: function () {
                return $scope.filter;
            }
        }
    })
    modalInstance.result.then(function (f) {
        filterService.createFilter(f).success(function (rsp) {
            Notification.success('Фильтр был успешно создан')
            loadFilters()
        }).error(function (rsp) {
            Notification.error('Ошибка при создании фильтра. Пожалуйста, повторите снова.')
        })
    });
}

$scope.openFilter = function (f, panel) {
    $scope.denyFilter(true);
    if (f.active) {
        deactivateFilters()
        refreshOrders()
        $localStorage.activeFilterId = null;
        $localStorage.activePanelId = null;
    } else {
        deactivateFilters();
        f.active = true;
        if (panel) {
            $localStorage.activePanelId = f.id;
        } else {
            $scope.period = f.datePeriod;
            $localStorage.activeFilterId = f.id;
            $scope.filter = f;
        }
        $scope.acceptFilter(false, panel);
    }
}

$scope.anyFilterActive = function () {

```

```

    for (var i = 0; i < $scope.filters.length; i++) {
        if ($scope.filters[i].active == true)
            return true;
    }
    return false;
}
$scope.removeFilter = function () {
    var filter = null;
    for (var i = 0; i < $scope.filters.length; i++) {
        if ($scope.filters[i].active == true) {
            filter = $scope.filters[i];
            break;
        }
    }
    if (filter == null) return;
    // remove filter via service
    filterService.deleteFilter(filter.id).success(function (rsp) {
        Notification.success('Фильтр был успешно удален')
        loadFilters()
    }).error(function (rsp) {
        Notification.error('Ошибка при удалении фильтра. Пожалуйста, повторите снова.')
    })
}

var statusColors = [];
//statuses multiselect
$scope.allStatuses = []
var extractStatuses = function (groups) {
    for (var i = 0; i < groups.length; i++) {
        for (var j = 0; j < groups[i].statuses.length; j++) {
            groups[i].statuses[j].color = groups[i].color;
            $scope.allStatuses.push(groups[i].statuses[j])
        }
    }
}

statusService.getStatuses(true).success(function (rsp) {
    $scope.statusesAll = rsp.items;
    extractStatuses(rsp.items)

}).error(function (rsp) {
    Notification.error('Ошибка при загрузке статусов')
})

$scope.filter = { };
$scope.selectedStatuses = []
$scope.localLang = {
    selectAll: "Включить все",
    selectNone: "Выключить все",
    reset: "Отмена",
    search: "Введите название статуса...",
    nothingSelected: "Любой статус" //default-label is deprecated and replaced with this.
}

//ordertype multiselect
$scope.localOrderType = {
    selectAll: "Включить все",
    selectNone: "Выключить все",
    reset: "Отмена",
    search: "Введите название типа заказа...",
    nothingSelected: "Любой тип заказа" //default-label is deprecated and replaced with this.
}

$scope.allOrderTypes = [{
    id: 0,
    name: 'Платный'

```

```

    },
    {
        id: 1,
        name: 'По гарантии'
    }
]
$scope.selectedOrderTypes = [];

//Managers multiselect
$scope.localManagers = {
    selectAll: "Включить все",
    selectNone: "Выключить все",
    reset: "Отмена",
    search: "Введите имя менеджера...",
    nothingSelected: "Любой менеджер" //default-label is deprecated and replaced with this.
}

employeeService.getManagers().success(function (rsp) {
    $scope.allManagers = rsp.items;
    angular.forEach($scope.allManagers, function (obj) {
        obj.name = obj.firstName + ' ' + obj.lastName;
    });
})
$scope.selectedManagers = [];

//Engineers multiselect
$scope.localEngineers = {
    selectAll: "Включить все",
    selectNone: "Выключить все",
    reset: "Отмена",
    search: "Введите имя инженера...",
    nothingSelected: "Любой инженер" //default-label is deprecated and replaced with this.
}

employeeService.getEngineers().success(function (rsp) {
    $scope.allEngineers = rsp.items;
    angular.forEach($scope.allEngineers, function (obj) {
        obj.name = obj.firstName + ' ' + obj.lastName;
    });
})
$scope.selectedEngineers = [];

deviceTypeService.getDeviceTypes().success(function (rsp) {
    $scope.allDeviceTypes = rsp.items;
})

//DATE PICKER
$scope.period = null;

// PAGINATION
$scope.currentPage = 1;

$scope.pageChanged = function (p) {
    $scope.currentPage = p
    updGrid()
};

$scope.maxSize = 30;
// top-right blocks
var refreshPanels = function () {
    orderService.getSummary().success(function (rsp) {
        $scope.panels = [];
        $scope.panels = [{
            id: 0,
            name: 'Общее количество устройств, относящихся к группам статусов "В работе" и "Новые"',
            color: '#fcd0f',

```

```

        value: rsp.result.working.count,
        active: false,
        icon: '<i class="fa fa-wrench" aria-hidden="true"></i>'
    },
    {
        id: 2,
        name: 'Устройства со статусом "Ждёт запчасть"',
        color: '#fcd0f',
        value: rsp.result.waiting.count,
        active: false,
        icon: '<i class="fa fa-truck" aria-hidden="true"></i>'
    },
    {
        id: 1,
        name: 'Устройства со статусом "Ждёт оплаты"',
        color: '#fcd0f',
        value: rsp.result.done.count,
        active: false,
        icon: '<i class="fa fa-money" aria-hidden="true"></i>'
    }
    ]
    openFilter(1);
}).error(function (rsp) {
    Notification.error('Ошибка при загрузке блоков-фильтров')
})
}
refreshPanels();
var updGrid = function () {
    var begin = ($scope.currentPage - 1) * $scope.maxSize;
    var end = $scope.maxSize;
    settings.skipCount = begin;
    settings.takeCount = end;
    refreshPanels();
    refreshOrders()
}

if ($localStorage.activeFilterId == null && $localStorage.activePanelId == null)
    updGrid()

// PERMISSIONS
$scope.canCreateOrder = accountService.canEditOrders();

// HEADER NAME
$scope.getHeaderName = function () {
    if ($localStorage.currentWorkshop == 0)
        return 'компании'
    else
        return '' + $localStorage.currentWorkshopName;
}

// RAZIO MULTISELECT
$scope.textDeclension = ['статус', 'статуса', 'статусов']

$scope.orderTextDeclension = ['тип заказа', 'типа заказа', 'типов заказа']

$scope.managerTextDeclension = ['менеджер', 'менеджера', 'менеджеров']

$scope.engineerTextDeclension = ['инженер', 'инженера', 'инженеров']

})

```

## 2. CashboxesController.

```

controllers.controller('cashboxesController', ['$scope', '$uibModal', 'cashboxService', 'cashflowService',
'$localStorage', 'accountService', '$rootScope', '$state', 'Notification', 'orderByFilter', '$http', 'host', function
($scope, $uibModal, cashboxService, cashflowService, $localStorage, accountService, $rootScope, $state, Notification,
orderByFilter, $http, host) {
    //DATE PICKER
    $scope.period = null;
    $scope.totalPlus = 0;
    $scope.totalMinus = 0;

    if ($localStorage.selectedCashbox == null)
        $localStorage.selectedCashbox = 0;
    var updateCashboxes = function () {
        $scope.totalPlus = 0;
        $scope.totalMinus = 0;
        cashboxService.getCashboxes().success(function (rsp) {
            $scope.cashboxes = null;
            $scope.cashboxes = orderByFilter(rsp.items, '+id');
            if (rsp.count > 0) {
                if ($localStorage.selectedCashbox > rsp.count)
                    $localStorage.selectedCashbox = 0;
                $scope.active($scope.cashboxes[$localStorage.selectedCashbox])
                $scope.openCashbox($scope.cashboxes[$localStorage.selectedCashbox],
                $scope.cashboxes.indexOf($scope.cashboxes[$localStorage.selectedCashbox]))
            } else {
                $scope.currentCashbox = null;
                $scope.flows = [];
            }
        }).error(function (rsp) {
            Notification.error('Произошла ошибка при загрузке касс. Повторите попытку снова')
        })
    }
    updateCashboxes()
    var settings = {
        takeCount: 15,
        skipCount: 0,
        sortField: 'Id',
        descending: true
    }
    $scope.openCashbox = function (c, id) {
        $scope.currentCashbox = c;
        if (typeof $scope.currentCashbox == 'undefined') return "
        updGrid();
        $localStorage.selectedCashbox = id;
    }

    $scope.active = function (c) {
        if ($scope.currentCashbox == c)
            return 'cb-box-item-active'
        else
            return "
    }

    $scope.getPlus = function (sum) {
        if (sum >= 0)
            return sum;
    }

    $scope.getMin = function (sum) {
        if (sum < 0)
            return sum;
    }

    $scope.isActive = function (c) {
        if ($scope.currentCashbox == c)
            return true;
        else

```



```

        return false;
    }

$scope.createCashbox = function () {

    var modalInstance = $uibModal.open({
        animation: $scope.animationsEnabled,
        templateUrl: 'app/views/modals/createcashbox.html',
        controller: 'createCashboxModalInstanceController',
        size: 'md'
    });

    modalInstance.result.then(function (newCashbox) {
        newCashbox.balance = 0;
        newCashbox.isRemoved = false;
        cashboxService.createCashbox(newCashbox).success(function (rsp) {
            Notification.success('Касса успешно создана!')
            updateCashboxes()
        }).error(function (rsp) {
            Notification.error('Произошла ошибка при создании кассы')
        })
    }, function () {});
}

$scope.sendCash = function (type) {
    var modalInstance = $uibModal.open({
        animation: $scope.animationsEnabled,
        templateUrl: 'app/views/modals/sendcash.html',
        controller: 'sendCashModalInstanceController',
        size: 'md',
        resolve: {
            type: function () {
                return {
                    id: type,
                    sum: 0,
                    exceptCash: $scope.currentCashbox.id
                };
            }
        }
    })
    modalInstance.result.then(function (flow) {
        if (type === 'trans') {
            //Запросить кассу из flow и засунуть ей второй флоу
            var cashId = flow.cb.id;
            flow.cashboxid = $scope.currentCashbox.id;
            var comment = null;
            if (typeof flow.comment !== 'undefined')
                comment = flow.comment;
            flow.comment = 'Перемещение денежных средств из кассы ' + $scope.currentCashbox.name + ' в кассу ' +
flow.cb.name;
            if (comment !== null)
                flow.comment += ' | ' + comment;
            cashflowService.addFlow(cashId, flow).success(function (rsp) {
                flow.sum = -flow.sum
                cashflowService.addFlow(flow.cashboxid, flow).success(function (rsp) {
                    Notification.success('Денежные средства были успешно перемещены')
                    updateCashboxes()
                }).error(function (rsp) {
                    Notification.error('Произошла ошибка при создании 2 ден. перемещения')
                })
            }).error(function (rsp) {
                Notification.error('Произошла ошибка при создании первого ден. перемещения')
            })
        }
    })
}

```

```

    } else {

        if (type == 'out')
            flow.sum = -flow.sum
        flow.cashboxid = $scope.currentCashbox.id;
        cashflowService.addFlow(flow.cashboxid, flow).success(function (rsp) {
            Notification.success('Кассовая операция произошла успешно!')
            updateCashboxes()
        }).error(function (rsp) {
            Notification.error('Произошла ошибка при создании денежного перемещения')
        })
    }
});
}

$scope.loadFlows = function () {
    settings.leftDate = $scope.period.dates.firstDate;
    settings.rightDate = $scope.period.dates.lastDate;
    cashflowService.getFlows($scope.currentCashbox.id, settings).success(function (rsp) {
        $scope.flows = rsp.items;
        $scope.totalItems = rsp.count;
        $scope.totalPlus = rsp.totalSumIn;
        $scope.totalMinus = rsp.totalSumOut;
    }).error(function (rsp) {
        Notification.error('Ошибка при загрузке движений денежных средств по кассе.')
    })
}

$scope.editCashbox = function (c) {

    var modalInstance = $uibModal.open({
        animation: $scope.animationsEnabled,
        templateUrl: 'app/views/modals/editcashbox.html',
        controller: 'editCashboxModalInstanceController',
        size: 'md',
        resolve: {
            cashbox: function () {
                return c;
            }
        }
    })
    modalInstance.result.then(function (newCashbox) {
        if (newCashbox == 'del') {
            //delete cashbox
            cashboxService.deleteCashbox(c.id).success(function (rsp) {
                $localStorage.selectedCashbox = 0;
                Notification.success('Касса "' + c.name + '" была успешно удалена')
                updateCashboxes()
            }).error(function (rsp) {
                Notification.error('При удалении корзины произошла ошибка. Повторите попытку снова')
            })
        } else {
            //update cashbox
            cashboxService.updateCashbox(newCashbox.id, newCashbox).success(function (rsp) {
                Notification.success('Касса "' + c.name + '" была успешно обновлена')
                updateCashboxes()
            }).error(function (rsp) {
                Notification.error('При обновлении кассы произошла ошибка. Повторите попытку снова')
            })
        }
    });
};

$scope.getComment = function (flow) {
    if (!flow.link) return flow.comment;

```

```

var text = flow.comment.substring(flow.link.indexOf + flow.link.length, flow.link.indexOf);
if (flow.link.type == 0) {
    return flow.comment.replace(text, '<a ng-click="openOrder(fl.link.entityId)" class="order-tr-item">' + text +
'</a>')
}
if (flow.link.type == 1) {
    return flow.comment.replace(text, '<a ng-click="openSale(fl.link.entityId)" class="order-tr-item">' + text + '</a>')
}
}

$scope.removeFlow = function (flow) {
    var modalInstance = $uibModal.open({
        animation: $scope.animationsEnabled,
        templateUrl: 'app/views/modals/warning.html',
        controller: 'warningModalInstanceController',
        size: 'md',
        windowClass: 'zindex',
        resolve: {
            data: {
                text: 'Вы действительно хотите удалить кассовое движение?'
            }
        }
    })
    modalInstance.result.then(function () {
        cashflowService.deleteFlow(flow.id).success(function (rsp) {
            Notification.success('Кассовое движение было отменено')
            updateCashboxes();
        }).error(function (rsp) {
            Notification.error('При отмене кассового движения возникла ошибка')
        })
    }, function () {});
}

// UPDATE WHEN FILIAL CHANGES
$scope.$on('update-page', function (event, data) {
    if ($state.current.name == 'cashboxes.list')
        updateCashboxes()
});

// PERMISSIONS
$scope.canSeeCashboxes = accountService.canSeeCashboxes()

$scope.canEditCashboxes = accountService.canEditCashboxes()

// HEADER NAME
$scope.$parent.getHeaderName = function () {
    if ($localStorage.currentWorkshop == 0)
        return 'компании'
    else
        return '' + $localStorage.currentWorkshopName;
}

//DATE PICKER
var getPeriod = function (id) {
    var lastDate = new Date();
    if (id == 0) {
        var lastDate = new Date();
        var firstDate = new Date();
        firstDate.setFullYear(2016, 1, 1);
        lastDate.setHours(23, 59, 59, 0);
        return {
            firstDate: firstDate,
            lastDate: lastDate
        }
    }
}

```

```

    }
    if (id == 1) {
        var firstDate = new Date();
        firstDate.setHours(0, 0, 0, 0);
        lastDate.setDate(firstDate.getDate())
        lastDate.setHours(23, 59, 59, 0);
        return {
            firstDate: firstDate,
            lastDate: lastDate
        }
    }
    if (id == 2) {
        var firstDate = new Date();
        firstDate.setDate(lastDate.getDate() - 1);
        firstDate.setHours(0, 0, 0, 0);
        lastDate.setDate(firstDate.getDate())
        lastDate.setHours(23, 59, 59, 0);
        return {
            firstDate: firstDate,
            lastDate: lastDate
        }
    }
    if (id == 3) {
        var lastDate = new Date();
        var d = new Date(lastDate);
        var day = d.getDay(),
            diff = d.getDate() - day + (day == 0 ? -6 : 1); // adjust when day is sunday
        var firstDate = new Date(d.setDate(diff));
        firstDate.setHours(0, 0, 0, 0);
        lastDate.setHours(23, 59, 59, 0);
        return {
            firstDate: firstDate,
            lastDate: lastDate
        }
    }
    if (id == 4) {
        var date = new Date();
        var firstDate = new Date(date.getFullYear(), date.getMonth(), 1);
        var lastDate = new Date();
        firstDate.setHours(0, 0, 0, 0);
        lastDate.setHours(23, 59, 59, 0);
        return {
            firstDate: firstDate,
            lastDate: lastDate
        }
    }
}
$scope.periods = [{
    id: 0,
    value: 'За все время',
    dates: getPeriod(0),
    showValue: 'За все время'
},
{
    id: 1,
    value: 'За сегодня',
    dates: getPeriod(1)
},
{
    id: 2,
    value: 'За вчера',
    dates: getPeriod(2)
},
{
    id: 3,
    value: 'С начала недели',

```

```

        dates: getPeriod(3)
    },
    {
        id: 4,
        value: 'С начала месяца',
        dates: getPeriod(4)
    },
    {
        id: 5,
        value: 'Выбрать даты',
        dates: {}
    },
]
var monthes = [
    'январ',
    'фев',
    'мар',
    'апр',
    'мая',
    'июн',
    'июл',
    'авг',
    'сен',
    'окт',
    'ноя',
    'дек'
]
var refreshShowValues = function () {
    for (var i = 0; i < $scope.periods.length; i++) {
        if (typeof $scope.periods[i].dates != 'undefined' && typeof $scope.periods[i].dates.firstDate != 'undefined' && i >
0) {
            var first = $scope.periods[i].dates.firstDate;
            var second = $scope.periods[i].dates.lastDate;
            first.setHours(1, 0, 0, 0);
            $scope.periods[i].showValue = first.toLocaleDateString() + ' - ' + second.toLocaleDateString();
            var f = first.getDate() + ' ' + monthes[first.getMonth()] + ' ' + first.getFullYear();
            var s = second.getDate() + ' ' + monthes[second.getMonth()] + ' ' + second.getFullYear();
            $scope.periods[i].showValue = f + ' - ' + s;
        }
    }
}
refreshShowValues()
$scope.period = $scope.periods[0];
$scope.datePickerNeeded = false;
$scope.selectDateType = function (id) {
    $scope.period = $scope.periods[id];
    if (id == 5) {
        $scope.datePickerNeeded = true;
    }
}
var initializeDates = function () {
    $scope.periods[5].dates.firstDate = new Date();
    $scope.periods[5].dates.firstDate.setHours(1, 0, 0);
    $scope.periods[5].dates.lastDate = new Date();
    $scope.periods[5].dates.lastDate.setHours(23, 59, 59);
}

initializeDates()
$scope.today = function () {
    $scope.filterStartDate = new Date();
    $scope.filterLastDate = new Date();
    $scope.filterLastDate.setHours(23, 59, 59);
};

```

```

$scope.clear = function () {
    $scope.filterStartDate = null;
    $scope.filterLastDate = null;
};

$scope.inlineOptions = {
    customClass: getDayClass,
    minDate: new Date(),
    showWeeks: true
};

$scope.dateOptions = {
    dateDisabled: disabled,
    formatYear: 'yy',
    maxDate: new Date(),
    minDate: new Date(2000, 1, 1),
    startingDay: 1
};

$scope.toggleMin = function () {
    $scope.inlineOptions.minDate = $scope.inlineOptions.minDate ? null : new Date();
    $scope.dateOptions.minDate = $scope.inlineOptions.minDate;
};

$scope.toggleMin();

$scope.open1 = function () {
    $scope.popup1.opened = true;
};

$scope.open2 = function () {
    $scope.popup2.opened = true;
};

$scope.setDate = function (year, month, day) {
    $scope.filterStartDate = new Date(year, month, day);
};

$scope.altInputFormats = ['M!/d!/yyyy'];

$scope.popup1 = {
    opened: false
};

$scope.popup2 = {
    opened: false
};

function disabled(data) {
    var date = data.date,
        mode = data.mode;
    return mode === 'day' && (date.getDay() === 0 || date.getDay() === 6);
}

function getDayClass(data) {
    var date = data.date,
        mode = data.mode;
    if (mode === 'day') {
        var dayToCheck = new Date(date).setHours(0, 0, 0, 0);

        for (var i = 0; i < $scope.events.length; i++) {
            var currentDay = new Date($scope.events[i].date).setHours(0, 0, 0, 0);

            if (dayToCheck === currentDay) {
                return $scope.events[i].status;
            }
        }
    }
}

```

```

    }
  }

  return "";
}

// PAGINATION
$scope.currentPage = 1;

$scope.pageChanged = function (p) {
  $scope.currentPage = p
  updGrid()
};

$scope.maxSize = 30;
var updGrid = function () {
  var begin = ($scope.currentPage - 1) * $scope.maxSize;
  var end = $scope.maxSize;
  settings.skipCount = begin;
  settings.takeCount = end;
  if ($scope.selectedFilters.length > 0)
    settings.tags = $scope.selectedFilters;
  $scope.loadFlows()
}

//TYPEAHEAD SETTINGS

$scope.getDetails = function (val) {
  if (val.length == 0) return;
  $scope.detsArray =
    $http.get(host + 'api/tag', {
      params: {
        input: val
      }
    }).then(function (rsp) {
      $scope.typeaheadArray = rsp.data.items;
      return rsp.data.items.map(function (item) {
        return item;
      });
    });
  return $scope.detsArray;
}

$scope.ngModelOptionsSelected = function (value) {
  if (arguments.length) {
    _selected = value;
  } else {
    return _selected;
  }
};

$scope.modelOptions = {
  debounce: {
    default: 500,
    blur: 250
  },
  getterSetter: true
};

$scope.selectedFilters = [];
$scope.cb = {
  filterText: ""
};
$scope.onSelect = function (a, b, c, d) {
  $scope.taSelectedItem = a;
  if ($scope.selectedFilters.indexOf(a.value) != -1) {

```

```

        $scope.cb.filterText = "";
        return;
    }
    $scope.selectedFilters.push(a.value);
    $scope.cb.filterText = "";
    updGrid();
}

$scope.removeFilter = function (f) {
    var i = $scope.selectedFilters.indexOf(f);
    $scope.selectedFilters.splice(i, 1);
    updGrid();
}

// activate tag
$scope.activateTag = function (tag) {
    if ($scope.selectedFilters.indexOf(tag.value) != -1) return;
    $scope.selectedFilters.push(tag.value);
    updGrid();
}
})

```

### 3. CreateOrderController.

```

controllers.controller('createOrderModalInstanceController', ['$scope', '$uibModalInstance', '$uibModal',
'orderService', 'employeeService', 'clientSourceService', 'deviceTypeService', 'documentService', 'cashflowService',
'$window', 'Notification', '$http', 'host', '$localStorage', 'settingsService', 'phoneInputService', 'clientService',
'accountService', function ($scope, $uibModalInstance, $uibModal, orderService, employeeService, clientSourceService,
deviceTypeService, documentService, cashflowService, $window, Notification, $http, host, $localStorage,
settingsService, phoneInputService, clientService, accountService) {
    $scope.addClassToObject = function (id) {
        phoneInputService.addClass(angular.element(id));
    }
    $scope.removeClassFromObject = function (id) {
        phoneInputService.removeClass(angular.element(id));
    }
    $scope.order = {}
    $scope.order.device = {}
    settingsService.getSettings(2).success(function (rsp) {
        if (rsp.order) {
            if (rsp.order.deviceLook)
                $scope.order.device.look = rsp.order.deviceLook;
        }
    })
    $scope.order.advance = 0;
    $scope.order.client = {};
    $scope.id = "";
    var loadLastId = function () {
        orderService.getLastId().success(function (rsp) {
            $scope.id = rsp;
        }).error(function (rsp) {
            Notification.error('Ошибка при загрузке последнего ID')
        })
    }
    loadLastId();
    var createOrder = function (ordModel) {
        // upd client typeahead
        if ($scope.clientSelected) {
            var id = $scope.order.client.id;
            $scope.order.client = { id: id }
        }
        orderService.createOrder(ordModel).success(function (rsp) {
            Notification.success('Заказ был успешно создан')
            $scope.printDocs();
            $uibModalInstance.close($scope.order);
        }).error(function (rsp) {

```



```

        Notification.error('Ошибка при создании заказа. Пожалуйста, попробуйте снова')
    })
}
var makePayment = function (o) {
    if ($scope.order.advance > 0) {
        // open dialog
        var modalInstance = $uibModal.open({
            animation: $scope.animationsEnabled,
            templateUrl: 'app/views/modals/orderpay.html',
            controller: 'orderPayModalInstanceController',
            size: 'md',
            windowClass: 'zindex',
            resolve: {
                order: function () {
                    return { customId: $scope.id, price: $scope.order.advance, isNew: true }
                }
            }
        })
        modalInstance.result.then(function (flow) {
            createOrder({ order: o, pay: flow })
        }, function () {
        });
    }
    else {
        createOrder({ order: o })
    }
}

$scope.managers = []
employeeService.getManagers().success(function (rsp) {
    for (var i = 0; i < rsp.items.length; i++) {
        rsp.items[i].name = rsp.items[i].firstName + ' ' + rsp.items[i].lastName;
    }
    $scope.managers = rsp.items;
    if (rsp.items.length > 0)
        $scope.order.manager = $scope.managers[0]
    else {
        var id = accountService.getUserId();
        for (var i = 0; i < $scope.managers.length; i++) {
            if ($scope.managers[i].id == id) {
                $scope.order.manager = $scope.managers[i]
                break;
            }
        }
    }
}).error(function (rsp) {
    Notification.error('Ошибка при загрузке списка менеджеров. Пожалуйста, перезагрузите страницу')
})

$scope.engineers = []
employeeService.getEngineers().success(function (rsp) {
    for (var i = 0; i < rsp.items.length; i++) {
        rsp.items[i].name = rsp.items[i].firstName + ' ' + rsp.items[i].lastName;
    }
    $scope.engineers = rsp.items;
}).error(function (rsp) {
    Notification.error('Ошибка при загрузке списка инженеров. Пожалуйста, перезагрузите страницу')
})

$scope.ok = function () {
    $scope.order.creationTime = new Date();
    var o = $scope.order;
    if (o.engineer.id == -1)
        o.engineer = null;
    makePayment(o);
};

```

```

$scope.cancel = function () {
    $uibModalInstance.dismiss('cancel');
};

// device types
var refreshDeviceTypes = function (upd) {
    $scope.deviceTypes = []
    deviceTypeService.getDeviceTypes().success(function (rsp) {
        $scope.deviceTypes = rsp.items;
        if (upd)
            $scope.order.device.deviceTypeId = $scope.deviceTypes[$scope.deviceTypes.length-1].id;
    }).error(function (rsp) {
        Notification.error('Ошибка при загрузке типов устройства. Пожалуйста, перезагрузите страницу')
    })
}
refreshDeviceTypes();

// promo sources
var refreshSources = function (upd) {
    $scope.clientSources = []
    clientSourceService.getSources().success(function (rsp) {
        $scope.clientSources = rsp.items;
        if (upd)
            $scope.order.client.sourceId = $scope.clientSources[$scope.clientSources.length - 1].id;
    }).error(function (rsp) {
        Notification.error('Ошибка при загрузке списка рекламных кампаний. Пожалуйста, перезагрузите страницу')
    })
}
refreshSources();

// PRINT
$scope.docs2Print = []
$scope.selectedDocsCount = 0;
documentService.getTemplates(1).success(function (rsp) {
    $scope.docs2Print = rsp.items;
    if ($localStorage.docs2Print) {
        for (var i = 0; i < $scope.docs2Print.length; i++) {
            for (var j = 0; j < $localStorage.docs2Print.length; j++) {
                if ($scope.docs2Print[i].id == $localStorage.docs2Print[j])
                    $scope.docs2Print[i].checked = true;
            }
        }
    }
}).error(function (rsp) {
    Notification.error('Ошибка при загрузке документов. Пожалуйста, перезагрузите страницу')
})
var getDocs2Print = function () {
    var docs = []
    for (var i = 0; i < $scope.docs2Print.length; i++) {
        if ($scope.docs2Print[i].checked)
            docs.push($scope.docs2Print[i].id)
    }
    $scope.selectedDocsCount = docs.length;
    return docs;
}

$scope.printDocs = function () {
    var docs = getDocs2Print();
    if (docs.length == 0) return;
    // Saving print docs to cache
    if (!$localStorage.docs2Print) $localStorage.docs2Print = []
    for (var i = 0; i < $scope.docs2Print.length; i++) {
        var index = $localStorage.docs2Print.indexOf($scope.docs2Print[i].id);
        if (index > -1) {
            $localStorage.docs2Print.splice(index, 1);
        }
    }
}

```

```

    }
  }
  for (var i = 0; i < docs.length; i++) {
    $localStorage.docs2Print.push(docs[i]);
  }
  // saving end
  angular.element('#print-dropdown').removeClass('open');
  // send ids & order id
  var pages = null;
  documentService.createDocument($scope.id, docs, 1).success(function (rsp) {
    pages = rsp;
    if (pages == null) {
      Notification.error('Нет документов для печати')
      return;
    }
    // get html data
    angular.element(document.querySelector('#printElement')).html(pages);
    $window.print();
  }).error(function (rsp) {
    Notification.error('Ошибка при создании документов. Пожалуйста, перезагрузите страницу')
  })
}

$scope.getDocumentString = function (number) {
  getDocs2Print()
  if (number == 0) return 'Нет выбранных документов'
  var titles = ['документ', 'документа', 'документов']
  cases = [2, 0, 1, 1, 1, 2];
  return titles[(number % 100 > 4 && number % 100 < 20) ? 2 : cases[(number % 10 < 5) ? number % 10 : 5]];
}

//TYPEAHEAD SETTINGS

$scope.getClients = function (val, type) {
  $scope.loadedClients =
  $http.get(host + 'api/client/' + type + '/' + val).then(function (rsp) {
    $scope.typeaheadArray = rsp.data;
    return rsp.data.map(function (item) {
      return item;
    });
  });
  return $scope.loadedClients;
}

$scope.ngModelOptionsSelected = function (value) {
  if (arguments.length) {
    _selected = value;
  } else {
    return _selected;
  }
};

$scope.modelOptions = {
  debounce: {
    default: 500,
    blur: 250
  },
  getterSetter: true
};

$scope.onSelect = function (a, b, c, d) {
  $http.get(host + 'api/client/' + a.id).success(function (rsp) {
    $scope.order.client = rsp;
    $scope.clientSelected = true;
    if (rsp.source != null) {

```

```

        $scope.order.client.sourceId = rsp.source.id;
    }
    });
}
$scope.denyClientChoice = function () {
    $scope.order.client = {};
    $scope.clientSelected = false;
}

// edit client
$scope.editClient = function (c) {
    var modalInstance = $uibModal.open({
        animation: $scope.animationsEnabled,
        templateUrl: 'app/views/modals/clients/clientmodal.html',
        controller: 'clientModalInstanceController',
        size: 'md',
        windowClass: 'zindex',
        resolve: {
            client: function () {
                return $scope.order.client;
            }
        }
    });

    modalInstance.result.then(function (cl) {
        clientService.updateClient(cl).success(function (rsp) {
            Notification.success('Клиент был успешно обновлен');
            $scope.order.client = rsp;
        }).error(function (rsp) {
            Notification.error('Произошла ошибка при обновлении клиента')
        })
    }, function () {
        //cancel
    });
}

// create device type
$scope.addDeviceType = function () {
    var modalInstance = $uibModal.open({
        animation: $scope.animationsEnabled,
        templateUrl: 'app/views/modals/settings/devicetype.html',
        controller: 'deviceTypeModalInstanceController',
        size: 'md',
        windowClass: 'zindex',
        resolve: {
            deviceType: function () {
                return null;
            }
        }
    });

    modalInstance.result.then(function (d) {
        deviceTypeService.createDeviceType(d).success(function (rsp) {
            Notification.success('Новый тип устройства был успешно создан')
            refreshDeviceTypes(true);
        }).error(function (rsp) {
            Notification.error('Произошла ошибка при создании типа устройств')
        })
    }, function () {
        //cancel
    });
}

// add promo source
$scope.addSource = function () {
    var modalInstance = $uibModal.open({

```

```

        animation: $scope.animationsEnabled,
        templateUrl: 'app/views/modals/settings/clientsource.html',
        controller: 'clientSourceModalInstanceController',
        size: 'md',
        windowClass: 'zindex',
        resolve: {
            src: function () {
                return null;
            }
        }
    });

    modalInstance.result.then(function (srcModel) {
        clientSourceService.createSource(srcModel).success(function (rsp) {
            Notification.success('Рекламный канал был успешно создан')
            refreshSources(true);
        }).error(function (rsp) {
            Notification.error('Ошибка при создании рекламного канала')
        })
    }, function () {
        //cancel
    });
}
})

```

## C# код приложения

### 1. OrderController

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using Microsoft.AspNetCore.Authorization;
using DeloCRM.Services.Interfaces;
using DeloCRM.Extensions;
using DeloCRM.ApiModels;
using DeloCRM.ApiModels.Settings;
using Newtonsoft.Json;
using DeloCRM.ApiModels.Lists;
using DeloCRM.Results;
using DeloCRM.Attributes;

namespace DeloCRM.Controllers
{
    [Route("api/order"), Authorize]
    public class OrderController : BaseController
    {
        private IOrderRepository _orderRepo;

        public OrderController(IOrderRepository repo)
        {
            _orderRepo = repo;
        }

        #region CRUD
        /// <summary>
        /// Получение списка заказов
        /// </summary>
        [HttpGet]
        public async Task<ActionResult> Get([FromQuery]ListSettings settings)
        {
            _orderRepo.User = this.GetUserInfo();
            var result = await _orderRepo.List(settings);
            return ExtractResult(result);
        }

        /// <summary>
        /// Получение заказа
        /// </summary>
        [HttpGet("{id:int}")]
        public async Task<ActionResult> Get(int id)
        {
            _orderRepo.User = this.GetUserInfo();

            var result = await _orderRepo.Get(id);

            return ExtractResult(result);
        }

        /// <summary>
        /// Создание заказа
        /// </summary>
        [HttpPost]
        public async Task<ActionResult> Post([FromBody]CreatingOrderModel model)
        {
            if (!ModelState.IsValid) return BadRequest(ModelState);
            _orderRepo.User = this.GetUserInfo();

```

```

        var result = await _orderRepo.Create(model);

        return ExtractResult(result);
    }

    /// <summary>
    /// Редактирование заказа
    /// </summary>
    [HttpPut("{id:int}")]
    public async Task<IActionResult> Edit(int id, [FromBody]EditingOrderModel model)
    {
        // нужна валидация
        _orderRepo.User = this.GetUserInfo();
        var result = await _orderRepo.Edit(id, model);

        return ExtractResult(result);
    }

    /// <summary>
    /// Частичное редактирование заказа
    /// </summary>
    /// <param name="orderId">id заказа</param>
    /// <param name="statusId">id статуса</param>
    /// <returns></returns>
    [HttpPut("{id:int}/partial")]
    public async Task<IActionResult> PartialEdit(int id, [FromBody]PartialEditOrderModel model)
    {
        _orderRepo.User = this.GetUserInfo();
        var result = await _orderRepo.PartialEdit(id, model);
        return ExtractResult(result);
    }

    /// <summary>
    /// Пометить заказ как удаленный
    /// </summary>
    [HttpDelete("{id:int}"), HasPermission("Order_Delete")]
    public async Task<IActionResult> Delete(int id)
    {
        _orderRepo.User = this.GetUserInfo();

        var result = await _orderRepo.Remove(id);

        return ExtractResult(result);
    }
}
#endregion

    /// <summary>
    /// Узнать следующий уникальный номер для нового заказа
    /// </summary>
    /// <returns></returns>
    [HttpGet("id")]
    public async Task<IActionResult> GetNextId()
    {
        _orderRepo.User = this.GetUserInfo();

        var result = await _orderRepo.GetNextId();

        return Ok(result);
    }

    /// <summary>
    /// Добавление комментария в историю заказа
    /// </summary>
    /// <param name="orderId"></param>
    /// <param name="model"></param>

```

```

    /// <param name="_eventSvc"></param>
    /// <returns></returns>
    [HttpPost("{orderId:int}/comment")]
    public async Task<IActionResult> AddComment(int orderId, [FromBody]CreatingOrderEventModel model,
[FromServices]IOrderEventRepository _eventSvc)
    {
        _eventSvc.User = this.GetUserInfo();
        var result = await _eventSvc.CreateComment(orderId, model);
        return ExtractResult(result);
    }

    /// <summary>
    /// Восстановление заказа
    /// </summary>
    /// <param name="orderId"></param>
    /// <returns></returns>
    [HttpPatch("{orderId:int}/restore")]
    public async Task<IActionResult> RestoreOrder(int orderId)
    {
        _orderRepo.User = this.GetUserInfo();
        var result = await _orderRepo.RestoreOrder(orderId);
        return ExtractResult(result);
    }

    /// <summary>
    /// Смена филиала у заказа
    /// </summary>
    /// <param name="orderId">id заказа</param>
    /// <param name="workshopId">id новой мастерской</param>
    /// <returns></returns>
    [HttpPatch("{orderId:int}/workshop/{workshopId:int}")]
    public async Task<IActionResult> ChangeWorkshop(int orderId, int workshopId)
    {
        _orderRepo.User = this.GetUserInfo();
        var result = await _orderRepo.ChangeWorkshop(orderId, workshopId);
        return ExtractResult(result);
    }

    /// <summary>
    /// Получение быстрой информации о заказах для 3 блоков
    /// </summary>
    /// <returns></returns>
    [HttpGet("summary")]
    public async Task<IActionResult> GetOrderQuickData()
    {
        _orderRepo.User = this.GetUserInfo();
        var result = await _orderRepo.GetQuickOrderCards();
        return Ok(result);
    }

    /// <summary>
    /// Удаление запроса товара из заказа
    /// </summary>
    /// <param name="orderId">id заказа</param>
    /// <param name="requestId">id запроса товара</param>
    /// <returns></returns>
    [HttpPatch("{orderId:int}/request/{requestId:int}/detach")]
    public async Task<IActionResult> DetachRequest(int orderId, int requestId, [FromServices]IDeliveryService svc)
    {
        var result = await svc.RemoveRequestAsync(requestId, orderId);
        return ExtractResult(result);
    }
}
}

```

## 2. AccountController.



```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using Microsoft.AspNetCore.Authorization;
using DeloCRM.Services.Interfaces;
using Microsoft.AspNetCore.Identity;
using DeloCRM.Models;
using DeloCRM.Database;
using DeloCRM.ApiModels;
using DeloCRM.Extensions;
using Microsoft.EntityFrameworkCore;
using DeloCRM.ApiModels.Lists;
using DeloCRM.ApiModels.Account;
using System.Net;
using Microsoft.Extensions.Configuration;

namespace DeloCRM.Controllers
{
    [Authorize]
    public class AccountController : BaseController
    {
        IUserService _userService;
        private UserManager<AppUser> _userManager;
        private AppDb _db;

        public AccountController(IUserService userService, AppDb db, UserManager<AppUser> um)
        {
            _db = db;
            _userService = userService;
            _userManager = um;
        }

        #region Users
        /// <summary>
        /// Регистрация пользователя
        /// </summary>
        [AllowAnonymous]
        [HttpPost("api/account/user")]
        public async Task<IActionResult> Register([FromBody]RegisterUserModel model)
        {
            if (!ModelState.IsValid) return BadRequest(ModelState);

            _userService.User = this.GetUserInfo();
            _userService.UserManager = _userManager;

            var result = await _userService.RegisterUser(model);

            return ExtractResult(result);
        }

        /// <summary>
        /// Редактирование пользователя
        /// </summary>
        [HttpPut("api/account/user"),]
        public async Task<IActionResult> Edit([FromBody]EditUserModel model)
        {
            if (!ModelState.IsValid) return BadRequest(ModelState);

            _userService.User = this.GetUserInfo();
            _userService.UserManager = _userManager;

            var result = await _userService.EditUser(model);

            return ExtractResult(result);
        }
    }
}

```

```

}

/// <summary>
/// Получение пользователей
/// </summary>
/// <returns></returns>
[HttpGet("api/account/user")]
public async Task<IActionResult> GetUsers()
{
    _userService.User = this.GetUserInfo();

    var result = await _userService.GetUsers();

    return ExtractResult(result);
}

/// <summary>
/// Получение упрощенных пользователей
/// </summary>
/// <returns></returns>
[HttpGet("api/account/user/simple")]
public async Task<IActionResult> GetSimpleUsers(RoleUserType role = default(RoleUserType))
{
    _userService.User = this.GetUserInfo();

    var result = await _userService.GetSimpleUsers(role);

    return ExtractResult(result);
}

/// <summary>
/// Пометить пользователя как удаленный
/// </summary>
[HttpDelete("api/account/user/{id}")]
public async Task<IActionResult> RemoveUser(int id)
{
    var result = await _userService.RemoveUser(id);
    return ExtractResult(result);
}

/// <summary>
/// Переключить аватар
/// </summary>
/// <param name="id">id аватара</param>
/// <returns></returns>
[HttpPost("api/account/user/avatar/{id:int}")]
public async Task<IActionResult> SetAvatar(int id)
{
    int userId = User.GetUserId("usr");
    if (userId == 0) return Unauthorized();
    _userService.User = new UserInfo { Id = userId };
    var result = await _userService.SetUserAvatar(id);
    return ExtractResult(result);
}

/// <summary>
/// Проверка существования логина и\или электронной почты
/// </summary>
/// <param name="login"></param>
/// <param name="email"></param>
/// <returns></returns>
[HttpGet("api/account/user/check"), AllowAnonymous]
public async Task<IActionResult> CheckLoginAndEmail(string login, string email)
{
    var result = await _userService.ExistAccount(login, email);
    return ExtractResult(result);
}

```

```

    }
#endregion
#region Roles
/// <summary>
/// Получение всех ролей
/// </summary>
[HttpGet("api/account/role")]
public async Task<IActionResult> RoleList()
{
    _userService.User = this.GetUserInfo();

    var result = await _userService.GetRoles();

    return ExtractResult(result);
}

/// <summary>
/// Создание роли
/// </summary>
[HttpPost("api/account/role")]
public async Task<IActionResult> CreateRole([FromBody]CreatingRoleModel model)
{
    if (!ModelState.IsValid) return BadRequest(ModelState);
    _userService.User = this.GetUserInfo();

    var result = await _userService.CreateRole(model);

    return ExtractResult(result);
}

/// <summary>
/// Редактирование роли
/// </summary>
[HttpPut("api/account/role/{id}")]
public async Task<IActionResult> EditRole(int id, [FromBody]EditingRoleModel model)
{
    if (!ModelState.IsValid) return BadRequest(ModelState);

    var result = await _userService.EditRole(id, model);

    return ExtractResult(result);
}

/// <summary>
/// Пометить роль как удаленную
/// </summary>
[HttpDelete("api/account/role/{id}")]
public async Task<IActionResult> RemoveRole(int id)
{
    var role = await _db.Roles
        .Include(r => r.Users)
        .SingleOrDefaultAsync(r => r.Id == id);
    if (role == null) return NotFound();
    if (role.Users.Count > 0) return BadRequest("Невозможно удалить роль, т.к. есть пользователи с этой ролью");
    _db.Roles.Remove(role);

    try { await _db.SaveChangesAsync(); }
    catch (Exception ex)
    {
        return BadRequest("Ошибка при удалении роли");
    }

    return Ok();
}
#endregion

```

```

#region Permissions
[HttpGet("api/user/access")]
public async Task<IActionResult> UserAccess()
{
    _userService.User = this.GetUserInfo();
    var result = await _userService.GetAccessInstance();
    return ExtractResult(result);
}

/// <summary>
/// Список всех разрешений
/// </summary>
/// <returns></returns>
[HttpGet("api/account/permission")]
public async Task<IActionResult> PermissionList()
{
    var entities = await _db.PermissionGroups
        .Include(g => g.Permissions)
        .ToListAsync();
    var result = new PermissionGroupListModel
    {
        Count = entities.Count,
        Items = entities.Select(gr =>
GettingPermissionGroupModel<GettingPermissionModel>.FromPermissionGroup(gr))
    };
    return Ok(result);
}

/// <summary>
/// Список разрешений указанной роли
/// </summary>
/// <param name="roleId">id роли</param>
/// <returns></returns>
[HttpGet("api/account/role/{roleId}/permission")]
public async Task<IActionResult> PermissionList(int roleId)
{
    var entities = await _db.PermissionGroups
        .Include(g => g.Permissions)
        .ToListAsync();
    var role = await _db.Roles
        .Include(r => r.Permissions).ThenInclude(p => p.Permission)
        .SingleOrDefaultAsync(r => r.Id == roleId);
    var rolePermissions = role.Permissions.Select(p => p.Permission.Id);
    var result = new PermissionGroupListModel
    {
        Count = entities.Count,
        Items = entities.Select(gr =>
GettingPermissionGroupModel<GettingPermissionModel>.FromPermissionGroup(gr, ids: rolePermissions.ToArray()))
    };
    return Ok(result);
}

/// <summary>
/// Список разрешений по состоянию приложения (state)
/// </summary>
/// <param name="state">Название состояния</param>
/// <returns></returns>
[HttpGet("api/state/permission")]
public async Task<IActionResult> PermissionStateList()
{
    var entity = await _db.States.Include(s => s.Permissions).ToListAsync();
    var permissions = entity.Select(i => new StatePermissionModel
    {
        State = i.Name,
        Permissions = i.Permissions.Select(p => p.Value).ToArray()
    });
    return Ok(new StatePermissionListModel
    {

```

```

        Count = permissions.Count(),
        Items = permissions
    });
}
#endregion

[HttpPost("api/account/reset"), AllowAnonymous]
public async Task<IActionResult> ResetPassword([FromBody]ResetPasswordModel model,
                                              [FromServices]IEmailService emailSvc)
{
    if (!ModelState.IsValid) return BadRequest(ModelState);
    try
    {
        await _userService.GeneratePasswordResetToken(model.Email, emailSvc, (token, userId) => { return new
Uri($"http://{this.Request.Host.Value}/reset?token={WebUtility.UrlEncode(token)}&userId={userId}",
UriKind.Absolute); });
        return Ok();
    }
    catch (System.Exception ex)
    {
        return StatusCode(500, ex);
    }
}

[HttpPost("api/account/password"), AllowAnonymous]
public async Task<IActionResult> ResetPassword([FromBody>PasswordChangeModel model)
{
    if (!ModelState.IsValid) return BadRequest(ModelState);
    var result = await _userService.ChangePassword(model);
    return ExtractResult(result);
}
}
}

```

### 3) OrderRepository:

```

using DeloCRM.ApiModels;
using DeloCRM.ApiModels.Lists;
using DeloCRM.ApiModels.Settings;
using DeloCRM.Database;
using DeloCRM.Models;
using DeloCRM.Results;
using DeloCRM.Services.Interfaces;
using Microsoft.EntityFrameworkCore;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using DeloCRM.Extensions;
using Microsoft.Extensions.Logging;
using Newtonsoft.Json;

namespace DeloCRM.Services
{
    public class OrderRepository : IOrderRepository
    {
        private ILogger _log;
        private AppDb _db;
        private Company _currentCompany;
        private OrderEventRepository _eventRepo;
        private StoreRepository _storeRepo;

        public OrderRepository(AppDb db, ILogger<OrderRepository> log)
        {
            _db = db;
            _log = log;
        }
    }
}

```

```

/// <summary>
/// Общая информация о текущем пользователе
/// </summary>
public UserInfo User { get; set; }
/// <summary>
/// Текущая компания
/// </summary>
public Company CurrentCompany
{
    get
    {
        return _currentCompany ?? _db.Companies.SingleOrDefault(c => c.Id == User.CompanyId.Value);
    }
    set { _currentCompany = value; }
}
/// <summary>
/// Загружает асинхронно текущую компанию
/// </summary>
/// <returns></returns>
public async Task LoadCompany() => _currentCompany = await _db.Companies.SingleOrDefaultAsync(c => c.Id
== User.CompanyId.Value);

/// <summary>
/// Получить все заказы с применением фильтра
/// </summary>
/// <param name="settings">настройки сортировки</param>
/// <param name="filter">настройки фильтрации</param>
/// <returns></returns>
public async Task<GeneralResult<OrderListModel>> List(ListSettings settings, FilterModel filter)
{
    var result = new GeneralResult<OrderListModel>();

    if (filter.DatePeriod != null) filter.DatePeriod.LeftDate =
filter.DatePeriod.LeftDate.Value.Subtract(TimeSpan.FromHours(1));
    Func<Order, bool> filterFunc = o =>
    {
        if (!o.IsRemoved &&
            (filter.ClientId.HasValue ? filter.ClientId.Value == o.ClientId : true) &&
            (filter.DatePeriod != null && filter.DatePeriod.PeriodId != 0 ? o.CreationTime >= filter.DatePeriod.LeftDate
&& o.CreationTime <= filter.DatePeriod.RightDate : true) &&
            (filter.DeviceTypeId.HasValue ? o.Device.TypeId == filter.DeviceTypeId : true) &&
            (filter.Model != null ? o.Device.Model == filter.Model : true) &&
            (filter.OrderTypes != null && filter.OrderTypes.Length == 1 ? (filter.OrderTypes[0] == 0 ?
o.GuaranteeInDays == 0 : o.GuaranteeInDays > 0) : true) &&
            (filter.Statuses != null ? filter.Statuses.Contains(o.StatusId) : true) &&
            (filter.Managers != null ? filter.Managers.Contains(o.ManagerId) : true) &&
            (o.EngineerId.HasValue && filter.Engineers != null ? filter.Engineers.Contains(o.EngineerId.Value) : true))
            return true;
        else return false;
    };
    var query = _db.Orders
        .Include(o => o.Device).ThenInclude(d => d.Type)
        .Include(o => o.Status).ThenInclude(s => s.Category)
        .Include(o => o.Client).ThenInclude(c => c.Source)
        .Include(o => o.Client).ThenInclude(c => c.Phones)
        .Include(o => o.Author).ThenInclude(u => u.Profile)
        .Include(o => o.Manager).ThenInclude(u => u.Profile)
        .Include(o => o.Engineer).ThenInclude(u => u.Profile)
        .Include(o => o.Services)
        .Include(o => o.Parts).ThenInclude(p => p.Debit)
        .Include(o => o.Parts).ThenInclude(p => p.Store)
        .Include(o => o.Parts).ThenInclude(p => p.Definition).ThenInclude(pd => pd.Prices)
        .Include(w => w.Workshop).ThenInclude(w => w.Icon)
        .Where(o => !o.IsRemoved &&
            (User.CurrentWorkshop > 0) ? o.WorkshopId == User.CurrentWorkshop : o.Workshop.CompanyId
== User.CompanyId)

```

```

        .Where(filterFunc);
    if (settings.Descending)
        query = query.OrderByDescending(o => o.CustomId);
    else query = query.OrderBy(o => o.CustomId);
    var count = await query.ToAsyncEnumerable().Count();
    query = query
        .Skip(settings.SkipCount)
        .Take(settings.TakeCount);
    var orders = await query.ToAsyncEnumerable().ToList();

    return result.Ok(new OrderListModel
    {
        Count = count,
        Items = orders.Select(o =>
        {
            var m = RowOrderModel.FromOrder(o, price: o.Services.Select(op => op.Price).Sum() + o.Parts.Select(p =>
p.SoldPrice).Sum());
            m.Workshop.Icon = o.Workshop != null && o.Workshop.Icon != null ?
FileModel.Instance(o.Workshop.Icon, User.HostValue) : null;
            return m;
        })
    });
}

/// <summary>
/// Получить все заказы
/// </summary>
/// <param name="settings"></param>
/// <returns></returns>
public async Task<GeneralResult<OrderListModel>> List(ListSettings settings)
{
    if (settings.Filter != null) return await List(settings, JsonConvert.DeserializeObject<FilterModel>(settings.Filter));
    var result = new GeneralResult<OrderListModel>();

    Func<Order, bool> wherePredicate = o =>
    {
        Func<Order, bool> quickFilterPredicate = null;
        if (settings.PanelId.HasValue)
        {
            switch (settings.PanelId.Value)
            {
                case QuickFilter.Working:
                {
                    quickFilterPredicate = order => order.Status.Category.Title == "Новые" || order.Status.Category.Title
== "В работе";
                    break;
                }
                case QuickFilter.WaitingToPay:
                {
                    quickFilterPredicate = order => order.Status.Category.Title == "Завершенные"; // &&
(order.Parts.Any() || order.Services.Any());
                    break;
                }
                case QuickFilter.WaitingSparepart:
                {
                    quickFilterPredicate = order => order.Status.Title == "Ждет запчасть";
                    break;
                }
            }
        }
        bool _list = false;
        if (!o.IsRemoved &&
            (settings.PanelId.HasValue ? quickFilterPredicate(o) : true) &&
            (User.CurrentWorkshop > 0 ? o.WorkshopId == User.CurrentWorkshop : o.Workshop.CompanyId ==
User.CompanyId) &&

```

```

        ((settings.LeftDate.HasValue && settings.RightDate.HasValue) ? (settings.LeftDate <= o.CreationTime
&& o.CreationTime <= settings.RightDate) : true))
        _list = true;

bool _search = false;

#region Поле поиска
if (!string.IsNullOrEmpty(settings.Input) && settings.SearchType.HasValue)
{
    Action<bool> Search = b => { if (_search && !b) return; else _search = b; }; // заглушка: поставив
однажды true, сменить нельзя

    var input = settings.Input;
    OrderSearchType type = (OrderSearchType)settings.SearchType.Value;
    switch (type)
    {
        case OrderSearchType.ByAnyFields:
        {
            if (o.Manager.Profile.Name.Contains(input, true)) Search(true); else Search(false);
            if (o.Manager.Profile.Surname.Contains(input, true)) Search(true); else Search(false);
            if (o.EngineerId.HasValue)
            {
                if (o.Engineer.Profile.Name.Contains(input, true)) Search(true); else Search(false);
                if (o.Engineer.Profile.Surname.Contains(input, true)) Search(true); else Search(false);
            }
            if (o.Client.Name.Contains(input, true)) Search(true); else Search(false);
            if (o.Client.Phones.Any() && o.Client.Phones.Any(phone => phone.Number.Contains(input, true)))
Search(true); else Search(false);
            if (o.Status.Title.Contains(input, true)) Search(true); else Search(false);
            if (o.Device.Model.Contains(input, true)) Search(true); else Search(false);
            if (o.Defect.Contains(input, true)) Search(true); else Search(false);
            if (o.CustomId.ToString().Contains(input, true)) Search(true); else Search(false);
            break;
        }
        case OrderSearchType.ByClient:
        {
            if (o.Client.Name.Contains(input, true)) Search(true); else Search(false);
            break;
        }
        case OrderSearchType.ByCustomId:
        {
            if (o.CustomId.ToString().Contains(input, true)) Search(true); else Search(false);
            break;
        }
        case OrderSearchType.ByDevice:
        {
            if (o.Device.Model.Contains(input, true)) Search(true); else Search(false);
            break;
        }
        case OrderSearchType.ByEngineer:
        {
            if (o.EngineerId.HasValue)
            {
                if (o.Engineer.Profile.Name.Contains(input, true)) Search(true); else Search(false);
                if (o.Engineer.Profile.Surname.Contains(input, true)) Search(true); else Search(false);
            }
            break;
        }
        default: { Search(true); break; }
    }
}
else _search = true; // to skip
#endregion

if (_list && _search) return true;
return false;

```



```

};

#region Query
var query = _db.Orders
    .Include(o => o.Device).ThenInclude(d => d.Type)
    .Include(o => o.Status).ThenInclude(s => s.Category)
    .Include(o => o.Client).ThenInclude(c => c.Source)
    .Include(o => o.Client).ThenInclude(c => c.Phones)
    .Include(o => o.Author).ThenInclude(u => u.Profile)
    .Include(o => o.Manager).ThenInclude(u => u.Profile)
    .Include(o => o.Engineer).ThenInclude(u => u.Profile)
    .Include(o => o.Services)
    .Include(o => o.Parts).ThenInclude(p => p.Debit)
    .Include(o => o.Parts).ThenInclude(p => p.Store)
    .Include(o => o.Parts).ThenInclude(p => p.Definition).ThenInclude(pd => pd.Prices)
    .Include(w => w.Workshop).ThenInclude(w => w.Icon)
    .Where(wherePredicate);
var count = await query.ToAsyncEnumerable().Count();
if (settings.Descending)
    query = query.OrderByDescending(o => o.CustomId);
else query = query.OrderBy(o => o.CustomId);
query = query
    .Skip(settings.SkipCount)
    .Take(settings.TakeCount);
#endregion
var items = await query.ToAsyncEnumerable().ToList();

return result.Ok(new OrderListModel
{
    Count = count,
    Items = items.Select(o =>
    {
        var m = RowOrderModel.FromOrder(o, price: o.Services.Select(op => op.Price).Sum() + o.Parts.Select(p =>
p.SoldPrice).Sum());
        m.Workshop.Icon = o.Workshop != null && o.Workshop.Icon != null ?
FileModel.Instance(o.Workshop.Icon, User.HostValue) : null;
        return m;
    })
});
}

/// <summary>
/// Получить конкретный заказ
/// </summary>
/// <param name="id"></param>
/// <returns></returns>
public async Task<GeneralResult<ShowOrderModel>> Get(int id)
{
    var result = new GeneralResult<ShowOrderModel>(_log);

    var order = await _db.Orders
        .Include(o => o.Device).ThenInclude(d => d.Type)
        .Include(o => o.Status).ThenInclude(s => s.Category)
        .Include(o => o.Client).ThenInclude(c => c.Source)
        .Include(o => o.Client).ThenInclude(c => c.Phones)
        .Include(o => o.Author).ThenInclude(u => u.Profile).ThenInclude(p => p.Avatar)
        .Include(o => o.Manager).ThenInclude(u => u.Profile).ThenInclude(p => p.Avatar)
        .Include(o => o.Engineer).ThenInclude(u => u.Profile).ThenInclude(p => p.Avatar)
        .Include(o => o.History).ThenInclude(h => h.Who)
        .Include(o => o.History).ThenInclude(h => h.Status).ThenInclude(s => s.Category)
        .Include(o => o.History).ThenInclude(h => h.Author)
        .Include(o => o.Parts).ThenInclude(p => p.Debit)
        .Include(o => o.Parts).ThenInclude(p => p.Store)
        .Include(o => o.Parts).ThenInclude(p => p.Definition).ThenInclude(pd => pd.Prices)
        .Include(o => o.Parts).ThenInclude(o => o.Request)
        .Include(o => o.Services)

```

```

.Include(o => o.Workshop)
.SingleOrDefaultAsync(o => o.Workshop.CompanyId == User.CompanyId && o.Id == id);

if (order == null) return result.NotFound("Не найден заказ с id=" + id);
return result.Ok(ShowOrderModel.FromOrder(order));
}

/// <summary>
/// Создать заказ
/// </summary>
/// <param name="model">модель заказа</param>
/// <returns></returns>
public async Task<GeneralResult<ShowOrderModel>> Create(CreatingOrderModel model)
{
    var result = new GeneralResult<ShowOrderModel>(_log);
    if (User.CurrentWorkshop == 0) return result.Error("Чтобы создать заказ нужно выбрать мастерскую");

    var company = await _db.Companies
        .Include(c => c.Statuses).ThenInclude(s => s.Category)
        .Include(c => c.Workshops)
        .Include(c => c.Employeers)
        .Include(c => c.Settings)
        .SingleOrDefaultAsync(c => c.Id == User.CompanyId.Value);

    #region Установка начального статуса
    if (model.Order.Status == null)
    {
        var status = CurrentCompany.Statuses.SingleOrDefault(i => i.Title == "Новый");
        if (status == null) status = CurrentCompany.Statuses.FirstOrDefault();
        if (status == null) return result.NotFound("Не найден ни единый статус. Чтобы создать заказ нужен как минимум иметь 1 статус.");
        model.Order.Status = new GettingStatusModel { Id = status.Id };
    }
    #endregion
    Order order = model.Order.ToOrder();
    order.GuaranteeInDays = company.Settings.Guarantee;

    #region Client checking
    if (!model.Order.Client.Id.HasValue)
    {
        var clientRepo = new ClientRepository(_db, null) { User = User };
        order.Client = clientRepo.CreateEntity(model.Order.Client.ToCreatingModel());
    }
    else
    {
        order.ClientId = model.Order.Client.Id.Value;
    }
    #endregion

    //var deviceRepo = new DeviceRepository(_db) { CurrentCompany = company };
    order.Device = model.Order.Device.ToDevice();//deviceRepo.AddOrUpdateDevice(model.Device);

    order.CustomId = await GetNextId();
    order.AuthorId = User.Id;
    order.ManagerId = order.AuthorId; // автор как менеджер? или выбранный?

    #region Pay advance
    if (order.Advance != 0f)
    {
        var cashRepo = new CashRepository(_db, null) { User = User };
        order.Paid += order.Advance - order.Advance * (model.Pay.Discount / 100);
        var cashFlow = cashRepo.Pay(order, model.Pay, PayModel.PayType.OrderAdvance);
        if (cashFlow == null) return result.Error("Произошла ошибка в создании кассового движения при оплате аванса заказа");
        cashFlow.Order = order;
    }
}

```

```

    }
    #endregion

    #region Запись событий в историю
    _eventRepo = new OrderEventRepository(_db, null);
    order.History.Add(_eventRepo.CreateEvent(new CreatingOrderEventModel
    {
        Type = OrderEventType.StatusChange,
        AuthorId = User.Id,
        StatusId = order.StatusId
    }, restoreProperty: true));
    order.History.Add(_eventRepo.CreateEvent(new CreatingOrderEventModel
    {
        Type = OrderEventType.System,
        AuthorId = User.Id,
        UserMentionId = order.ManagerId,
        Text = "Назначен менеджер"
    }, restoreProperty: true));
    if (order.EngineerId != null)
        order.History.Add(_eventRepo.CreateEvent(new CreatingOrderEventModel
        {
            Type = OrderEventType.System,
            AuthorId = User.Id,
            UserMentionId = order.EngineerId.Value,
            Text = "Назначен инженер"
        }, restoreProperty: true));
    #endregion
    company.Workshops.SingleOrDefault(w => w.Id == User.CurrentWorkshop).Orders.Add(order);
    try { await _db.SaveChangesAsync(); }
    catch (Exception ex) { return result.Error("Не удалось создать новый заказ", ex); }

    #region Восстановление навигационных свойств
    order.Device.Type = await _db.GetDeviceTypeAsync(order.Device.TypeId.Value);
    order.Author = company.Employees.SingleOrDefault(u => u.Id == order.AuthorId);
    order.Manager = order.Author; // автор как менеджер? или выбранный?
    order.Engineer = order.EngineerId != null ? company.Employees.SingleOrDefault(i => i.Id == order.EngineerId) :
null;
    if (order.Client == null && order.ClientId.HasValue)
        order.Client = await _db.Clients.SingleOrDefaultAsync(c => c.Id == order.ClientId);
    order.Client.Source = order.Client.SourceId != null ? _db.ClientSources.Single(i => i.Id == order.Client.SourceId)
: null;
    #endregion

    var resp = ShowOrderModel.FromOrder(order);
    return result.Ok(resp, LogLevel.Debug, "Успешно создан новый заказ");
}

/// <summary>
/// Редактирование существующего заказа
/// </summary>
/// <param name="orderId">id заказа</param>
/// <param name="model">модель с заказом, работами и запчастями</param>
/// <returns>модель заказа</returns>
public async Task<GeneralResult<RowOrderModel>> Edit(int orderId, EditingOrderModel model)
{
    var result = new GeneralResult<RowOrderModel>(_log);

    if (model.PartList.Distinct().Count() != model.PartList.Length)
        return result.Error("Ошибка: в заказ нельзя присвоить одну и ту же запчасть несколько раз");

    Order order = await _db.Orders
        .Include(o => o.Device).ThenInclude(d => d.Type)
        .Include(o => o.Status).ThenInclude(s => s.Category)
        .Include(o => o.Client).ThenInclude(c => c.Phones)
        .Include(o => o.Author).ThenInclude(u => u.Profile)
        .Include(o => o.Manager).ThenInclude(u => u.Profile)

```

```

.Include(o => o.Engineer).ThenInclude(u => u.Profile)
.Include(o => o.History).ThenInclude(h => h.Who)
.Include(o => o.History).ThenInclude(h => h.Status).ThenInclude(s => s.Category)
.Include(o => o.History).ThenInclude(h => h.Author)
.Include(o => o.Parts).ThenInclude(p => p.Definition).ThenInclude(pd => pd.Prices)
.Include(o => o.Services)
.Include(o => o.Workshop)
.Include(o => o.Credits)
.IncludeOrDefaultAsync(o => o.Workshop.CompanyId == User.CompanyId && o.Id == orderId);

if (order == null) return result.NotFound($"Не удалось найти заказ с id={orderId}");
var engineer = model.Order.Engineer != null ? await _db.GetUserAsync(model.Order.Engineer.Id) : null;
var manager = await _db.GetUserAsync(model.Order.Manager.Id);
if (manager == null) return result.Error($"Не найден менеджер или не был указан при редактировании
{orderId} заказа");

#region Обновление запчастей
// новые и старые, для обновления списания
var newParts = new List<int>();
var oldParts = order.Parts.Select(p => p.Id).ToList();

foreach (var partId in model.PartList)
{
    var match = oldParts.SingleOrDefault(v => v == partId);
    if (match > 0) // сошлось - оставляем нетронутые запчасти
    {
        oldParts.Remove(match);
        continue;
    }
    else // добавляем новую запчасть
    {
        var part = await _db.GetProductAsync(partId);
        if (part == null) return result.NotFound($"Не найдена запчасть с id={partId} при обновлении списка
запчастей в заказе с id={orderId}");
        var lastPrice = await _db.Prices.LastAsync(p => p.DefinitionId == part.DefinitionId);
        part.SoldPrice = lastPrice.Value; // фиксирование цены
        order.Parts.Add(part);
        newParts.Add(partId);
    }
}
// убираем удалённые запчасти
oldParts.ForEach((item) =>
{
    var oldPart = order.Parts.SingleOrDefault(p => p.Id == item);
    oldPart.SoldPrice = null;
    order.Parts.Remove(oldPart);
});
#endregion Обновляем списание
_storeRepo = new StoreRepository(_db, null) { User = User };
if (order.Parts.Count == 0) // если никаких запчастей нет, удаляем списание
{
    var ids = order.Credits.Select(c => c.Id).ToArray();
    foreach (var id in ids)
    {
        await _storeRepo.RemoveCredit(id); // !!!!!!! SaveChanges inside (it's bad)
    }
    // check for success
}
if (order.Parts.Count > 0) // если есть запчасти, то обновляем списания
{
    await _storeRepo.UpdateCreditsForOrder(order, new EditingCreditModel
    {
        NewProducts = newParts.ToArray(),
        DeletedProducts = oldParts.ToArray()
    });
}
}

```

```

#endregion
#endregion

#region Обновление работ
order.Services.Clear();
foreach (var work in model.WorkList)
{
    if (!work.Id.HasValue) // добавление
    {
        var operation = _db.Services.Add(new Operation
        {
            Name = work.Name,
            Price = work.Price
        }).Entity;
        //if (!_db.ApplyChanges()) return null;
        order.Services.Add(operation);
    }
    else
    {
        var operation = await _db.GetServiceAsync(work.Id.Value);
        order.Services.Add(operation);
    }
}
#endregion

#region Добавление событий
_eventRepo = new OrderEventRepository(_db, null) { User = User };
_eventRepo.UpdateOrderHistory(order, model.Order);
#endregion

#region [Paying]
if ((await _db.GetStatusAsync(model.Order.Status.Id)).Title == "Закрыт")
{
    order.ClosedTime = DateTime.UtcNow;
    var toPay = order.CountPrice() - order.Paid;
    if (toPay > 0f) // есть за что платить
    {
        if (model.Pay == null) return result.Error("Модель оплаты заказа null. Невозможно закрыть заказ.");
        var cashRepo = new CashRepository(_db, null) { User = User };
        var cashFlow = cashRepo.Pay(order, model.Pay, PayModel.PayType.Order);
        if (cashFlow == null) return result.Error("Произошла ошибка в создании кассового движения при закрытии заказа");
        cashFlow.Order = order;
    }
}
#endregion

#region [Pay advance]
if (model.PayAdvance != null)
{
    if (!model.PayAdvance.Sum.HasValue && model.PayAdvance.Sum.Value <= 0) return result.Error("Модель оплаты аванса: sum должна быть больше 0");
    order.Paid += model.PayAdvance.Sum.Value;
    order.Advance += model.PayAdvance.Sum.Value;
    var cashRepo = new CashRepository(_db, null) { User = User };
    var cashFlow = cashRepo.Pay(order, model.PayAdvance, PayModel.PayType.OrderAdvance);
    if (cashFlow == null) return result.Error("Произошла ошибка в создании кассового движения при оплате аванса заказа");
    cashFlow.Order = order;
}
#endregion

if (!order.Remaked && !int.Equals(model.Order.Status.Id, order.StatusId) &&
    (await _db.GetStatusAsync(model.Order.Status.Id)).Title == "На доработке")
    order.Remaked = true;

```

```

#region [Удаление привязанных запросов]
if (!int.Equals(model.Order.Status.Id, order.StatusId) &&
    (await _db.GetStatusAsync(model.Order.StatusId)).Title == "Закрыт")
{
    var deliverySvc = new DeliveryService(_db, null) { User = User };
    if (!deliverySvc.DetachAllRequests(order)) return result.Error("Произошла ошибка при удалении запросов,
привязанных к заказу");
}
#endregion

model.Order.UpdateOrder(order);
order.Manager = manager;
order.Engineer = engineer;

try { await _db.SaveChangesAsync(); }
catch (Exception ex) { return result.Error($"Не удалось сохранить изменения в заказе с id={orderId}", ex); }

order.Device.Type = await _db.DeviceTypes.SingleOrDefaultAsync(d => d.Id == order.Device.TypeId);
order.Status.Category = await _db.StatusCategories.SingleOrDefaultAsync(c => c.Id == order.Status.CategoryId);

var resp = RowOrderModel.FromOrder(order);
return result.Ok(resp, LogLevel.Debug, $"Успешно отредактирован заказ с id={orderId}");
}

/// <summary>
/// Удалить заказ
/// </summary>
/// <param name="id"></param>
/// <returns></returns>
public async Task<GeneralResult> Remove(int id)
{
    var result = new GeneralResult(_log);
    var order = await _db.Orders
        .Include(o => o.Workshop)
        .SingleOrDefaultAsync(o => o.Workshop.CompanyId == User.CompanyId && o.Id == id);

    if (order == null) return result.NotFound($"Не найден заказ с id={id} внутри компании");
    order.IsRemoved = true;

    try { await _db.SaveChangesAsync(); }
    catch (Exception ex) { return result.Error($"Не удалось удалить заказ с id={id}", ex); }
    return result.Ok(LogLevel.Debug, $"Успешно удален заказ с id={id}");
}

/// <summary>
/// Узнать следующий уникальный номер для нового заказа
/// </summary>
/// <returns></returns>
public async Task<int> GetNextId()
{
    var orders = await _db.Orders
        .Include(w => w.Workshop)
        .Where(o => o.Workshop.CompanyId == User.CompanyId)
        .ToListAsync();

    var nextId = orders.Select(o => o.CustomId).DefaultIfEmpty().Max();
    return nextId + 1;
}

/// <summary>
/// Редактировать заказ частично
/// </summary>
/// <param name="orderId"></param>
/// <param name="model">модель для частичного редактирования</param>
/// <returns></returns>
public async Task<GeneralResult> PartialEdit(int orderId, PartialEditOrderModel model)

```

```

{
    var result = new GeneralResult(_log);

    Order order = await _db.Orders
        .Include(o => o.Parts).ThenInclude(p => p.Definition).ThenInclude(pd => pd.Prices)
        .Include(o => o.Services)
        .SingleOrDefaultAsync(o => o.Id == orderId);
    if (order == null) return result.NotFound($"Не найден заказ {orderId}");

    if (order.StatusId != model.StatusId) // произошла действительно смена статуса
    {
        order.StatusId = model.StatusId;
        var orderEventRepo = new OrderEventRepository(_db, null) { User = User };
        var statusEvent = orderEventRepo.CreateStatusChangeEvent(model.StatusId);
        order.History.Add(statusEvent);

        #region Paying
        if ((await _db.GetStatusAsync(model.StatusId)).Title == "Закрыт")
        {
            order.ClosedTime = DateTime.UtcNow;
            var toPay = order.CountPrice() - order.Paid;
            if (toPay > 0f) // есть за что платить
            {
                if (model.Pay == null) return result.Error("Модель оплаты заказа null. Невозможно закрыть заказ.");
                var cashRepo = new CashRepository(_db, null) { User = User };
                var cashFlow = cashRepo.Pay(order, model.Pay, PayModel.PayType.Order);
                if (cashFlow == null) return result.Error("Произошла ошибка в создании кассового движения при  
быстром закрытии заказа");
                cashFlow.Order = order;
            }
        }
        #endregion

        try { await _db.SaveChangesAsync(); }
        catch (Exception ex) { return result.Error($"Не удалось частично отредактировать заказ {orderId}", ex); }
    }

    return result.Ok(LogLevel.Debug, $"Заказ {orderId} частично отредактирован");
}

/// <summary>
/// Восстановить заказ по гарантии
/// </summary>
/// <param name="orderId"></param>
/// <returns></returns>
public async Task<GeneralResult<object>> RestoreOrder(int orderId)
{
    var result = new GeneralResult<object>(_log);

    var order = await _db.GetOrderAsync(orderId);
    if (order == null) return result.NotFound("Не найден заказ с таким id");

    var status = await _db.Statuses.Include(s => s.Category).SingleOrDefaultAsync(s => s.Title == "На доработке"
    && s.CompanyId == User.CompanyId);
    if (status == null) return result.NotFound(@"Статус ""На доработке"" не найден");
    order.StatusId = status.Id;

    var orderEventRepo = new OrderEventRepository(_db, null) { User = User };
    var statusEvent = orderEventRepo.CreateStatusChangeEvent(status.Id);
    order.History.Add(statusEvent);

    try { await _db.SaveChangesAsync(); }
    catch (Exception ex) { return result.Error("Не удалось возобновить заказ по гарантии", ex); }

    var resp = new
    {

```

```

        status = GettingStatusModel.FromStatus(status),
        @event = GettingOrderEventModel.FromOrderEvent(statusEvent)
    };
    return result.Ok(resp, LogLevel.Information, "Заказ возобновлен по гарантии");
}
/// <summary>
/// Сменить мастерскую у заказа
/// </summary>
/// <param name="orderId">id заказа</param>
/// <param name="workshopId">id мастерской</param>
/// <returns></returns>
public async Task<GeneralResult> ChangeWorkshop(int orderId, int workshopId)
{
    var result = new GeneralResult(_log);

    var order = await _db.GetOrderAsync(orderId);
    if (order == null) return result.NotFound("Не найден заказ с таким id");

    if (order.WorkshopId.Value == workshopId) return result.Ok(LogLevel.Information, "Смена мастерской у заказа: пропуск");

    order.WorkshopId = workshopId;

    try { await _db.SaveChangesAsync(); }
    catch (Exception ex)
    {
        ex.Data.Add("orderId", orderId);
        ex.Data.Add("workshopId", workshopId);
        return result.Error("Не удалось сменить мастерскую в заказе", ex);
    }

    return result.Ok(LogLevel.Information, "Смена мастерской у заказа: успешно");
}

/// <summary>
/// Получить быструю информацию о заказах
/// </summary>
/// <returns></returns>
public async Task<GeneralResult<OrderCardsModel>> GetQuickOrderCards()
{
    var result = new GeneralResult<OrderCardsModel>(_log);

    var query = _db.Orders
        .Include(o => o.Status).ThenInclude(s => s.Category)
        .Include(o => o.Workshop)
        .Where(o => !o.IsRemoved && (User.CurrentWorkshop != 0 ? o.WorkshopId == User.CurrentWorkshop :
o.Workshop.CompanyId == User.CompanyId));
    var working = await query.Where(o => o.Status.Category.Title == "Новые" ||
o.Status.Category.Title == "В работе").CountAsync();
    var color1 = await (from cat in _db.StatusCategories
        where cat.Title == "Новые" ||
        //cat.Title == "В работе"
        select cat.Color).SingleAsync();
    var waiting = await query.Where(o => o.Status.Title == "Ждет запчасть").CountAsync();
    var color2 = await (from cat in _db.StatusCategories
        where cat.Title == "Отложенные"
        select cat.Color).FirstAsync();
    var done = await query
        //.Include(o => o.Parts)
        //.Include(o => o.Services)
        .Where(o => o.Status.Category.Title == "Завершенные"/* &&*/
/*(o.Parts.Any() || o.Services.Any())*/).CountAsync();
    var color3 = await (from cat in _db.StatusCategories
        where cat.Title == "Завершенные"
        select cat.Color).FirstAsync();

```



```
var resp = new OrderCardsModel
{
    Working = new OrderCardModel { Count = working, Color = color1 },
    Waiting = new OrderCardModel { Count = waiting, Color = color2 },
    Done = new OrderCardModel { Count = done, Color = color3 }
};
return result.Ok(resp, LogLevel.Information, "Получена быстрая информация о заказах: 3 блока");
}
}
```