МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РФ

Федеральное государственное бюджетное образовательное учреждение высшего образования
«ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»
ПОЛИТЕХНИЧЕСКИЙ ИНСТИТУТ

Факультет	*Кафедра*
вычислительной техники	МОиПЭВМ

Направление подготовки 09.03.02 «Информационные системы и технологии»

Направленность Информационные системы и технологии

БАКАЛАВРСКАЯ РАБОТА

на тему

Виджет для автоматизации генерации и согласования документов в CRM-системе

Студент _____ Севостьянова Анастасия Олеговна
(подпись, дата) (ФИО полностью)

Руководитель _____ Михалев А.Г.
(подпись, дата) (фамилия, инициалы)

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

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

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

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

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

Пенза, 2025

Федеральное государственное бюджетное образовательное учреждение
высшего образования
ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ПОЛИТЕХНИЧЕСКИЙ ИНСТИТУТ

Факультет	*Кафедра*
Вычислительной техники	МО и ПЭВМ

«Утверждаю»
Заведующий кафедрой
_____А.Ю. Козлов
«20» декабря 2024 г

ЗАДАНИЕ
НА ВЫПУСКНУЮ КВАЛИФИКАЦИОННУЮ РАБОТУ
БАКАЛАВРА

1. Студент Севостьянова Анастасия Олеговна гр. 21ВИ1 факультета ВТ
<Фамилия, имя, отчество полностью>

2.Тема работы Виджет для автоматизации генерации и согласования документов в CRM-системе
Тема утверждена приказом ПГУ № 12 от " 09 " 06 2025

3. Руководитель работы доцент кафедры «МОиПЭВМ» Михалев А.Г.

4. Задание на работу (назначение разработки, исходные данные и т.п.)
Назначение: виджет обеспечивает генерацию документов и процесс их согласования в CRM-системе amoCRM
Основные функции:
1) Генерация документов
2) Использование формы счета для генерации документов
4) Редактирование документов
5) Согласование документов
Технология разработки: ООП
Язык программирования: TypeScript
Среда программирования: MS Visual Studio

5. Перечень подлежащих разработке вопросов:
1) Анализ предметной области и постановка задачи
2) Проектирование программных средств
3) Реализация программных средств
4) Тестирование программных средств
5) Оформление пояснительной записки
6. Календарный график выполнения работы

№ п/п	Наименование этапов работы	Объем работы	Срок выполнения	Подпись руководителя
1	Анализ предметной области и требований к системе	10%	08.04.2025 – 14.04.2025	
2	Проектирование программных средств	20%	15.04.2025 – 25.04.2025	
3	Реализация программных средств	30%	26.04.2025 – 12.05.2025	
4	Тестирование программных средств	20%	13.05.2025 – 22.05.2025	
5	Оформление пояснительной записки	20%	23.05.2025 – 03.06.2025	

Дата выдачи задания «20» декабря 2024г.

Руководитель бакалаврской работы _____
(подпись, дата)

Задание к исполнению принял студент _____
(подпись, дата)

Работу к защите допустить
Декан ФВТ _____ Л.Р. Фионова
(подпись, дата)

Реферат

Пояснительная записка содержит: 92 листа, 34 рисунка, 40 таблиц, 2 листинга, 4 приложения.

ВИДЖЕТ, POSTGRESQL, TYPESCRIPT, CRM, ГЕНЕРАЦИЯ ДОКУМЕНТОВ, СОГЛАСОВАНИЕ ДОКУМЕНТОВ

Объект разработки: Виджет для автоматизации генерации и согласования документов CRM-системе.

Цель работы: разработать виджет для автоматизации генерации и согласования документов в CRM-системе.

Результаты работы: в результате проекта представлен Виджет для автоматизации генерации и согласования документов в CRM-системе.

Средства разработки: PostgreSQL, TypeScript.

Содержание

Введение 7
1 Анализ предметной области и требований 8
1.1 Анализ предметной области 8
1.2 Сравнительный анализ программных продуктов-аналогов 9
1.3 Анализ требований 12
1.3.1 Анализ функциональных требований 12
1.3.2 Анализ нефункциональных требований 18
1.4 Технологии и средства 19
1.5 Постановка задачи 20
1.6 Планирование разработки и оценка бюджета 21
2 Проектирование виджета для автоматизации генерации и согласования документов в CRM-системе 23
2.1 Проектирование архитектуры системы 23
2.2 Проектирование пользовательского интерфейса виджета для автоматизации генерации и согласования документов в CRM-системе 24
2.3 Проектирование базы данных для виджета для автоматизации генерации и согласования документов в CRM-системе 27
2.2.1 Концептуальное проектирование базы данных 28
2.2.2 Логическое проектирование базы данных 30
2.2.3 Физическое проектирование базы данных 33
3 Реализация виджета для автоматизации генерации и согласования документов в CRM-системе 38
3.1 Реализация пользовательского интерфейса 38
3.2 Реализация функционала 48
3.2.1 Реализация генерации документа 48
3.2.2 Реализация сценариев 49
3.2.3 Реализация интеграции с Яндексом 51
3.2.4 Развертывание системы 53

4 Тестирование виджета для автоматизации генерации и согласования документов в CRM –системе 55
4.1 Тестирование черным ящиком 55
4.2 Метрики Холстеда..... 59
Заключение..... 64
Список использованный источников 66
Приложение А (обязательное) Листинг программы 69
Приложение Б (рекомендуемое) База данных..... 87
Приложение В (рекомендуемое) Диаграммы..... 89
Приложение Г (рекомендуемое) Акт внедрения дипломной работы 97

Введение

Автоматизация процессов формирования документов имеет большое значения для повышения эффективности организационного процесса управления и повышения качества работы сотрудника. Документы в современных бизнес-процессах являются неотъемлемой частью любого бизнеса. На ручное формирование типовых форм документов уходит большое количество времени, при этом не исключается человеческий фактор, при котором в важных документах могут присутствовать ошибки.

CRM система – это система, которая собирает данные о клиенте и формирует на основе этих данных профиль клиента [1]. Из преимуществ CRM можно выделить возможность объединить в себе различные каналы связи: мессенджеры, онлайн-чаты, почта.

Компании, внедряющие себе CRM системы, стремятся к работе в рамках «одного окна», т. е. в одной вкладке. CRM-система позволяет прозрачно вести клиента по воронке продаж, конечным пунктом которой является совершенная продажа. Одним из этапов продажи является подготовка определенных документов. Это может быть договор на оказание услуг, счет на оплату, акт выполненных работ или сертификат, если речь идет о образовательном курсе. Важные документы необходимо согласовывать с юристами и руководителями компании.

Решение, которые позволяют цифровизировать и автоматизировать эти бизнес-процессы представляет актуальность и интерес для практического внедрения в информационные системы.

Объектом исследования бакалаврской работе является автоматизация генерации и согласования документов в CRM-системе.

Предметом исследования бакалаврской работы является виджет для автоматизации генерации и согласования документов в CRM-системе.

Цель выпускной квалификационной работы – создание виджета для автоматизации генерации и согласования документов в CRM-системе.

1 Анализ предметной области и требований

1.1 Анализ предметной области

Генерация документов – это операция, позволяющая создать документ на основе заранее размеченного шаблону и введенных пользователям данных. Добавляя возможность согласовать документы решение значительно освобождает время сотрудников и уменьшает вероятность возникновения ошибок.

Данное решение является виджетным дополнением к уже существующей CRM-системе – amoCRM. Виджет интегрирован с интерфейсом amoCRM следующими способами:

– отдельная группа полей в интерфейсе сделки;

– триггер в цифровой воронке;

– отдельный раздел в настройках, доступ к которой имеет администратор.

Пользователь, имеющий в системе права администратора, в разделе настройки может самостоятельно настроить шаблон и сопоставить переменные в документе. Так пользователь с правами администратора может настроить сценарии согласования: добавить этапы согласования, назначить ответственного за каждый этап, поставить уведомление на этап в виде задачи. Помимо этого, он может самостоятельно в цифровой воронке установить триггер на этапе. При переходе сделки на этап будет происходить автоматическая генерация выбранного документа.

Пользователь, имеющий доступ к разделу «Сделки», может выбрать шаблон и запустить генерацию документа, выбрать уже сгенерированный.

документ и запустить сценарий согласования и самостоятельно редактировать документ.

1.2 Сравнительный анализ программных продуктов-аналогов

На российском рынке существует множество CRM-систем, одними из наиболее популярных являются amoCRM, Битрикс24, RetailCRM [2]. Данные CRM-системы подходят для разных сценариев использования.

AmoCRM позиционирует себя как CRM-система для малого и среднего бизнеса, что определяет её направление развития – нет лишней информации, все заточено для того, чтобы не отвлекаясь вести работу с клиентской базой. Интерфейс заточен под быстрое совершение действий, не прибегая к работе с другими разделами.

Битрикс24 – CRM-система для среднего и большого бизнеса, больше является корпоративны порталом с различным уровнем автоматизаций, чем CRM-системой. На данный момент включает в себя обширное множество различных функций: CRM-система, корпоративный чат, маркетинг, ведение проектов и многое другое.

RetailCRM хорошо подходит для розничных продаж: работа с розничными сетями, e-commerce, маркетплейсами. Хорошо реализована работа с заказами, управление складом.

В данной дипломной работе выбрана CRM-система amoCRM, потому что её минималистичный интерфейс дает возможность внедрения собственного интерфейса (в Битрикс24 интерфейс перегружен другими функциями, что усложняет работу в системе), а RetailCRM слишком узкоспециализирован и ориентирован на B2C-сегмент и управление заказами.

В соответствии с разрабатываемым виджетом, было решено более подробно изучить виджеты для генерации и согласования документов и провести сравнительный анализ популярных аналогов.

В таблице 1 представлены ключевые характеристики аналогичных решений в amoCRM, включая их функциональные возможности, интеграции с другими сервисами и использование API для взаимодействия с базовым функционалом amoCRM.

Таблица 1 – Сравнение виджетов от интеграторов amoCRM

Название виджета Сравнение	Документы от Интроверт	Документы от Команды F5	Документы 2.0 от Генезис	Документооборот от Nova	Доку
Возможность работать по сценариям с документами внутри компании	Нет	Да	Нет	Нет	Да
Свои юр. лица	Нет	Нет	Да	Нет	Да
Форма счета	Нет	Нет	Да	Нет	Да
Возможность добавлять шаблоны	Да	Да	Да	Да	Да
Есть ли реестр документов в amoCRM	Нет	Да	Да	Нет	Да
Есть ли возможность создавать документы из ЦВ (только для тарифов Расширенный и выше)	Да	Да	Да	Да	Да
Работает с полями сделки\контакта\компании	Да	Да	Да	Да	Да
Счетчики документов	Да	Да	Да	Да	Да
Запись ссылок на сгенирированный документ в поля сделок	Нет	Да	Да	Да	Да
Можно ли загружать документы без шаблонов	Да	Нет	Нет	Нет	Да
Есть ли возможность работать с шаблонами таблиц (Excel)	Да	Нет	Нет	Да	Нет
Возможность ограничить редактирования документов некоторыми пользователями	Да	Нет	Нет	Да	Да
Интеграция с Яндекс диском	Нет	Нет	Нет	Нет	Да
Интеграция с Яндекс документами	Нет	Нет	Нет	Нет	Да
Интеграция с Google Docs	Нет	Нет	Нет	Нет	Да
Свои товары	Нет	Нет	Нет	Нет	Да
Общие плюсы	- Включает в себя возможности генерации и хранение документов на Google Drive	- Есть общий сценарий реестров, возможность работы из SalesBot	- Есть общий сценарий реестров, возможность работы из SalesBot	- Есть возможность работать с шаблонами и excel и презентациями	- Есть возможность согласования документов, возможность работы из Digital Pipeline

Продолжение таблицы 1

Название виджета Сравнение	Документы от Интроверт	Документы от Команды F5	Документы 2.0 от Генезис	Документооборот от Nova	Доку
Общие минусы	-Нет возможности работать с шаблонами в excel, не работает из SalesBot	- Работать можно только с шаблонами, для интеграции с Google Drive нужно ставить доп.решение	- Работать можно только с шаблонами	- Работать можно только с шаблонами	- Нет возможности генерировать документы в формате excel, не работает из SalesBot

Анализ четырех популярных решений в amoCRM показывает сильные и слабые стороны виджета. Выбор оптимального варианта зависит от сценария использования.

Ключевые функции, необходимые для разработки актуального для рынка виджета:

- умение работать со сценариями;
- возможность выставлять счета от разных юр. лиц;
- использование формы счета для выбора позиций товаров;
- наличие реестра сгенерированных документов;
- интеграция с сервисами Яндекс;
- использование своего списка товаров.

Виджет должен являться универсальным решением, закрывающим большинство потребностей потребителя. Сравнение с текущими решениями показывает наличие большого количества преимуществ перед ними. Основные функции при разработке решения будут учтены при постановке задачи на дипломное проектирование.

1.3 Анализ требований

1.3.1 Анализ функциональных требований

Функциональные требования позволяет выявить прикладные функции системы. Они описывают какие действия, в какой последовательности должен совершить пользователь и как на эти действия должна отреагировать система [3].

В виджете существует два типа пользователей: сотрудник и администратор.

Основными функциями администратора являются:

– авторизацию/ деавторизацию в Яндекс.Диске и Google Disk;

– загрузка, удаление, редактирование, просматривание шаблон документа;

– сопоставление переменных в шаблоне;

– создание, удаление, редактирование папки с шаблонами;

– создание, удаление, редактирование сценариев;

– добавление, удаление, редактирование товаров;

– добавление, удаление редактирование юр. лиц;

– добавление, удаление, редактирование документа;

– выбор шаблона для генерации документа;

– генерирование документа;

– заполнение формы счета перед генерацией документа;

– просматривание, редактирование, удаление сгенерированного документа;

– редактирование формы счета;

– создание на основе сгенерированного документа новый (используя уже заполненную форму счета);

– скачивание документа;

– загрузка документа с компьютера;

– копирование ссылки на документ на Яндекс.Диске или Google Disk;

– запуск сценариев.

Основными функциями сотрудника являются:

– выбор шаблона для генерации документа;

– генерирование документа;

– заполнение формы счета перед генерацией документа;

– просмотр, редактирование, удаление сгенерированного документа;

– редактирование формы счета;

– создание на основе сгенерированного документа новый (используя уже заполненную форму счета);

– скачивание сгенерированного документа;

– загрузка документа с компьютера;

– копирование ссылки на документ на Яндекс.Диске или Google Disk;

– запуск сценариев.

Диаграмма вариантов использования позволяет отобразить последовательность действий пользователя в системе и наглядно показать различия вариантов использования для каждой роли (актера) [4].

Анализ предметной области позволил составить требования к системе и составить диаграмму вариантов использования.

Данная диаграмма описывает создаваемую систему на концептуальном уровне [5]. Было выделено два актера:

– пользователь с ограниченными настройками к системе;

– администратор, имеющий возможность редактирования и настройки системы.

Так же в диаграмме вариантов использования были отражены основные функции, доступные пользователям.

Диаграмма вариантов использования представлена на рисунке 1.

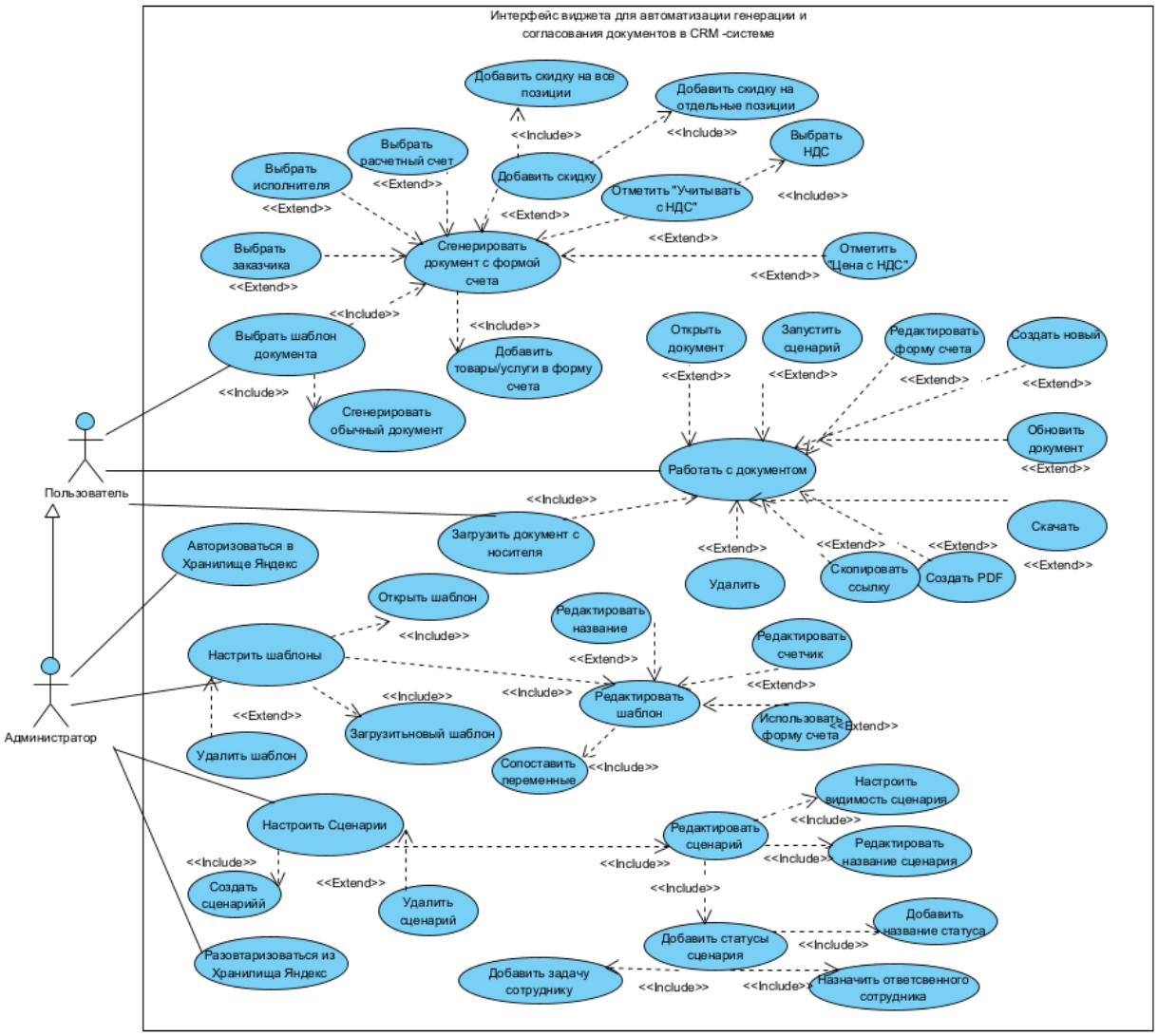


Рисунок 1 – Диаграмма вариантов использования

Для данной диаграммы вариантов использования был описана спецификация прецедентов. Спецификация прецедентов описывает прецедент, краткое описание, главных актеров, второстепенных актеров, предусловия для выполнения сценария, основной поток действий, постусловия и альтернативные потоки [6].

Спецификации прецедентов «Сгенерировать обычный документ» представлена на таблице 2.

Таблица 2 – Сценарий прецедента «Сгенерировать обычный документ»

Название	Описание
Прецедент	Сгенерировать обычный документ
ID	1

Продолжение таблицы 2

Название	Описание
Краткое описание	Пользователь генерирует документ без формы счета
Главные актеры	Пользователь
Второстепенные актеры	Нет
Предусловия	1. Пользователь открывает форму виджета 2. Пользователь выбирает шаблон документа без формы счет
Основной поток	1. Система генерирует документ, подставляет вместо переменных в документе, заранее введенные в CRM значения 2. Система сохраняет сгенерированный документ на Яндекс.Диске
Постусловия	Документ сгенерирован
Альтернативные потоки	Нет

Спецификации прецедентов «Сгенерировать документ с формой счета» представлена на таблице 3.

Таблица 3 – Сценарий прецедента «Сгенерировать документ с формой счета»

Название	Описание
Прецедент	Сгенерировать документ с формой счета
ID	2
Краткое описание	Пользователь генерирует документ без формы счета
Главные актеры	Пользователь
Второстепенные актеры	Нет
Предусловия	1. Пользователь открывает форму виджета 2. Пользователь выбирает шаблон документа с формой счета
Основной поток	1. Система открывает форму счета 2. Пользователь заполняет форму счета и нажимает «сохранить» 3. Система генерирует документ, подставляет вместо переменных в документе, заранее введенные в CRM значения 4. Система сохраняет сгенерированный документ на Яндекс.Диске
Постусловия	Внесенные изменения сохранены
Альтернативные потоки	Нет

Спецификации прецедентов «Редактировать шаблон» представлена на таблице 4.

Таблица 4 – Сценарий прецедента редактирование шаблона

Название	Описание
Прецедент	Редактировать шаблон
ID	1
Краткое описание	Администратор редактирует шаблон документа
Главные актеры	Администратор
Второстепенные актеры	Нет

Продолжение таблицы 4

Название	Описание
Предусловия	1. Администратор открыл форму виджета 2. В виджет загружен шаблон документа
Основной поток	1. Система: отображает в интерфейсе переменные из загруженного документа 2. Администратор сопоставляет переменные с системными полями CRM-системы. 3. Администратор редактирует название шаблона 4. Администратор редактирует счетчик сгенерированных документов 5. Администратор сохраняет внесенные изменения
Постусловия	Внесенные изменения сохранены
Альтернативные потоки	Нет

Спецификации прецедентов «Настроить шаблон» представлена на таблице 5.

Таблица 5 – Сценарий прецедента настроить шаблон

Название	Описание
Прецедент	Настроить шаблон
ID	1
Краткое описание	Администратор настраивает шаблон документа
Главные актеры	Администратор
Второстепенные актеры	Нет
Предусловия	1. Администратор открыл форму виджета
Основной поток	1. Система: отображает в интерфейсе все загруженные шаблоны 2. Администратор может выбрать существующий шаблон, удалить существующий шаблон или загрузить новый= 3. Администратор производит действие над шаблоном 4. Администратор сохраняет внесенные изменения
Постусловия	Внесенные изменения сохранены
Альтернативные потоки	Нет

Спецификации прецедентов «Настроить сценарий» представлена на таблице 6.

Таблица 6 – Сценарий прецедента настроить сценарий

Название	Описание
Прецедент	Настроить сценарий
ID	1
Краткое описание	Администратор настраивает сценарий документа
Главные актеры	Администратор
Второстепенные актеры	Нет
Предусловия	1. Администратор открыл форму виджета

Продолжение таблицы 6

Название	Описание
Основной поток	1. Система: отображает в интерфейсе все добавленные сценарии 2. Администратор может выбрать существующий сценарий, удалить существующий сценарий или загрузить новый 3. Администратор производит действие над сценарием 4. Администратор сохраняет внесенные изменения
Постусловия	Внесенные изменения сохранены
Альтернативные потоки	Нет

Актер «Пользователь» использует варианты использования «Выбрать шаблон документа», «Работать с документом», «Загрузить документ с носителя».

Вариант использования «Выбрать шаблон документа» включает вариант использования «Сгенерировать документ с формой счета» и «Сгенерировать обычный документ». Варианты использования «Выбрать заказчика», «Выбрать исполнителя», «Выбрать расчетный счет», «Добавить скидку», «Отметить «Учитывать с НДС»», «Отметить «Цена с НДС»» расширяют вариант использования «Сгенерировать документ с формой счета». Вариант использования «Добавить скидку» включает в себя «Добавить скидку на все позиции», «Добавить скидку на отдельные позиции». Вариант использования «Отметить «Учитывать с НДС»» включает в себя «Выбрать НДС». Вариант использования «Сгенерировать документ с формой счета» включает в себя вариант использования «Добавить товары/услуги в форму счета».

Варианты использования «Открыть документ», «Запустить сценарий», «Редактировать форму счета», «Создать новый», «Обновить документ», «Скачать», «Создать PDF», «Скопировать ссылку», «Удалить» расширяют вариант использования «Работать с документом».

Вариант использования «Загрузить документ с носителя» включает вариант «Работать с документом».

Актер «Администратор» наследует варианты использования от актера «Пользователь».

Актер «Администратор» использует варианты использования «Авторизоваться в хранилище Яндекс», «Настроить шаблоны», «Настроить сценарии».

Вариант использования «Удалить шаблон» расширяет вариант использования «Настроить шаблон». Варианты использования «Настроить шаблон» включает в себя варианты использования «Открыть шаблон», «Загрузить новый шаблон», «Редактировать шаблон». Варианты использования «Редактировать название», «Редактировать счетчик», «Использовать форму счета» расширяют вариант использования «Редактировать шаблон». Вариант использования «Редактировать шаблон» включает вариант использования «Сопоставить переменные».

Вариант использования «Настроить сценарии» включает в себя варианты использования «Создать сценарий», «Удалить сценарий», «Редактировать сценарий». Вариант использования «Редактировать сценарий» включает в себя варианты использования «Настроить видимость сценария», «Редактировать название сценария», «Добавить название статусов»,», «Добавить статусы сценариев». Вариант использования «Добавить статусы сценариев» включает в себя варианты «Добавить задачу сотруднику», «Добавить название статусов», «Назначить ответственного сотрудника».

Диаграмма вариантов использования в полной мере отображает возможности пользователей с разными правами взаимодействовать с системой.

1.3.2 Анализ нефункциональных требований

Нефункциональные требования описывают условия, которые не относятся к поведению и функциональности системы, но обеспечивают их на уровне компонентов архитектуры программного продукта [7].

Список нефункциональный требований к «Виджету для автоматизации генерации и согласования документов в CRM -системе»:

– интерфейс должен быть интуитивно понятный для пользователя;

– клиентская часть интеграции (виджет) не должна перекрывать элементы управления amoCRM и мешать пользователю взаимодействовать с интерфейсом amoCRM [8];

– виджет должен интегрироваться в интерфейс amoCRM и соответствовать общей стилистике;

– интерфейс должен корректно отображаться на различных устройствах в браузере;

– передача данных между серверной и клиентской частью должна осуществляться через JSON;

– не должно быть зацикленных запросов к API amoCRM;

– ответ сервера не должен превышать 3 мс;

– генерация документа не должна превышать 30 секунд;

– возникновение ошибки загрузки/генерации документа у одного пользователя, не должна влиять на работу виджета других пользователей.

Нефункциональные требования отображают требования к системе, которые не относятся к логике работы виджета, но влияют на выбор архитектуры, базы данных и алгоритмов при написании кода.

1.4 Технологии и средства

Разрабатываемый виджет для amoCRM будет представлять из себя веб-приложение, встраиваемое в интерфейс amoCRM. Для написания программы интерфейса и сервера был выбран язык программирования TypeScript, который расширяет возможности JavaScript и позволяет отлавливать базовые ошибки на этапе компиляции кода.

Интерфейс будет построен с использованием библиотеки React. React позволяет работать с компонентами, что значительно упрощает интеграцию с интерфейсом amoCRM. Серверная логика будет реализована с использованием фреймворка NestJS. Разработка будет производиться в MS Visual Studio. В качестве основной базы данных была выбрана PostgreSQL из-за её удобства и

высокой производительности. Для более удобной работы с базой выбрана TypeORM. Тестирование будет производиться в рабочем интерфейсе amoCRM.

1.5 Постановка задачи

Виджет для автоматизации генерации и согласования документов в CRM–системе должен иметь следующие функции:

– интеграцию с Яндекс.Диском, которая позволит хранить на диске шаблоны документов, сформированные на основе шаблонов документы клиентов, загруженные документы клиентов;

– загрузку, удаление и настройку шаблона (сопоставление переменных, работа с функциями, датой, счетчиком, формой счета);

– создание, скачивание, редактирование документов на основе шаблонов;

– загрузку документов с компьютера;

– работу со списками: товарами, юр. лицами и расчетными счетами;

– работу с формой для формирования документа;

– реестр сгенерированных документов;

– работу со сценариями согласования документов;

– работу с папками шаблонов на стороне виджета.

Не функциональные требования:

– интерфейс должен быть интуитивно понятный для пользователя;

– клиентская часть интеграции (виджет) не должна перекрывать элементы управления amoCRM и мешать пользователю взаимодействовать с интерфейсом amoCRM [9];

– виджет должен интегрироваться в интерфейс amoCRM и соответствовать общей стилистике;

– интерфейс должен корректно отображаться на различных устройствах в браузере;

– передача данных между серверной и клиентской частью должна осуществляться через JSON;

– не должно быть зацикленных запросов к API amoCRM;

– ответ сервера не должен превышать 3 мс;

– генерация документа не должна превышать 30 секунд;

– возникновение ошибки загрузки/генерации документа у одного пользователя, не должна влиять на работу виджета других пользователей.

Для исполнения поставленной задачи в данной дипломной работе был выбран язык программирования TypeScript, фреймворк NestJS и база данных PostgreSQL.

1.6 Планирование разработки и оценка бюджета

Планирование проекта включает в себя постановку основных задач, установку сроков выполнения и ответственных лиц. Список поставленных задач приведен на рисунке 2. Дата начала проекта: 01.05.2025. Дата завершения: 30.05.2025.

ID	Task Name	Start	End	Duration	Resources
1	Начало проекта	2025-05-01	2025-05-01	1 day	
2	▾ Анализ требований и постановка задачи	2025-05-02	2025-05-07	4 days	Аналитик
3	Анализ предметной области	2025-05-02	2025-05-06	3 days	Аналитик
4	Сравнение программ-аналогов	2025-05-05	2025-05-06	2 days	Аналитик
5	Постановка задачи	2025-05-06	2025-05-07	2 days	Аналитик
6	▾ Проектирование программных средств	2025-05-06	2025-05-13	6 days	
7	Проектирование архитектуры системы	2025-05-06	2025-05-08	3 days	Программист 1,Аналитик
8	Проектирование базы данных	2025-05-07	2025-05-09	3 days	Программист 1,Аналитик
9	Проектирование пользовательского интерфейса	2025-05-12	2025-05-13	2 days	Программист 2
10	▾ Реализация	2025-05-14	2025-05-20	5 days	
11	Реализация серверной части приложения	2025-05-14	2025-05-20	5 days	Программист 1
12	Реализация клиентской части приложения	2025-05-14	2025-05-20	5 days	Программист 2
13	▾ Тестрирование	2025-05-20	2025-05-29	8 days	
14	Тестирование серверной части	2025-05-20	2025-05-29	8 days	Программист 1,Аналитик
15	Тестирование клиенткой части	2025-05-20	2025-05-29	8 days	Программист 2,Аналитик
16	Составление и оформление документации	2025-05-26	2025-05-29	4 days	Аналитик
17	Конец проекта	2025-05-30	2025-05-30	1 day	

Рисунок 2 – Основные задачи проекта

Диаграмма Ганта проекта представлена на рисунке 3.

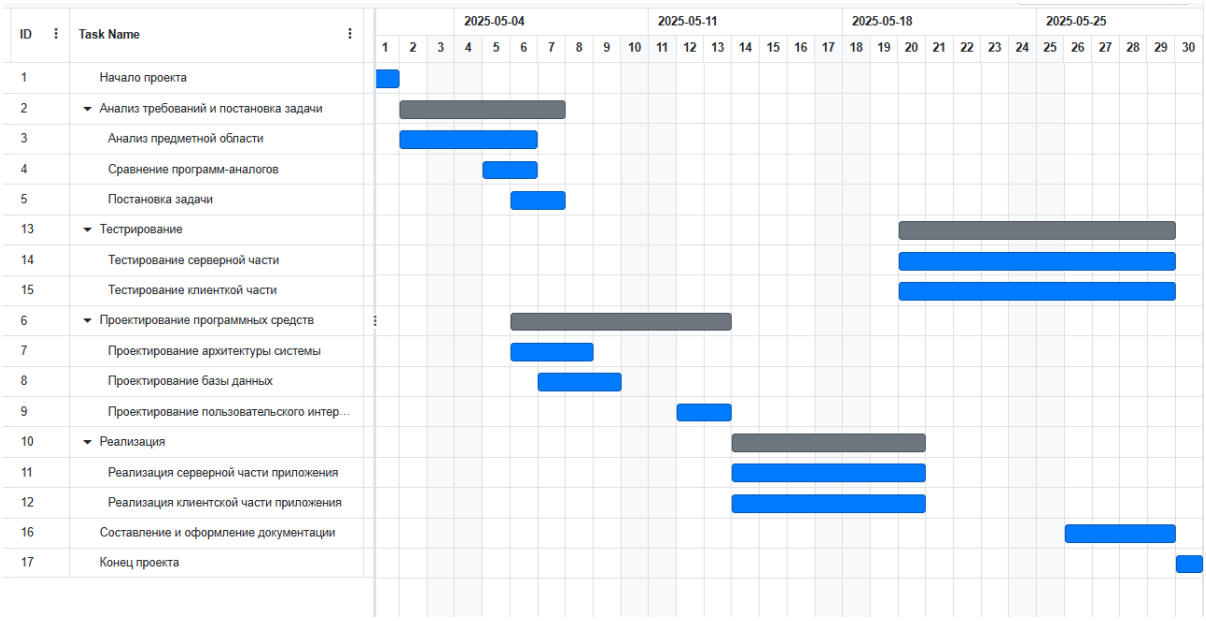


Рисунок 3 – Диаграмма Ганта

Список ресурсов представлены в таблице 7.

Таблица 7 – Список ресурсов

Название ресурса	Тип	МаксЕд.,%	Стандартная ставка, руб./час	Ставка сверхурочная руб./час	Начисление
Аналитик	Трудовой	100	1200.00	2400.00	Пропорциональное
Программист 1	Трудовой	100	1600.00	3200.00	Пропорциональное
Программист 2	Трудовой	100	1400.00	2800.00	Пропорциональное

В результате предварительной оценки бюджета примерная стоимость разработки составила 385900 рублей. Общая продолжительность разработки составляет 192 человеко-часа и займет 22 календарных дня при условии соблюдения составленного графика.

2 Проектирование виджета для автоматизации генерации и согласования документов в CRM -системе

2.1 Проектирование архитектуры системы

В данной дипломной работе используется клиент-серверная архитектура (рисунок 4). При этом для разработки серверной части используется монолитная архитектура.

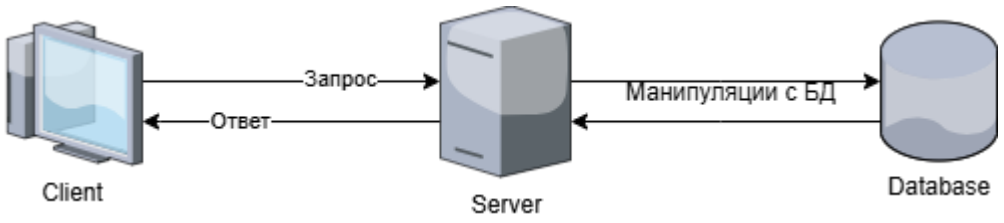


Рисунок 4 – Схема клиент-серверной архитектуры

Используется классическая трехуровневая архитектура, разделенная на три логический уровня: слой клиента, слой логики, слой данных [10].

Клиент представляет из себя интерфейс для взаимодействия пользователя и сервера. Сервер обрабатывает запросы клиента и производит манипуляции с базой данных. В базе данных хранятся данные сгенерированного документа, связи на шаблон, файл и форму счета.

Спроектированная диаграмма компонентов представлена на рисунке 5.

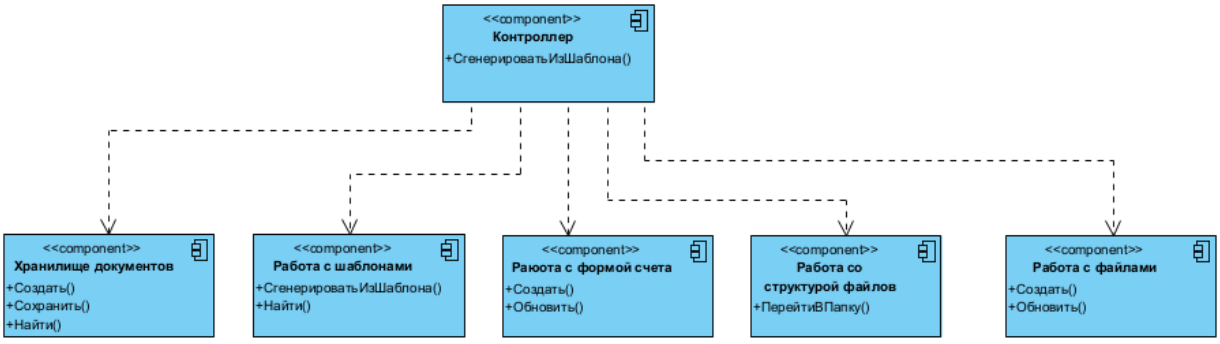


Рисунок 5 – Диаграмма компонентов

На диаграмме присутствуют следующие компоненты:

– контроллер – функция, принимающая HTTP запросы;

– хранилище документов – реализует операции над документом в БД;

– работа с шаблонами – реализует генерацию документа из шаблона;

– работа с формой счета – создание/обновление записей формы счета;

– работа со структурой – создание структуры директорий в хранилище;

– работа с файлами – сохранение и обновление файлов.

Метод СгенерироватьИзШаблона будет позволять создавать и обновлять документ, учитывая переданные параметры. Сам документ будет обновляться путем повторной генерации с сохранением существующих метаданных, но с измененными переменными и формой счета. Диаграмма компонентов отображает взаимодействие с шаблонами, формами и файлами, что обеспечивает согласованность данных.

2.2 Проектирование пользовательского интерфейса виджета для автоматизации генерации и согласования документов в CRM -системе

Пользовательский интерфейс – это совокупность элементов и средств для управления определенным предметом, программой, или системой [11]. Представляет из себя набор компонентов (кнопки, иконки, поля, разделы и пр.) позволяющие пользователю взаимодействовать с системой, посылать определенные команды [12].

В результате анализа предметной области в дипломной работе «Виджет для автоматизации генерации и согласования документов в CRM-системе» были выделены следующие элементы пользовательского интерфейса:

– отдельная группа «Доку» в карточке сделки;

– форма счета для шаблонов, предполагающих добавление товаров/услуг;

– список доступных пользователю функций для работы с документом.

В карточке сделка должна быть реализована группа «Доку» (рисунок 6). Группа создаётся автоматчики после подключения виджета в аккаунт

пользователя. После подключения интеграции с облачным хранилищем и добавлением шаблонов в данной группе появляется возможность выбрать шаблон для генерации или загрузить уже существующий документ.

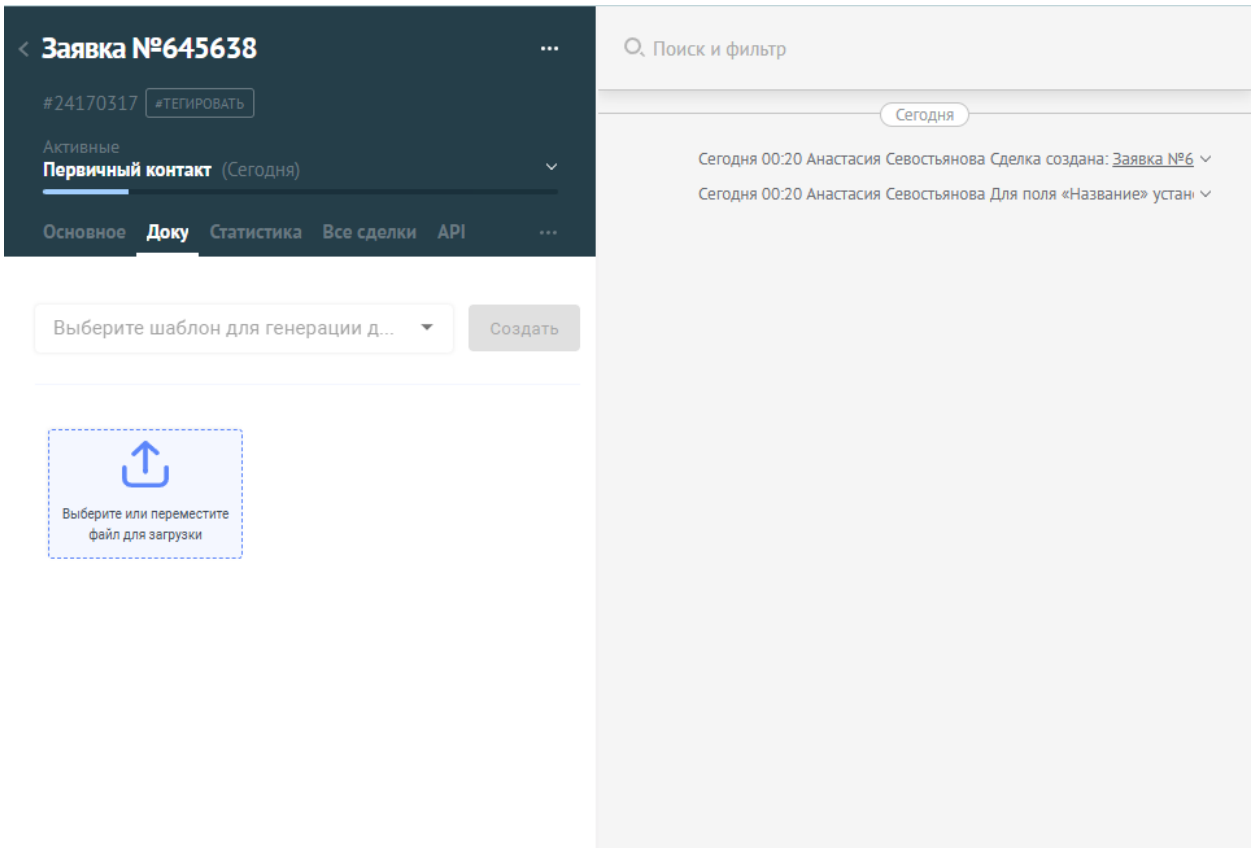


Рисунок 6 – Интерфейс группы «Доку»

На рисунке 7 макет списка шаблонов. У некоторых шаблонов будет возможность добавить товары/услуги. Серой иконкой будут помечены документы без формы счета. Они будут генерироваться сразу. В обоих случаях, при наличии переменных в шаблоне, будет произведена подстановка значений из заполненных полей в CRM.

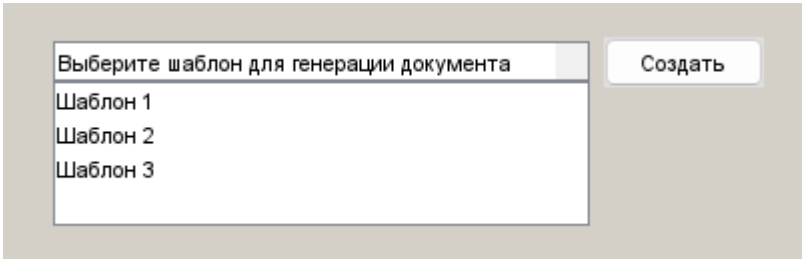


Рисунок 7 – Макет списка шаблонов

При выборе шаблона, использующего форму счета, откроется окно (рисунок 8). В данном окне будет возможность редактировать дату, заказчика, исполнителя, реквизиты. Будет возможность применить скидку, учитывать НДС, добавлять и редактировать товары/услуги.

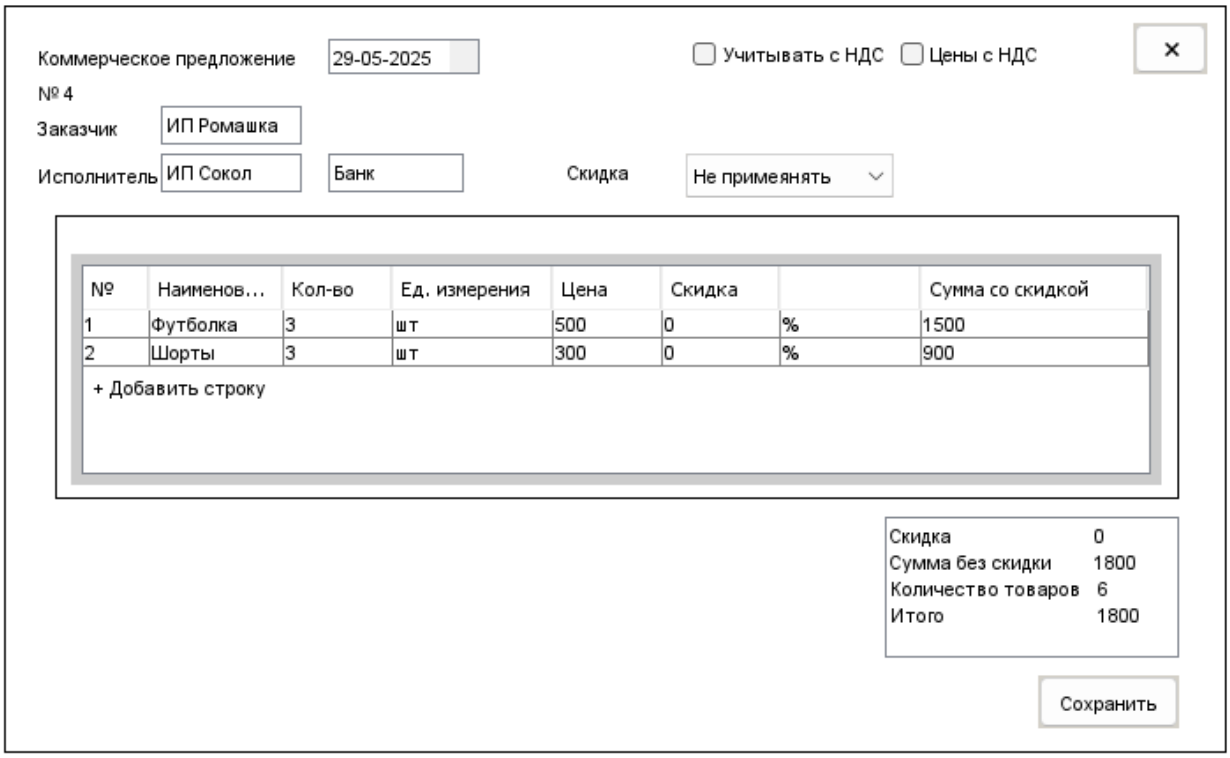


Рисунок 8 – Макет интерфейса формы счета

После генерации документа, он станет доступным в группе «Доку». Доступные действия над документом (рисунок 9):

– открыть документ;
– редактировать форму;
– создать новый;
– обновить документ;
– скачать;
– создать PDF;
– скопировать ссылку;
– отправить на согласование;
– удалить.

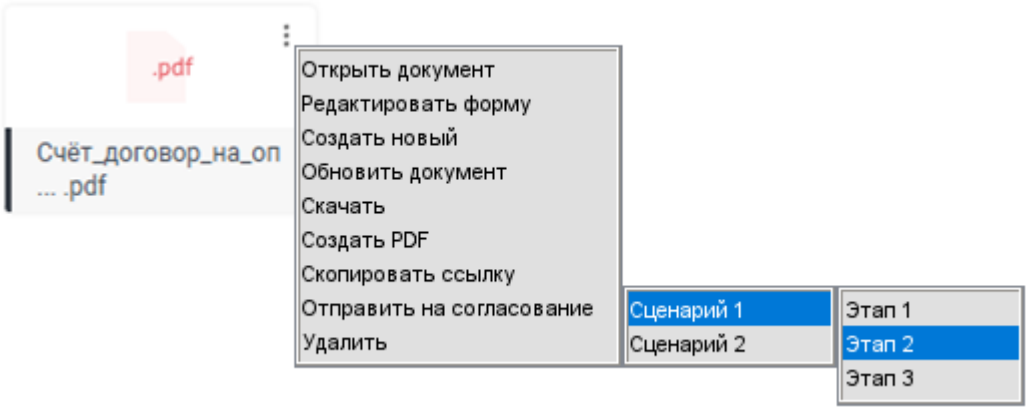


Рисунок 9 – Макет интерфейса доступных функций

Отдельное внимание следует уделить процессу согласования. У пользователя должна быть возможность выбрать сценарий согласования. Этапы сценария настраиваются администратором. За каждый этап назначается конкретный ответственный пользователь. При запуске сценария пользователю будет поставлена задача. Далее пользователю в списке будет доступен для редактирования тот этап, за который он ответственный. Этот этап можно «принять» и тогда процесс согласования пойдет дальше, либо «отклонить», тогда процесс остановится.

2.3 Проектирование базы данных для виджета для автоматизации генерации и согласования документов в CRM -системе

Проектирование базы данных является важным аспектом в разработке приложения. Неправильно спроектированная база может увеличить время обработки запросов и объем хранимых данных, которые не будут использоваться системой [13].

Проектирование длится на три этапа: концептуальное, логическое, физическое [14].

В данной дипломной работе в результате анализа предметной области были разработаны концептуальная, логическая и физическая модели базы данных. Разработка производилась при помощи Visual Paradigm.

2.2.1 Концептуальное проектирование базы данных

Концептуальное проектирование – это процесс представления модели данных, не зависимо от реализации.

На рисунке 10 представлена концептуальная модель базы данных.

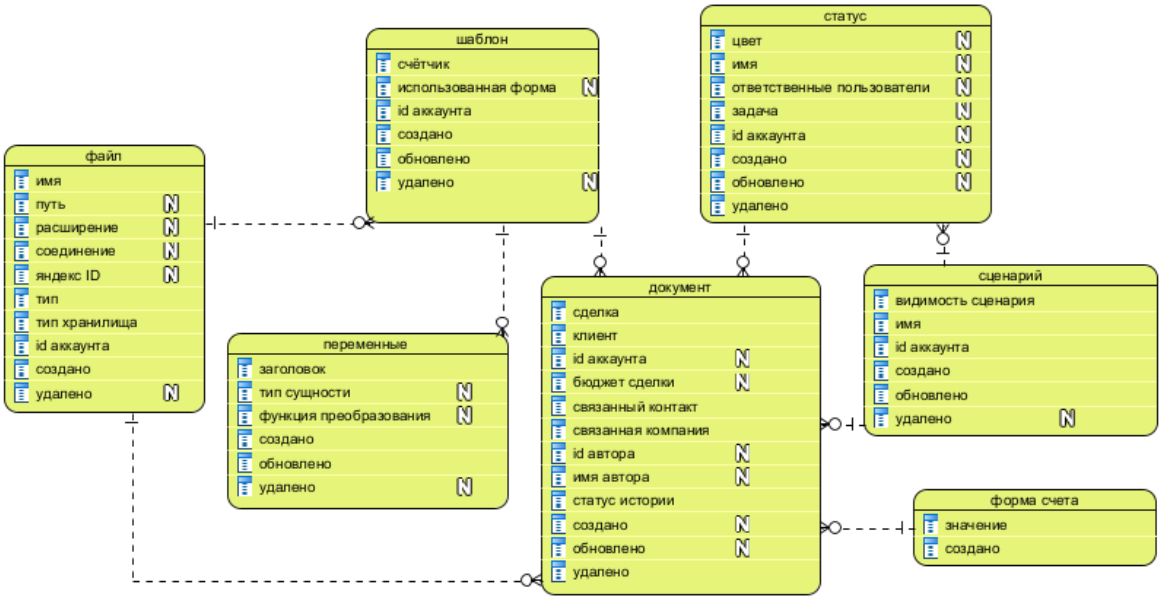


Рисунок 10 – Концептуальная модель базы данных

Сущность «Документ» изображена в таблице 8.

Таблица 8 – Документ

Название столбца	Тип данных	Длина	Nullable
id автора	integer	10	Истина
id аккаунта	integer	10	Истина
бюджет сделки	double	10	Истина
имя автора	varchar	255	Истина
клиент	jsonb		Ложь
обновлено	timestamp	0	Истина
связанная компания	jsonb		Ложь
связанный контакт	jsonb		Ложь
сделка	jsonb		Ложь
создано	timestamp	0	Истина
статус истории	jsonb		Ложь
удалено	timestamp	0	Ложь

Сущность «Переменные» изображена в таблице 9.

Таблица 9 – Переменные

Название столбца	Тип данных	Длина	Nullable
заголовок	varchar	255	Ложь

Продолжение таблицы 9

Название столбца	Тип данных	Длина	Nullable
Обновлено	timestamp	0	Ложь
создано	timestamp	0	Ложь
тип сущности	enum		Ложь
удалено	timestamp	0	Ложь
функция преобразования	integer	10	Ложь

Сущность «Статус» изображена в таблице 10.

Таблица 10 – Статус

Название столбца	Тип данных	Длина	Nullable
id аккаунта	integer	10	Истина
задача	integer	10	Истина
имя	varchar	255	Истина
обновлено	timestamp	0	Истина
ответственные пользователи	integer	10	Истина
создано	timestamp	0	Истина
удалено	timestamp	0	Ложь
цвет	varchar	255	Истина

Сущность «Сценарий» изображена в таблице 11.

Таблица 11 – Сценарий

Название столбца	Тип данных	Длина	Nullable
id аккаунта	integer	10	Ложь
видимость сценария	jsonb		Ложь
имя	varchar	255	Ложь
обновлено	timestamp	0	Ложь
создано	timestamp	0	Ложь
удалено	timestamp	0	Истина

Сущность «Файл» изображена в таблице 12.

Таблица 12 – Файл

Название столбца	Тип данных	Длина	Nullable
id аккаунта	timestamp	0	Ложь
имя	varchar	255	Ложь
путь	varchar	255	Истина
расширение	enum		Истина
соединение	varchar	255	Истина
создано	timestamp	0	Ложь
тип	enum		Ложь
тип хранилища	enum		Ложь
удалено	timestamp	0	Истина

Продолжение таблицы 12

Название столбца	Тип данных	Длина	Nullable
яндекс ID	varchar	255	Истина

Сущность «Форма счета» изображена в таблице 13.

Таблица 13 – Форма счета

Название столбца	Тип данных	Длина	Nullable
значение	jsonb		Ложь
создано	integer	10	Ложь

Сущность «Шаблон» изображена в таблице 14.

Таблица 14 – Шаблон

Название столбца	Тип данных	Длина	Nullable
id аккаунта	integer	10	Ложь
использованная форма	bit	0	Истина
обновлено	timestamp	0	Ложь
создано	timestamp	0	Ложь
счётчик	integer	10	Ложь
удалено	timestamp	0	Истина

В ходе анализа предметной области при проектировании данной дипломной работы были выделены ключевые сущности: документ, переменные, статус, сценарий, файл, форма счета и шаблон. В таблица определенны основные атрибуты, их типы данных, допустимая длина, возможность NULL-значения.

2.2.2 Логическое проектирование базы данных

Логическое проектирование – процесс проектирования модели данных, на основе конкретной модели данных (реляционной, сетевой, иерархической).

Логическая модель базы данных представлена на рисунке 11.

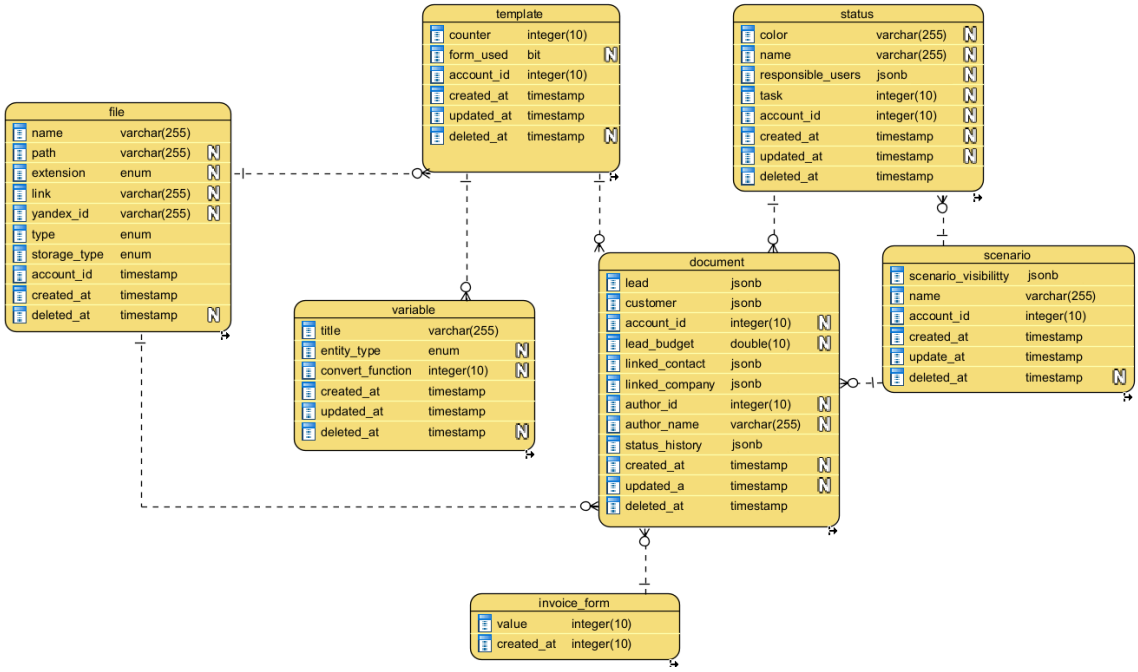


Рисунок 11 – Логическая модель базы данных

Отношение «Scenario» изображено в таблице 15.

Таблица 15 – Scenario

Название столбца	Тип данных	Длина	Nullable
account_id	integer	10	Истина
author_id	integer	10	Истина
author_name	varchar	255	Истина
created_at	timestamp	0	Истина
customer	jsonb		Ложь
deleted_at	timestamp	0	Ложь
lead	jsonb		Ложь
lead_budget	double	10	Истина
linked_company	jsonb		Ложь
linked_contact	jsonb		Ложь
status_history	jsonb		Ложь
updated_a	timestamp	0	Истина

Отношение «Variable» изображено в таблице 16.

Таблица 16 – Variable

Название столбца	Тип данных	Длина	Nullable
convert_function	integer	10	Истина
created_at	timestamp	0	Ложь
deleted_at	timestamp	0	Истина
entity_type	enum		Истина
title	varchar	255	Ложь

Продолжение таблицы 16

Название столбца	Тип данных	Длина	Nullable
updated_at	timestamp	0	Ложь

Отношение «Status» изображено в таблице 17.

Таблица 17 – Status

Название столбца	Тип данных	Длина	Nullable
account_id	integer	10	Истина
color	varchar	255	Истина
created_at	timestamp	0	Истина
deleted_at	timestamp	0	Ложь
name	varchar	255	Истина
responsible_users	integer	10	Истина
task	integer	10	Истина
updated_at	timestamp	0	Истина

Отношение «Scenario» изображено в таблице 18.

Таблица 18 – Scenario

Название столбца	Тип данных	Длина	Nullable
account_id	integer	10	Ложь
created_at	timestamp	0	Ложь
deleted_at	timestamp	0	Истина
name	varchar	255	Ложь
scenario_visibilitty	jsonb		Ложь
update_at	timestamp	0	Ложь

Отношение «File» изображено в таблице 19.

Таблица 19 – File

Название столбца	Тип данных	Длина	Nullable
account_id	timestamp	0	Ложь
created_at	timestamp	0	Ложь
deleted_at	timestamp	0	Истина
extension	enum		Истина
link	varchar	255	Истина
name	varchar	255	Ложь
path	varchar	255	Истина
storage_type	enum		Ложь
type	enum		Ложь
yandex_id	varchar	255	Истина

Отношение «Invoice_form» изображено в таблице 20.

Таблица 20 – Invoice_form

Название столбца	Тип данных	Длина	Nullable
created_at	integer	10	Ложь
value	jsonb		Ложь

Отношение «Template» изображено в таблице 21.

Таблица 21 – Template

Название столбца	Тип данных	Длина	Nullable
account_id	integer	10	Ложь
counter	integer	10	Ложь
created_at	timestamp	0	Ложь
deleted_at	timestamp	0	Истина
form_used	bit	0	Истина
updated_at	timestamp	0	Ложь

Логическая модель в данном разделе является уточнением концептуальной модели в разделе 2.2.1. определены таблицы document, variable, status, scenario, file, invoice_form, template. Для каждой таблицы указаны названия столбцов, их типы данных, допустимая длина, возможность NULL-значения.

2.2.3 Физическое проектирование базы данных

Физическое проектирование – создание описание конкретной базы данных, размещающейся в СУБД.

Физическая модель базы данных представлена на рисунке 12.

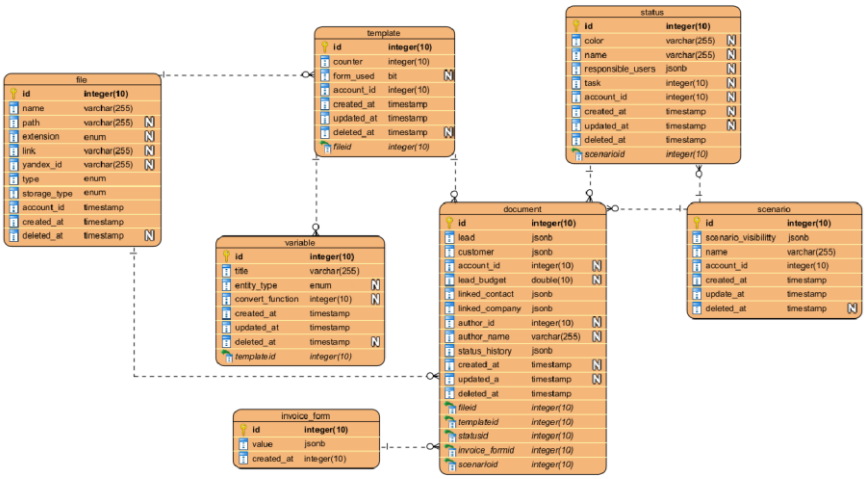


Рисунок 12 – Физическая модель базы данных

Таблица «Document» представлена в таблице 22.

Таблица 22 – Document

Название столбца	Тип данных	Длина	Nullable	Primary Key
id	integer	10	Ложь	Истина
account_id	integer	10	Истина	Ложь
author_id	integer	10	Истина	Ложь
author_name	varchar	255	Истина	Ложь
created_at	timestamp	0	Истина	Ложь
customer	jsonb		Ложь	Ложь
deleted_at	timestamp	0	Ложь	Ложь
lead	jsonb		Ложь	Ложь
lead_budget	double	10	Истина	Ложь
linked_company	jsonb		Ложь	Ложь
linked_contact	jsonb		Ложь	Ложь
status_history	jsonb		Ложь	Ложь
updated_a	timestamp	0	Истина	Ложь
fileid	integer	10	Ложь	Ложь
invoice_formid	integer	10	Ложь	Ложь
scenarioid	integer	10	Ложь	Ложь
templateid	integer	10	Ложь	Ложь

Таблица «Variable» представлена в таблице 23.

Таблица 23 – Variable

Название столбца	Тип данных	Длина	Nullable	Primary Key
id	integer	10	Ложь	Истина
convert_function	integer	10	Истина	Ложь
created_at	timestamp	0	Ложь	Ложь
deleted_at	timestamp	0	Истина	Ложь
entity_type	enum		Истина	Ложь
title	varchar	255	Ложь	Ложь
updated_at	timestamp	0	Ложь	Ложь
templateid	integer	10	Ложь	Ложь

Таблица «Status» представлена в таблице 24.

Таблица 24 – Status

Название столбца	Тип данных	Длина	Nullable	Primary Key
id	integer	10	Ложь	Истина
account_id	integer	10	Истина	Ложь
color	varchar	255	Истина	Ложь
created_at	timestamp	0	Истина	Ложь
deleted_at	timestamp	0	Ложь	Ложь
name	varchar	255	Истина	Ложь
responsible_users	integer	10	Истина	Ложь
task	integer	10	Истина	Ложь

Продолжение таблицы 24

Название столбца	Тип данных	Длина	Nullable	Primary Key
updated_at	timestamp	0	Истина	Ложь
scenarioid	integer	10	Ложь	Ложь

Таблица «Scenario» представлена в таблице 25.

Таблица 25 – Scenario

Название столбца	Тип данных	Длина	Nullable	Primary Key
id	integer	10	Ложь	Истина
account_id	integer	10	Ложь	Ложь
created_at	timestamp	0	Ложь	Ложь
deleted_at	timestamp	0	Истина	Ложь
name	varchar	255	Ложь	Ложь
scenario_visibilitty	jsonb		Ложь	Ложь
update_at	timestamp	0	Ложь	Ложь

Таблица «File» представлена в таблице 26.

Таблица 26 – File

Название столбца	Тип данных	Длина	Nullable	Primary Key
id	integer	10	Ложь	Истина
account_id	timestamp	0	Ложь	Ложь
created_at	timestamp	0	Ложь	Ложь
deleted_at	timestamp	0	Истина	Ложь
extension	enum		Истина	Ложь
link	varchar	255	Истина	Ложь
name	varchar	255	Ложь	Ложь
path	varchar	255	Истина	Ложь
storage_type	enum		Ложь	Ложь
type	enum		Ложь	Ложь
yandex_id	varchar	255	Истина	Ложь

Таблица «Invoice_form» представлена в таблице 27.

Таблица 27 – Invoice_form

Название столбца	Тип данных	Длина	Nullable	Primary Key
id	integer	10	Ложь	Истина
created_at	integer	10	Ложь	Ложь
value	jsonb		Ложь	Ложь

Таблица «Template» представлена в таблице 28.

Таблица 28 – Template

Название столбца	Тип данных	Длина	Nullable	Primary Key
id	integer	10	Ложь	Истина
account_id	integer	10	Ложь	Ложь
counter	integer	10	Ложь	Ложь
created_at	timestamp	0	Ложь	Ложь
deleted_at	timestamp	0	Истина	Ложь
form_used	bit	0	Истина	Ложь
updated_at	timestamp	0	Ложь	Ложь
fileid	integer	10	Ложь	Ложь

Физическая модель данных включает в себя конкретные детали и является продолжением логической модели, отражая то, как данные будут хранится в конкретной СУБД.

Выделены следующие элементы: первичные ключи, внешние ключи, индексы и то как связаны между собой таблицы.

Отношения «Document» представлена в таблице 29.

Таблица 29 – Отношения Document

Отношения	From	To
1:N	invoice_form	document
1:N	file	document
1:N	status	document
1:N	scenario	document
1:N	template	document

Отношения «Variable» представлена в таблице 30.

Таблица 30 – Отношения Variable

Отношения	From	To
1:N	template	variable

Отношения «Status» представлена в таблице 31.

Таблица 31 – Отношения Status

Отношения	From	To
1:N	status	document
1:N	scenario	status

Отношения «Scenario» представлена в таблице 32.

Таблица 32 – Отношения «Scenario»

Отношения	From	To
1:N	scenario	status
1:N	scenario	document

Отношения «File» представлена в таблице 33.

Таблица 33 – Отношения File

Отношения	From	To
1:N	file	document
1:N	file	template

Отношения «Invoice_form» представлена в таблице 34.

Таблица 34 – Отношения Invoice_form

Отношения	From	To
1:N	invoice_form	document

Отношения «Template» представлена в таблице 35.

Таблица 35 – Отношения template

Отношения	From	To
1:N	template	variable
1:N	template	document
1:N	file	template

В таблицах 8 – 35 определены связи между сущностями базы данных, отражая реальные зависимости объектов. В данной дипломной работе была спроектирована БД для хранения в PostgreSQL.

Полная версия БД реализованная в виджете представлена в приложении Б.

3 Реализация виджета для автоматизации генерации и согласования документов в CRM-системе

3.1 Реализация пользовательского интерфейса

В ходе выполнения выпускной квалификационной работы был выполнен следующий пользовательский интерфейс.

На рисунке 13 можно увидеть, как выглядит группа полей. В данной группе должна быть возможность выбрать из списка шаблон для генерации или загрузить документ с носителя.

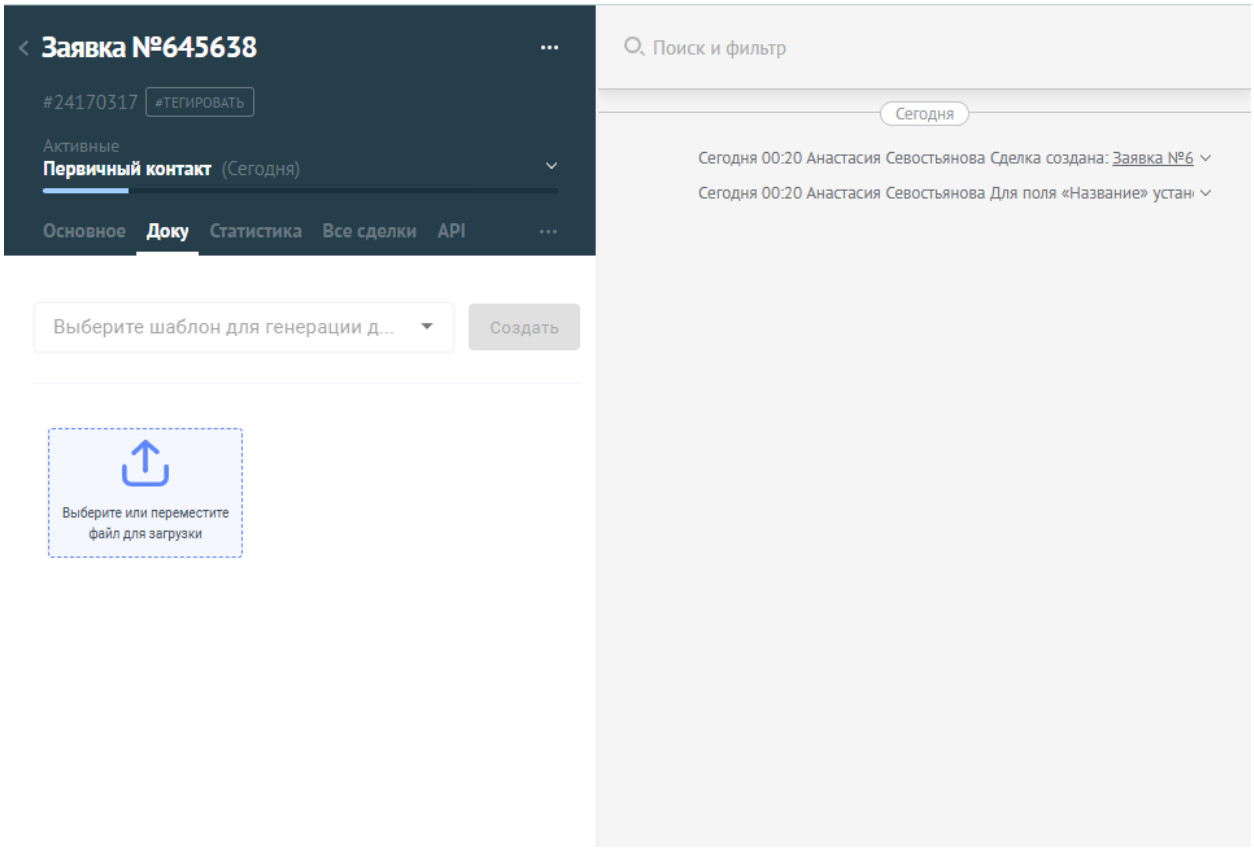


Рисунок 13 – Интерфейс группы «Доку»

При выборе списка шаблона синей иконкой помечены документы с формой счета (рисунок 14). В таком шаблоне есть возможность добавить товары/услуги. Серой иконкой помечены документы без формы счета. Они генерируются сразу.

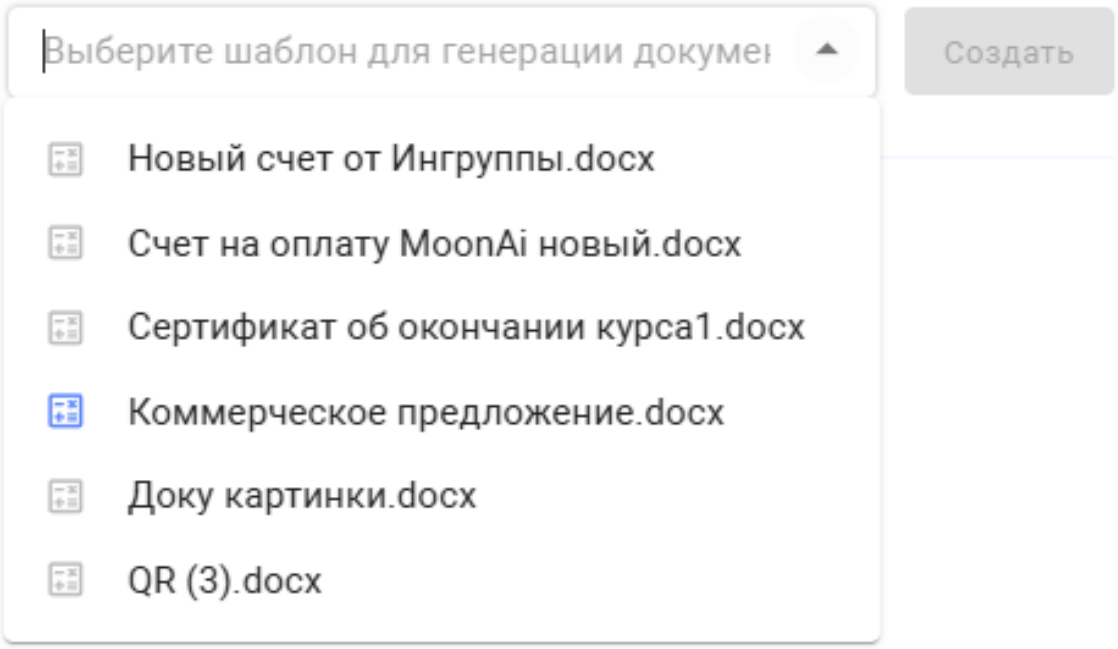


Рисунок 14 – Список шаблонов

На рисунке 15 изображена реализованная форма счета. В данном окне есть возможность редактировать дату, заказчика и исполнителя, реквизиты, а также есть возможность применить скидку, учитывать НДС.

Для работы с товарами и услугами предусмотрена функция добавления позиций через выпадающий список. Для быстрого поиска нужной позиции предусмотрена поисковая строка с динамическим отображением позиций по мере ввода текста.

Каждую позицию можно редактировать непосредственно в форме – изменять наименование, количество, цену и скидку. При обновлении документа или при создании нового документа на основе текущего измененные данные в форме счета будут использоваться повторно.

В правом нижнем углу находится блок с автоматическим расчетом, который включает сумму скидки, сумму без скидки, количество товаров, и итоговую сумму к оплате.

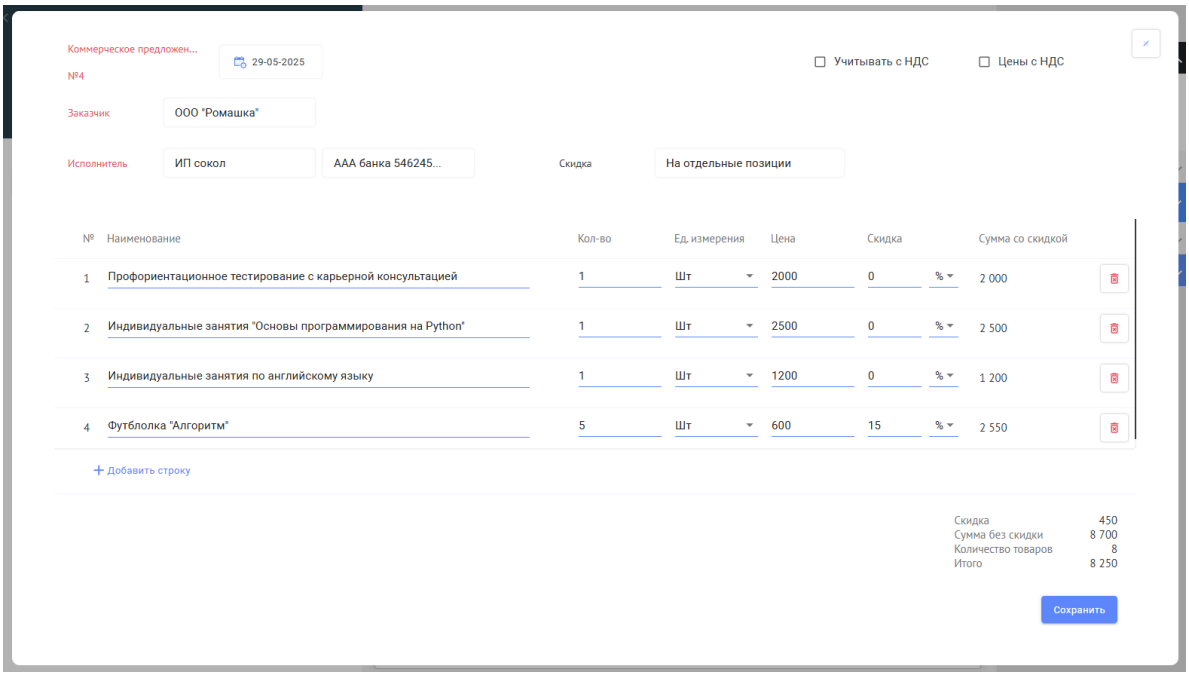


Рисунок 15 – Интерфейс формы счета

После генерации документа, он становится доступным в группе «Доку» (рисунок 16). Так же на этом рисунке видно отображение доступных функций над документами.

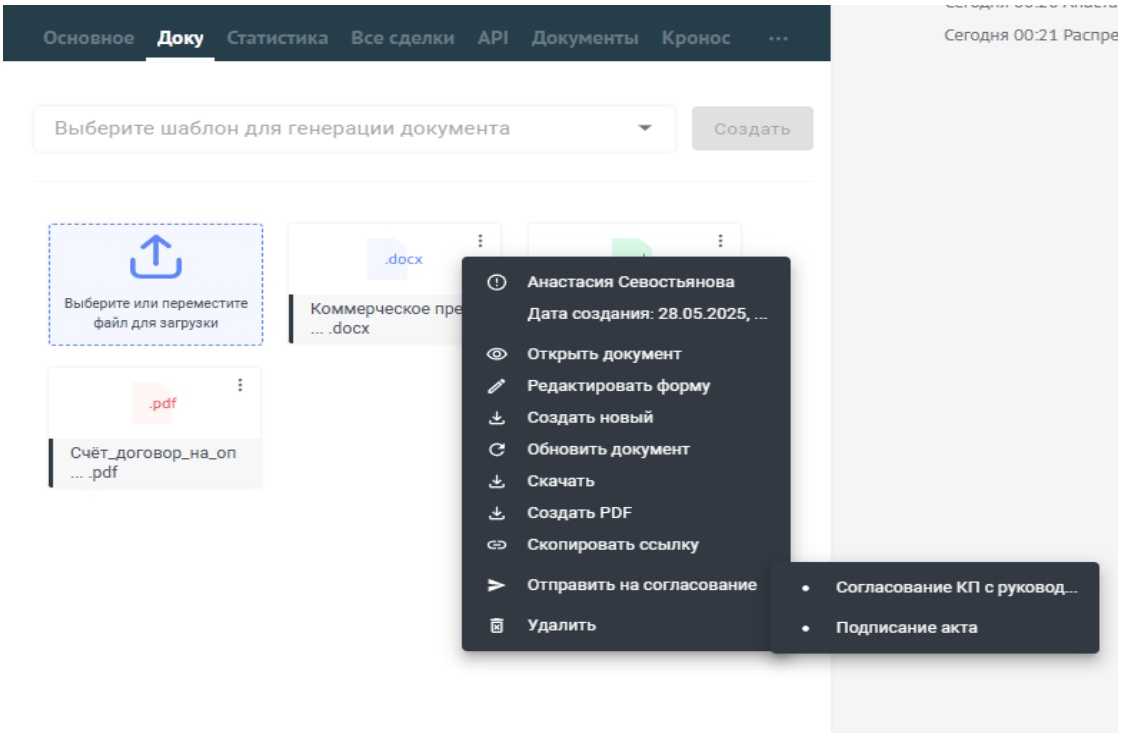


Рисунок 16 – Интерфейс доступных функций

Доступные действия над документом:

– открыть документ;
– редактировать форму;
– создать новый;
– обновить документ;
– скачать;
– создать PDF;
– скопировать ссылку;
– отправить на согласование;
– удалить.

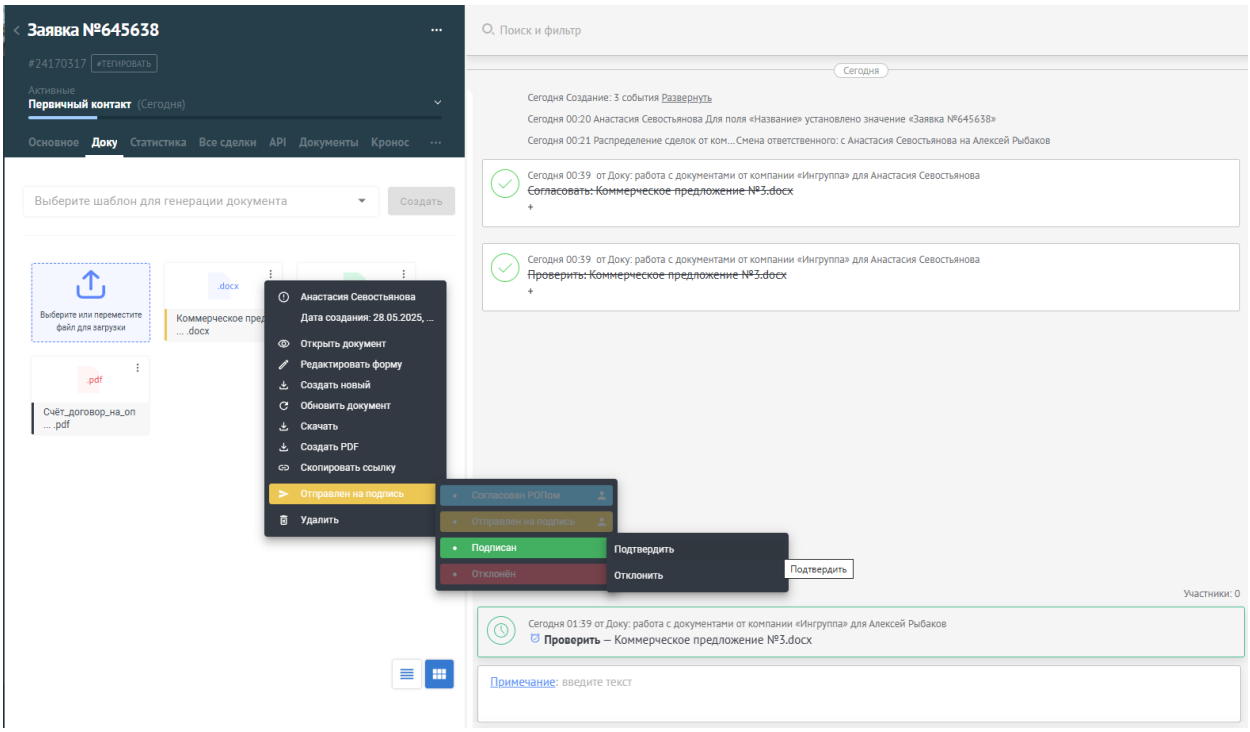


Рисунок 17 – Процесс согласования документа

На рисунке 17 виден процесс согласования документа. Пользователь выбирает необходимый сценарий, это может быть как проверка документов руководителем, так и согласование бюджета сделки, согласование машины с логистом и многое другое. На каждом этапе на ответственного пользователя

ставится задача. Все действия пользователей фиксируются в истории CRM-системы.

Кроме того, в виджете реализован так же раздел «Настройки», реестр документов реестр товаров и реестр юр. лиц, доступные администратору. В разделе «Настройки» (рисунок 18) есть возможность провести авторизацию с хранилищем Яндекс или хранилищем Google. Добавить и настроить шаблоны документа. Добавить и настроить сценарии.

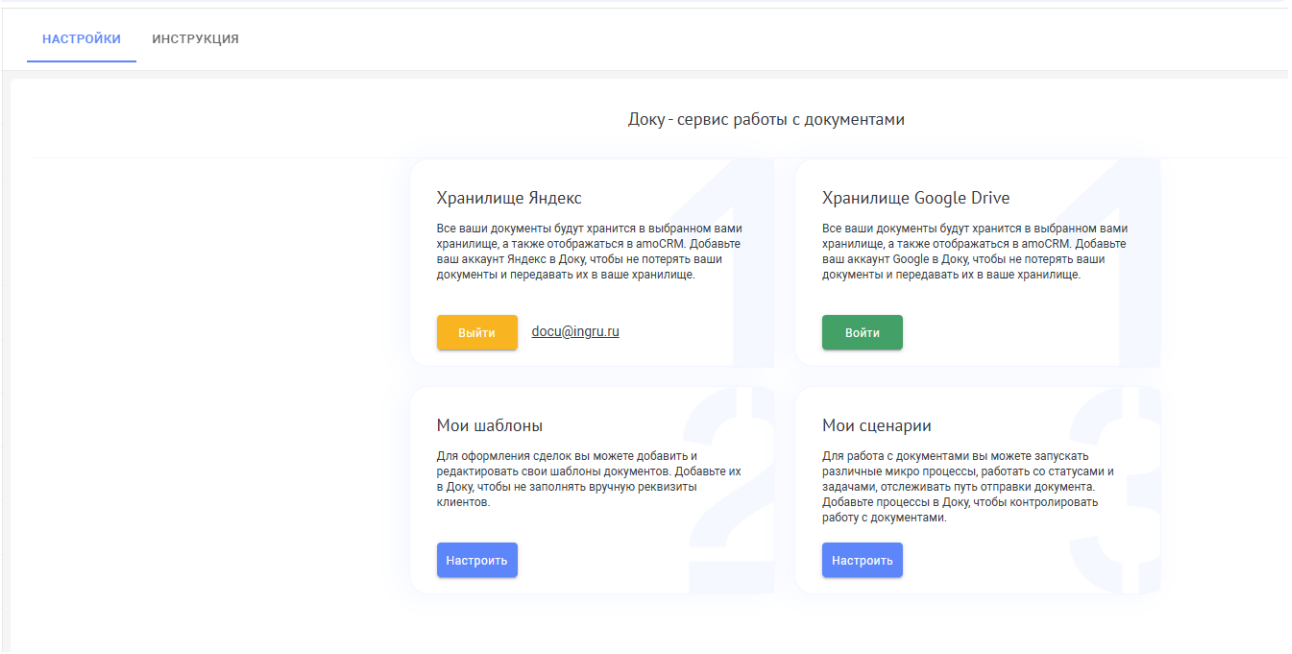


Рисунок 18 – Раздел настройки

Раздел «Мои шаблоны» позволяет добавлять шаблоны документов и работать с ними (рисунок 19). Здесь реализован следующий функционал:

– сопоставление переменных с полями amoCRM;
– настройка нумерации документов – задание начального счетчика;
– возможность установить использование формы счета.

В тексте шаблона переменные заключаются в двойные решетки (##переменная##). При загрузке шаблона система автоматически распознает все переменные и отображает их в первой колонке таблицы, во второй колонке пользователь выбирает соответствующее поле и amoCRM. При генерации

документа значение из выбранных полей автоматически подставляются вместо переменных.

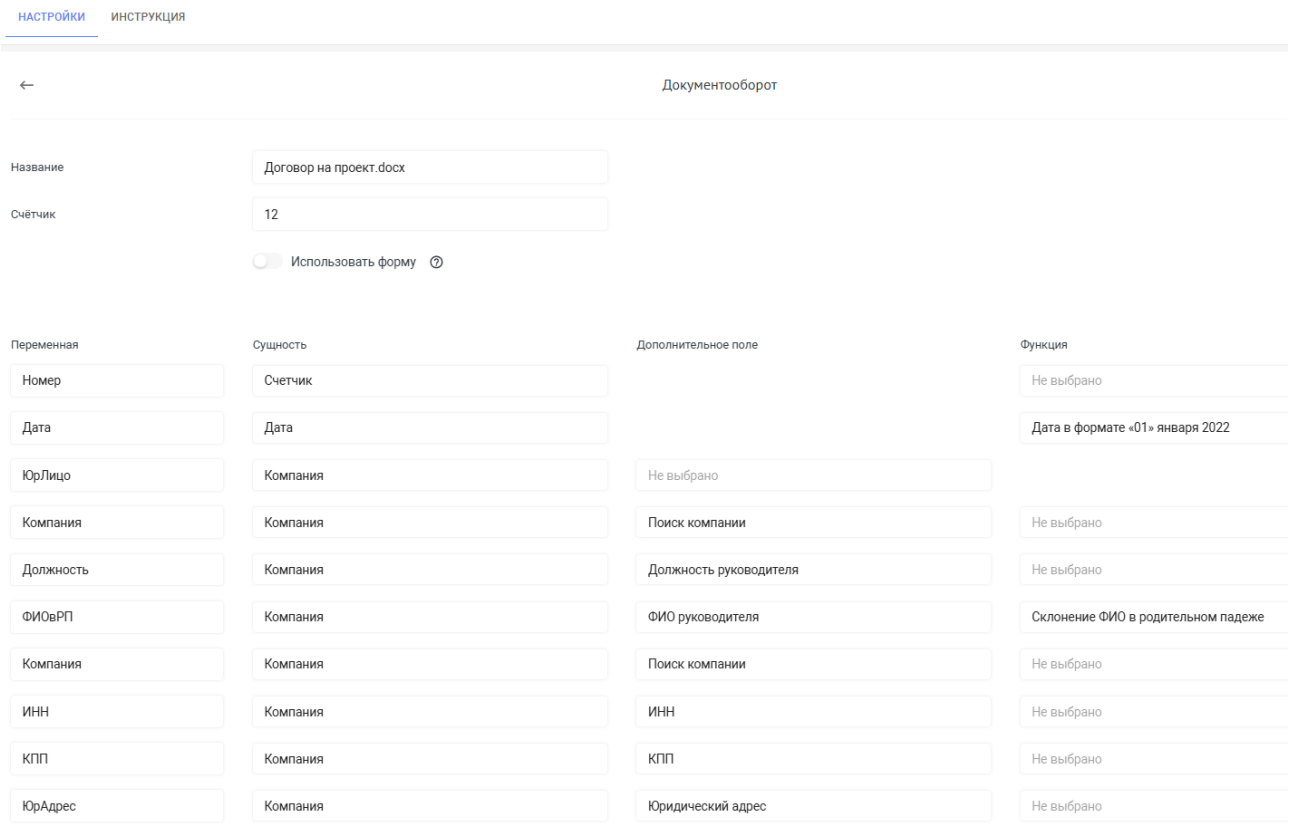


Рисунок 19 – Сопоставление переменных в шаблоне

В разделе «Мои сценарии» можно выбрать добавленный сценарий и изменить/добавить этапы согласования и ответственное лицо (рисунок 20).

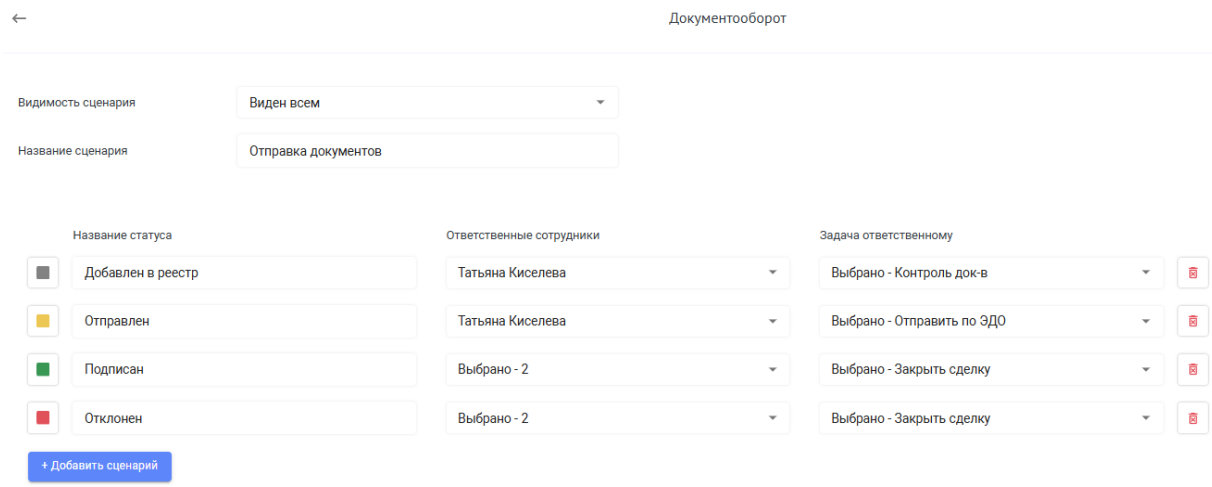


Рисунок 20 – Раздел «Мои Сценарии»

Реестр документов хранит в себе информацию о всех добавленных документах в amoCRM (рисунок 21).

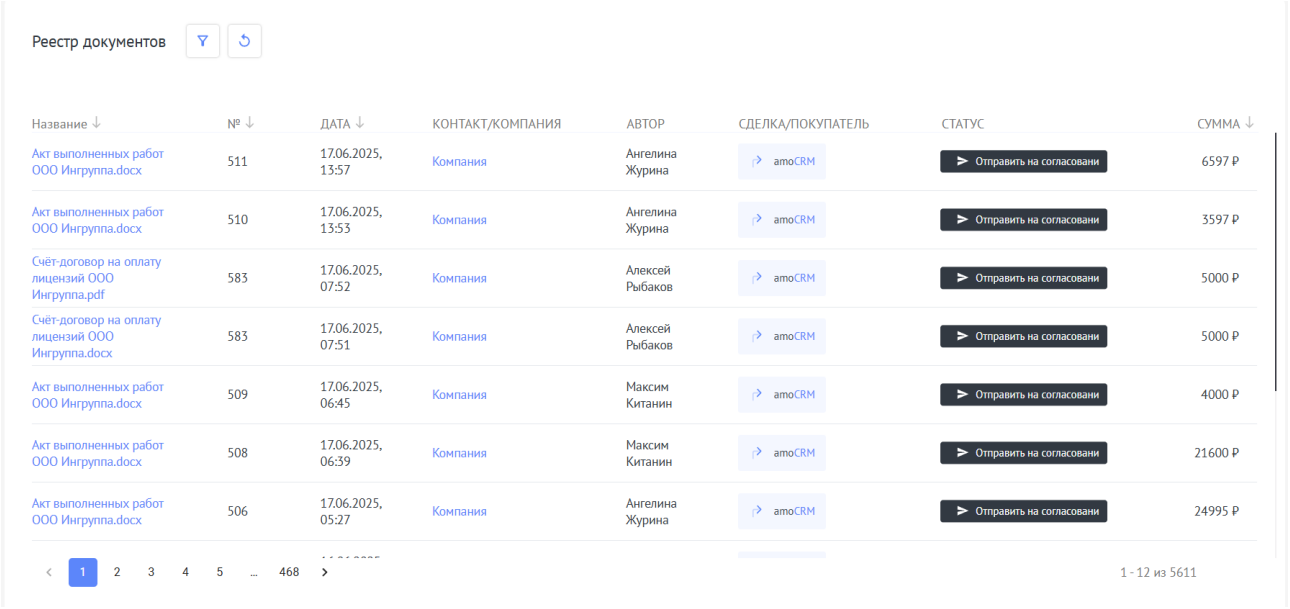


Рисунок 21 – Реестр документов

В реестре документов можно увидеть информацию о том, что за шаблон использовался, кто и когда сгенерировал документ, к какой сделке он прикреплен, а также, какой сценарий используется на документе.

Помимо этого, в реестре можно использовать фильтр и отсортировать необходимые документы (рисунок 22).

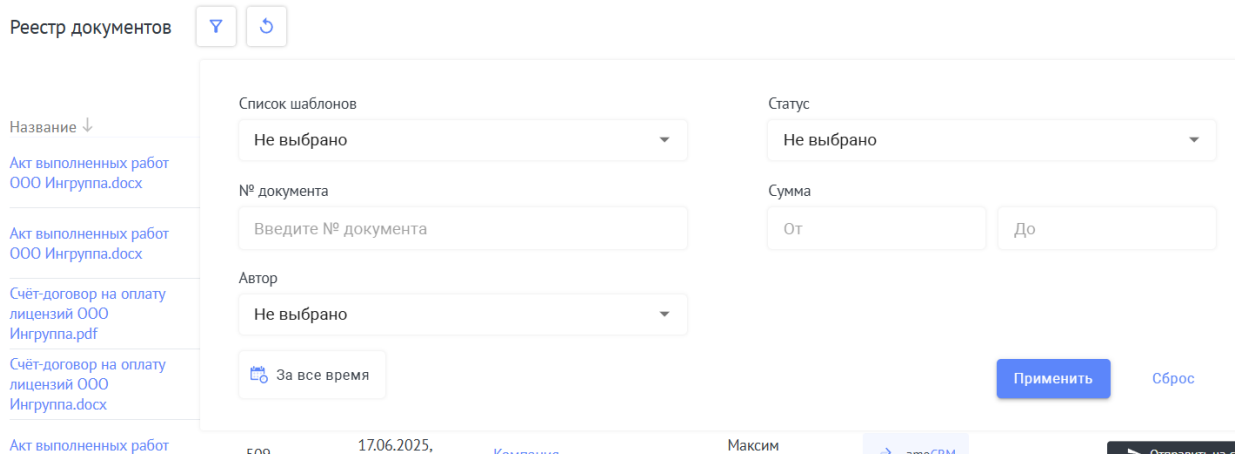


Рисунок 22 – Фильтр реестра

Раздел товары включает в себя товары и услуги, которые можно использовать в форме счета документа (рисунок 23).

Рисунок 23 – Реестр товаров

Есть форма добавления товаров (рисунок 24).

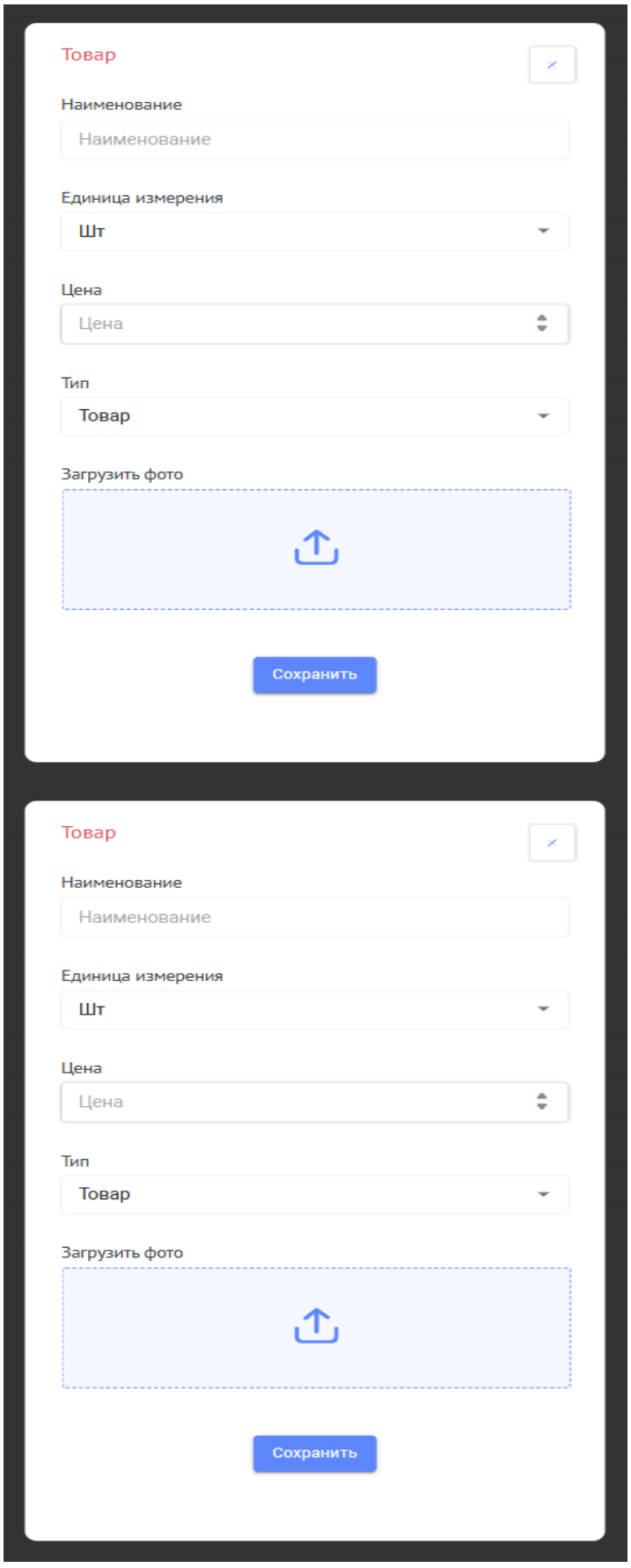


Рисунок 24 – Форма добавления товаров

И так же реестр юр. лиц с информацией о юр. лицах компании (рисунок 25), из-под которых можно выставить счет и информация о реквизитах (рисунок 26).

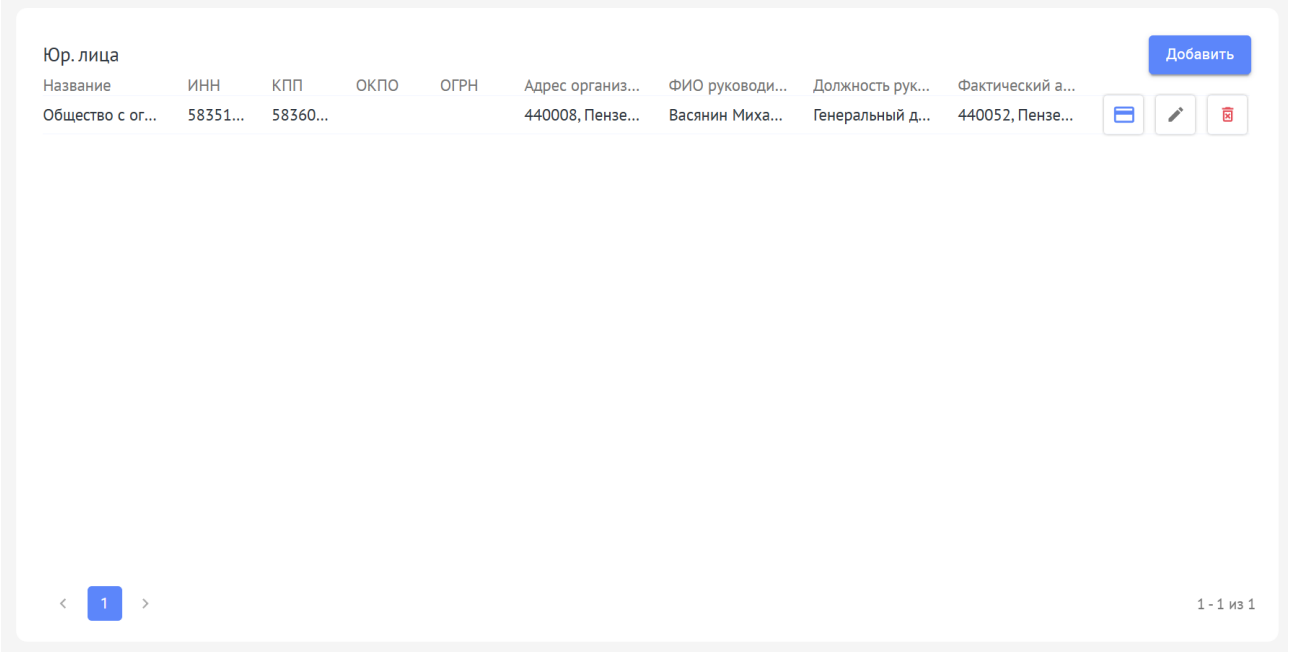


Рисунок 25 – Юридические лица компании

Информацию о юридических лицах и реквизитах можно изменять.

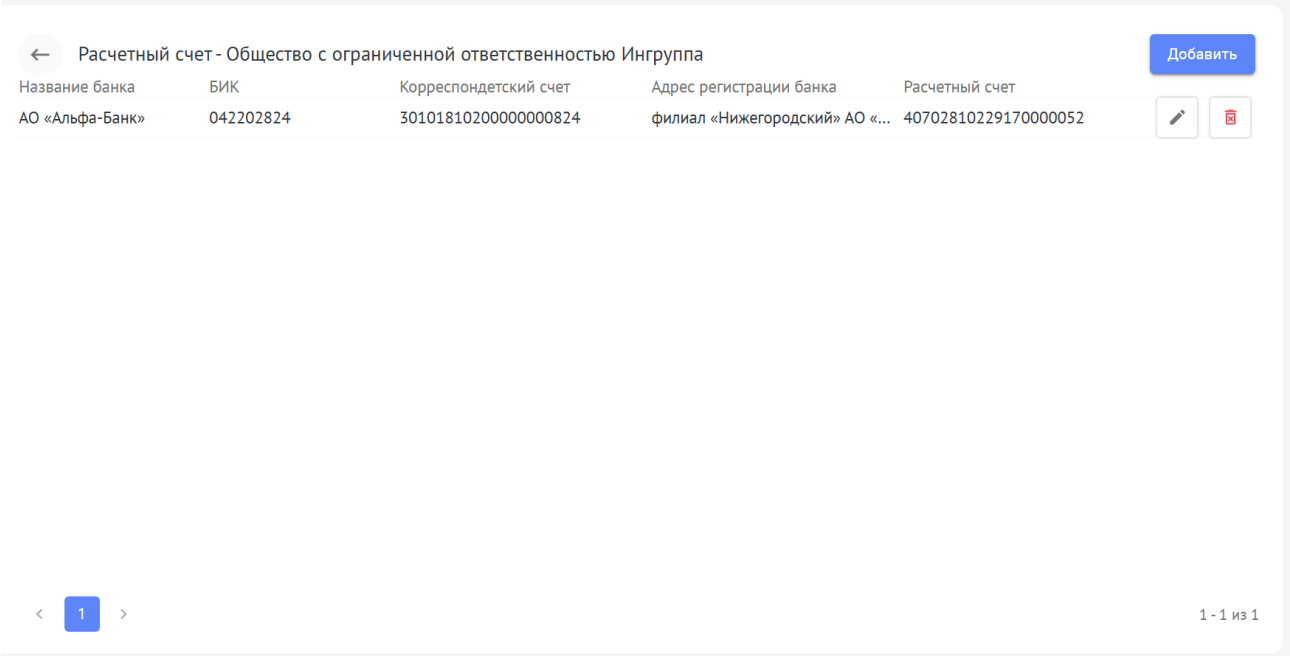


Рисунок 26 – Реквизиты

Функционал данного виджета закрывает многие сценарии использования в различных компаниях. Данное решение является достаточно универсальным и не требует долгого обучения сотрудников. Весь интерфейс интуитивно понятен пользователю.

3.2 Реализация функционала

3.2.1 Реализация генерации документа

Метод generateFromTemplate (рисунок 27) представляет из себя единую точку входа для создания или обновления документа [15]. Ему приходят данные info, где указаны идентификатор шаблона, флаг обновления refresh, данные для формы счета и прочие параметра.

```
async generateFromTemplate(req: any, info: GenerateFromTemplateDto): Promise<Document> {
  /** Если происходит обновление (повторная генерация) существующего документа,
   *  добавляем в объект info объект документа */
  if (info.hasOwnProperty('refresh') && info.refresh) {
  if (info.hasOwnProperty('documentId') && info.documentId) {
    info.document = await this.findOneFull(info.documentId)
  }
}
```

Рисунок 27 – Фрагмент кода generateFromTemplate

Если приходит флаг обновления refresh и передан documentId, метод findOneFull находит информацию о документе и загружает его в info.document, чтобы изменить уже существующую запись вместо создания новой.

Далее происходит генерация документа при помощи обращения к templateService.generateFromTemplate. Обновление документа является повторной генерацией документа, с использованием существующих данных. То есть связи шаблон, ID документа, его расположение в базе данных и на Яндекс.Диске остается прежними. Мы обновляем только переменные, и форму счета.

Если в info переданы данные формы счета Invoice_form, то создается либо новая запись формы (в случае если это первая генерация документа) через

метод InvoiceFormService.create, либо обновляется существующая через InvoiceFormService.update.

На рисунке 28 изображена диаграмма компонентов.

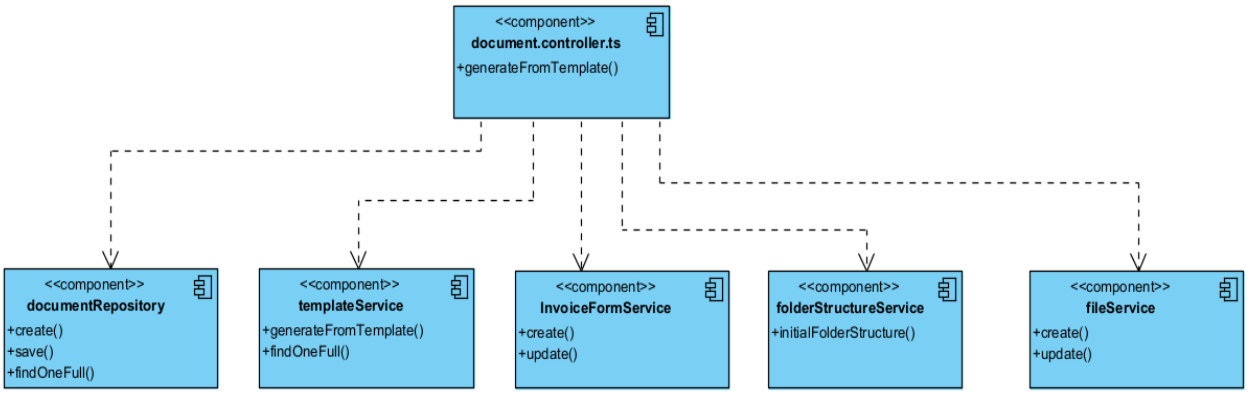


Рисунок 28 – Диаграмма компонентов

На диаграмме присутствуют следующие компоненты:

– document.controllet.ts – принимает HTTP запросы;

– documentRepository – операции над документом в БД;

– templateService – генерация документа из шаблона;

– invoiceFromService – создание/обновление записей формы счета;

– folderStructureService – создание структуры директорий в хранилище;

– fileService – сохранение и обновление файлов.

Метод generateFromTemplate позволяет создать и обновить документ, учитывая переданные параметры. Сам документа обновляется путем повторной генерации с сохранением существующих метаданных, но с измененными переменными и формой счета. Диаграмма компонентов отображает взаимодействие с шаблонами, формами и файлами, что обеспечивает согласованность данных [16].

3.2.2 Реализация сценариев

Работа со сценариями реализована по следующей логике.

Пользователь генерирует или добавляет документ, далее выбирает необходимый сценарий и запускает его. Ответственному за этап согласования пользователю ставиться системная задача. Тип задачи, как и этапы сценария настраиваются заранее администратором. Пользователь, ответственный за этап, открывает раздел «Доку», выбирает нужный файл и либо подтверждает свой этап, либо отклоняет его. При этом статус документа можно увидеть по цвету угла документа (рисунок 29).

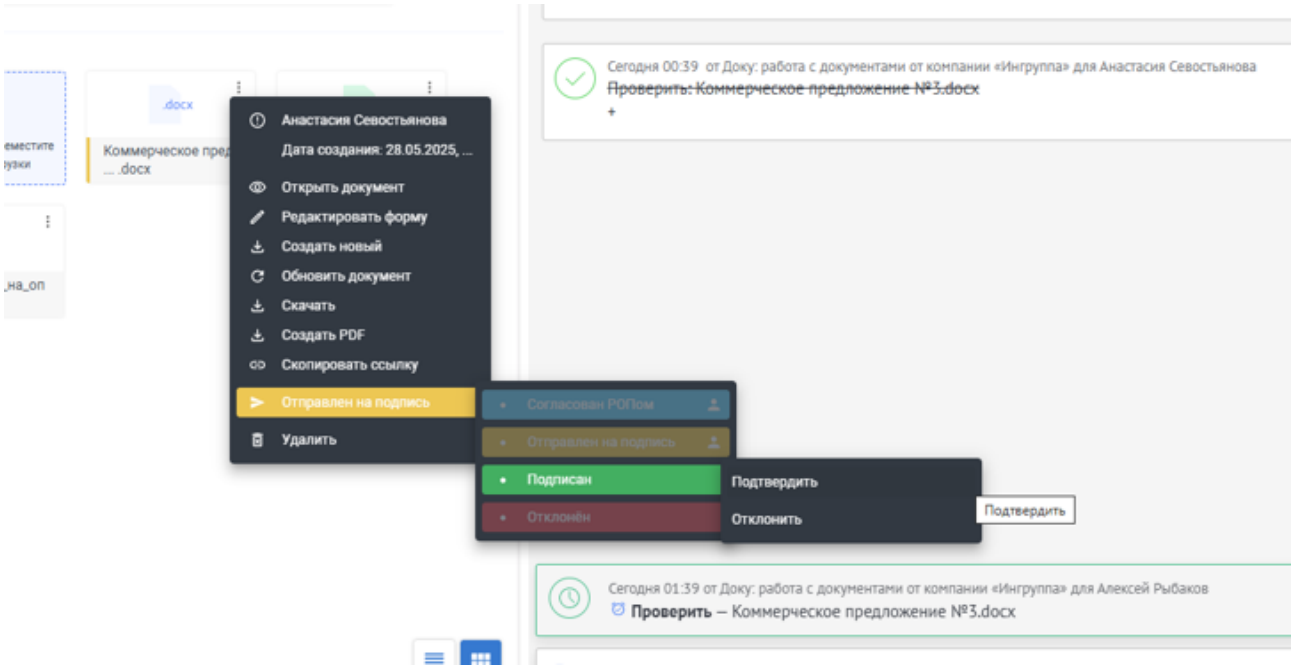


Рисунок 29 – Работа со сценариями

В dto хранится информация об объекте: document_id, scenario_id, user_name, lead_id. Сервер запрашивает документ а базе данных при помощи функции findOneFull(document_id). Если у документа отсутствует statusHistory, то сервер его формирует и заполняет в базе данных и создает на ответственного в амоCRM задачу. Обновленный документ при успехе сервер возвращает клиенту с кодом 200 и отображением запущенного сценария. После этого система автоматически переводить документ на следующий этап согласования, если текущий этап был подтвержден. В случае отклонения документа ответственным пользователем, статус изменяется на «Отклонен», и все следующие этапы документа блокируются.

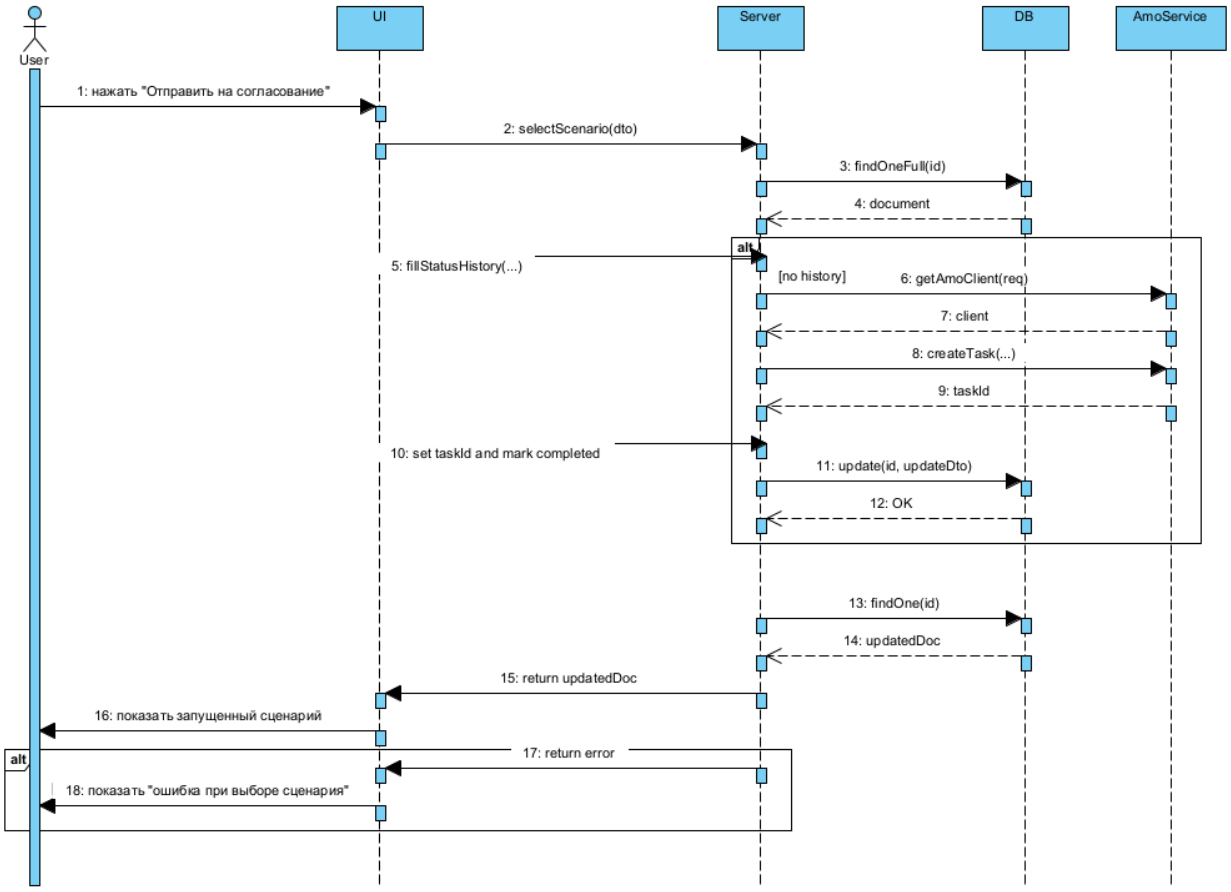


Рисунок 30 – Диаграмма последовательности

На основе логики обмена данными была реализована диаграмма последовательности, изображенная на рисунке 30.

3.2.3 Реализация интеграции с Яндексом

Для хранения файлов используется интеграция с Яндекс.Диском. Код представленный в листbнге 1, реализует авторизацию в Яндексе для дальнейшей возможности сохранения на Яндекс.Диске документов.

Листинг 1 – Авторизация в Яндекс

```
import React, { useEffect, useState } from 'react';
import './YandexAuth.css';

export const YandexAuth = ({clientID, Uri, onSuccess }) => {
    const auth = () => {
const url = new URL(“https://oauth.yandex.ru/authorize”);
        url.searchParams.append('response_type', 'token');
        url.searchParams.append('client_id', clientID);
        url.searchParams.append('Uri', Uri);
```

```
window.open(url, '_blank', 'width=500,height=600');
    };
```

Продолжение листинга 1

```
    window.open(url, '_blank', 'width=500,height=600');
    };
useEffect(() => {
        const handleMessage = (e) => {
if (e.origin === window.location.origin && e.data?.token) {
                onSuccess(e.data.token);
            }
        };
        window.addEventListener('message', handleMessage);
        return () => window.removeEventListener('message',
handleMessage);
    }, [onSuccess]);
    return <button onClick = {auth}>Войти</button>;
};
```

В листинге 1 описан компонент React «YandexAuth», реализующий OAuth-авторизацию через Яндекс.

На вход поступает три аргумента: clientID является идентификатором приложения, зарегистрированного в Яндекс, Url содержит url на который перенаправляет после авторизации, onSuccess содержит ответ, вызываемый при успешной авторизации.

Компонент содержит функции: auth и useEffect.

Функция auth создает url для реализации запроса авторизации и открывает этот url в новом окне при помощи window.open.

Функция useEffect подписывает на события message в созданном окне. При успешной авторизации вызывается onSuccess с полученным токеном.

Компонент YandexAuth обеспечивает безопасную OAuth-аутентификацию через протокол Яндекс. OAuth, соответсвуя современным стандартам авторизации внешних сервисов. Решение демонстрирует корректную работу с API Яндекс, включая получение токена доступа и обработку ответа сервера. Данный механизм позволяет в дальнейшем реализовать возможность работы с облачным хранилищем Яндекс, что сильно расширяет функционал в amoCRM по работе с документами.

3.2.4 Развертывание системы

Диаграммы развертывания используются для визуализации аппаратных процессоров/узлов/устройств системы, каналов связи между ними и размещения программных файлов на этом аппаратном обеспечении [17].

Общие программные средства были реализованы на языке TypeSrcript в среде Visual Studio с использованием современных подходов к проектированию архитектуры.

Основные компоненты программе отражены на рисунке 31.

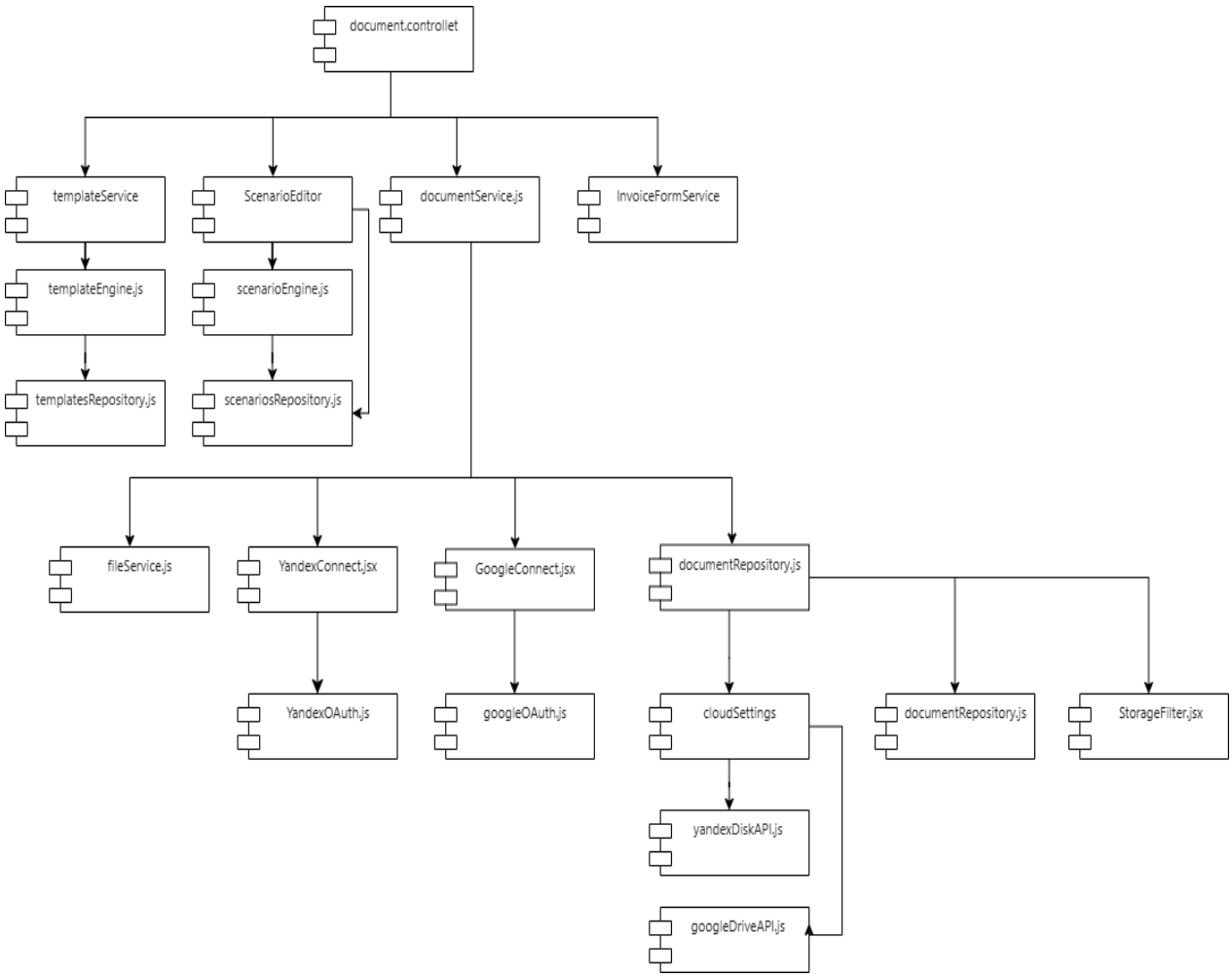


Рисунок 31 – Диаграмма компонентов

В данной дипломной работе были реализованы следующие части:

– document.controllet.ts – принимает HTTP запросы;

– documentRepository – операции над документом в БД;

– templateService – генерация документа из шаблона;

– invoiceFromService – создание/обновление записей формы счета;

– fileService – сохранение и обновление файлов.

Визуализации физического развертывания системы отображена на рисунке 32 в виде диаграммы развертывания.

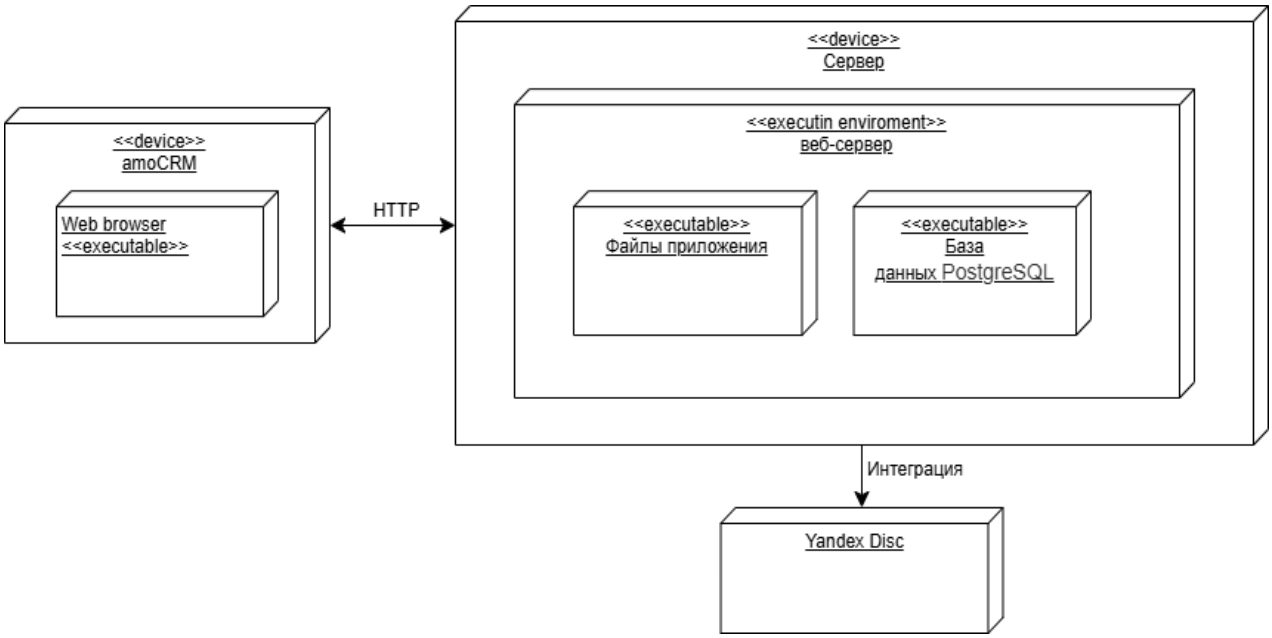


Рисунок 32 – Диаграмма развертывания

Система amoCRM является веб-сервисом с которым интегрируется виджет «Доку». Команды приходят из интерфейса клиента на сервер, который вместе с базой данных развернут на сервере Рег.Ру. Помимо этого, приложение взаимодействует и с внешним сервисом Яндекс.Диск куда сохраняет для пользователя прикрепленные или сгенерированные документы.

Архитектура системы включает компоненты для обработки HTTP-запросов, работы с базой данных, генерации документов и управления файлами. Диаграмма развертывания отражает физическую структуру системы, включая сервер Рег.Ру и взаимодействие с внешними сервисами. Реализация на TypeScript с использованием современных подходов обеспечила модульность и масштабируемость решения.

4 Тестирование виджета для автоматизации генерации и согласования документов в CRM-системе

4.1 Тестирование черным ящиком

Тестирование является заключающим этапом разработки. Оно проверяет соответствие реальных и ожидаемых результатов поведения программы и обеспечивает стабильность работы готового продукта [18].

Тестирование черным ящиком один из наиболее популярных методов оценки качества и соответствия ПО [19]. Оно происходит путем тестирования продукта через интерфейсную часть вне зависимости от внутренней архитектуры [20].

Важным аспектом при тестировании черным ящиком является рассмотрение всевозможных путей пользователя при использовании системы.

Результатом успешного выполнения программы является сгенерированный документ, согласно заполненным данным в виджете. Рассмотрим какие состояния могут быть у виджета на рисунке 33.

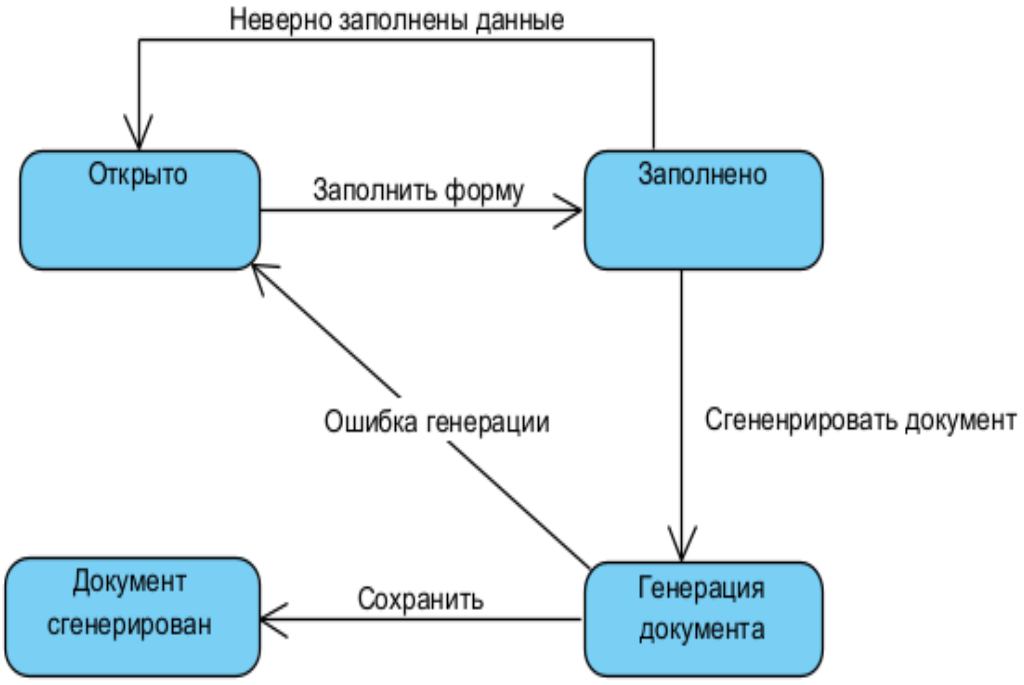


Рисунок 33 – Диаграмма состояний

Сам виджет существует в интерфейсе CRM системы. И для пользователя его работа начинается в момент запуска виджеты через кнопку генерации. При генерации документа с формой счета открывается диалоговое окно. Далее оно ожидает заполнения данных в форме. После заполнения пользователь нажимает кнопку «Сохранить» и происходит генерация документа. После генерации документа виджет закрывает диалоговое окно. Документ отображается в группе «Доку» и доступен к просмотру на Яндекс.Диске.

Наиболее уязвимым в работе виджета является форма ввода данных при формировании счет-договора. Оно требуется заполнения некоторых данных вручную.

Проверим реакцию программы на пользовательский ввод. При некорректных значениях должно появится оповещение.

Ожидаем следующую реакцию программы на пользовательский ввод:

- программа должна сообщить о некорректной записи заказчика;
- программа должна сообщить о некорректной записи исполнителя;
- программа должна сообщить об некорректном применении скидки;
- программа должна сообщения об некорректной записи товара;
- программа должна сообщить об успешно созданном документе.

Рассмотрим причины и следствия составленные в таблице 36. В причинах укажем события, которые ведут к успешному результату (генерация документа). В следствиях укажем события, которые должны произойти при нарушении одной из причин.

Таблица 36 – Причины и следствия

П1	Корректно прописан заказчик
П2	Корректно прописан исполнитель
П3	Корректно прописана скидка
П4	Товар выбран
С1	Сообщение: «Не заполнено поле заказчик»
С2	Сообщение: «Не заполнено поле исполнитель»
С3	Сообщение: «Скидка прописана не корректно»
С4	Сообщение: «Товар не выбран»

Отразим взаимосвязи в функциональной диаграмме представленной на рисунке 34.

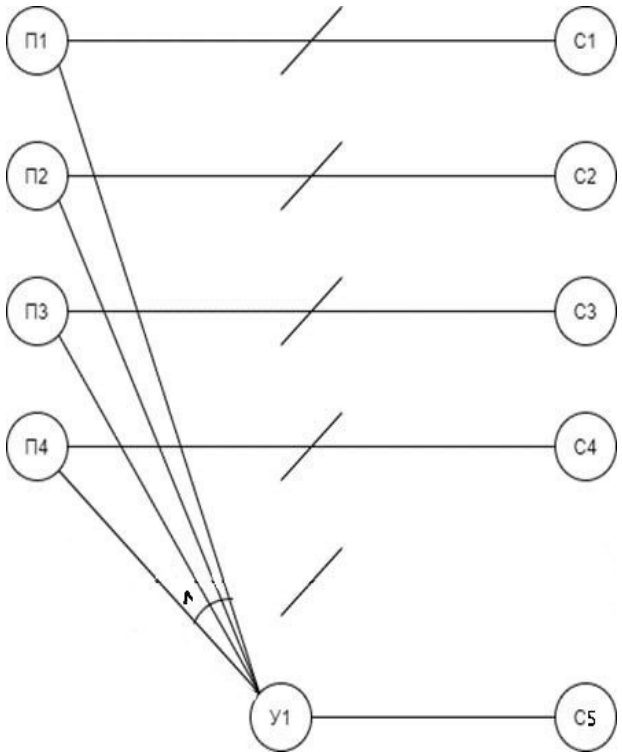


Рисунок 34 – Функциональная диаграмма

На основе диаграммы составляем таблицу решений представленную на рисунке 37. В верхней части – комбинация причин, в нижней – соответствующее следствие на каждую комбинацию причин.

Таблица 37 – Таблица решений

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
П1	0	1	0	0	0	1	1	1	1	1	0	0	1	0	0	1
П2	0	0	1	0	0	1	0	0	1	1	0	1	0	1	1	1
П3	0	0	0	1	0	0	1	0	1	0	1	0	0	1	1	1
П4	0	0	0	0	1	0	0	1	0	1	1	1	1	0	1	1
С1	1	0	1	1	1	0	0	0	0	0	1	1	0	1	1	0
С2	1	1	0	1	1	0	1	1	0	0	1	0	1	0	0	0

Продолжение таблицы 38

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
С3	1	1	1	0	1	1	0	1	0	1	0	1	1	0	0	0
С4	1	1	1	1	0	1	1	0	1	0	0	0	0	1	0	1
С5	1	1	1	1	1	1	1	1	1	1	1	0	1	1	1	1

На основе таблицы решений строим наборы тест-кейсов которые может ввести пользовател (рисунок 38).

Таблица 38 – Тест-кейсы

№	Входные данные	Выходные данные
1	Заказчик: Рога и Копыта Исполнитель: Ромашка Скидка: -10% Товар: .	Неверно записан заказчик. Неверно записан исполнитель. Скидка не может иметь отрицательное значение. Не заполнен товар
2	Заказчик: ООО «Рога и копыта» Исполнитель: Ромашка Скидка: -10% Товар: .	Неверно записан исполнитель. Скидка не может иметь отрицательное значение. Не заполнен товар
3	Заказчик: Рога и Копыта Исполнитель: ООО «Ромашка» Скидка: -10% Товар: .	Неверно записан заказчик. Скидка не может иметь отрицательное значение. Не заполнен товар
4	Заказчик: Рога и Копыта Исполнитель: Ромашка Скидка: 10% Товар: .	Неверно записан заказчик. Неверно записан исполнитель. Не заполнен товар
5	Заказчик: Рога и Копыта Исполнитель: Ромашка Скидка: -10% Товар: Хлеб ц/з, 1 шт	Неверно записан заказчик. Неверно записан исполнитель. Скидка не может иметь отрицательное значение.
6	Заказчик: ООО «Рога и копыта» Исполнитель: ООО «Ромашка» Скидка: -10% Товар: .	Скидка не может иметь отрицательное значение. Не заполнен товар
7	Заказчик: ООО «Рога и копыта» Исполнитель: Ромашка Скидка: 10% Товар: .	Неверно записан исполнитель. Не заполнен товар
8	Заказчик: Рога и Копыта Исполнитель: Ромашка Скидка: -10% Товар: Хлеб ц/з, 1 шт	Неверно записан заказчик. Неверно записан исполнитель. Скидка не может иметь отрицательное значение. Не заполнен товар
9	Заказчик: ООО «Рога и Копыта» Исполнитель: Ромашка Скидка: -10% Товар: .	Неверно записан исполнитель. Скидка не может иметь отрицательное значение.

Продолжение таблицы 38

10	Заказчик: Рога и Копыта Исполнитель: Ромашка Скидка: -10% Товар: Хлеб ц/з, 1 шт	Неверно записан заказчик. Неверно записан исполнитель.
11	Заказчик: ООО «Рога и Копыта» Исполнитель: Ромашка Скидка: 10% Товар: Хлеб ц/з, 1 шт	Неверно записан заказчик. Не заполнен товар
12	Заказчик: Рога и Копыта Исполнитель: ООО «Ромашка» Скидка: 10% Товар: .	Неверно записан заказчик. Не заполнен товар
13	Заказчик: Рога и Копыта Исполнитель: ООО «Ромашка» Скидка: 10% Товар: Хлеб ц/з, 1 шт	Неверно записан заказчик.
14	Заказчик: ООО «Рога и копыта» Исполнитель: ООО «Ромашка» Скидка: 10% Товар: Хлеб ц/з, 1 шт	Документ успешно сгенерирован

Анализ тест-кейсов показал, что для успешной генерации документов необходимо корректное заполнение всех полей, в противном случае пользователь будет уведомлен о соответсвующей ошибке.

4.2 Метрики Холстеда

Программные метрики Холстеда – это набор показателей измеряющих сложность кода с помощью формул, которые основаны на количестве операторов и операндов в коде.

В метриках программного обеспечения Халстеда компьютерная программа определяется как набор токенов, которые можно описать как операторы или операнды. Эти токены используются для анализа сложности и объёма программы. Операторы – это символы, обозначающие действия, а операнды – это объекты, на которые воздействуют операторы. Все метрики в области разработки программного обеспечения могут быть определены с помощью этих базовых символов. Эти символы называются токенами.

Подсчитывая и анализируя эти токены, метрики Халстеда дают представление о сложности, трудозатратах и качестве программного кода.

В «Метриках программного обеспечения» Холстеда:

– n1 – количество различных операторов;

– n2 – количество различных операндов;

– N1 – общее количество вхождений операторов;

– N2 – общее количество вхождений операндов.

Используя данные характеристики, можно провести расчёты по формулам.

Длина фрагмента программы, которая рассчитывается по формуле:

$$N = N_1 + N_2.$$

Словарь фрагмента программы, который рассчитывается по формуле:

$$n = n_1 + n_2.$$

Объем фрагмента программы, который рассчитывается по формуле:

$$V = N \times \log_2 n.$$

Сложность фрагмента программы, которая рассчитывается по формуле:

$$HDiff = (n_1 \div 2) \times (N_2 \div n_2).$$

Для метода «selectScenario» был произведён расчёт количественных показателей с использованием метрики Холстеда. Код процедуры приведён в листинге 1.

Листинг 2 – Код метода «selectScenario»

```
async selectScenario(req: any, selectScenarioDto:
SelectScenarioDto): Promise<Document> { const document = await
this.findOneFull(selectScenarioDto.document_id)
if(!document.status_history) {
        const updateDocumentDto = await
this.scenarioService.fillStatusHistory(
            selectScenarioDto.user_name,
            selectScenarioDto.scenario_id
        )
        if
(updateDocumentDto.status_history[1].responsible_users.length < 2
&&
updateDocumentDto.status_history[1].responsible_users.length > 0
        ) {
            let userId =
Number(updateDocumentDto.status_history[1].responsible_users[0])
            const amo = await this.amocrmService.getAmoClient(req)
```

Продолжение листинга 2

```
            if (userId === StatusResponsible.LEAD_OWNER) { const
lead = await amo.leads.get(selectScenarioDto.lead_id)
userId = lead.responsible_user_id }
            let task = await amo.tasks.create({
                entity_type: 'leads',
                entity_id: selectScenarioDto.lead_id,
                responsible_user_id: userId,
                complete_till: Math.floor(Date.now() / 1000) +
3600,
                text: document.file.name,
                task_type_id:
updateDocumentDto.status_history[1].task_type_id,
            })
            if (!task['_embedded']) {
                throw new Error(task['response']['statusText'])
            }
            task = task['_embedded']['tasks'][0]
            updateDocumentDto.status_history[0].amo_task_id =
task.id
        }
        updateDocumentDto.status_history[0].user_name =
selectScenarioDto.user_name
        updateDocumentDto.status_history[0].is_completed = true
        updateDocumentDto.scenario_id =
selectScenarioDto.scenario_id
        await this.update(selectScenarioDto.document_id,
updateDocumentDto)
        return this.findOne(selectScenarioDto.document_id)
    } else {
        return document
    }
}
```

Все операторы, используемые в методе «selectScenario» приведены в таблице 39.

Таблица 39 – Операторы метода

Название	Количество
=	12
if / else	3
await	8
return	2
throw new	1
()	7
[]	5
+	1
new	1
Создание объекта	2
Вызов методов	6

Все операнды и их количество, используемые в методе «selectScenario» представленный в таблице 40.

Таблица 40 - Операнды метода

Название	Количество
document	5
selectScenarioDto	6
updateDocumentDto	7
userId	5
amo	3
lead	3
task	6
StatusResponsible.LEAD_OWNER	1
this.scenarioService	1
this.amocrmService	1
scenario_id	2
document.file.name	1
updateDocumentDto.status_history	6
selectScenarioDto.user_name	2
selectScenarioDto.lead_id	3
selectScenarioDto.document_id	2
Date.now()	1
Math.floor	1

После подсчёта количества операторов и операндов были рассчитаны значения перечисленных выше показателей:

– длина программы (N): 104;

– словарь программы (n): 29;

– объём программы (V): 505.22;

– сложность программы (HDiff): 17.1.

Таким образом, по результатам расчёта метрик Холстеда для метода «selectScenario» можно сделать вывод, что данный метод имеет среднюю длину (N = 104), умеренный словарь (n = 29) и средний уровень информационной сложности (V = 505). Полученное значение сложности программы (HDiff = 17.1) говорит о том, что метод остаётся читаемым и поддерживаемым. Структура и логика метода соответствуют типичным требованиям для реализации бизнес-логики с использованием внешнего API и обработкой сценариев.

Тестирование методом «чёрного ящика» подтвердило, что виджет корректно обрабатывает пользовательский ввод, надёжно выявляет ошибки заполнения полей и обеспечивает формирование документа только при корректном вводе данных. Реализованные сценарии проверки позволяют значительно повысить устойчивость системы и предотвратить возможные ошибки на этапе использования. По результатам тестирования виджет продемонстрировал соответствие заявленным требованиям и готов к внедрению в эксплуатационную CRM-среду.

В целом, проведённый анализ качества кода и тестирование функциональности виджета показали его надёжность, соответствие требованиям и готовность к практическому использованию. Метод «selectScenario» реализован с оптимальной сложностью и хорошей читаемостью, а тестирование интерфейсных сценариев подтвердило устойчивость и корректность работы системы при различных вариантах пользовательского ввода. Полученные результаты демонстрируют высокий уровень готовности разработанного решения к интеграции в CRM-систему и дальнейшей эксплуатации.

Заключение

В ходе дипломного проектирования был реализован виджет для автоматизации генерации и согласования документов в CRM-системе. Данное решение позволило произвести автоматизацию процесса создания и согласования документов, сократить время обработки заявок, минимизировать ошибки при формировании документов и повысить эффективность сотрудников.

В процессе работы над дипломным проектом проведен сравнительный анализ программных продуктов-аналогов CRM систем и решений по автоматизации работы с документами. Были поставленные функциональные и не функциональные требования к виджету.

Разработка виджета велась в соответствии с поставленными требованиями при анализе предметной области проекта. Программные средства реализованы на основе клиент-серверной архитектуры и при помощи языка программирования TypeScript. База данных реализована в СУБД PostgreSQL Разработка велась в среде MS Visual Studio.

При разработке интерфейса виджета учитывался пользовательский опыт сотрудников amoCRM, что позволило сделать его интуитивно понятным и простым для обучения.

Виджет интегрирован в интерфейс amoCRM, что позволяет сотрудникам вести работу «из одного окна», не переходя в другие вкладки браузера или приложения на компьютере для работы с документами.

Был реализован следующий функционал:

- создание, удаление, редактирование сценариев;
- создание, удаление, редактирование документа;
- создание, удаление, редактирование шаблонов;
- создание, удаление, редактирование сценариев;
- генерация документа;
- просмотр, удаление, редактирование сгенерированного документа;

– редактирование формы счета;

– создание на основе сгенерированного документа новый (используя уже заполненную форму счета.

Тестирование системы подтвердило соответствие разработанного решения заявленным нефункциональным требованиям. Виджет демонстрирует стабильную работу и производительность в amoCRM.

Результаты дипломной работы были внедрены в бизнес-процессы компании «Ингруппа» и успешно применяются в её работе. Разработанный виджет интегрирован в CRM-систему компании и обеспечивает:

– автоматизацию создания и согласования документов;

– сокращение времени обработки заявок;

– минимизацию ошибок при формировании документов;

– повышение эффективности бизнес-процессов.

Кроме того, данный виджет был коммерциализирован и реализуется компанией как самостоятельный продукт под названием «Доку».

Проведенная работа показывает актуальность выбранной темя для дипломного проектирования и демонстрирует значимость решения для автоматизации бизнес-процессов.

Список использованный источников

1. CRM-системы: что это, характеристики, задачи – как выбрать CPM-систему управления клиентами для бизнеса и внедрить в работу // Блог Яндекс Практикума [Электронный ресурс]. URL: https://practicum.yandex.ru/blog/chto-takoe-crm-sistema-i-kak-ih-vybirat/ (дата обращения: 08.04.25).

2. Рейтинг CRM-систем [Электронный ресурс]. URL: https://crmrating.ru/crm-system2024 (дата обращения: 08.04.25).

3. Функциональные и нефункциональные требования: различия и особенности // Блог Яндекс Практикума [Электронный ресурс]. URL: https://practicum.yandex.ru/blog/funkcionalnye-i-nefunkcionalnye-trebovaniya/ (дата обращения: 14.04.25).

4. Петрова И. Р. Методология объектно-ориентированного моделирования. Язык UML / И. Р. Петрова, Р. Х. Фахртдинов, А. А. Сулейманова, И. О. Разживин, А. Г. Фазулзянов. – Казань, 2018. – URL: Metodichka_UML.pdf (дата обращения: 23.05.2025).

5. Буч Г. Язык UML. Руководство пользователя / Г. Буч, Д. Рамбо, И. Якобсон; пер. с англ. Н. Мухина. – 2-е изд. – М.: ДМК Пресс, 2022. – 268 с.

6. Диаграмма прецедентов (use case диаграмма) – testengineer.ru // UML диаграммы [Электронный ресурс] URL: https://testengineer.ru/use-case diagram/?ysclid=mamu42s9a4540631962 (Дата обращения 23.05.2025).

7. Функциональные и нефункциональные требования [Электронный ресурс].URL: https://intuit.ru/studies/curriculums/20891/courses/751/lecture/29036?page=3 (дата обращения: 15.04.25).

8. Требования к публичным интеграциям [Электронный ресурс]. URL: https://www.amocrm.ru/developers/content/integrations/requirements (дата обращения: 18.04.25).

9. Трехуровневая клиент-серверная архитектура [Электронный ресурс]. URL: https://sky.pro/wiki/sql/trehurovnevaya-klient-servernaya-arhitektura/ (дата обращения: 25.04.25).

10. Что такое UI, этапы проектирования пользовательского интерфейса // «Искра» [Электронный ресурс]. URL: https://iskra.st/blog/proektirovanie-polzovatelskogo-interfejsa/ (дата обращения: 02.05.25).

11. Макаров И. Д. Проектирование и разработка пользовательских интерфейсов / И. Д. Макаров. – М.: Дашков и К°, 2021. – 192 с.

12. Основы работы с базами данных. [Электронный ресурс]. URL: https://intuit.ru/studies/professional_skill_improvements/17123/courses/681/lecture/14021?page=2 (дата обращения: 06.05.25).

13. Проектирование программного обеспечения / Хабр // Хабр Проектирование ПО [Электронный ресурс] URL: https://habr.com/ru/companies/edison/articles/267569/ (Дата обращения 23.05.2025).

14. Проектирование: Начало / Хабр // Хабр Проектирование ИС [Электронный ресурс] URL: https://habr.com/ru/articles/907794/ (Дата обращения 23.05.2025).

15. Проектирование баз данных // DecoSystems [Электронный ресурс]. URL: https://www.decosystems.ru/proektirovanie-baz-dannykh/ (дата обращения: 06.05.25).

16. Простое руководство к диаграммам развертывания UML [Электронный ресурс]. URL: https://creately.com/blog/ru/uncategorized-ru/учебное-пособие-по-диаграмме-развёртки/ (дата обращения: 15.05.25).

17. Фундаментальная теория тестирования [Электронный ресурс]. URL: https://habr.com/ru/articles/549054/ (дата обращения: 21.05).

18. Тестирование методом «чёрного ящика» [Электронный ресурс]. URL: https://tquality.ru/blog/testirovanie-metodom-chernogo-yashchika/ (дата обращения: 22.05.25).

19. Агапов А. И. Тестирование программного обеспечения: практика разработки и методология / А. И. Агапов. – СПб.: БХВ-Петербург, 2021. – 352 с.

20. Обзорная статья о видах тестирования IT-продутов / Хабр // Хабр Тестирования ИС, виды тестирования [Электронный ресурс] URL: https://habr.com/ru/companies/otus/articles/720664/ (Дата обращения 23.05.2025).

Приложение А

(обязательное)

Листинг программы

Листинг А.1 – generate.js

```
import { GenerateFromTemplateDto } from
"./widgets/src/docu/template/dto/generate-from-template.dto";
import { Document } from
"./widgets/src/docu/document/entities/document.entity";
import { UpdateFileDto } from "./widgets/src/docu/file/dto/update-
file.dto";
import { StorageType } from
"./widgets/src/docu/file/enums/storage-type.enum";
import { Extension } from
"./widgets/src/docu/file/enums/extension.enum";
import { FileType } from "./widgets/src/docu/file/enums/file-
type.enum";
async generateFromTemplate(req: any, info:
GenerateFromTemplateDto): Promise < Document > {
      if (info.hasOwnProperty('refresh') && info.refresh) {
      if (info.hasOwnProperty('documentId') && info.documentId) {
        info.document = await this.findOneFull(info.documentId)
      }
    }
    const generatedFile = await
this.templateService.generateFromTemplate(req, info)

    const createDocumentDto =
this.makeCreateDocDtoAtGenerating(info)
    if (info.hasOwnProperty('invoice_form')) {
      if (info.hasOwnProperty('refresh') && info.refresh) {
        if (info.hasOwnProperty('document') && info.document) {
          const invoiceForm = await
this.invoiceFormService.update(
            info.document.invoice_form_id,
            {
              value: info.invoice_form,
            }
          )
          createDocumentDto.invoice_form_id = invoiceForm.id
        }
      } else {
        const invoiceForm = await this.invoiceFormService.create({
          value: info.invoice_form,
        })
        createDocumentDto.invoice_form_id = invoiceForm.id
      }
    }
    const template = await
this.templateService.findOneFull(info.templateId)
    const generatedDocumentName =
      template.file.name.split('.')[0] + `
№${info.templateCounter}.docx`
```

```
    const path = this.generateDocumentPath(generatedDocumentName,
createDocumentDto)
    await this.initialFolderStructure({
      account_id: info.account_id,
      path: path,
    })
    if (info.hasOwnProperty('refresh') && info.refresh) {
      const updateFileDto: UpdateFileDto = {
        id: info.document.file.id,
        path: info.document.file.path,
        name: info.document.file.name,
        type: info.document.file.type,
        extension: info.document.file.extension,
        account_id: createDocumentDto.account_id,
      }
      if (info.document.file.storage_type ===
StorageType.YANDEX_DISK) {
          updateFileDto.yandex_id = info.document.file.yandex_id
      }
      await this.fileService.update(generatedFile, updateFileDto)
      return this.findOneFull(info.documentId)
    } else {
      const uploadedGeneratedFile = await
this.fileService.create(generatedFile, {
        account_id: createDocumentDto.account_id,
        extension: Extension.DOCX,
        name: generatedDocumentName,
        path: path,
        type: FileType.GENERATED_BY_TEMPLATE,
      })
      createDocumentDto.file_id = uploadedGeneratedFile.id
      return
this.documentRepository.save(this.documentRepository.create(create
DocumentDto))
    }
}
```

Листинг А.2 – InvoiceForm.tsx

```
import React, { useEffect, useLayoutEffect } from 'react'
import s from './InvoiceForm.module.scss'
import { useAppDispatch, useAppSelector } from '@Hooks/redux'
import { Controller, FormProvider, useFieldArray, useForm,
useWatch } from 'react-hook-form'
import * as yup from 'yup'
import { FormControlLabel } from '@mui/material'
import AddIcon from '@mui/icons-material/Add'
import { CustomButton, CustomButtonIcon } from
'@ComponentsUI/Button'
import { SelectCustomer, SelectExecutor, SelectExecutorAccount }
from './Controls'
import { setCloseForm } from '@Store/Navi/InvoiceFormSlice'
import Footer from '@Components/Init/Leads/InvoiceForm/Footer'
```

```
import { documentsAPI } from '@Services/DocumentsService'
import { Notifies } from '@ComponentsUI/Notifies'
import { Loader } from '@ComponentsUI/Loader'
import Close from '@Icons/close.svg'
import { StyleCheckBox } from
'@ComponentsUI/CheckBox/StyleCheckBox'
import { TypeOfPositions, TypeRublePercent } from
'@/ENUMS/TypeOfPositions'
import MerchRow from './MerchRow'
import { AutoSelectStyle, TextFieldStyle } from
'@ComponentsUI/AutoCompleteSelect/StyleAutocompleteComponents'
import { StyleTextField } from
'@/componentsUI/TextField/StyleTextField'
import { createNotifies } from '@Store/Notifies/NotifiesSlice'
import { isObjectEmpty, sliceExtensions, transformNumbers } from
'@Utils/helpers'
import moment from 'moment'
import { Datepicker } from '@ComponentsUI/Datepicker'
import {
    useLazyGetCompanyByIdQuery,
    useLazyGetContactByIdQuery,
    useLazyGetLeadByIdQuery
} from '@Services/AmocrmService'
const schema = yup.object().shape({
    settings: yup.object().shape({
        status_id: yup.number().required('Поле обязательно к
заполнению!').typeError('Поле обязательно к заполнению!'),
        doublicate_fields: yup.object().shape({
            contacts: yup.number().required('Поле обязательно к
заполнению!').typeError('Поле обязательно к заполнению!'),
            companies: yup.number().required('Поле обязательно к
заполнению!').typeError('Поле обязательно к заполнению!'),
        }),
    })
})
const defaultValues = {
    invoice_form: {
        date: moment().format('YYYY-MM-DD'),
        merchandise: [],
        juridicalPerson: {
            inn_company_name: ''
        },
        settlementAccount: {
            bik_bank_name: '',
            settlement_account: ''
        },
        discount: {
            id: 1,
            option: 'Не предоставлена',
            count: 0,
            measure: '%'
        },
        useVAT: false,
```

```
        useSumWithVAT: false,
        sumVAT: 0,
        finalSum: 0,
        sumClear: 0,
        formCustomer: null
    }
}
const InvoiceForm = () => {
    const dispatch = useAppDispatch()
    const template = useAppSelector((store) =>
store.invoiceFormSlice.template)
    const invoice_form = useAppSelector((store) =>
store.invoiceFormSlice.invoice_form)
    const refreshDocument = useAppSelector((store) =>
store.invoiceFormSlice.refresh)
    const documentId = useAppSelector((store) =>
store.invoiceFormSlice.documentId)
    const isFormActive = useAppSelector((state) =>
state.invoiceFormSlice.isFormActive)
    const [generateFromTemplate, { isLoading:
loadingGenerateFormFromTemplate }] =
        documentsAPI.useGenerateFormFromTemplateMutation()
    const [getLeadById] = useLazyGetLeadByIdQuery()
    const [getContactById] = useLazyGetContactByIdQuery()
    const [getCompanyById] = useLazyGetCompanyByIdQuery()
    const methods = useForm({
        mode: 'onChange',
        defaultValues: defaultValues
    })
    const { control } = methods
    const { fields, append, remove, move } = useFieldArray({ name:
'invoice_form.merchandise', control })
    useEffect(() => {
        fields.map((item, index) => remove(index))
        if (invoice_form && isFormActive) {
            methods.reset({
                invoice_form: {
                    ...invoice_form,
                    merchandise: invoice_form.merchandise,
                    date: moment().format('YYYY-MM-DD')
                }
            })
        } else if (isFormActive) {
            methods.reset(defaultValues)
            append({
                id: 999,
                name: '',
                measure: 'Шт',
                price: 0,
                priceWithVAT: 0,
                priceWithoutVAT: 0,
                type: 'Услуга',
                image_id: 0,
```

```
                    count: 1,
                    vatPercent: 20,
                    sumVAT: 0,
                    discount: 0,
                    discountMeasure: '%',
                    finalSum: 0,
                    sumWithVAT: 0,
                    sumWithoutVAT: 0,
                    sumWithVATWithDiscount: 0,
                    sumWithoutVATWithDiscount: 0
                })
            }
        }, [isFormActive, documentId])
        let useVAT = useWatch({ control, name: 'invoice_form.useVAT',
defaultValue: false })
        let useSumWithVAT = useWatch({
            control,
            name: 'invoice_form.useSumWithVAT',
            defaultValue: false
        })
        let discountOption = useWatch({
            control,
            name: 'invoice_form.discount.option',
            defaultValue: invoice_form?.discount?.option ?? 'Не
предоставлена'
        })
        let discountCount = useWatch({
            control,
            name: 'invoice_form.discount.count',
            defaultValue: invoice_form?.discount?.count ?? 0
        })
        let discountMeasure = useWatch({
            control,
            name: 'invoice_form.discount.measure',
            defaultValue: invoice_form?.discount?.measure ?? '%'
        })
        const handleCloseInvoiceForm = () => {
            dispatch(setCloseForm())
        }
        const handleSaveInvoiceForm = async (data) => {
            let hasError = false
            if (data.invoice_form.merchandise.length === 1 &&
data.invoice_form.merchandise[0].id === 0) {
                dispatch(createNotifies({ type: 'warning', message:
'Добавьте товар в форме', time: 20000 }))
                hasError = true
            }
            if (!data.invoice_form.formCustomer) {
                dispatch(
                    createNotifies({
                        type: 'warning',
                        message: 'Добавьте контакт или компанию к
карточке сделки и выберете исполнителя в форме',
```

```
                    time: 20000
                })
            )
            hasError = true
        }
        if (hasError) return



        const merchandiseWithSerialNums =
data.invoice_form.merchandise.map((item, index) => ({
             ...item,
             serial_number: index + 1
         }))
        const dataForSend = { ...data, invoice_form: {
...data.invoice_form } }
        console.log('D', dataForSend)
        dataForSend.invoice_form.merchandise =
merchandiseWithSerialNums
        if (refreshDocument) {
            dataForSend.documentId = documentId
            dataForSend.refresh = refreshDocument
        }
        dataForSend.templateId = template.id
        @ts-ignore
        dataForSend.author_name = APP.constant('user').name
        @ts-ignore
        dataForSend.author_id = APP.constant('user').id
        dataForSend.invoice_form.formCustomer = {
            id: dataForSend.invoice_form?.formCustomer?.id,
            entityType: 'lead'
        }
        if (dataForSend.invoice_form.discount.id) {
            delete dataForSend.invoice_form.discount.id
        }
        if (!dataForSend.invoice_form.date) {
            delete dataForSend.invoice_form.date
        }
        const leadData = await
getLeadById(APP.data.current_card.id)
        const contactsId =
leadData.data?._embedded?.contacts[0]?.id
        const companiesId =
leadData.data?._embedded?.companies[0]?.id
        const contactData = contactsId ? await
getContactById(contactsId) : null
        const companyData = companiesId ? await
getCompanyById(companiesId) : null
        function getCustomFields(data): { phones: string[];
emails: string[] } {
            let customFieldsEmails: string[] = []
            let customFieldsPhones: string[] = []
            data?.map((cfv) => {
```

```
                if (cfv.field_code == 'EMAIL') {
                    cfv.values.map((value) =>
customFieldsEmails.push(value.value))
                }
                if (cfv.field_code == 'PHONE') {
                    cfv.values.map((value) =>
customFieldsPhones.push(value.value))
                }
            })
            return {
                emails: customFieldsEmails,
                phones: customFieldsPhones
            }
        }
        let lead = {
            id: leadData.data.id,
            title: leadData.data.name
        }
        const companiesCustomFields =
companyData?.data?.custom_fields_values
        let customFieldsCompany = companiesCustomFields ?
getCustomFields(companiesCustomFields) : null
        let linkedCompany
        if (companyData?.data?.id) {
            linkedCompany = {
                id: companyData?.data?.id,
                title: companyData?.data?.name,
                phone: customFieldsCompany?.phones,
                email: customFieldsCompany?.emails
            }
        }

        const contactsCustomFields =
contactData?.data?.custom_fields_values
        let customFieldsContacts = contactsCustomFields ?
getCustomFields(contactsCustomFields) : null
        let linkedContact
        if (contactData?.data?.id) {
            linkedContact = {
                id: contactData?.data?.id,
                title: contactData?.data?.name || 'Название не
указано',
                phone: customFieldsContacts?.phones,
                email: customFieldsContacts?.emails
            }
        }
        typeof linkedCompany?.id ? (dataForSend.linked_company =
linkedCompany) : ''
        typeof linkedContact?.id ? (dataForSend.linked_contact =
linkedContact) : ''
        dataForSend.entityType = 'lead'
        dataForSend.entityId = APP.data.current_card.id
        const resultData = transformNumbers(dataForSend)
```

```
        console.log('R', resultData)
        generateFromTemplate(resultData)
            .unwrap()
            .then((message) =>
                dispatch(
                    createNotifies({
                        type: 'success',
                        message: 'Форма успешно создана',
                        time: 3000
                    })
                )
            )
            .then(handleCloseInvoiceForm)
            .catch(({ data: { error } }) => {
                if (error) {
                    error.details.map((errorInstance) => {
                        dispatch(
                            createNotifies({
                                type: 'error',
                                message: 'Ошибка при создании
формы',
                                time: 3000
                            })
                        )
                    })
                } else {
                    dispatch(
                        createNotifies({
                            type: 'error',
                            message: 'Неизвестная ошибка',
                            time: 3000
                        })
                    )
                }
            })
    }
    let currentId = 1
    const handleAddRow = () =>
        append({
            id: currentId++,
            name: '',
            measure: 'Шт',
            price: 0,
            priceWithVAT: 0,
            priceWithoutVAT: 0,
            type: 'Услуга',
            image_id: 0,
            count: 1,
            vatPercent: 20,
            sumVAT: 0,
            discount: 0,
            discountMeasure: '%',
            finalSum: 0,
```

```
            sumWithVAT: 0,
            sumWithoutVAT: 0,
            sumWithVATWithDiscount: 0,
            sumWithoutVATWithDiscount: 0
        })
    return (
        isFormActive && (
            <FormProvider {...methods}>
                <div className={s.main}>
                    <div className={s.background} />
                    <div className={s.wrapper}>
                        <CustomButtonIcon
                            sx={{
                                position: 'absolute',
                                right: '25px',
                                top: '25px'
                            }}
                            color={`secondary`}
                            onClick={handleCloseInvoiceForm}
                        >
                            <Close />
                        </CustomButtonIcon>
                        {loadingGenerateFormFromTemplate &&
<Loader />}
                        <form
onSubmit={methods.handleSubmit(handleSaveInvoiceForm)}
className={s.content}>
                            <div className={s.formHeader}>
                                <div className={s.contentRow}>
                                    <div className={`${s.col}
${s.__330px}`}>
                                        <div
className={`${s.contentRow}`}>
                                            <div
className={`${s.col} ${s.__250px}`}>
                                                <span

title={template?.file ? sliceExtensions(template?.file?.name) :
''}

className={`${s.header} ${s.__red}`}
                                                >

{template?.file
                                                    ?
`${sliceExtensions(template?.file?.name, 22)} №${

template?.counter + 1
                                                    }`
                                                    :
`Документ №${template?.counter + 1}`}
                                                </span>
                                            </div>
```

```
                                        <div
className={`${s.col} ${s.__90px}`}>
                                            <Datepicker
startDateName={'invoice_form.date'} />
                                        </div>
                                    </div>
                                </div>
                                <div classNameName={s.contentRow}>
                                    <div classNameName={`${s.col}
${s.__85}`} />
                                    <div classNameName={`${s.col}
${s.__200px}`}>
                                        <FormControlLabel
                                            control={
                                                <StyleCheckBox
   {...methods.register('invoice_form.useVAT')}
checked={methods.getValues('invoice_form.useVAT')}
size={'small'}
                                                />
                                            }
                                            label={'Учитывать
с НДС'}
                                        />
                                    </div>
                                    <div classNameName={`${s.col}
${s.__200px}`}>
                                        <FormControlLabel
                                            control={
                                                <StyleCheckBox
{...methods.register('invoice_form.useSumWithVAT')}
checked={methods.getValues('invoice_form.useSumWithVAT')}
size={'small'}
                                                />
                                            }
                                            label={'Цены с
НДС'}
                                        />
                                    </div>
                                </div>
                            </div>
                            <div classNameName={`${s.contentRow}
${s.__bigMargin}`}>
                              <div classNameName={`${s.col}
${s.__330px}`}>
                                    <div
className={`${s.contentRow}`}>
                                        <div
className={`${s.col} ${s.__90px}`}>
                                            <span
className={`${s.subHeader} ${s.__red}`}>Заказчик</span>
```

```
                                        </div>

                                        <div
className={`${s.col} ${s.__200px}`}>
                                            <SelectCustomer
name={'invoice_form.formCustomer'} />
                                        </div>
                                    </div>
                                </div>
                                <div className={`${s.col}
${s.__30}`}>
                                    <div
className={s.contentRow}></div>
                                </div>
                            </div>
                            <div className={`${s.contentRow}
${s.__bigMargin}`}>
                                <div className={`${s.col}
${s.__330px}`}>
                                    <div
className={`${s.contentRow}`}>
                                        <div
className={`${s.col} ${s.__90px}`}>
                                            <span
className={`${s.subHeader} ${s.__red}`}>Исполнитель</span>
                                        </div>
                                        <div
className={`${s.col} ${s.__200px}`}>
                                            <SelectExecutor

subLabel={'inn_company_name'}
name={'invoice_form.juridicalPerson'}
placeholder={'Юр.лицо'}
                                            />
                                        </div>
                                    </div>
                                </div>
                                <div className={`${s.col}
${s.__200px}`}>
                                    <SelectExecutorAccount

name={'invoice_form.settlementAccount'}

placeholder={'Расчётный счет'}
                                    />
                                </div>
                                <div className={`${s.col}
${s.__discountCol}`}>
                                    <div
className={s.contentRow}>
                                        <div
className={`${s.col} ${s.__90px}`}>
```

```
                                                    <span
className={`${s.subHeader}`}>Скидка</span>
                                                </div>
                                                <div
className={`${s.col} ${s.__250px}`}>
                                                    {/* ВЫБОР СКИДКИ
НЕ ПРЕДОСТАВЛЕНА/ НА ВСЕ ПОЗИЦИИ / НА ОТДЕЛЬНЫЕ ПОЗИЦИИ */}

                                                    <Controller

name={'invoice_form.discount.option'
control={methods.control}
render={({ field: { value } }) => (
<AutoSelectStyle
freeSolo
disableClearable={true}
getOptionLabel={(option) => option?.value}
isOptionEqualToValue={(option, value) => {
return option.value === String(value)
options={TypeOfPositions}
value={{ value: discountOption }}
defaultValue={TypeOfPositions[0]}
onChange={(e, value) =
methods.setValue('invoice_form.discount', {
option: value.value,
count: 0,
measure: '%'
renderInput={(params) => (
<TextFieldStyle
{...params}
placeholder={'Выберите тип'}
                                                                    />
                                                                )}
                                                            />
                                                        )}
                                                    />
                                                </div>
                                                {discountOption != ''
&& discountOption == 'На все позиции' && (
                                                    <>
                                                        {/* ВЫБОР ТИПА
СКИДКИ ПРОЦЕНТ ИЛИ РУБЛЬ */}
                                                        <div
className={`${s.col} ${s.__250px}`}>

<StyleTextField

{...methods.register('invoice_form.discount.count')}

placeholder={'Сумма/процент скидки'}

size='small'
```

```
value={+discountCount}
inputProps={{
min: 0,
type: 'number',
step: '.001'
                                                                    }}
                                                                />
                                                            </div>
                                                            {/* ВЫБОР
СУММЫ ДЛЯ СКИДКИ */}
                                                            <div
className={`${s.col} ${s.__250px}`}>
<AutoSelectStyle
disableClearable={true}
getOptionLabel={(option) => option?.value}
isOptionEqualToValue={(option, value) =>
option.id === value?.id
                                                                }
options={TypeRublePercent}
defaultValue={TypeRublePercent[0]}
onChange={(e, value) =>
methods.setValue(
'invoice_form.discount.measure',
value?.value
                                                                )
                                                                }
renderInput={(params) => (
<TextFieldStyle
{...params}
{...methods.register(
'invoice_form.discount.measure'
```

```
)}

placeholder={'Выберите тип'}
                                                                    />
                                                                )}
                                                            />
                                                        </div>
                                                    </>
                                                )}
                                            </div>
                                        </div>
                                    </div>
                                    <div className={s.itemsList}>
                                        <div className={s.rows}>
                                            <div className={`${s.row}
${s.headerList}`}>
                                                <div className={`${s.col}
${s.__1}`}>№</div>
                                                <div className={`${s.col}
${s.__700px}`}>Наименование</div>
                                                <div className={`${s.col}
${s.__2}`} />
                                                <div className={`${s.col}
${s.__90px}`}>Кол-во</div>
                                                <div className={`${s.col}
${s.__90px}`}>Ед. измерения</div>
                                                <div className={`${s.col}
${s.__90px}`}>
                                                    {useVAT ?
(useSumWithVAT ? 'Цена с НДС' : 'Цена без НДС') : 'Цена'}
                                                </div>
                                                {useVAT && <div
className={`${s.col} ${s.__90px}`}>НДС</div>}
                                                {useVAT && <div
className={`${s.col} ${s.__90px}`}>Сумма НДС</div>}
                                                {discountOption &&
                                                    discountOption != ''
&&
                                                    discountOption == 'На
отдельные позиции' && (
                                                        <div
className={`${s.col} ${s.__130px}`}>Скидка</div>
                                                    )}
                                                <div className={`${s.col}
${s.__184px}`}>
                                                    {discountOption &&
                                                        discountOption ==
'Не предоставлена' &&
                                                        !useVAT &&
                                                        !useSumWithVAT &&
                                                        'Сумма'}
```

```
                                                {discountOption &&
                                                    discountOption ==
'Не предоставлена' &&
                                                    !useVAT &&
                                                    useSumWithVAT &&
                                                    'Сумма'}
                                                {discountOption &&
                                                    discountOption ==
'Не предоставлена' &&
                                                    useVAT &&
                                                    'Сумма с НДС'}
                                                {discountOption &&
                                                    discountOption !=
'Не предоставлена' &&
                                                    !useVAT &&
                                                    !useSumWithVAT &&
                                                    'Сумма со
скидкой'}
                                                {discountOption &&
                                                    discountOption !=
'Не предоставлена' &&
                                                    !useVAT &&
                                                    useSumWithVAT &&
                                                    'Сумма со
скидкой'}
                                                {discountOption &&
                                                    discountOption !=
'Не предоставлена' &&
                                                    useVAT &&
                                                    'Сумма с НДС со
скидкой'}
                                            </div>
                                            <div classNameName={`${s.col}
${s.__deleteButtonCol}`} />
                                        </div>
                                        {fields.map((row, index) => (
                                            <MerchRow
                                                key={row.id}
                                                index={index}
                                                merchRow={row}
                                                remove={remove}
                                                move={move}
                                            />
                                        ))}
                                    </div>
                                    <div classNameName={s.addRows}>
                                        <CustomButton
color={`primary`} onClick={handleAddRow}>
                                            <AddIcon color={`primary`}
/>
                                            Добавить строку
                                        </CustomButton>
                                    </div>
```

```
                                </div>
                                <Footer />
                            </form>
                        </div>
                        <Notifies />
                    </div>
                </FormProvider>
            )
        )
}
export default InvoiceForm
```

Листинг А.3 - selectScenario.js

```
async selectScenario(req: any, selectScenarioDto:
SelectScenarioDto): Promise < Document > {
    const document = await
this.findOneFull(selectScenarioDto.document_id)
        if(!document.status_history) {
    const updateDocumentDto = await
this.scenarioService.fillStatusHistory(
        selectScenarioDto.user_name,
        selectScenarioDto.scenario_id
    if (
updateDocumentDto.status_history[1].responsible_users.length < 2
&&
updateDocumentDto.status_history[1].responsible_users.length > 0
    ) {
        let userId =
Number(updateDocumentDto.status_history[1].responsible_users[0])
        const amo = await this.amocrmService.getAmoClient(req)
        if (userId === StatusResponsible.LEAD_OWNER) {
            const lead = await
amo.leads.get(selectScenarioDto.lead_id)
            userId = lead.responsible_user_id
        }
        let task = await amo.tasks.create({
            entity_type: 'leads',
            entity_id: selectScenarioDto.lead_id,
            responsible_user_id: userId,
            complete_till: Math.floor(Date.now() / 1000) + 3600,
            text: document.file.name,
            task_type_id:
updateDocumentDto.status_history[1].task_type_id,
        })
        if (!task['_embedded']) {
            throw new Error(task['response']['statusText'])
        }
        task = task['_embedded']['tasks'][0]
        updateDocumentDto.status_history[0].amo_task_id = task.id
    }
```

```
    updateDocumentDto.status_history[0].user_name =
selectScenarioDto.user_name
    updateDocumentDto.status_history[0].is_completed = true
    updateDocumentDto.scenario_id = selectScenarioDto.scenario_id
    await this.update(selectScenarioDto.document_id,
updateDocumentDto)
    return this.findOne(selectScenarioDto.document_id)
} else {
    return document
}
    }
```

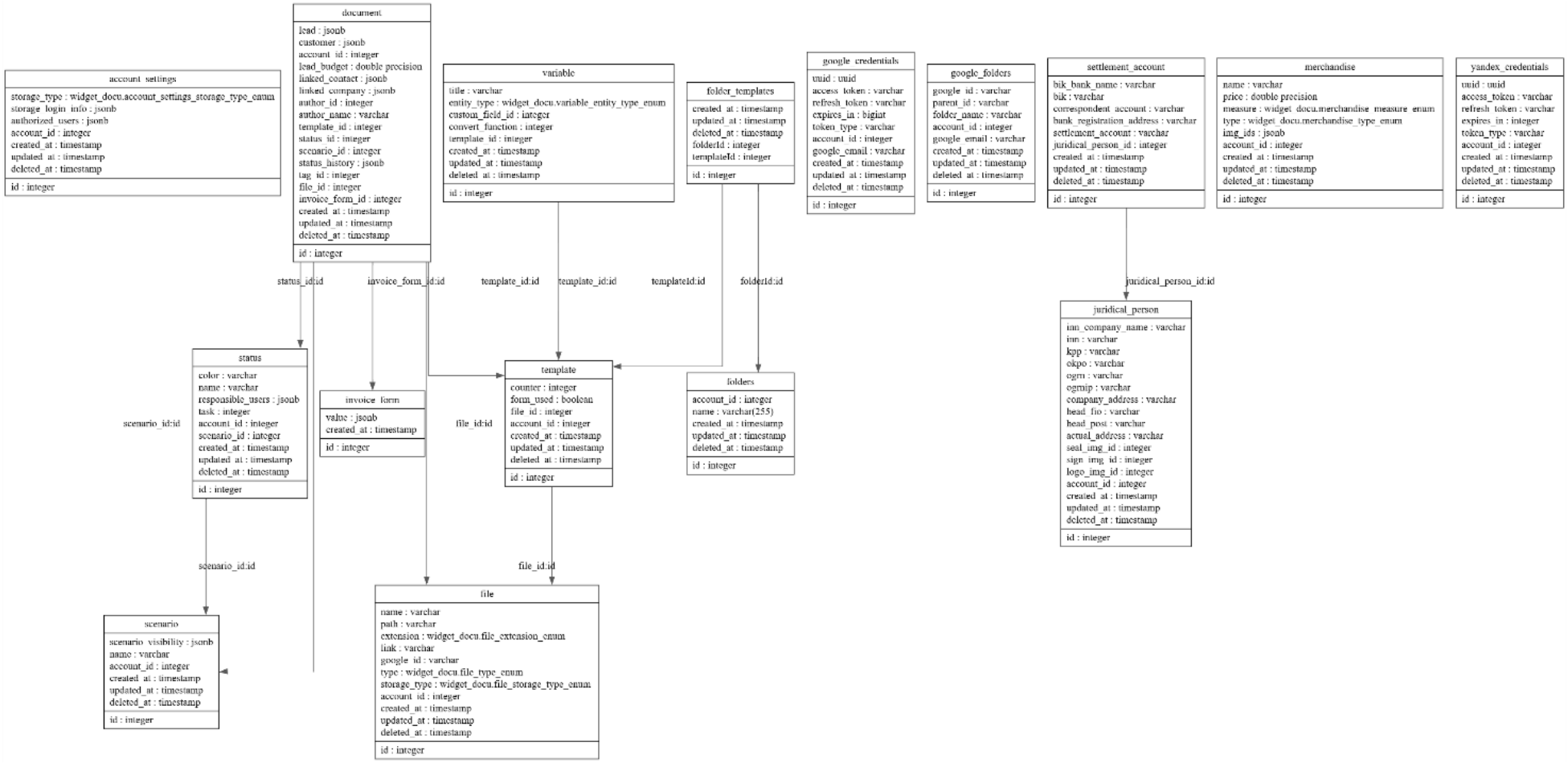
Листинг А.4 - handleSelectScenario.js

```
const handleSelectScenario = (id: number) => {
    selectScenario({
        scenario_id: id,
        document_id: document.id,
        user_name: APP.constant('user').name,
        lead_id: document.lead.id
    })
        .unwrap()
        .then((message) => {
            dispatch(
                createNotifies({
                    type: 'success',
                    message: 'Сценарий успешно выбран',
                    time: 3000
                })
            )
        })
        .catch((error) => {
            dispatch(
                createNotifies({
                    type: 'error',
                    message: 'Ошибка при выборе сценария',
                    time: 100000
                })
            )
        })
}
```

Приложение Б

(рекомендуемое)

База данных


account_settings
storage_type : widget_docu.account_settings_storage_type_enum
storage_login_info : jsonb
authorized_users : jsonb
account_id : integer
created_at : timestamp
updated_at : timestamp
deleted_at : timestamp
id : integer
document
lead : jsonb
customer : jsonb
account_id : integer
lead_budget : double precision
linked_contact : jsonb
linked_company : jsonb
author_id : integer
author_name : varchar
template_id : integer
status_id : integer
scenario_id : integer
status_history : jsonb
tag_id : integer
file_id : integer
invoice_form_id : integer
created_at : timestamp
updated_at : timestamp
deleted_at : timestamp
id : integer
variable
title : varchar
entity_type : widget_docu.variable_entity_type_enum
custom_field_id : integer
convert_function : integer
template_id : integer
created_at : timestamp
updated_at : timestamp
deleted_at : timestamp
id : integer
folder_templates
created_at : timestamp
updated_at : timestamp
deleted_at : timestamp
folderId : integer
templateId : integer
id : integer
google_credentials
uuid : uuid
access_token : varchar
refresh_token : varchar
expires_in : bigint
token_type : varchar
account_id : integer
google_email : varchar
created_at : timestamp
updated_at : timestamp
deleted_at : timestamp
id : integer
google_folders
google_id : varchar
parent_id : varchar
folder_name : varchar
account_id : integer
google_email : varchar
created_at : timestamp
updated_at : timestamp
deleted_at : timestamp
id : integer
settlement_account
bik_bank_name : varchar
bik : varchar
correspondent_account : varchar
bank_registration_address : varchar
settlement_account : varchar
juridical_person_id : integer
created_at : timestamp
updated_at : timestamp
deleted_at : timestamp
id : integer
merchandise
name : varchar
price : double precision
measure : widget_docu.merchandise_measure_enum
type : widget_docu.merchandise_type_enum
img_ids : jsonb
account_id : integer
created_at : timestamp
updated_at : timestamp
deleted_at : timestamp
id : integer
yandex_credentials
uuid : uuid
access_token : varchar
refresh_token : varchar
expires_in : integer
token_type : varchar
account_id : integer
created_at : timestamp
updated_at : timestamp
deleted_at : timestamp
id : integer
status_id:id
invoice_form_id:id
template_id:id
template_id:id
templateId:id
folderId:id
juridical_person_id:id
status
color : varchar
name : varchar
responsible_users : jsonb
task : integer
account_id : integer
scenario_id : integer
created_at : timestamp
updated_at : timestamp
deleted_at : timestamp
id : integer
invoice_form
value : jsonb
created_at : timestamp
id : integer
template
counter : integer
form_used : boolean
file_id : integer
account_id : integer
created_at : timestamp
updated_at : timestamp
deleted_at : timestamp
id : integer
folders
account_id : integer
name : varchar(255)
created_at : timestamp
updated_at : timestamp
deleted_at : timestamp
id : integer
juridical_person
inn_company_name : varchar
inn : varchar
kpp : varchar
okpo : varchar
ogrn : varchar
ogrnip : varchar
company_address : varchar
head_fio : varchar
head_post : varchar
actual_address : varchar
seal_img_id : integer
sign_img_id : integer
logo_img_id : integer
account_id : integer
created_at : timestamp
updated_at : timestamp
deleted_at : timestamp
id : integer
scenario_id:id
file_id:id
scenario_id:id
file_id:id
scenario
scenario_visibility : jsonb
name : varchar
account_id : integer
created_at : timestamp
updated_at : timestamp
deleted_at : timestamp
id : integer
file
name : varchar
path : varchar
extension : widget_docu.file_extension_enum
link : varchar
google_id : varchar
type : widget_docu.file_type_enum
storage_type : widget_docu.file_storage_type_enum
account_id : integer
created_at : timestamp
updated_at : timestamp
deleted_at : timestamp
id : integer

Приложение В

(рекомендуемое)

Диаграммы

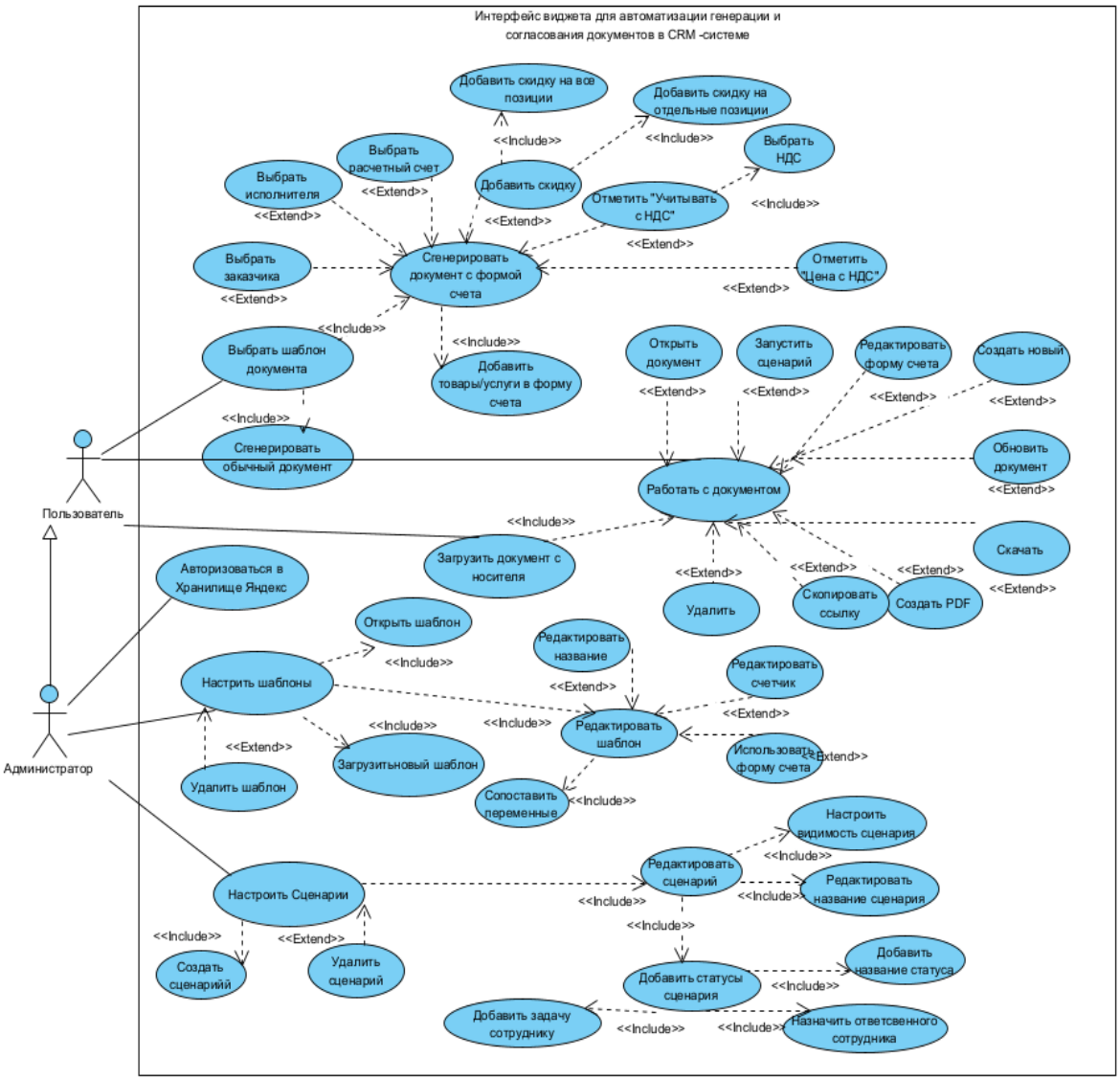


Рисунок В.1 – Диаграмма вариантов использования

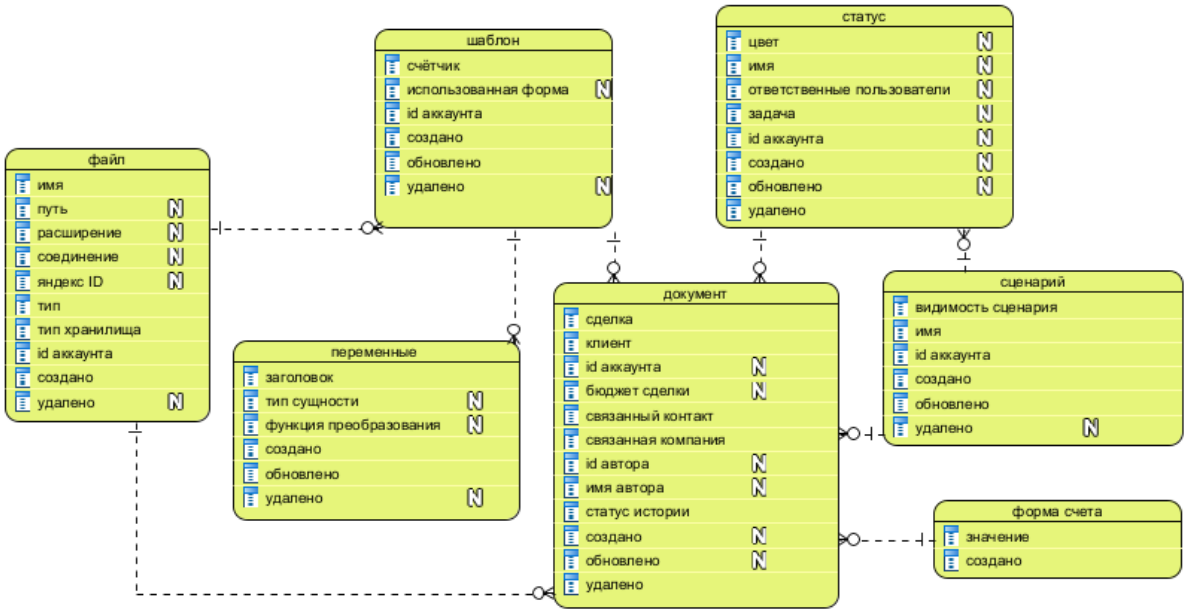


Рисунок В.2 – Концептуальная модель базы данных

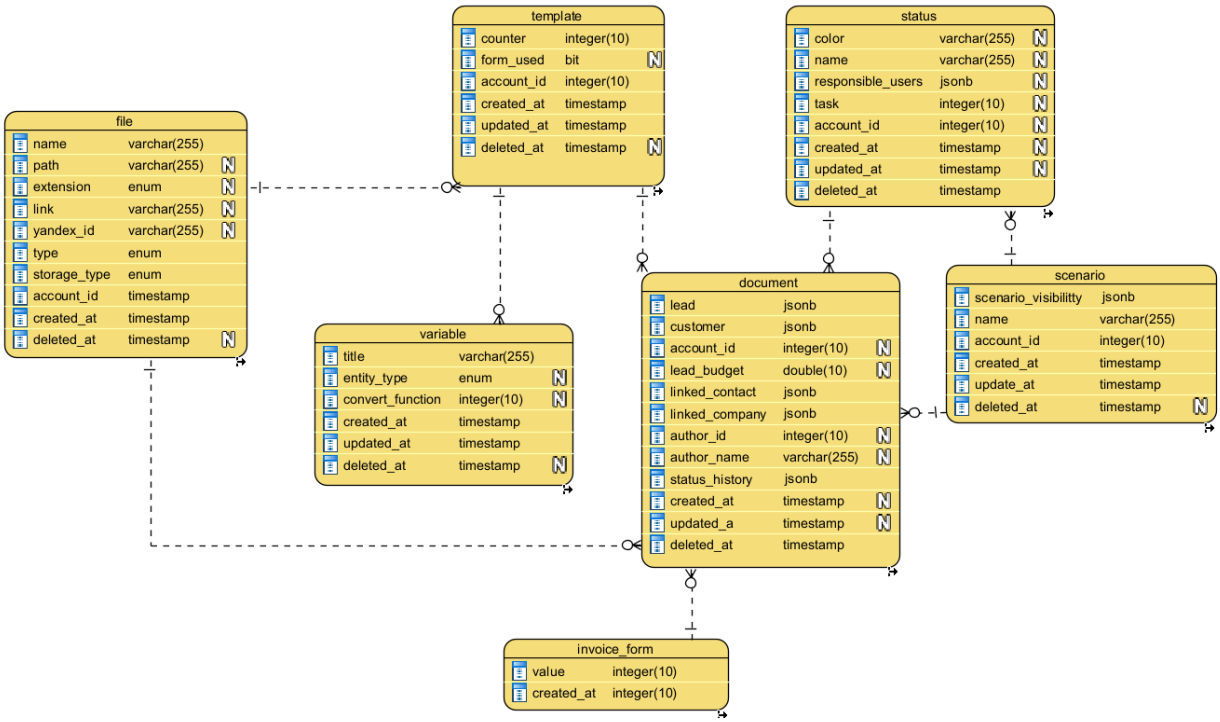


Рисунок В.3 – Логическая модель базы данных

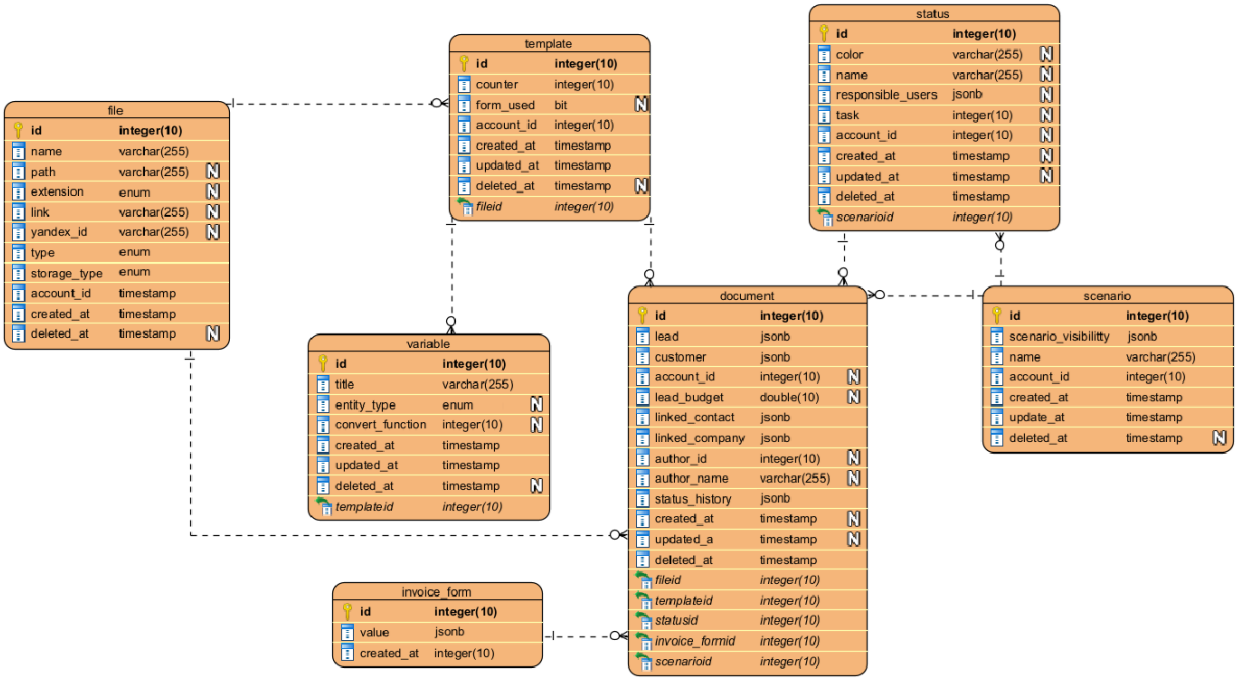


Рисунок В.4– Физическая модель базы данных

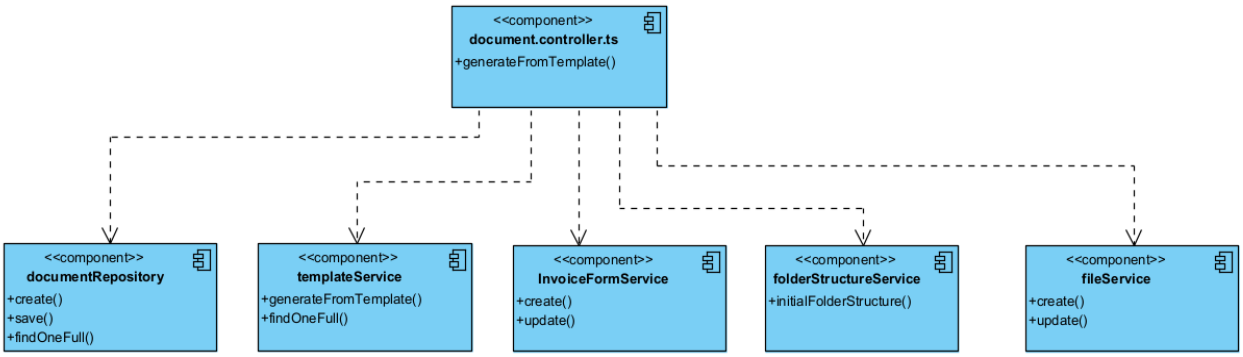


Рисунок В.5 – диаграмма компонентов

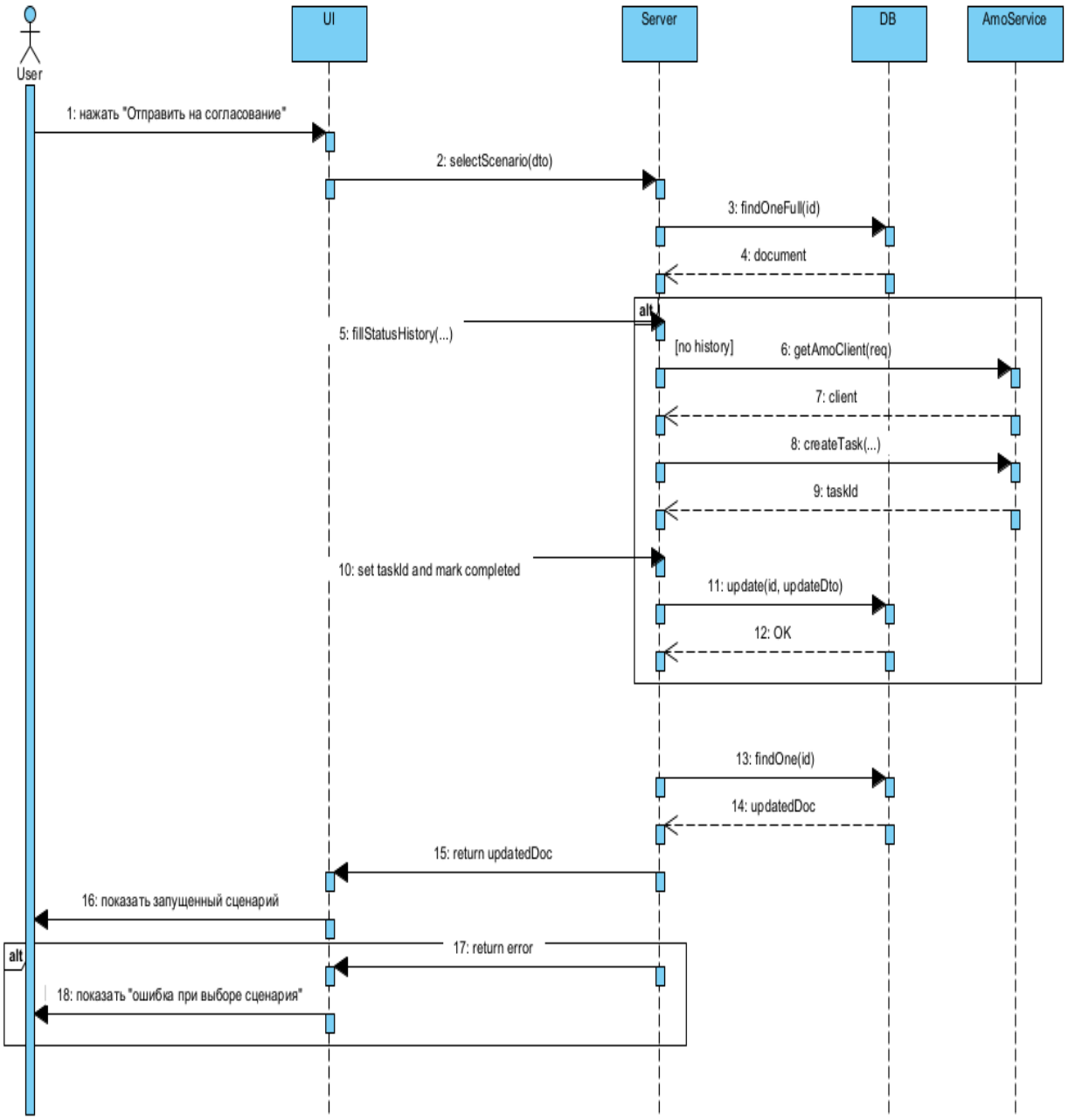


Рисунок В.6 – Диаграмма последовательности

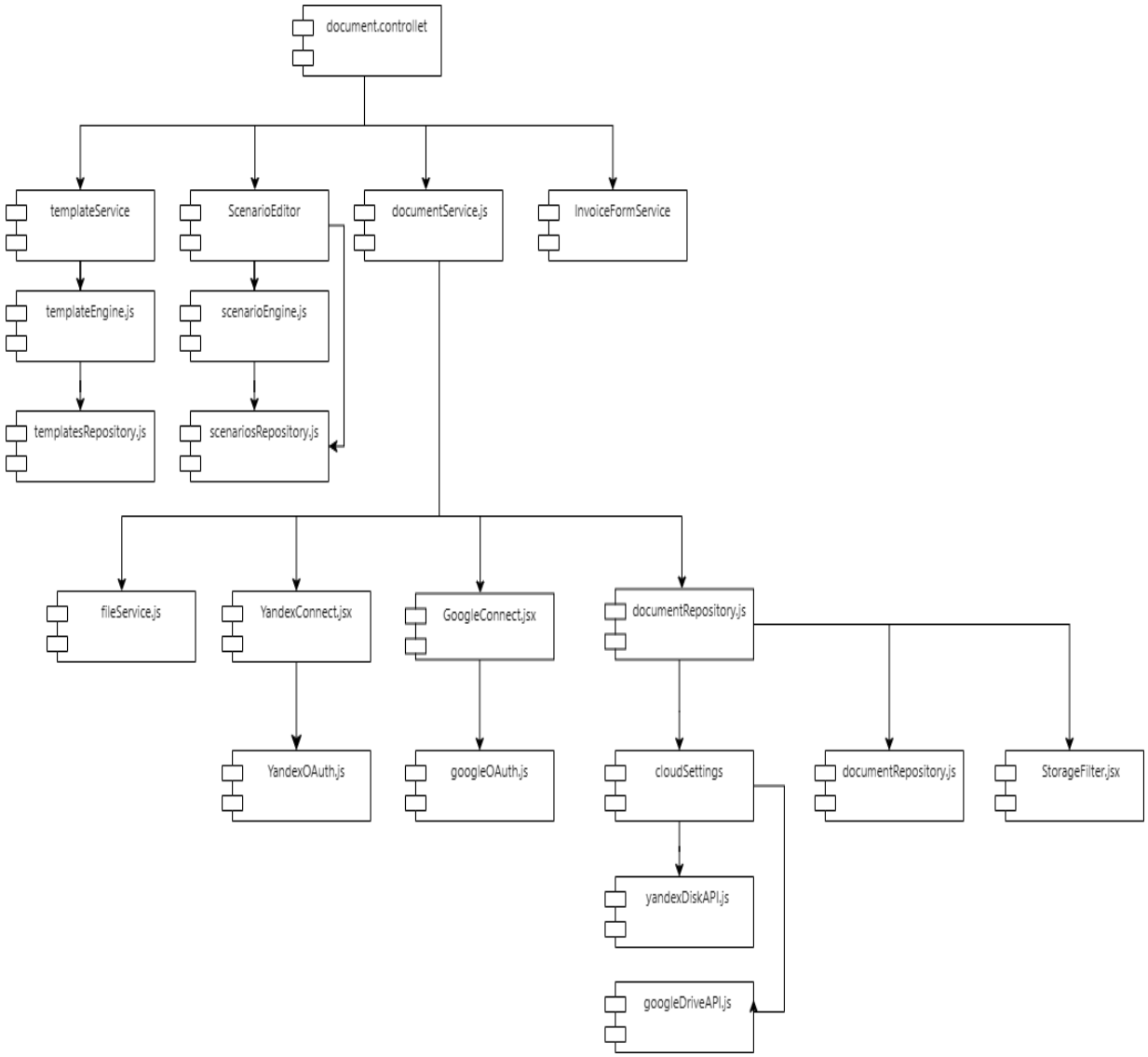


Рисунок В.7 – Диаграмма компонентов

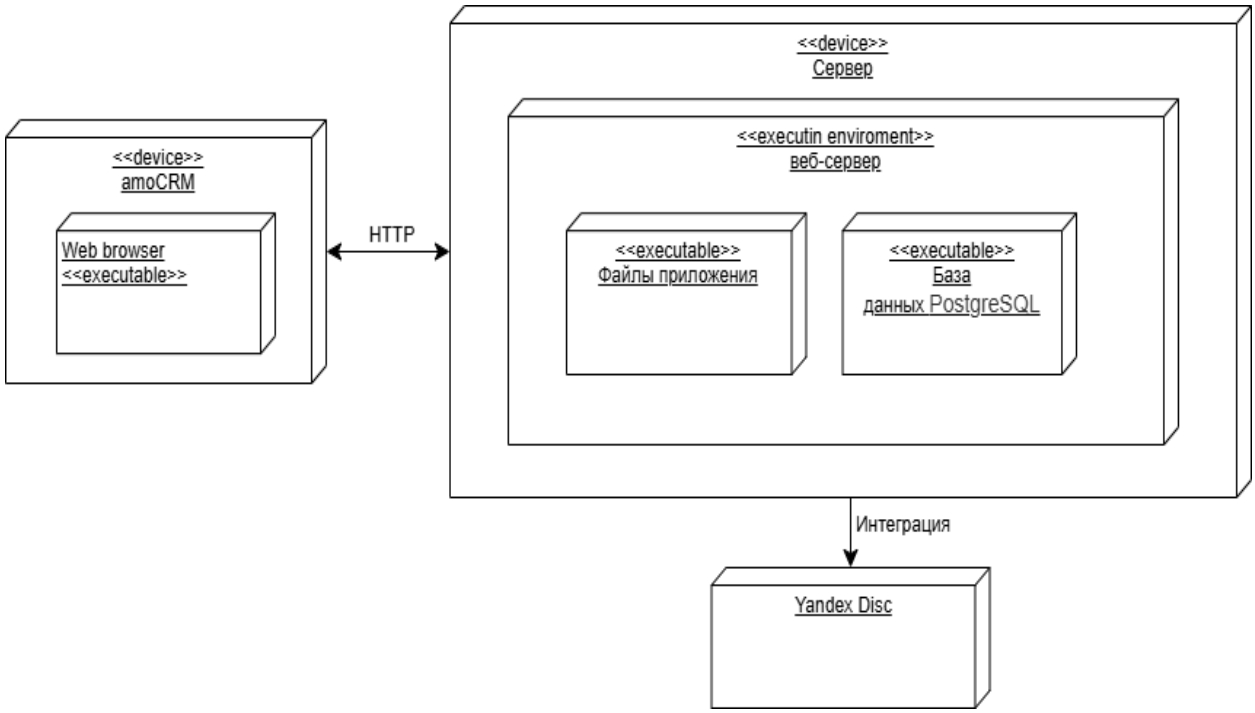


Рисунок В.8 – Диаграмма развертывания

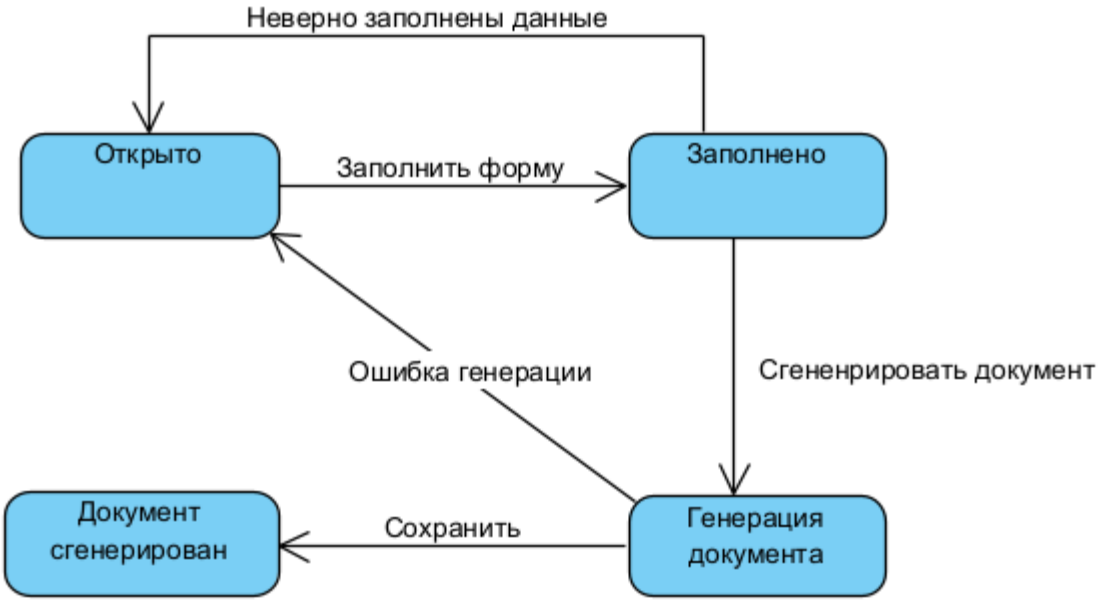


Рисунок В.9 – Диаграмма состояний

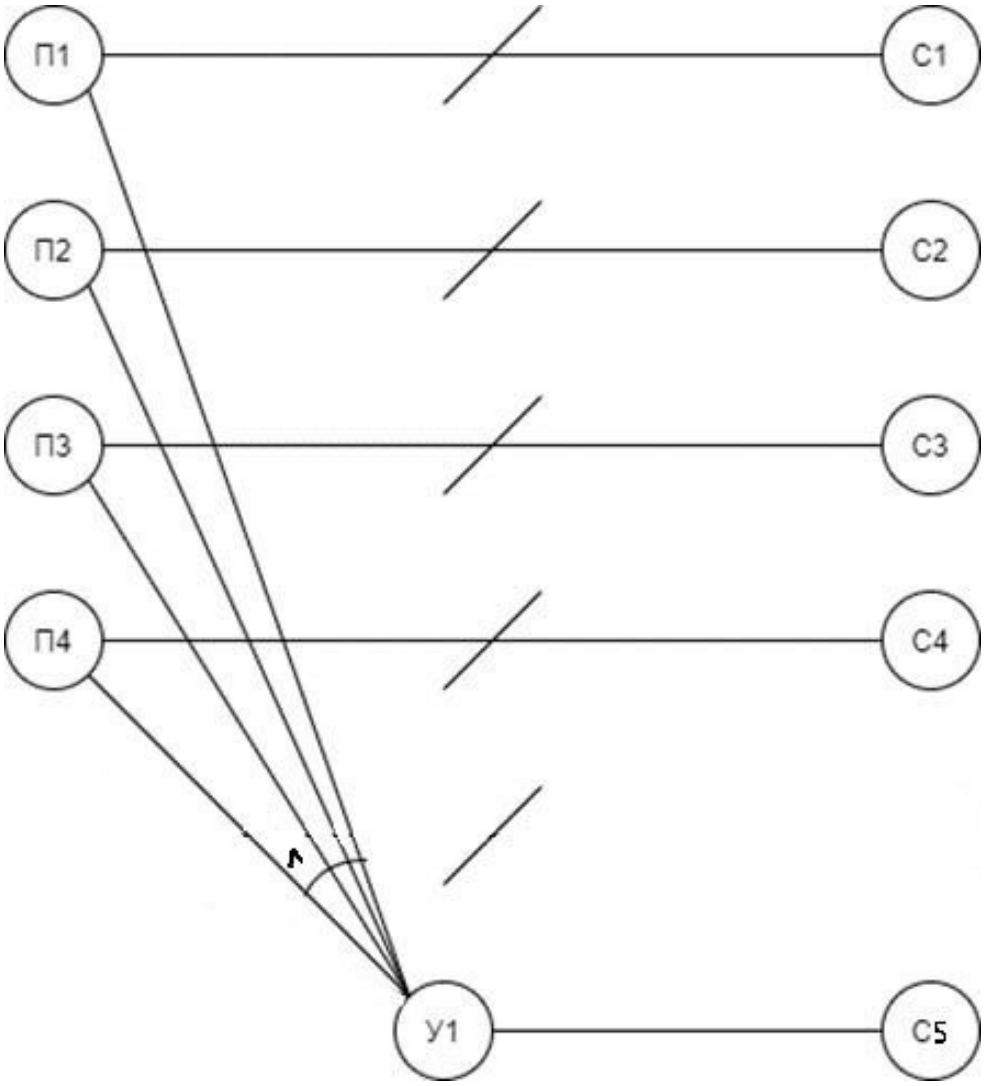


Рисунок В.10 - Функциональная диаграмма

Приложение Г

(рекомендуемое)

Акт внедрения дипломной работы

УТВЕРЖДАЮ
Генеральный директор
ООО «Ингруппа»
Васянин М.С.

(подпись)
“ 19 ” 06 2025 г.

АКТ
о внедрении результатов исследования, полученных в дипломной работе «Виджет для автоматизации генерации и согласования документов в CRM-системе» выполненной студентом Севостьяновой Анастасией Олеговной

Настоящий акт составлен о том, что результат выпускной квалификационной роботы студентки ПГУ группы 21ВИ1 очной формы обучения Севостьяновой Анастасии Олеговны на тему «**Виджет для автоматизации генерации и согласования документов в CRM-системе**» внедрены в компанию «Ингруппа» и успешно применяются в ее работе с 13.04.2025. Разработанный виджет интегрирован в CRM-систему компании и обеспечивает:

автоматизацию создания и согласования документов;

сокращение времени обработки заявок;

минимизацию ошибок при формировании документов;

повышение эффективности бизнес-процессов.

Кроме того, данный виджет был коммерциализирован и реализуется компанией **ООО «Ингруппа»** как самостоятельный продукт.

Генеральный директор
ООО «Ингруппа»
Васянин М.С.

Ингруппа