Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение высшего образования
АМУРСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
(ФГБОУ ВО «АмГУ»)

Институт компьютерных и инженерных наук
Кафедра информационных и управляющих систем
Направление подготовки 09.03.04 – Программная инженерия
Направленность (профиль) образовательной программы – Программная инженерия

ДОПУСТИТЬ К ЗАЩИТЕ
Зав. кафедрой
_____ А.В. Бушманов
«____»_____2025 г.

БАКАЛАВРСКАЯ РАБОТА

на тему: Разработка программного обеспечения мультичат для взаимодействия с клиентами ООО «ИнфоКом»

Исполнитель
студент группы 1105-об _____ В.С. Подорван
(подпись, дата)

Руководитель
доцент, канд. техн. наук _____ О.В. Жилиндина
(подпись, дата)

Консультант
по безопасности и
экологичности
доцент, канд. техн. наук _____ А.Б. Булгаков
(подпись, дата)

Нормоконтроль
инженер кафедры _____ В.Н. Адаменко
(подпись, дата)

Благовещенск 2025

Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение высшего образования
АМУРСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
(ФГБОУ ВО «АмГУ»)

Институт компьютерных и инженерных наук
Кафедра информационных и управляющих систем

УТВЕРЖДАЮ
Зав. кафедрой
_____ А.В. Бушманов
подпись И.О. Фамилия
«____»_____2025 г.

З А Д А Н И Е

К выпускной квалификационной работе студента Подорван Владислава Сергеевича

1. Тема выпускной квалификационной работы: Разработка программного обеспечения мультичат для взаимодействия с клиентами ООО «ИнфоКом»
(утверждена приказом от 14.04.2025 № 980-уч)

2. Срок сдачи студентом законченной работы (проекта): 10.06.2025

3. Исходные данные к выпускной квалификационной работе: предметная область, перечень литературных источников, отчёт по преддипломной практике.

4. Содержание выпускной квалификационной работы: (перечень подлежащих разработке вопросов): анализ деятельности предприятия, проектирование мультичата, разработка программного обеспечения.

5. Перечень материалов приложения: (наличие чертежей, таблиц, графиков, схем, программных продуктов, иллюстративного материала и т.п.): техническое задание.

6. Консультанты по выпускной квалификационной работе (с указанием относящихся к ним разделов): консультант по безопасности и экологичности Булгаков А.Б., доцент, канд. техн. наук.

7. Дата выдачи задания: 14.02.2025 г.

Руководитель выпускной квалификационной работы: Жилиндина О. В. доцент, канд. техн. наук
(фамилия, имя, отчество, должность, ученая степень, ученое звание)

Задание принял к исполнению (дата): 14.02.2025 г.
(подпись студента)

РЕФЕРАТ

Бакалаврская работа содержит 78 с., 20 рисунков, 7 таблиц, 36 источников, 9 приложений.

МИКРОСЕРВИСНАЯ АРХИТЕКТУРА ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ, ПРОЕКТИРОВАНИЕ АРХИТЕКТУРЫ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ, ПРОЕКТИРОВАНИЕ БАЗЫ ДАННЫХ, ПРОЕКТИРОВАНИЕ WEB-ИНТЕРФЕЙСА, ТЕХНИЧЕСКОЕ ЗАДАНИЕ, UML ДИАГРАММЫ

В данной работе рассматриваются современные коммуникационные системы и их использование в качестве средства деловой коммуникации в ООО «ИнфоКом».

Цель работы – разработать программное обеспечение мультичат для взаимодействия с клиентами ООО «ИнфоКом».

Для достижения цели работы необходимо: произвести анализ деятельности организации ООО «ИнфоКом», рассмотреть API сервисов, с которыми будет взаимодействовать программное обеспечение, спроектировать программное обеспечение, выбрать средства реализации программного обеспечения, разработать web-интерфейс и реализовать проект.

В ходе выполнения работы на этапе проектирования программного обеспечения для реализации программы была выбрана микросервисная архитектура, которая была реализована с использованием языков программирования Python и JavaScript.

В результате выполнения работы был получен мультичат, обеспечивающей централизованную обработку клиентских обращений, поступающих из Telegram и ВКонтакте.

СОДЕРЖАНИЕ

НОРМАТИВНЫЕ ССЫЛКИ

В настоящей бакалаврской работе использованы ссылки на следующие стандарты:

ГОСТ 19.201-78 ЕСПД Техническое задание. Требования к содержанию и оформлению

ГОСТ Р 50571.5.54-2013 Электроустановки низковольтные. Выбор и монтаж электрооборудования. Заземляющие устройства, защитные проводники и защитные проводники уравнивания потенциалов

ГОСТ Р 50948-2001 Средства отображения информации индивидуального пользования. Общие эргономические требования и требования безопасности

ГОСТ Р 52872-2019 Интернет-ресурсы и другая информация, представленная в электронно-цифровой форме. Приложения для стационарных и мобильных устройств, иные пользовательские интерфейсы. Требования доступности для людей с инвалидностью и других лиц с ограничениями жизнедеятельности

Постановление Правительства РФ от 16.09.2020 N 1479 «Об утверждении правил противопожарного режима в Российской Федерации»

СанПиН 1.2.3685-21 Гигиенические нормативы и требования к обеспечению безопасности и (или) безвредности для человека факторов среды обитания

СП 52.13330.2016 Естественное и искусственное освещение

Федеральный закон «Об отходах производства и потребления» от 24.06.1998 N 89-ФЗ

Федеральный закон «Технический регламент о требованиях пожарной безопасности» от 22.07.2008 N 123-ФЗ

ISO/IEC 19505-2012 Information technology – Object Management Group Unified Modeling Language (OMG UML) Part 2: Superstructure

ОПРЕДЕЛЕНИЯ, ОБОЗНАЧЕНИЯ, СОКРАЩЕНИЯ

В настоящей бакалаврской работе применяют следующие термины с соответствующими определениями:

ПК – персональный компьютер;

Хэш-сумма – уникальная строка или идентификатор, получаемый в результате применения хеш-функции к данным, как правило, используется для проверки целостности и идентификации данных;

ЭДО – система обмена юридически значимыми документами в электронном виде между организациями и государственными структурами;

API – программный интерфейс, предоставляющий набор функций и протоколов для взаимодействия между программными компонентами;

CRM – система управления взаимоотношениями с клиентами, предназначенная для автоматизации процессов продаж, маркетинга, поддержки и обслуживания;

HTTP – протокол передачи гипертекста, используемый для обмена данными между клиентом и сервером в сети Интернет;

HTTPS – защищённая версия HTTP, использующая шифрование (TLS/SSL) для безопасной передачи данных;

JSON – текстовый формат обмена данными, основанный на синтаксисе JavaScript;

JWT – стандарт для обмена данными в формате JSON, часто используемый для авторизации и аутентификации;

S3 – облачное хранилище объектов, предназначенное для хранения и извлечения любых объемов данных в любое время и из любой точки сети;

SSL – криптографический протокол, обеспечивающий защищённую передачу данных в интернете;

TLS – криптографический протокол, обеспечивающий безопасность передачи данных в интернете, пришедший на смену SSL;

UML – унифицированный язык моделирования, используемый для визуализации, спецификации, проектирования и документирования программных систем;

URL – унифицированный указатель ресурса, адрес, по которому можно получить доступ к ресурсу в Интернете;

Webhook – механизм, при котором сервер отправляет данные на заданный URL по событию, без необходимости опроса со стороны клиента;

Long polling – метод периодической отправки запросов от клиента к серверу для проверки обновлений данных;

WebSocket – коммуникационный протокол, обеспечивающий двустороннюю передачу данных между клиентом и сервером в режиме реального времени поверх одного TCP-соединения.

ВВЕДЕНИЕ

В рамках данной бакалаврской работы рассматриваются современные коммуникационные системы, среди которых особое место занимают системы мгновенного обмена сообщениями. Они представляют собой удобные и гибкие платформы для обмена информацией между людьми. Наиболее популярными из них являются: Telegram, ВКонтакте и WhatsApp. Они предлагают разнообразный функционал: групповые чаты, видеозвонки и многое другое, что делает их мощным инструментом межличностного общения.

Именно из-за такого удобства и широкого распространения подобных систем, в след за межличностными отношениями, в них стали перемещаться деловые отношения между организациями и их клиентами, что позволило упростить взаимодействие клиентов с организациями и улучшить клиентский опыт.

Однако, так как клиенты организаций пользуются различными системами мгновенного обмена сообщениями, организациям стало не очень удобно взаимодействовать с клиентами из-за необходимости одновременно использовать несколько различных систем мгновенного обмена сообщениями.

Так, зачастую, сотрудникам приходится часто переключаться между разными мессенджерами, тратя время на переходы между интерфейсами, входы в аккаунты и смены контекста, что снижает эффективность и повышает риск ошибок. Одним из вариантов решения данной проблемы является объедение различных систем мгновенного обмена сообщениями в единый интерфейс – мультичат.

Кроме того, мультичат позволяет оперативно реагировать на обращения благодаря единому потоку уведомлений, исключая необходимость постоянно проверять несколько приложений. Это значительно снижает время, прошедшее от получения обращения клиента до ответа на него, что позволяет вызвать чувство внимания и важности обращения клиента, приводящее в конечном счёте к ощущению значимости для компании его обращения и, следовательно, к улучшению клиентского опыта и лояльности к организации.

Также стоит отметить, что при использовании мультичата клиенту не нужно будет проходить дополнительную регистрацию, так как он уже зарегистрирован в используемой им системе мгновенного обмена сообщениями. Это позволит ему сразу начать диалог без необходимости создания новой учётной записи, сделав процесс обращения более удобным и быстрым.

Таким образом, актуальность бакалаврской работы на тему «Разработка программного обеспечения мультичат для взаимодействия с клиентами ООО «ИнфоКом» обоснована тем, что она отвечает современным вызовам бизнеса, способствует улучшению клиентского опыта, оптимизации работы сотрудников и поддерживает долгосрочные цели по цифровизации и повышению конкурентоспособности организации.

В связи с чем в качестве цели бакалаврской работы ставится задача разработать программное обеспечение мультичат для взаимодействия с клиентами ООО «ИнфоКом».

Предметная область бакалаврской работы охватывает системы мгновенного обмена сообщениями, в частности ВКонтакте и Telegram. Объектом исследования является ООО «ИнфоКом», а предметом исследования выступает процесс делового взаимодействия сотрудников организации с клиентами.

Для достижения поставленной цели необходимо решить следующие задачи:

– произвести анализ деятельности организации ООО «ИнфоКом»;

– рассмотреть Telegram bot API и API ВКонтакте, с которыми будет взаимодействовать проектируемое программное обеспечение;

– спроектировать программное обеспечение;

– выбрать средства реализации программного обеспечения;

– спроектировать базу данных;

– разработать web-интерфейс и реализовать проект.

1 ПРЕДПРОЕКТНЫЙ АНАЛИЗ

1.1 Анализ деятельности предприятия ООО «ИнфоКом»

1.1.1 Общие сведения о предприятии

В качестве объекта исследования выступает ООО «ИнфоКом» – коммерческое предприятие, зарегистрированное 24 апреля 2007 года. Руководителем предприятия является Иванов Евгений Александрович.

Основная деятельность предприятия заключается в предоставлении широкого спектра услуг, направленных на повышение эффективности и прибыльности работы других организаций. За годы своей деятельности предприятие зарекомендовало себя как надёжный партнёр, способный находить индивидуальные решения под конкретные запросы клиентов.

Организация предоставляет следующие услуги:

– внедрение и настройка CRM-системы Битрикс24;

– разработка веб-приложений и интеграций для Битрикс24;

– описание бизнес-процессов;

– автоматизация бизнес-процессов;

– построение системы управления задачами;

– построение системы аналитики и контроля продаж;

– аудит текущих процессов и формирование предложений по их оптимизации;

– автоматизация внешней коммуникации с клиентами, включая автоматизированное ведение диалога с клиентами по средствам программного обеспечения;

– оптимизация процесса сбора информации о клиентах, которая предполагает объединение информации о клиентах из писем, сообщений, социальных сетей и других источников в единую базу данных;

– систематизация информации о клиентах;

– разработка сайтов.

1.1.2 Организационная структура предприятия

Организационная структура предприятия представляет собой формальную систему, которая определяет, как управляются и координируются различные функциональные направления и подразделения в организации, а также определяет иерархические отношения между сотрудниками, структуру управления, потоки коммуникации, полномочия и обязанности сотрудников.

В ходе выполнения анализа организационной структуры предприятия ООО «ИнфоКом» была выявлена структура, представленная на рисунке 1.1.

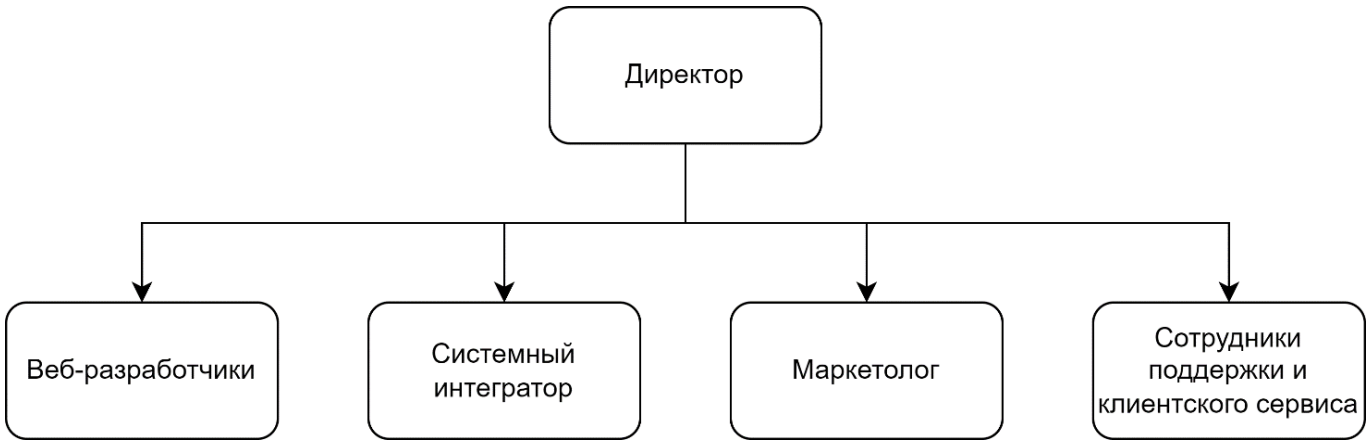

Рисунок 1.1 – Организационная структура предприятия ООО «ИнфоКом»

Как и в большинстве предприятий, во главе данного предприятия стоит директор, выполняющий следующие функции:

– осуществляет оперативное руководство деятельностью учреждения;

– представляет предприятие во всех учреждениях и организациях, в судах, как на территории России, так и за ее пределами;

– заключает сделки, договоры, соответствующие целям деятельности организации;

– обеспечивает сохранность и эффективное использование имущества;

– предоставляет в установленные сроки все виды отчетности, предусмотренные органами статистики, финансовыми и налоговыми органами;

– осуществляет бухгалтерский учёт;

– утверждает организационную структуру;

– принимает на работу и увольняет работников предприятия, заключает, изменяет и расторгает трудовые договоры с ними.

В подчинении директора находятся все работники предприятия: веб-разработчики, системный интегратор, маркетолог, сотрудники поддержки и клиентского сервиса.

Обязанности веб-разработчиков включают:

– разработку серверной части приложения;

– работу с базами данных, включая создание и управление базами данных;

– разработку и реализацию архитектуры приложения;

– создание веб-страниц.

Деятельность системного интегратора включает в себя:

– оценку потребностей бизнеса, анализ существующих систем и инфраструктуры, выявление потенциальных проблем и прогнозирование рисков;

– проектирование интегрированных решений с максимально подходящими компонентами к используемой инфраструктуре предприятия;

– разработку и внедрение решений, установку, настройку и объединение отдельных компонентов в единую систему;

– составление методических рекомендаций для пользователей продукции предприятия;

– тестирование и устранение проблем, которые возникают из-за внесенных изменений;

– внедрение Битрикс24.

Область ответственности маркетолога включает в себя:

– оценку эффективности рекламных каналов;

– анализ и подбор оптимальных рекламных каналов для клиентов;

– исследование конкурентов, включающее анализ их каналов продвижения, сильных и слабых сторон, позиционирования;

– исследование рынка, его объёма, трендов и составление прогнозов;

– разработку комплексной маркетинговой стратегии.

Сотрудники поддержки и клиентского сервиса отвечают за:

– обеспечение технической поддержки пользователей;

– консультацию клиентов по вопросам использования продуктов компании;

– проведение обучения и предоставление необходимой информации для эффективного использования программного обеспечения.

1.1.3 Документооборот предприятия

Документооборот предприятия представляет собой движение документов в организации с момента их создания или получения до завершения исполнения или отправки.

Документооборот предприятия разделяют на внешний и внутренний. Внешний документооборот представляет собой все входящие и исходящие документы компании, которыми она обменивается с организациями, клиентами и контролирующими органами, а внутренний описывает движение документов между структурными подразделениями и сотрудниками.

Схема внешнего документооборота данного предприятия приведена в приложении А.1.

В ней предприятие обменивается информацией с клиентами, ООО «Яндекс», управлением пенсионного фонда, фондом социального страхования, территориальным фондом обязательного медицинского страхования, отделением ПАО «СберБанк», межрайонной инспекцией ФНС, ООО «1С-Битрикс».

Данное предприятие осуществляет взаимодействие с клиентами по средствам оставляемых ими заявок на автоматизацию и требований, предъявляемых к проводимым работам, на основании которых предприятием формируется техническое задание, которое после формирования передаётся заказчику для согласования. После того как заказчик согласовал техническое задание начинается процесс разработки, по окончанию которого осуществляется приёмка работ. В следствии чего заказчиком в организацию предоставляется акт о приёме работ, а также заказчику передаётся счёт за предоставленные услуги.

Далее предприятие взаимодействует с ООО «Яндекс», арендуя у него вычислительные мощности. Для чего организация ООО «ИнфоКом» предоставляет ООО «Яндекс» заявку на аренду вычислительных мощностей. Затем ООО «Яндекс» предоставляется договор об аренде вычислительных мощностей и политику конфиденциальности, а также счёт за предоставленные услуги.

Также, как и любая другая организация ООО «ИнфоКом» взаимодействует с такими структурами как управление пенсионного фонда, фонд социального страхования, фонд обязательного медицинского страхования и инспекцией федеральной налоговой службы, которые предоставляют данной организации требования к отчётности, а организация ООО «ИнфоКом», непосредственно предоставляет им отчёты.

Кроме того, данное предприятие взаимодействует с отделением ПАО «СберБанк», предоставляя ему платёжные поручения и получая от него банковские выписки.

Также организация ООО «ИнфоКом» находится в технологическом партнёрстве с ООО «1С-Битрикс», для чего ООО «1С-Битрикс» предоставляет ООО «ИнфоКом» партнёрское предложение, а обратно получает согласованное партнёрское предложение.

Внутренний документооборот ООО «ИнфоКом» приведён в приложении А.2.

В данной организации ключевую роль во внешнем взаимодействии организацией играет директор и поэтому он осуществляет взаимодействие с ООО «Яндекс» арендуя у него вычислительные мощности, получает требования к отчётности и создаёт отчёты для различных государственных учреждений, согласовывает техническое задание, полученное от клиентов организации, а также рассматривает партнёрские предложения от ООО «1С-Битрикс». Кроме того, он издаёт указы, распоряжения и поручения, которые определяют порядок работы сотрудников и устанавливают цели и задачи для них.

Сотрудники, в свою очередь, следуют указаниям и распоряжениям директора, исполняя возложенные на них задачи и составляют отчёты о выполнении указаний и распоряжений.

Веб-разработчики получают согласованное техническое задание или распоряжения директора на основании которых осуществляется разработка программного обеспечения. Также разработчики составляют и передают журналы изменений программного обеспечения системному интегратору.

Системный интегратор в свою очередь, описывает изменения в программном обеспечении и предаёт их маркетологу и сотруднику поддержки и клиентского сервиса. Корме того системный интегратор получает от клиентов заявку на автоматизацию и требования к программному обеспечению, после чего рассматривает их и создаёт техническое задание, передовая его заказчику для согласования.

Маркетолог, получив описания обновлений программного обеспечения, проводит анализ этих нововведений и разрабатывает коммерческие предложения, после чего он направляет эти предложения клиентам.

Сотрудник поддержки и клиентского сервиса непосредственно взаимодействует с клиентами организации, рассматривая и отвечая на их обращения.

1.1.4 Анализ используемого программного обеспечения предприятием

В организации ООО «ИнфоКом» для обеспечения эффективного рабочего процесса используются различные ИТ-сервисы.

Одним из ключевых инструментов является CRM-система, предназначенная для управления взаимоотношениями с клиентами. Она предоставляет возможности для работы с маркетингом, продажами, обслуживанием клиентов и анализа данных. В организации используется CRM-система Битрикс-24.

Также в рабочем процессе задействована система управления версиями программного обеспечения GitHub, которая позволяет отслеживать изменения в исходном коде, возвращаться к предыдущим версиям и координировать работу между разработчиками.

Кроме того, в организации используется Enterprise Resource Planning-система (далее – ERP-система) 1С:ERP. Она объединяет данные организации в единую информационную среду, способствуя эффективному управлению ресурсами и повышению

общей продуктивности. Для организации электронного документооборота в ООО «ИнфоКом» применяется система ЭДО Контур.Диадок, интегрированная с 1С.

Для обеспечения информационной безопасности и защиты данных используются решения АО «Лаборатория Касперского», включая антивирусы и системы предотвращения несанкционированного доступа.

Также в ООО «ИнфоКом» активно используются популярные платформы для взаимодействия с клиентами, в частности Telegram и ВКонтакте. Для работы с ними применяются стандартные клиентские приложения, однако в условиях высокой нагрузки и большого количества обращений их использование становится неудобным. Отсутствие единого интерфейса, необходимость постоянного переключения между окнами и ограниченные возможности по контролю и отслеживанию переписки затрудняют работу сотрудников.

Это приводит организацию к необходимости разработки мультичата – системы для объедения различных систем, на подобие, Telegram и ВКонтакте, в единый интерфейс. Целью создания мультичата является организация централизованной обработки обращений клиентов, повышение скорости реагирования операторов и снижение риска пропуска сообщений. Основная функция мультичата заключается в сборе и пересылке входящих сообщений из указанных систем мгновенного обмена сообщений в единое информационное пространство, доступное сотрудникам компании.

1.2 Обзор существующих аналогов программного обеспечения

Перед началом разработки собственного программного обеспечения крайне важно провести анализ существующих решений, уже представленных на рынке. Анализ позволит оценить целесообразность создания программного обеспечения и выявить наиболее успешные решения, а также определить ключевые недостатки текущих решений, которые могут быть устранены в разрабатываемом программного обеспечения.

Аналогами разрабатываемого программного обеспечения являются мультичаты Jivo и Callibri, основные интерфейсы которых приведены на рисунках 1.2 и 1.3.

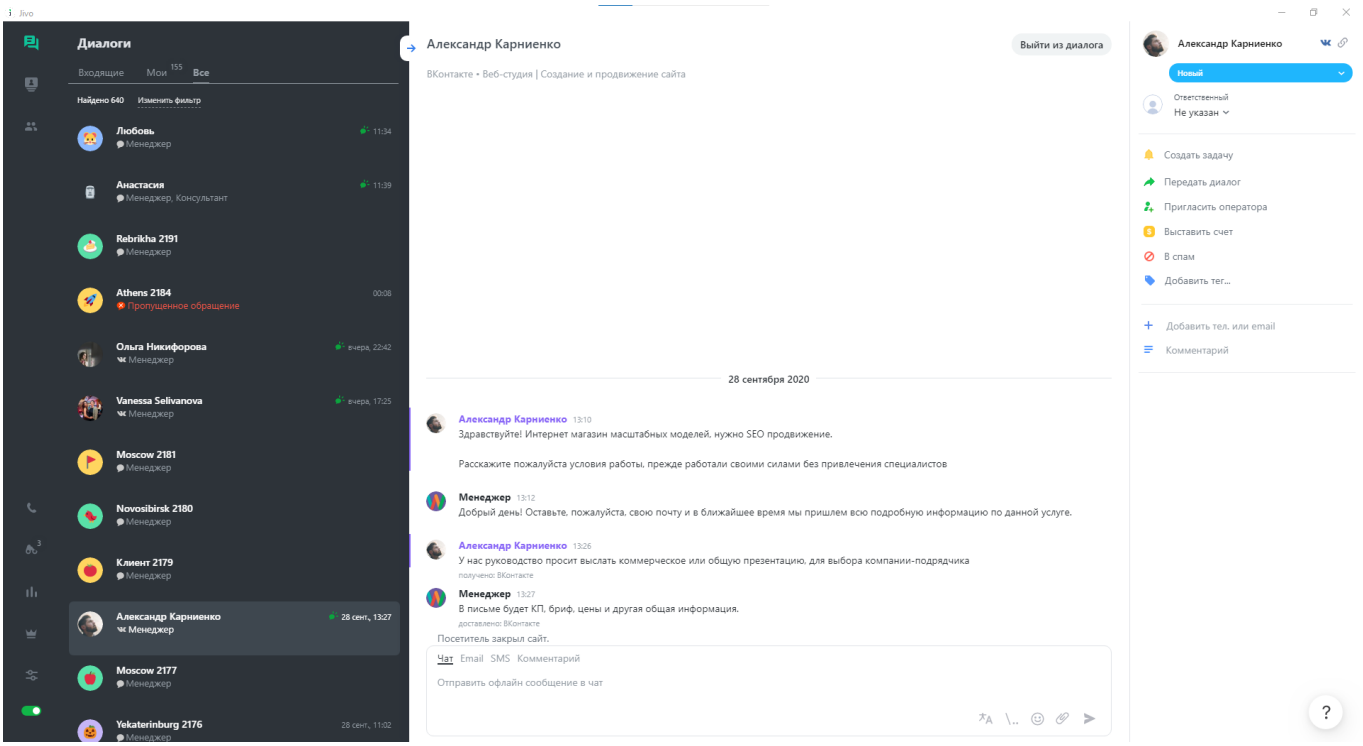

Рисунок 1.2 – Основной интерфейс Jivo

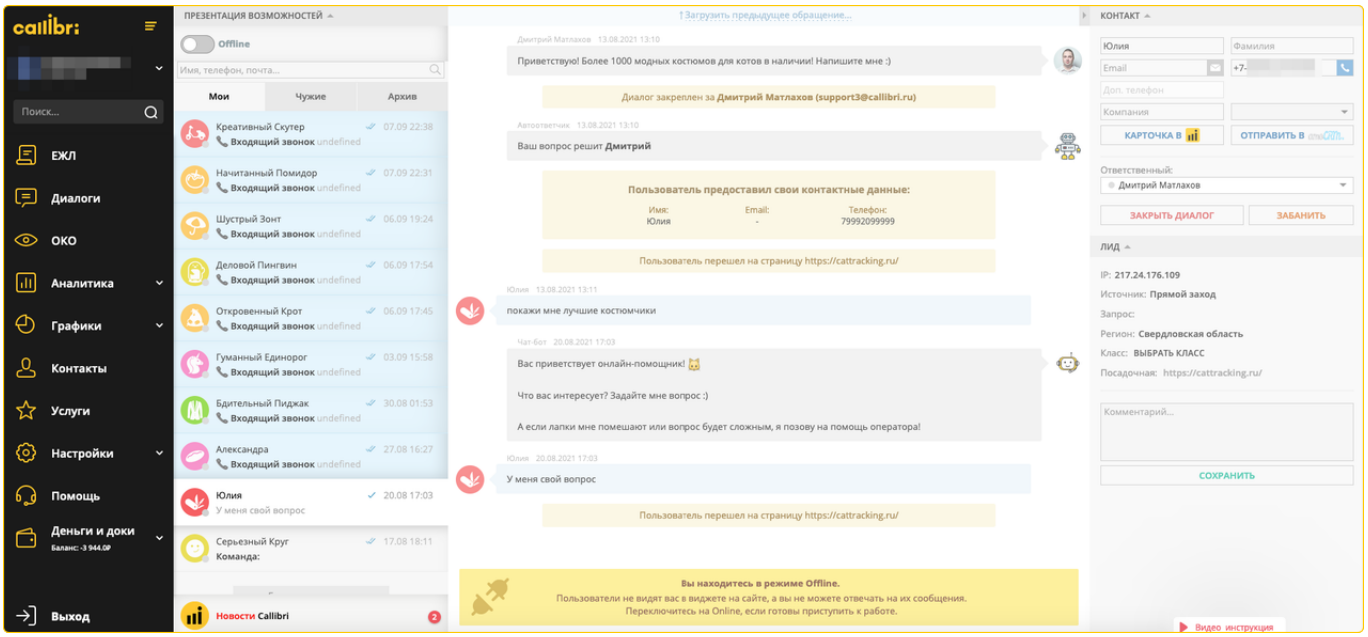

Рисунок 1.3 – Основной интерфейс Callibri

Оба решения предоставляют обширный набор инструментов для работы с клиентами, однако их функциональный фокус различается:

– Jivo ориентирован на омниканальную обработку сообщений, интеграцию с телефонией и улучшение качества обслуживания клиентов;

– Callibri предоставляет инструменты сквозной аналитики, оценку эффективности рекламных кампаний.

Для большей наглядности сравнение функциональных возможности обоих сервисов было оформлено таблице 1.1.

Таблица 1.1 – Сравнение функциональных возможностей Jivo и Callibri

Функция	Callibri	Jivo
Отслеживание рекламы	Коллтрекинг, email-трекинг, перехватчик форм, сквозная аналитика.	Коллтрекинг.
Чат для сайта	Есть.	Есть.
Интеграция с мессенджерами и социальными сетями	ВКонтакте, Telegram, WhatsApp, Одноклассники.	ВКонтакте, Telegram, WhatsApp, Viber.
Интеграция с электронной почтой	Нет.	Есть.
Интеграция SMS в чат	Нет.	Есть.
Чат-бот	Есть.	Есть.
AI-ассистент	Нет.	Есть.
Телефония	Есть.	Есть.
Обратный звонок	Есть.	Есть.
Таргетированные SMS-рассылки	Есть.	Есть.
Конструктор квизов	Есть.	Нет.
CRM	RetailCRM, Битрикс 24, amoCRM, Мегаплан.	Собственная.

Из сравнительного анализа видно, что Callibri лучше подходит для маркетинговой аналитики, тогда как Jivo делает упор на омниканальные коммуникации.

Однако при выборе оптимального решения важно учитывать не только функциональные возможности, но и стоимость использования каждого сервиса. Для большей наглядности сравнение тарифных планов сервисов было оформлено в таблице 1.2.

Таблица 1.2 – Сравнение тарифных планов Jivo и Callibri

Услуга	Callibri	Jivo
Мультичат	Пакет «Старт», 4.500 рублей в месяц, включает в себя: чат; форму заявки и отображение адресов; WhatsApp без доплат; подробную аналитику обращений	Пакет «Начальный» 0 рублей в месяц на человека, включает в себя: Чат для сайта; каналы Telegram, VK, Viber, Email; обратный звонок; онлайн-форму.

Продолжение таблицы 1.2

Услуга	Callibri	Jivo
	и работы менеджеров; автоматизацию диалогов. Пакет «Бизнес», 8.000 рублей в месяц, включает в себя: возможности пакета «Старт»; группы менеджеров; обратный звонок; квизы и чат-бот на сайт. Пакет «Премиум» – определяется по соглашению.	Пакет «Базовый», 742 рубля в месяц на человека, включает в себя: пакет «Начальный»; активные приглашения и кнопки быстрого начала чата; интеграция с Авито; шаблоны ответов для оператора; Jivo CRM. Пакет «Профессиональный», 1.117 рубля в месяц на человека, включает в себя: пакет «Базовый»; Интеграция с WhatsApp; кастомные атрибуты клиентов. Пакет «Корпаротивный», 2.617 рубля в месяц на человека, включает в себя: пакет «Профессиональный»; маршрутизация чатов; видео звонки.
Мульти трекинг	Пакет «S+» – 2.400 рублей в месяц. Пакет «M» – 5.000 рублей в месяц. Пакет «L» – 12.000 рублей в месяц. Пакеты отличаются объёмами доступных услуг.	Нет.
Чат-бот	Включён в основной тариф.	Пакет «Стартовый», 999 рублей в месяц, ограничен до 1500 пользователей в месяц. Пакет «Гибкий», (2.400 – 25.000) рублей в месяц, ограничен до 20.000 пользователей в месяц.
Обратный звонок	800 рублей в месяц за подключение и 10 рублей за минуту.	1.88 рублей за минуту.

Далее перейдём к сравнению возможностей обмена сообщениями, представленному в таблице 1.3.

Таблица 1.3 – Сравнение возможностей обмена сообщениями Jivo и Callibri

Функция	Callibri	Jivo
Отправка сообщений	Есть.	Есть.
Отправка файлов	Есть.	Есть.
Удаление сообщений со стороны клиента	Нет.	Нет.
Удаление сообщений со стороны сотрудника	Нет.	Нет.
Редактирование сообщений со стороны клиента	Нет.	Есть, но реализовано как отправка нового сообщения.
Редактирование сообщений со стороны сотрудника	Нет.	Нет.

Из данной таблицы видно, что ни у одного из решений нету возможности удаления сообщений, а также отсутствует возможность редактирования сообщений.

1.3 Обоснование необходимости разработки программного обеспечения

Несмотря на наличие на рынке мультичат-платформ, таких как Jivo и Callibri, разработка собственного решения остаётся обоснованной. Анализ этих сервисов показывает, что они предлагают достаточно широкий функционал, например, интеграции с телефонией, CRM и сквозная аналитика. Такой функционал избыточен для задач, ограниченных только обменом сообщениями. Пользователи переплачивают за ненужные функции и сталкиваются с перегруженным интерфейсом. Более того, обе платформы не поддерживают базовые функции редактирования и удаления сообщений, а в Jivo редактирование реализовано нестандартным образом – сообщение повторно отправляется в изменённом виде, что может запутать пользователя.

Также важно отметить, что собственное решение позволяет гибко адаптировать систему под внутренние процессы организации: изменить интерфейс, логику обработки сообщений и бизнес-логику, а также добиться тесной интеграции с корпоративными системами, такими как Битрикс24.

Кроме того, важно заметить, что Jivo и Callibri являются Software as a Service-решениями (далее – SaaS-решениями), то есть работают на стороне провайдера, что может приводить к проблемам с информационной безопасностью. В отличие от этого, разрабатываемый мультичат является коробочным решением, которое устанавливается на инфраструктуру заказчика, что обеспечивает полный контроль над данными, позволяет соответствовать внутренним стандартам безопасности и корпоративной политике, а также избавляет организацию от абонентских платежей.

Также важно отметить, что некоторые организации, использующие Битрикс24, осторожны в вопросах безопасности и неохотно устанавливают сторонние интеграции, опасаясь потенциальных утечек данных. Это особенно актуально для компаний с повышенными требованиями к конфиденциальности. В таких случаях коробочное решение не только предпочтительно, но и зачастую является единственно допустимым вариантом, соответствующим внутренней политике безопасности.

Таким образом, несмотря на стартовые издержки, собственный мультичат представляет собой более эффективное, гибкое и безопасное решение для задач, ограниченных только взаимодействием с Telegram и ВКонтакте. Он позволяет избежать избыточной сложности, обеспечить интеграцию с корпоративными системами и гарантировать полный контроль над всей системой – включая вопросы безопасности и доверия.

1.4 Рассмотрение API сторонних сервисов

Перед началом проектирования какого-либо программного обеспечения, взаимодействующего со сторонними сервисами, необходимо изучить API этих сторонних сервисов.

В данном пункте будут рассмотрены ключевые аспекты взаимодействия с API сторонних сервисов, включая механизмы аутентификации, принципы взаимодействия с сервисами, их функциональные возможности и ограничения.

1.4.1 Рассмотрение Telegram bot API

Telegram Bot API предоставляет разработчикам возможность взаимодействовать с Telegram посредствам HTTPS-запросов. Для вызова его методов необходимо использовать URL-адрес вида https://api.telegram.org/«token»/ «method», где «token» – токен, полученный от бота @BotFather, а «method» – название метода API. Все запросы осуществляются с помощью глагола POST. Передача параметров осуществляется в тете запроса в формате JSON, а ответы также возвращаются в теле ответа в формате JSON. [1]

Существует два способа получения событий от Telegram:

– Webhook – метод, при котором Telegram отправляет события на заранее указанный URL, как только они происходят.

– Long polling – метод, предполагающий периодическую отправку запросов к серверу Telegram для получения событий.

При использовании webhook, перед началом работы необходимо вызвать метод «setWebhook», предназначенный для указания URL-адреса на котором будут ожидаться события от Telegram, в его параметрах указывается обязательный параметр

«url» – URL-адрес на котором будут ожидаться события. Этот адрес обязательно должен использовать протокол HTTPS. Дополнительно можно передать параметр «secret_token» – строку длиной до 256 символов, которая будет включена в заголовок запроса «X-Telegram-Bot-Api-Secret-Token» для подтверждения подлинности запроса. После успешного выполнения метода, на указанный URL-адрес начнут поступать события в формате структуры Update, описание которой можно найти в документации.

При использовании long polling, события получают путём регулярных вызовов метода «getUpdates». В запрос можно включить параметр «offset», равный идентификатору последнего обработанного события плюс один, если такой идентификатор не известен, то в запросе параметр «offset» не указывается, а в результате будут получены события начиная с первого не полученного события. В результате успешного выполнения метода ответе будет возвращён массив структур Update, содержащих произошедшие события.

Далее важно отметить, что обычно, перед первым взаимодействием пользователя с ботом, боту автоматически отправляется команда «/start». Это позволяет извлечь основные сведения о пользователе, такие как имя, фамилия и идентификатор пользователя, из структуры Update в которую вложена структура Message, содержащая данные о пользователе в структуре данных User. Владея полученными данными, можно получить дополнительную информацию, например, фотографии профиля пользователя, можно получить методом «getUserProfilePhotos», однако этот метод вернёт массив структур UserProfilePhotos, содержащих только идентификаторы файлов фотографий, а для получения самого файла необходимо сначала получить URL-адрес файла путём вызова метода «getFile» который вернёт структуру «File» в которой будет содержатся поле «file_path», указывающее URL-адрес по которому можно скачать нужный файл.

Для отправки сообщений используется метод «sendMessage». Для отправки изображений, видео и документов применяются методы «sendPhoto», «sendVideo» и

«sendDocument», при этом файлы передаются через multipart/form-data. Также доступен метод «deleteMessage» для удаления ранее отправленного сообщения.

1.4.2 Рассмотрение API ВКонтакте

Аналогично Telegram, API ВКонтакте предоставляет разработчикам возможность взаимодействовать с системой посредствам вызова методов с помощью HTTPS-запросов с глаголом POST по URL-адресу формируемому по следующему шаблону: https://api.vk.com/method/«method», где «method» – название вызываемого метода. Передача параметров осуществляется в теле запроса в формате JSON. Обязательно с каждым запросом передают ключи «access_token» – аутентификационный токен сообщества, который можно получить в разделе «управление сообществом» в пункте «работа с API», и «v» – версия используемого API. Результат выполнения метода будет доступен в теле ответа в формате JSON. [2]

Как и в Telegram, в ВКонтакте для получения событий используются long polling и webhook, который в ВКонтакте называют «Callback API». Работа с API ВКонтакте имеет свои особенности.

Так настройки webhook задаются в разделе «управления сообществом» в пункте «работа с API» во вкладке «Callback API». Там необходимо указать адрес, на котором будут ожидаться события, а также секретный ключ, гарантирующий подлинность источника выполняемых запросов.

В отличии от Telegram API, API ВКонтакте проверяет правильность введённого адреса отправляя тестовый запрос типа «confirmation». В ответе на запрос, ВКонтакте будет ожидать секретную строку, указанную чуть ниже поля ввода адреса, если не отправить данную строку в ответе, то будет считается, что вы допустили ошибку и адрес не будет установлен.

Так же важно отметить что в подпункте «Типы событий» необходимо указать какие события должны приходить по данному адресу. Пример настроек webhook приведён на рисунке 1.4.

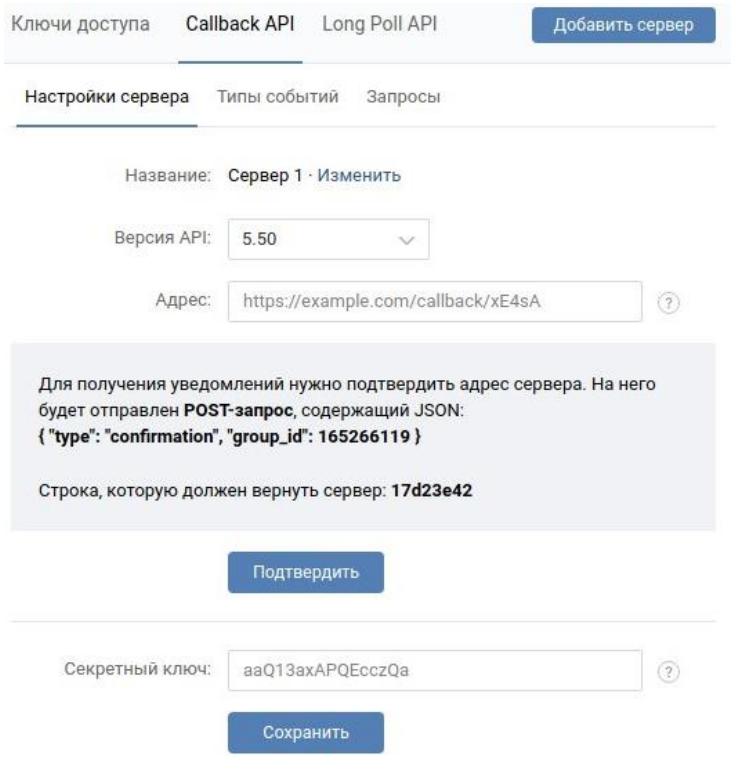

Рисунок 1.4 – Настройка webhook в ВКонтакте

После успешной установки адреса для webhook на него начнут приходить запросы с произошедшими событиями, указанными в теле запроса в формате JSON.

Для использования long polling необходимо открыть раздел «Управление сообществом» и на вкладке «Работа с API» в пункте «Long Poll API» установить флаг «Long Poll API» в положение «Включён». Также в пункте «Типы событий» нужно выбрать необходимые вам типы событий.

Далее для получения событий необходимо получить секретный ключ сессии и URL-адрес сервера, к которому будет производится запрос на получение событий, а также номер последнего события, начиная с которого нужно получать события. Собственно, чтобы получить данную информацию необходимо вызвать метод «groups.getLongPollServer». В результате успешного выполнения запроса будет получен ответ с ключами «key», «server» и «ts», секретный ключ, адрес сервера и номер события, соответственно.

Теперь используя, полученные данные, необходимо собрать адрес по следующему шаблону: «server»?act=a_check&key=«key»&ts=«ts»&wait=«wait», где «key» –

секретный ключ сессии, «server» – адрес сервера, «ts» – номер последнего события, начиная с которого нужно получать данные, «wait» – максимальное время ожидания ответа от сервера. Затем необходимо выполнить запрос по составленному адресу с глаголом POST. В ответе будет содержатся массив событий и ключ «ts» – номер последнего события, который нужно будет указать при выполнении следующего запроса.

В отличии от Telegram, во ВКонтакте перед началом взаимодействия пользователя с сообществом, не приходит событий начала диалога, поэтому о начале диалога можно узнать только после получения нового сообщения в событии типа «message_new».

Информацию о пользователях можно получить с помощью метода «users.get», указав идентификаторы пользователей, сведения о которых необходимо получить, и необходимые поля, например, «photo_max» для получения URL фотографии профиля.

Для отправки сообщений используется метод «messages.send». Важно отметить, что, чтобы прикрепить файл, необходимо сначала узнать URL-адрес для загрузи файла путём вызова метода «docs.getMessagesUploadServer». Затем выполнить загрузку файла по полученному адресу и сохранить файл методом «docs.save». Практически аналогично осуществляется отправка изображений и видео, но загрузка видео от имени сообщества не поддерживается. Для удаления сообщений применяется метод «messages.delete».

2 ПРОЕКТИРОВАНИЕ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

2.1 Общие сведения о программном обеспечении

Как было написано ранее, в рамках работы осуществляется проектирование мультичата, который представляет собой комплексную систему управления деловыми коммуникациями организации, объединяющей в едином интерфейсе обращения клиентов из различных каналов взаимодействия с организацией, например, мессенджеров и социальных сетей.

Целями реализации данного программного обеспечения являются:

– предоставление клиентам удобного способа асинхронного обращения в организацию через системы мгновенного обмена сообщениями;

– унификация процесса обмена информацией между клиентом и организацией с целью повышения оперативности и качества обслуживания клиентов организации;

– оптимизация внутренних процессов организации и снижение нагрузки на сотрудников.

Требования к проектируемому программному обеспечению были оформлены в виде технического задания в соответствии с ГОСТ 19.201. Полученное техническое задание приведено в приложении Б.

Исходя из требований к составу выполняемых функций проектируемого программного обеспечения была построена UML-диаграмма вариантов использования системы, представленная на рисунке В.1.

На данной диаграмме приведён пользователь, который может выполнять основные действия, необходимые для работы с чатом: просматривать сообщения в чате и отправлять сообщения в чат.

При этом важно отметить, что чат может находится в одном из трёх состояний:

– «Входящий» – состояние чата, соответствующее новому обращению клиента в организацию.

– «Активный» – состояние чата, в котором он находится в процессе обработки поступившего обращения.

– «Архив» – состояние чата, в котором обращение клиента считается завершённым.

Условия переходов между состояниями чата приведены на UML-диаграмме состояний, изображённой на рисунке 2.1.

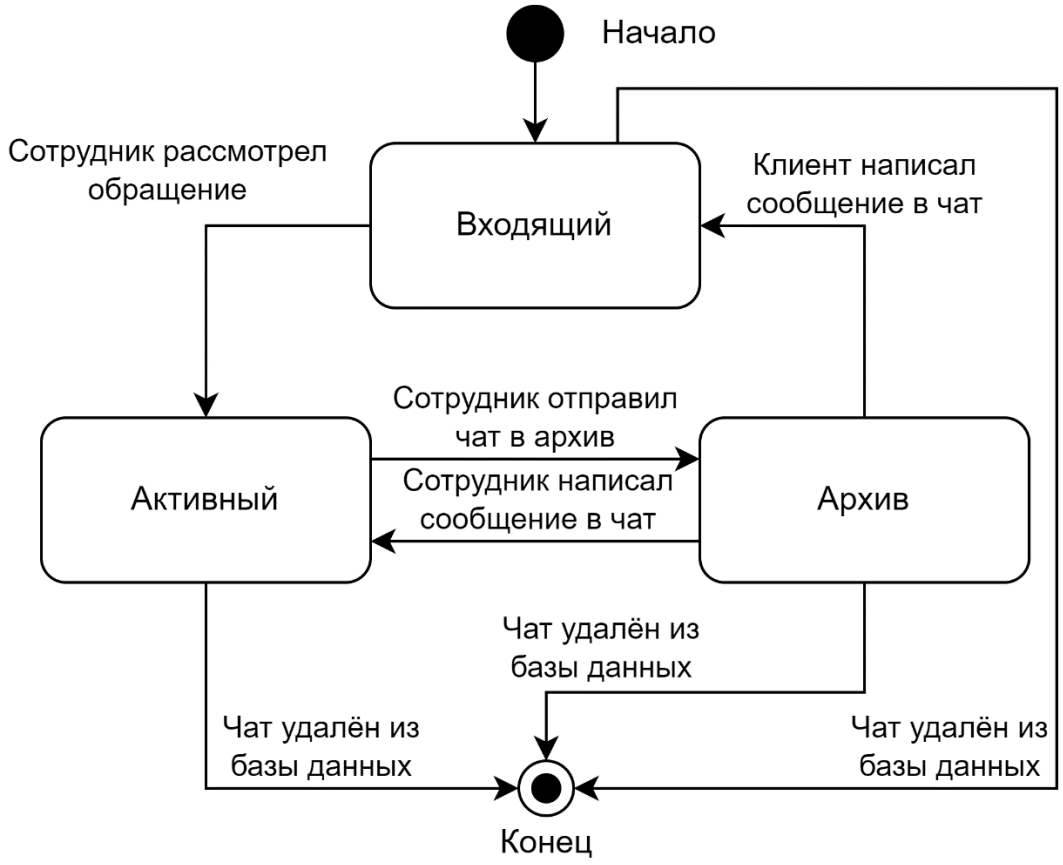

Рисунок 2.1 – UML-диаграмма состояний чата

Далее рассмотрим актёра «Клиент», который является пользователем и, соответственно, расширяет его варианты использования системы. Кроме этого клиент может обратится в организацию, что приведёт к созданию нового чата, если его ещё не существует, и переводу его в состояние «Входящий».

Сотрудник организации также является пользователем и, следовательно, может просматривать сообщения в чате и отправлять сообщения в чат. Также перед взаимодействием с программным обеспечением он должен пройти аутентификацию, после чего он может просмотреть список обращений клиентов и рассмотреть обращение,

что переведёт соответствующий чат в состояние «Активный» и добавлению сотрудника в чат. У сотрудника имеется возможность удалять отправленные им сообщения. При завершении обработки обращения сотрудник может убрать чат в архив. Также сотрудник может редактировать профиль, например, для смены аутентификационной информации.

Администратор системы представляет собой сотрудника, ответственного за управление пользователями. Он может просматривать список сотрудников, создавать их и редактировать сведения о них, а также удалять сотрудников из системы, если в этом возникает необходимость, например, при увольнении.

Также для удобства управления пользователями была предусмотрена система ролей пользователей, в которой администратор может просматривать имеющиеся роли, создавать, редактировать и удалять роли.

2.2 Выбор архитектуры web-приложения

Перейдём к выбору архитектуры web-приложения – ключевому аспекту проектирования, определяющему структуру программного обеспечения. Наиболее распространёнными видами архитектур web-приложений являются монолитная и микросервисная архитектуры.

Монолитная архитектура представляет собой концепцию разработки web-приложения как единого, неделимого, автономного web-сервиса, в котором одна кодовая база используется для выполнения нескольких бизнес-функций. Такое приложение может быть построено на основе принципов объектно-ориентированного подхода, соответственно, использующего встроенные механизмы взаимодействия внутри системы. [3]

Так, например, при создании онлайн-магазина в рамках монолитной архитектуры web-приложение будет представлять собой единый сервис, включающий в себя следующие модули: управления пользователями, каталога товаров, обработки заказов, оплаты.

Общая структура web-приложения на основе монолитной архитектуры приведена на рисунке 2.2.

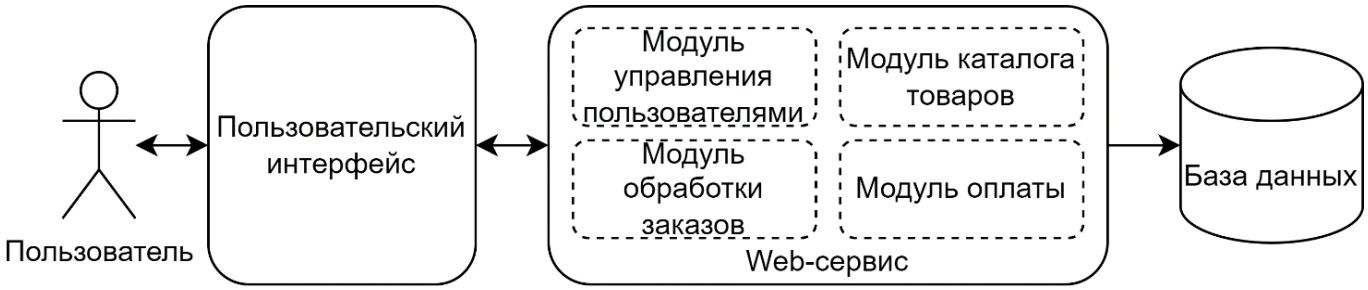

Рисунок 2.2 – Монолитная архитектура web-приложения

К преимуществам монолитной архитектуры относятся:

– простота разработки, отладки и тестирования;

– лёгкость развёртывания;

– высокая производительность за счёт локального взаимодействия компонентов.

Основными недостатками такой архитектуры могут выступать:

– затруднённость модификации отдельных частей системы, поскольку любое изменение кода влияет на всю систему и требует повторного её тестирования;

– сложности с горизонтальным масштабированием;

– высокая связность компонентов системы, из-за чего сбой в одном компоненте может повлиять на всю систему.

Микросервисная архитектура предлагает несколько иной подход к проектированию web-приложения. В ней осуществляется разбиение web-приложения на небольшие автономные, слабо связанные сервисы, каждый из которых имеет свою кодовую базу и выполняет определённую часть бизнес логики всего переложения, взаимодействуя с другими сервисами через четко определённый интерфейс, как правило, построенный на основе сетевого взаимодействия. [4]

Также важно отметить что при использовании микросервисной архитектуры может использоваться шлюз программного интерфейса, который будет объединять интересы различных сервисов в один для удобства их использования. Шлюз также может обеспечивать балансировку нагрузки и защиту от несанкционированного доступа.

Возвращаясь к примеру, с онлайн магазином, система будет разбита на 4 микросервиса: микросервис управления пользователями, микросервис каталога товаров, микросервис обработки заказов и микросервис оплаты.

Общая структура программного обеспечения на основе микросервисов приведена на рисунке 2.3.

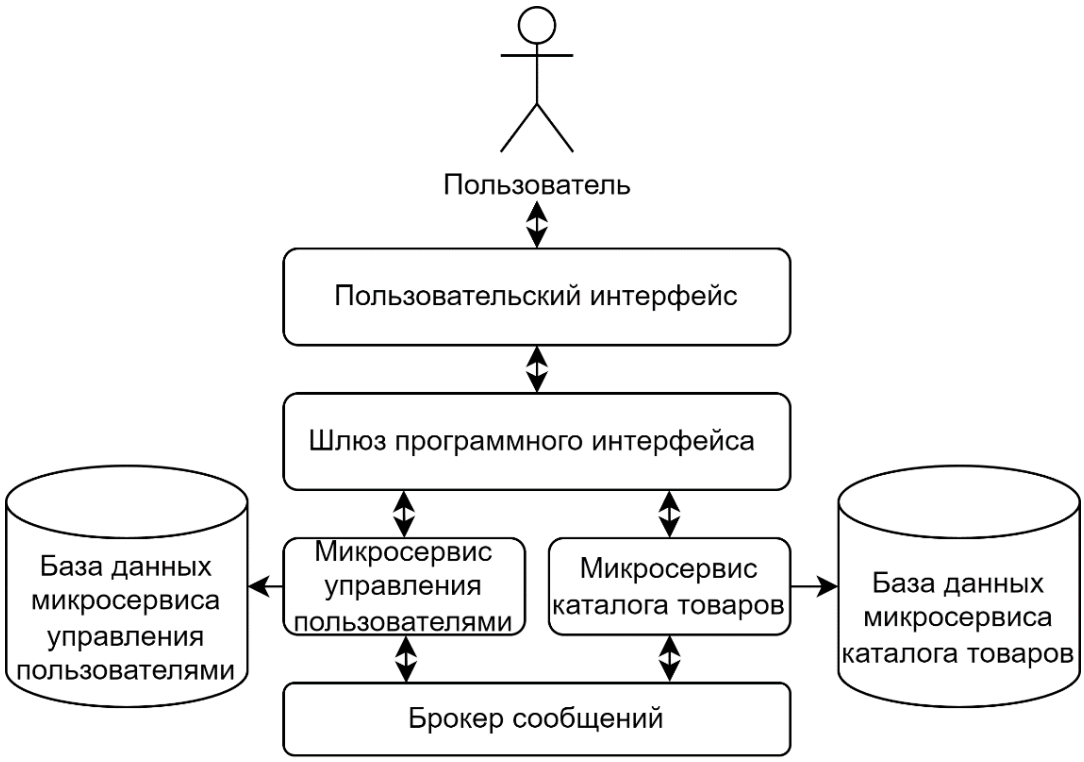

Рисунок 2.3 – Микросервисная архитектура web-приложения

К достоинствам микросервисной архитектуры относят:

– лёгкость в модификации приложения в силу его декомпозиции на отдельные сервисы, независимые друг от друга;

– независимость от конкретного языка программирования;

– повышение отказоустойчивости, в силу слабой связанности сервисов. [5]

Основными недостатками микросервисной архитектуры являются:

– сложность отладки и тестирования;

– сложность управления программным обеспечением;

– меньшая эффективность по сравнению с монолитной архитектурой, за счёт накладных расходов на межсервисное взаимодействие.

Таким образом, из рассмотренных архитектур была выбрана микросервисная архитектура в силу того, что:

– Монолитный подход оказался менее предпочтительным, поскольку он не обеспечивает достаточной гибкости при разработке программного обеспечения. Например, добавление новой системы мгновенного обмена сообщениями требует внесения изменений и развертывания новой версии приложения, что может привести к приостановке обслуживания клиентов, что недопустимо для организации.

– Монолитная архитектура уступает в отказоустойчивости, так как сбой одного компонента может привести к выходу из строя всей системы. Микросервисный подход, напротив, обеспечивает изоляцию сервисов, позволяя им функционировать независимо друг от друга и снижая риск полного отказа.

– Микросервисная архитектура обладает преимуществом в плане масштабируемости. Возможность разворачивать, масштабировать и обновлять каждый сервис отдельно облегчает адаптацию системы к изменяющимся требованиям и росту нагрузки, сводя к минимуму необходимость серьёзных изменений во всей архитектуре приложения.

2.3 Общая архитектура web-приложения

Следуя общей структуре микросервисной архитектуры, можно разбить проектируемое программное обеспечение на несколько микросервисов, представленных на рисунке Г.1.

На данном рисунке представлено четыре микросервиса, основным из которых является микросервис мгновенного обмена сообщениями. Он предназначен для формирования собственной системы мгновенного обмена сообщениями с целью организации обмена сообщениями пользователей Telegram и ВКонтакте с пользователями web-клиента разрабатываемого программного обеспечения. Основной задачей данного микросервиса является организация обработки и маршрутизации событий в системе, а также хранение информации необходимой для организации системы мгновенного обмена сообщениями.

Микросервис Telegram бота предназначен для организации взаимодействия с клиентами организации, посредствам Telegram. Основными назначениями данного микросервиса являются:

– аутентификация и авторизация пользователей Telegram;

– приём сообщений от пользователей Telegram и их преобразование во внутренний формат системы с последующей их пересылкой микросервису мгновенного обмена сообщениями;

– приём сообщений от микросервиса мгновенного обмена сообщениями и их отправка пользователям Telegram от имени бота.

Микросервис бота ВКонтакте выполняет те же функции, что и микросервис Telegram бота, но взаимодействует не с Telegram, а с ВКонтакте.

Web-клиент данной системы разделён на две части бэкенд и фронтенд – серверную и клиентскую части программы, соответственно.

Основным назначением бэкенда web-клиента является аутентификация и авторизация пользователя web-клиента, а также обработка запросов, поступающих с фронтенда с последующей их модификацией и перенаправлением микросервису мгновенного обмена сообщениями. Ещё одними предназначением бэкенда web-клиента, является обработка и маршрутизация сообщений, направляемых микросервисом мгновенного обмена сообщениями к фронтенду web-клиента.

Фронтенд представляет собой пользовательский интерфейс web-приложения, которым пользуются сотрудники организации для взаимодействия с клиентами.

Каждый микросервис имеет собственную базу данных, предназначенную для хранения информации, необходимой для его функционирования. Кроме того, в системе предусмотрено S3 хранилище, обеспечивающее хранение файлов, доступных для всех микросервисов.

2.4 Проектирование API микросервисов

Проектирование API микросервисов было выполнено на основе вариантов использования системы. Для каждого ключевого сценария взаимодействия между ком-

понентами системы были разработаны UML-диаграммы последовательностей, приведённые в приложении Е. Используемые структуры данных и описания методов находятся в приложениях Ж и Й, соответственно.

Также важно отметить что API бота ВКонтакте, практически идентичен API Telegram бота, за исключением того, что в место вызовов API ВКонтакте происходит вызов API Telegram, поэтому дальнейшее проектирование будет проводится на примере Telegram бота.

Работа системы начинается с инициализации компонентов. Так Telegram бот при запуске регистрирует webhook в Telegram API и платформу в микросервисе мгновенного обмена сообщениями. Аналогично бэкенд web-клиента осуществляет регистрацию платформы и дополнительно проверяет наличие администратора в системе, создавая его при необходимости. Эти процессы отражены в диаграммах последовательностей, приведённых на рисунках Д.1 и Д.4 в последовательностях 1.

Также в web-интерфейсе системы должна быть предусмотрена аутентификация сотрудников организации, реализуемая вариантом использования «Аутентифицироваться». Он осуществляется посредствам ввода логина и пароля в web-интерфейс. Введённые данные передаются на бэкенд web-клиента, с помощью вызова метода «Аутентификация сотрудника», возвращающего JWT токен, который в дальнейшем будет использоваться для авторизации. Осуществление аутентификации сотрудника приведено на рисунке Д.4 в последовательности 2.

Теперь перейдём к управлению пользователями в системе. Так при открытии раздела управления пользователями, для формирована графического интерфейса необходимо получить списки пользователей и ролей. Для этого фронтенд вызывает соответствующие методы бэкенда web-клиента, который извлекает нужные данные из базы данных. Этот процесс отображён на рисунке Д.4 в последовательности 3. Далее администратор может создавать, редактировать или удалять роли и пользователей – эти действия отражены в диаграммах Д.5 и Д.6.

Далее перейдём к сотруднику организации. Он после успешной аутентификации попадает в раздел обращений клиентов. Для формирования, графического интерфейса которого фронтенд получает список доступных чатов обращаясь к бекенду web-клиента, который, в свою очередь, запрашивает данные у микросервиса мгновенного обмена сообщения в последовательности 2 на рисунке Д.7.

Поскольку в системе пока не зарегистрировано ни одного клиента, перейдём к процессу его регистрации. Когда клиент начинает диалог с Telegram ботом, Telegram автоматически отправляет команду «/start» боту. Это инициирует сценарий «Обратиться в организацию»: бот проверяет наличие пользователя, собирает информацию об обратившемся клиенте, загружает фотографию его профиля в S3 хранилище, и регистрирует пользователя в микросервисе мгновенного обмена сообщениями. Также важно отметить, что в процессе регистрации пользователя будет создан новый чат для созданного пользователя. Необходимость создания нового чата обусловлена спецификой работы бота, так как фактически у пользователя в интерфейсе Telegram будет один чат с ботом и у него не будет возможности сменить чат. После регистрации происходит распространение события «Чат обновлён». Данный процесс описан последовательностями 2 и 6, приведёнными на рисунках Д.1 и Д.8, соответственно.

После регистрации пользователя, Telegram бот будет готов к приёму сообщений от клиента. Когда клиент отправит сообщение боту, Telegram передаёт его с помощью вызова метода «webhook» Telegram бота, который в свою очередь преобразует сообщение во внутренний формат, загрузит файлы в S3 хранилище и вызовет метод «Отправить сообщение» микросервиса мгновенного обмена сообщениями. Важно отметить, что, если чат, в который было отправлено сообщение, находился в состоянии «Архив», микросервис мгновенного обмена сообщениями изменит его состояние на «Входящий», если сообщение было отправлено клиентом организации, или «Активный», если сообщение было отправлено сотрудником организации. В случае изменения состояния чата микросервис мгновенного обмена сообщениями распространит событие «Чат обновлён». Осуществление данного процесса приведено на рисунках Д.2, Д.8 и Д.9 в последовательностях 3, 6 и 8.

После получения сообщения от клиента, сотрудник неизбежно захочет просмотреть сообщения в чате, что приведёт к тому, что он откроет некоторый чат в web-интерфейсе, для формирования которого фрнтенд web-клиента загрузит сообщения чата и состав его участников. Данный процесс приведён на рисунке Д.7 в последовательности 3.

После просмотра сообщений чата сотрудник может захотеть отправить ответ, однако для этого он должен быть участником чата. В случае отсутствия в чате, фронтенд web-клиента инициирует вызов метода «Присоединиться к чату» бэкенда web-клиента, который в свою очередь вызывает одноимённый метод микросервиса мгновенного обмена сообщениями. После добавления пользователя в чат микросервис мгновенного обмена сообщениями распространяет событие «Пользователь добавлен в чат», что приведено в последовательностях 4 и 5 на рисунке Д.8.

После присоединения к чату сотрудник может отправить сообщение. Он введёт текст и, при необходимости, прикрепит файлы. После чего фронтенд web-клиента сначала загрузит файлы в S3 хранилище через метод «Загрузить файл» бэкенда web-клиента, затем сформирует сообщение и вызывает метод «Отправить сообщение». Бэкенд web-клиента перешлёт его в микросервис мгновенного обмена сообщениями, который в свою очередь вызовет событие «Новое сообщение». Этот процесс описан на рисунке Д.9 в последовательностях 7 и 8 и рисунке Д.3 в последовательности 4.

Также у сотрудника имеется возможность удалить отправленное им сообщение. В этом случае фронтенд вызовет метод «удалить сообщение» бэкенда web-клиента, что приведёт к распространению события «Сообщение удалено». Данные действия приведены на рисунке Д.10 в последовательностях 10 и 11, а также на рисунке Д.3 в последовательности 5.

Завершая рассмотрение вариантов использования рассмотрим последний вариант использования «Отправить чат в архив». Данный вариант использования может понадобится сотруднику кода общение с клиентом будет прекращено, чтобы он не мешался при работе с системой. Для этого фронтенд web-клиента вызывает метод

«Отправить чат в архив» бэкенда, который передаёт соответствующий запрос в микросервис обмена сообщениями. Тот изменит состояние чата на «Архив» и удалит всех сотрудников из его состава, а также инициирует распространение события «Чат обновлён». Этот процесс отражён на рисунке Д.10 в последовательности 11 и рисунке Д.8 в последовательности 6.

2.5 Выбор средств реализации проекта

Средства реализации программного проекта оказывают существенное влияние на эффективность разработки и удобство сопровождения программного обеспечения. В данном пункте обосновывается выбор языков программирования, фреймворков, библиотек и вспомогательных инструментов, использованных при создании мультичата.

2.5.1 Выбор средств реализации серверной части

При выборе языка программирования для серверной части проекта было важно найти оптимальный баланс между удобством разработки, производительностью и поддержкой современных подходов к построению web-приложений. В качестве основных вариантов рассматривались JavaScript или TypeScript с Node.js, Go и Python.

Node.js предлагает высокую производительность и является распространённым выбором для web-приложений, однако его экосистема ориентирована на JavaScript, который сложнее поддерживать в больших проектах из-за отсутствия строгой типизации, если не использовать TypeScript.

Go демонстрирует отличную производительность, особенно при работе с сетевыми соединениями, но требует большего времени на освоение и более сложен в отладке.

Python поддерживает асинхронную обработку с помощью asyncio, имеет мощную экосистему, обладит удобным и читаемым синтаксисом, а также отлично подходит для быстрого прототипирования и реализации основой бизнес логики.

Таким образом, Python стал оптимальным выбором с точки зрения баланса между удобством разработки и техническими возможностями.

Далее, после выбора языка программирования, необходимо выбрать серверный фреймворк. Наиболее популярными решениями для Python являются Django, Flask и FastAPI.

Django – мощный и зрелый фреймворк, отлично подходящий для монолитных приложений, однако в нём ограничена поддержка асинхронности. Django требует настройки дополнительных компонентов для работы с WebSocket, что усложняет архитектуру.

Flask – более лёгкий фреймворк, но не имеет встроенной поддержки асинхронного кода и WebSocket. Для подобных задач необходимо использовать сторонние библиотеки, что может повлиять на надёжность и поддержку.

FastAPI изначально разрабатывался с акцентом на асинхронность и высокую производительность. Он поддерживает как Representational State Transfer Application Programming Interface (далее – REST API), так и WebSocket, легко масштабируется и предоставляет встроенную документацию.

Учитывая требования проекта к скорости, эффективности, асинхронной обработке соединений и простоте разработки, FastAPI стал наиболее подходящим выбором.

Теперь перейдём к выбору, используемых баз данных.

Для хранения информации о пользователях, сообщениях и другой структурированной информации рассматривались следующие СУБД: PostgreSQL, MySQL и MongoDB.

MongoDB – нереляционная СУБД, ориентированная на хранение документов. Она эффективна при работе с неструктурированными данными, но не обеспечивает должного уровня надёжности и целостности, необходимого для работы с пользовательской информацией и историей сообщений.

MySQL является популярной реляционной СУБД, однако в некоторых аспектах уступает PostgreSQL по гибкости запросов, поддержке современных стандартов и расширяемости.

PostgreSQL обеспечивает надёжное хранение структурированных данных, поддержку индексов, полнотекстового поиска и масштабируемость, а также высокую стабильность.

Учитывая это, PostgreSQL была выбрана как оптимальная СУБД для проекта.

Для реализации кэша в системе необходима резидентная база данных. Наиболее популярными из которых являются Redis и Memcached.

Memcached представляет собой простую, но эффективную систему кэширования, предназначенную для хранения данных в памяти. Memcached использует модель хранения данных по ключу-значению, что делает его очень простым и быстрым инструментом для кэширования. Однако его возможности ограничены, поскольку он не поддерживает работу с расширенными структурами данных, такими как списки, множества или хэш-суммы, что ограничивает его применение в более сложных сценариях.

Redis – это высокопроизводительная система хранения данных в памяти, которая поддерживает множество структур данных и предоставляет расширенные возможности для работы с временными данными и кэшированием. В отличие от Memcached, Redis не ограничивается лишь хранением данных по ключу-значению, а поддерживает разнообразные структуры, такие как строки, списки, множества, хэш-суммы, битовые карты и геопространственные индексы.

Redis является наиболее универсальным решением, так как:

– поддерживает разнообразные структуры данных;

– работает исключительно в памяти, обеспечивая высокую скорость;

– имеет механизм издатель-подписчик, что позволяет использовать его для реализации простого механизма рассылки сообщений.

В следствии этого Redis выбран как основная резидентная база данных, идеально подходящая для задач чата.

Также для хранения файлов необходимо S3 хранилище. Для которого предъявляется основное требование в виде совместимости с Amazon Web Services Command

Line Interface (далее – AWS CLI). Наиболее популярными решениями являются: Amazon S3, локальное файловое хранилище и MinIO.

Amazon S3 – облачное хранилище от Amazon, индустриальный стандарт. Предоставляет высокую доступность, отказоустойчивость и безопасность. Однако его использование требует оплату облачной инфраструктуры.

Локальное файловое хранилище – простой вариант, но плохо масштабируется, не поддерживает распределённое хранение и не даёт возможности тонко управлять доступом к файлам.

MinIO – лёгкое, высокопроизводительное AWS CLI-совместимое хранилище с открытым исходным кодом. Может быть развёрнуто локально или в облаке, поддерживает отказоустойчивость и шифрование.

В качестве S3 хранилища был выбран MinIO в силу того, что:

– он поддерживает API Amazon S3, в следствии чего хорошо интегрируется с FastAPI и Python-стеком;

– идеально подходит для проектов с ограниченным бюджетом, в следствии открытого исходного кода и наличия уже готового контейнера Docker.

Для взаимодействия с PostgreSQL в проекте была выбрана связка SQLAlchemy и Alembic. SQLAlchemy обеспечивает гибкую и мощную работу с базой данных, позволяя удобно описывать модели и строить запросы. Alembic дополняет её возможностями по управлению миграциями базы данных. В качестве альтернатив SQLAlchemy рассматривались: Peewee, Tortoise ORM и Django ORM.

Peewee – лёгкая ORM, которая хорошо подходит для небольших проектов, но обладает ограниченной функциональностью, особенно при работе со сложными связями и запросами.

Tortoise ORM – асинхронная ORM, более простая, но менее зрелая по сравнению с SQLAlchemy, с меньшей гибкостью и менее развитыми средствами работы с миграциями.

Django ORM – встроена в Django и тесно с ним связана. Не является универсальным решением, не используется вне экосистемы Django.

2.5.2 Выбор средств реализации клиентской части

Для реализации клиентской части был выбран язык программирования JavaScript, поскольку он является стандартом для разработки web-интерфейсов и поддерживается всеми современными браузерами без необходимости в дополнительных надстройках.

В качестве основного JavaScript-фреймворка был выбран Vue.js. В качестве его альтернатив рассматривались React и Angular.

React отличается широкой экосистемой и гибкостью, однако требует больше ручной настройки архитектуры и менее прозрачен для начинающих разработчиков.

Angular, в свою очередь, предоставляет комплексное решение «всё в одном», но является избыточным для проекта подобного масштаба и требует значительного времени на освоение.

А Vue.js был выбран за его низкий порог входа, лаконичную компонентную модель, встроенную реактивность и высокую интеграцию с современными инструментами разработки.

Для ускорения разработки пользовательского интерфейса и получения визуально завершенного результата была использована библиотека PrimeVue. В качестве её альтернатив рассматривались Vuetify и Element Plus.

Vuetify, реализующий концепцию Material Design, отличается тяжеловесной структурой и требует значительных усилий при использовании.

Element Plus предоставляет приятный визуальный стиль, но уступает по полноте компонентной базы и гибкости.

PrimeVue был выбран как наиболее сбалансированное решение: оно предоставляет широкий набор визуальных компонентов, хорошо интегрируется с Vue и TailwindCSS и активно поддерживается.

В качестве инструмента сборки проекта был выбран Vite. Основными альтернативами были Webpack и Parcel.

Webpack обладает мощными возможностями настройки, но значительно уступает по скорости и удобству в современных условиях.

Parcel предлагает автоматическую конфигурацию, но менее гибок и медленнее при работе с Vue.

Vite обеспечивает моментальный запуск проекта, горячую перезагрузку без задержек, нативную поддержку Vue и простую настройку, что делает его оптимальным решением для быстрого и комфортного процесса разработки.

Для управления глобальным состоянием приложения применена библиотека Pinia – официальное решение для Vue. Среди её альтернатив есть Vuex, однако он ранее был стандартом в экосистеме Vue, но требовал большого количества шаблонного кода и в связи с чем морально устарел.

2.5.3 Выбор средств развёртывания программного обеспечения

Для развёртывания программного обеспечения была выбрана контейнерная платформа Docker, обеспечивающая высокую степень изоляции, повторяемость окружения и удобство масштабирования. Контейнеризация позволила развернуть каждую составляющую системы – серверную часть, клиентское приложение, базы данных, кэш, файловое хранилище и обратный прокси – в отдельных изолированных средах, что минимизирует вероятность конфигурационных конфликтов и упрощает управление зависимостями.

Основными преимуществами Docker стали лёгкость сборки образов, быстрое развертывание на различных платформах, поддержка Docker Compose для оркестрации нескольких сервисов, а также широкое сообщество и наличие готовых образов для большинства используемых компонентов. Контейнеры позволяют с высокой точностью воспроизводить среду развёртывания сервера при локальной разработке.

В качестве альтернатив Docker рассматривались решения Podman и LXC.

Podman обеспечивает улучшенную безопасность и отказ от фонового демона, однако пока уступает Docker в зрелости инструментов и распространённости в промышленной среде.

LXC даёт более низкоуровневый контроль, но требует существенных усилий при конфигурировании и не имеет столь развитой экосистемы.

Учитывая это, Docker стал оптимальным выбором для обеспечения гибкости, стабильности и удобства развёртывания.

Для маршрутизации запросов и управления входящим трафиком был выбран Traefik – динамический обратный прокси-сервер, специально разработанный для микросервисных и контейнерных сред. Он автоматически обнаруживает новые сервисы, запущенные в Docker, и обновляет конфигурацию маршрутизации без прерывания работы системы. Это позволяет сократить объём ручной настройки и ускорить внедрение изменений.

Среди альтернатив Traefik рассматривались Nginx и Caddy.

Nginx отличается высокой производительностью и широким применением, однако требует статической конфигурации и перезапуска при каждом изменении маршрутов.

Caddy предлагает простую настройку и автоматическую работу с TLS, но уступает Traefik по гибкости и возможностям интеграции с Docker. В свою очередь, Traefik обеспечивает автоматическое получение и продление TLS-сертификатов через Let's Encrypt, имеет удобную систему меток для маршрутов и визуальную панель мониторинга.

2.6 Проектирование базы данных

2.6.1 Инфологическое проектирование

Проектирование базы данных состоит из трёх этапов: инфологического, логического и физического проектирования.

Первым этапом в инфологическом проектировании базы данных является выделение сущностей базы данных из предметной области. [6] Так из архитектурного проекта можно выделить следующие сущности:

– «Платформа», описывает конвертирующие микросервисы, подключённые к микросервису мгновенного обмена сообщениями.

– «Пользователь» – сведения о конкретном пользователе.

– «Пользователь Telegram» – расширение сущности «Пользователь» дополнительными сведениями, необходимыми для работы с Telegram.

– «Пользователь VK» – расширение сущности «Пользователь» дополнительными сведениями, необходимыми для работы с ВКонтакте.

– «Пользователь Web» – расширение сущности «Пользователь» дополнительными сведениями, необходимыми для организации web-интерфейса системы.

– «Роль» – разрешения пользователей web-интерфейса в системе.

– «Чат» – сведения о чатах.

– «Участники чата» – ассоциативная сущность, описывающая наличие пользователя в чате.

– «Сообщение» – отправленное пользователем сообщение.

– «Сообщение Telegram» – сущность, используемая для сопоставления внутренних сообщений с сообщениями Telegram.

– «Сообщение VK» – сущность, используемая для сопоставления внутренних сообщений с сообщениями ВКонтакте.

Следующим этапом инфологического проектирования базы данных является определение атрибутов сущностей, а также выделение ключевых атрибутов. По результатам выполнения данных действий были составлены таблицы с описаниями атрибутов сущностей. Данные таблицы приведены в приложении И.

Также важно отметить, что сущности «Пользователь Telegram», «Пользователь VK» и «Пользователь Web» представляют собой производные сущности от сущности «Пользователь». Данные зависимости можно реализовать в базе данных следующими способами:

– объединить атрибуты всех сущностей в одной сущности;

– оставить все сущности неизменными и независимыми;

– оставить базовую сущность неизменной, но в производных сущностях оставить только уникальные атрибуты, не содержащиеся в базовой сущности.

Выбор конкретного решения исходил из того, что первый вариант требует много дискового пространства, второй усложняет работу из-за фрагментации данных по различным таблицам, а третий позволяет сохранить целостность и удобство, поэтому был выбран именно он.

Далее в инфологическом проектировании базы данных необходимо определить связи между сущностями. Так между представленными сущностями можно выделить следующие связи: «Платформа – Пользователь», «Платформа – Чат», «Пользователь – Пользователь Telegram», «Пользователь – Пользователь VK», «Пользователь Telegram – Чат», «Пользователь VK – Чат», «Пользователь – Пользователь Web», «Пользователь Web – Роль», «Пользователь – Участники чата», «Чат – Участники чата», «Участники чата – Сообщение», «Чат – Сообщение», «Сообщение – Сообщение VK», «Сообщение – Сообщение Telegram», «Сообщение – Пользователь». Описание данных связей представлено в таблице И.12, а полученная в результате инфологического проектирования ER-диаграмма приведена на рисунке И.1.

2.6.2 Логическое и физическое проектирование

Следующим этапом проектирования базы данных является логическое проектирование, которое включает в себя отображение концептуальной модели базы данных на реляционную и нормализацию отношений.

Для отображения концептуальной модели базы данных на реляционную необходимо преобразовать сущности в отношения и связи между сущностями в связи между отношениями.

Преобразование связей происходит по следующим правилам:

– Для связей кратности 1 к 1 с обязательными классами принадлежности обоих сущностей образуется одно отношение, содержащие все атрибуты двух сущностей, а первичным ключом отношения становится любой первичный ключ этих сущностей.

– Для связей кратности 1 к 1 с разными классами принадлежности сущностей образуется два отношения с атрибутами, соответствующими атрибутам сущностей, при этом сущность класс принадлежности, которой не обязательный будет является родительским отношением, а другая дочерним и для установления связи первичный ключ родительского отношения добавляется в дочернее.

– Для связей кратности 1 к 1 и не обязательным классом принадлежности для обоих сущностей, образуется три отношения по одному отношению на сущность и одно отношение для связи, содержащее первичные ключи двух сущностей.

– В случае связей кратности 1 к M, если для многосвязной сущности класс принадлежности является обязательным, то родительской сущностью становится односвязная сущность, а дочерней много связная и тогда образуется два отношения, соответствующих сущностям и в дочернюю сущность, добавляется первичный ключ родительской сущности, иначе, в случае если многосвязная сущность имеет не обязательный класс принадлежности, образуется три отношения по одному отношению на сущность и одно отношение для связи, содержащее первичные ключи двух сущностей.

– В случае связи кратности M к N, независимо от класса принадлежности обоих сущностей, образуется три отношения по одному отношению на сущность и одно отношение для связи, содержащее первичные ключи двух сущностей. [7]

Класс принадлежности сущности определяется следующим образом:

– Если каждый экземпляр сущности A связан с экземпляром сущности B, то класс принадлежности сущности A является обязательным.

– Если не каждый экземпляр сущности A связан с экземпляром сущности B, то класс принадлежности сущности A является не обязательным.

Далее были рассмотрены все связи между сущностями. Результаты данного рассмотрения представлены в таблице К.1, а полученные отношения приведены на рисунке К.1.

Следующим этапом логического проектирования базы данных является нормализация отношений базы данных. Для нормализации отношений реляционной базы данных достаточно привести все её отношения к третьей нормальной форме.

Так для приведения отношения в первую нормальную форму необходимо, чтобы все атрибуты данного отношения были атомарными, то есть хранили единственное значение и не являлись ни списком, ни множеством значений.

Так все представленные атрибуты отношений атомарны, однако сомнения могут вызывать атрибуты: «Разрешения» отношения «Роль», «Вложения» отношения «Сообщение». Эти атрибуты содержат абстрактные данные с которыми не взаимодействует база данных и, следовательно, они являются атомарными. Таким образом

все отношения находятся в первой нормальной форме.

Для приведения отношений во вторую нормальную форму, отношения должны находится в первой нормальной форме и каждый не ключевой атрибут должен полностью завесить от первичного ключа. [8]

Нахождение отношения в третьей нормальной форме требует выполнения условий нахождения во второй нормальной форме и отсутствия транзитных зависимостей, заключающихся в том, что, если атрибут А является ключом, а атрибуты Б и В – нет, то транзитной зависимостью называется зависимость В от А при том, что В зависит от Б и Б зависит от А.

Зависимости атрибутов отношений приведены на рисунке К.2. Из рисунка видно, что все не ключевые атрибуты полностью зависят от первичного ключа и отсутствуют транзитные зависимости, следовательно, все отношения находятся во второй и третьей нормальных формах.

В итоге нормализации отношений базы данных были получены отношения, приведённые на рисунке К.1. Итоговая IDEF1X-диаграмма базы данных после этапа логического проектирования приведена на рисунке К.3.

Последний, заключительный этап проектирования базы данных заключается в формировании физической модели базы данных на основании логической. Для чего необходимо сформировать структуры данных на основании отношений, полученных на этапе логического проектирования. [9]

В результате выполнения физического этапа проектирования базы данных была получена физическая модель базы данных, приведённая на рисунке К.4.

3 РАЗРАБОТКА ИНТЕРФЕЙСА ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

Хорошей практикой перед началом разработки графического интерфейса программного обеспечения является рассмотрение интерфейсов аналогичных систем. Это позволит выявить успешные и неудачные решения, создать интуитивно понятный интерфейс, отвечающий потребностям целевой аудитории.

Одним из примеров, на который можно опираться при разработке интерфейса, является мультичат «Callibri». Рассмотрим, форму входа в систему в этом сервисе, приведённую на рисунке 3.1.

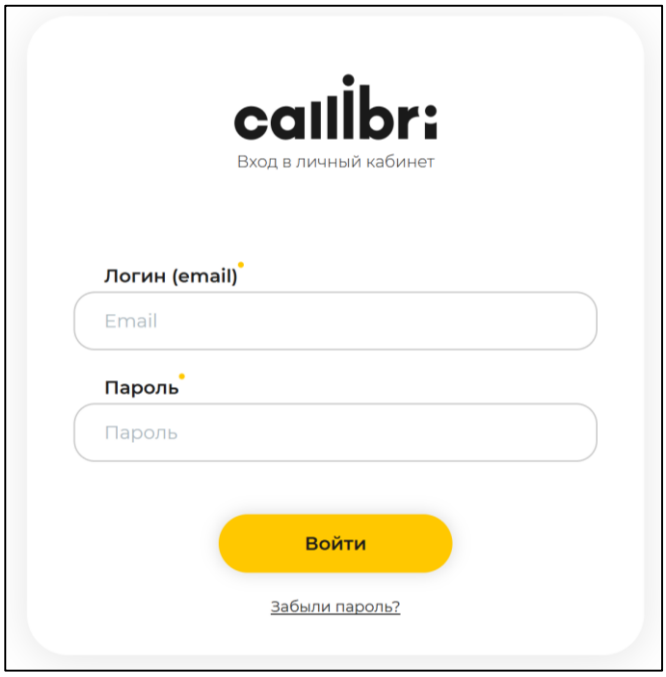

Рисунок 3.1 – Форма входа в «Callibri»

На данном рисунке приведена форма входа, которая является первым что видит пользователь при взаимодействии с системой, и от ее удобства и логичности во многом зависит, насколько легко и быстро пользователь сможет приступить к работе.

В данной форме представлено всего два поля – логин и пароль, что делает процесс входа максимально простым и понятным. Минимализм в количестве полей исключает возможность путаницы и снижает нагрузку на пользователя, особенно если это его первое взаимодействие с системой.

Форма входа содержит всего две кнопки: «Войти» и «Забыли пароль?».

Кнопка «Войти» является основной кнопкой, она ярко выделена на фоне остального интерфейса ярким жёлтым цветом, который интуитивно подсказывает пользователю, куда нажать, чтобы перейти на следующую страницу.

Кнопка «Забыли пароль?» расположена под кнопкой «Войти» и предназначена для восстановления доступа в случае, если пользователь забыл свои учётные данные.

Форма входа, используемая в «JIVO» аналогична форме входа «Callibri», за исключением наличия других способов входа. Изображение формы входа приведено на рисунке 3.2.

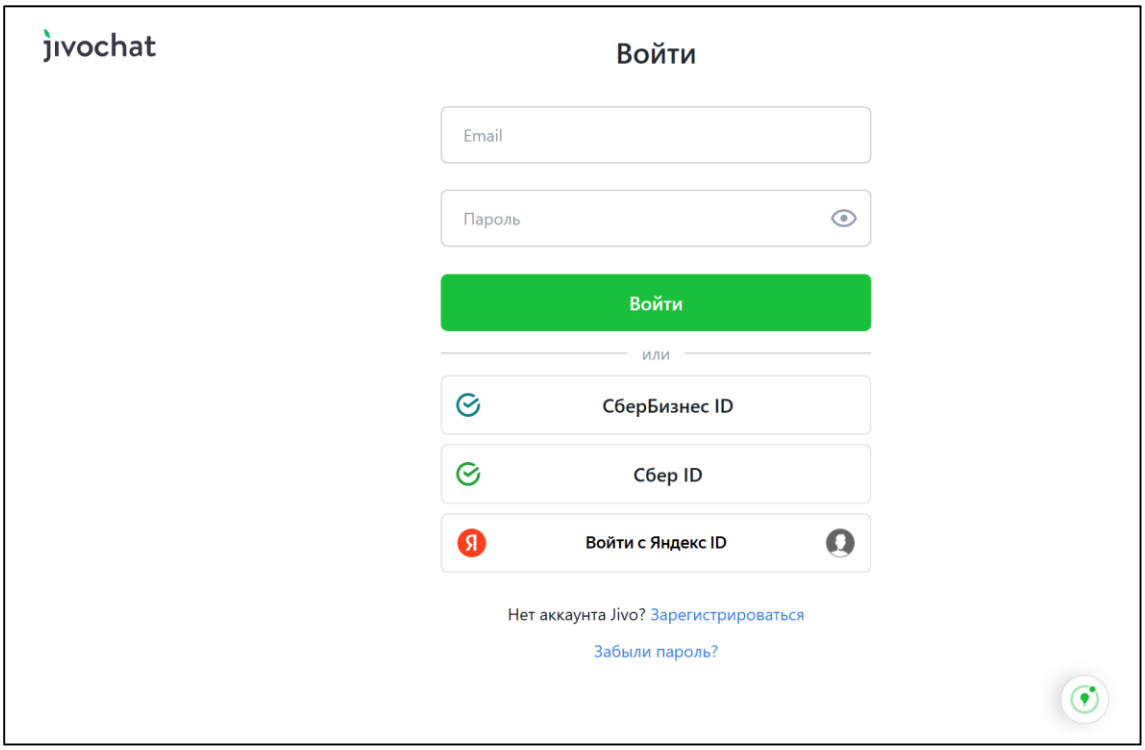

Рисунок 3.2 – Форма входа в «JIVO»

Далее перейдём к рассмотрению интерфейса взаимодействия с чатами в «Callibri». Данный интерфейс приведён на рисунке 1.3.

Интерфейс на рисунке выполнен в виде трех основных секций:

– Левой панели навигации – включает в себя основные разделы такие как чаты, документы, звонки, контакты и др., что упрощает переход между различными функциями программы.

– Списка чатов, который расположен в левой части основного блока интерфейса, содержит все чаты пользователя, что позволяет ему видеть недавние переписки и быстро обращаться к ним.

– Блока переписки – занимает центральную часть экрана и отображает сообщения выбранного чата. В этом блоке также присутствует поле для отправки сообщений и кнопка для загрузки файлов.

Эргономика интерфейса «Callibri» разработана с учетом удобства пользователя. Размещение основных разделов и функций продумано так, чтобы они были легко доступны без лишних кликов. Структура с навигацией слева и окном чата справа, знакомая по интерфейсам мессенджеров, позволяет пользователю быстро адаптироваться к работе с системой.

В интерфейсе используются цветовые акценты и иконки, которые помогают пользователю различать контакты и действия, упрощая ориентацию по интерфейсу.

Также рассмотрим интерес «JIVO», изображение которого приведено на рисунке 1.2.

Данный интерфейс похож на интерес «Callibri». Он также включает в себя основные компоненты, такие как панель навигации, список чатов и блок переписки.

Исходя из рассмотренных интересов было решено оформить форму входа в минималистичном стиле в похожем на форму входа «Callibri». Полученная форма входа приведена на рисунке 3.3.

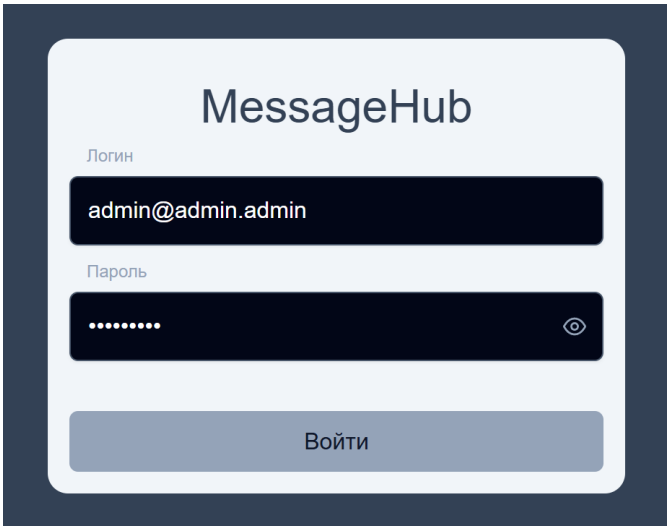

Рисунок 3.3 – Разработанная форма входа

В данном интерфейсе в центре экрана находится простая форма входа, содержащая поля для ввода логина и пароля. Поле ввода пароля скрывает введённые символы, как это обычно принято для защиты данных. Также в низу формы содержится кнопка «Войти» необходимая для проведения аутентификации, она выделена сланцевым цветом, который интуитивно подсказывает пользователю, куда нужно нажать, чтобы перейти на следующую страницу.

Таким образом страница входа выполнена достаточно просто и удобно. Высокий контраст между темным фоном и светлой формой делает интерфейс легко читаемым. Также в интерфейсе используются только необходимые элементы, что позволяет сделать интерфейс простым для освоения.

Основной интерфейс программы было решено выполнить в таком же формате, как, например, на рисунке 1.2. Полученный интерфейс приведён на рисунке 3.4.

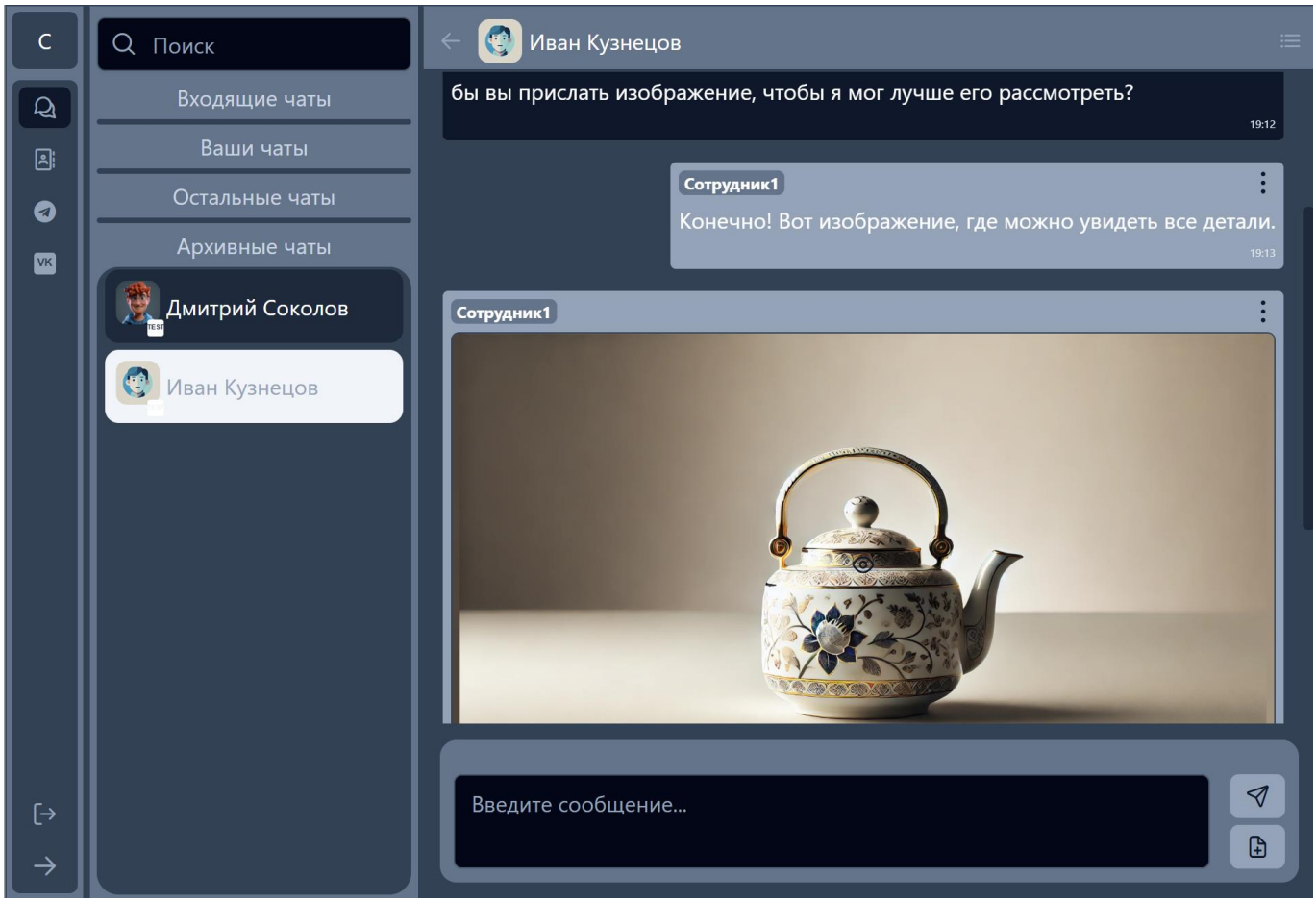

Рисунок 3.4 – Разработанный интерфейс программы

Собственно, интерфейс программы был поделён на два основных блока: блок навигации и блок отображения содержимого раздела пользовательского интерфейса. Первый блок предназначен для перемещения между различными разделами пользовательского интерфейса, а второй раздел предназначен для отображения содержимого конкретного раздела, так на приведённом рисунке открыт блок «Диалоги».

Блок навигации расположен в левой части интерфейса. Как было написано ранее, он предназначен для перемещения между основными разделами пользовательского интерфейса. В верхней его части расположены сведения о текущем пользователе, в данном случае это администратор. По нажатии на строку с именем текущего пользователя откроется диалог редактирования сведений о текущем пользователе, изображение данного диалога приведено на рисунке 3.5.

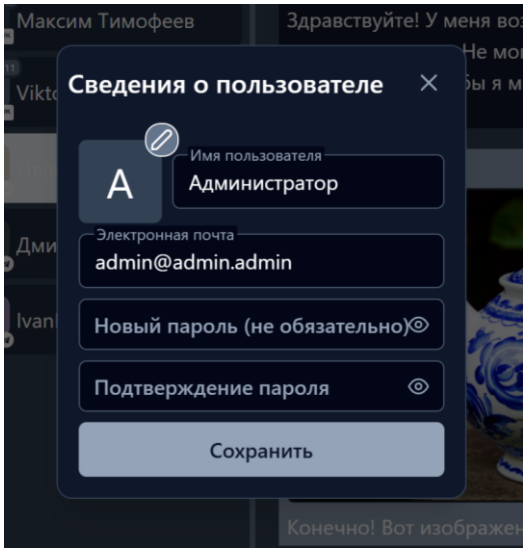

Рисунок 3.5 – Диалог редактирования сведений о пользователе

В данном диалоге пользователь может отредактировать сведения о себе и сменить ранее установленный пароль для входа в учётную запись.

Далее в панели навигации представлены следующие разделы:

– «Диалоги» – раздел, в котором непосредственно осуществляется общение с клиентами.

– «Контроль доступа» – раздел, предназначенный для управления пользователями в системе.

– «Telegram бот» – ссылка на бота Telegram.

– «Сообщество ВКонтакте» – ссылка на сообщество ВКонтакте.

– «Выйти» – кнопка для выхода из учётной записи.

– «Свернуть» – кнопка для уменьшения размера блока навигации.

Перейдём к первому пункту – «Диалоги». Интерфейс данного пункта представлен на рисунке 3.4. Он так же разбит на две части: блок отображения чатов и блок переписки.

Блок отображения чатов расположен в левой части интерфейса раздела, как и в большинстве мессенджеров. Он предназначен для отображения и переключения между всеми доступными пользователю чатами, которые разделены на следующие категории:

– «Входящие чаты» – категория в которую попадают чаты если клиент в первый раз обратился в организацию или чат находился в архиве и в него вновь написал клиент. Нахождение чата в данной группе означает то, что клиент обратился в организацию и ещё не один сотрудник не ответил данному клиенту. Как только какой-нибудь сотрудник напишет сообщение в этот чат, он перейдёт в категорию «Ваши чаты» или «Остальные чаты», в зависимости от пользователя, написавшего сообщение.

– «Ваши чаты» – категория чатов, в которых ведётся обработка обращения клиента и данный пользователь находится в этих чатах.

– «Остальные чаты» – категория чатов, в которых ведётся обработка обращения клиента и данный пользователь не состоит в чатах этой группы.

– «Архивные чаты» – категория в которую помещаются чаты после завершения обработки обращения.

Также в верхней части данного блока расположена поисковая строка, осуществляющая поиск чатов по их названию.

Блок переписки занимает центральную часть и отображает сообщения чата, а также содержит блок ввода сообщения, расположенный в нижней его части, как и в большинстве мессенджеров. Он включает в себя поле ввода текстового сообщения и

кнопки «Прикрепить файл» и «Отправить», соответственно, для прикрепления файлов к сообщению и отправки сообщения в чат.

Также в верхнем левом углу интерфейса была предусмотрена кнопка для открытия меню управления чатом. Изображение открытого меню приведено на рисунке 3.6.

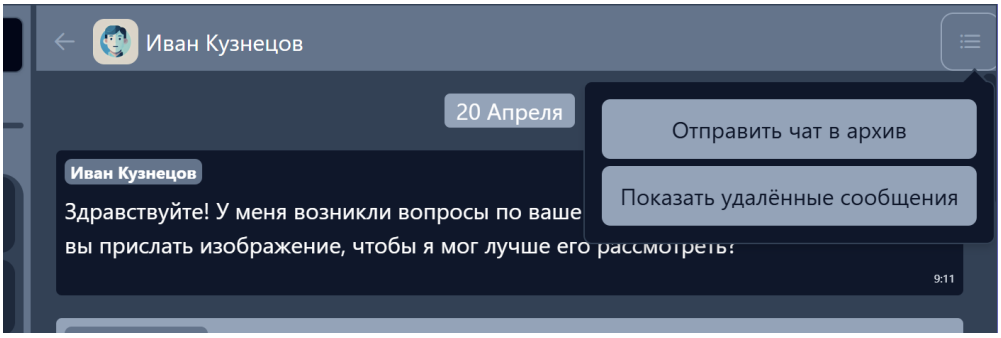

Рисунок 3.6 – Меню управления чатом

Меню управления чатом включает две кнопки. Первая – кнопка «Отправить чат в архив». Она позволяет завершить обработку обращения и принести его во вкладку «Архивные чаты». Вторая – кнопка «Показать удалённые сообщения», предоставляет возможность просмотреть ранее удалённые сообщения в данном чате.

Далее перейдём к разделу «Контроль доступа» пользовательского интерфейса. Интерфейс данного раздела пользовательского интерфейса приведён на рисунке 3.7.

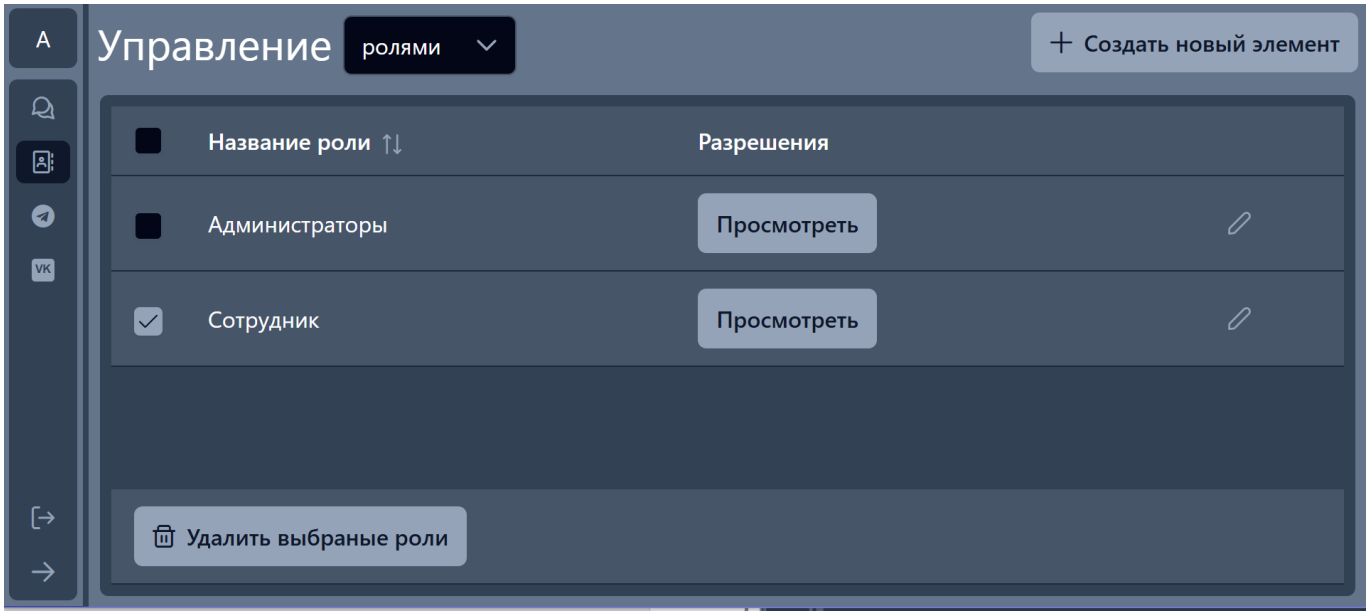

Рисунок 3.7 – Раздел «Управление пользователями» пользовательского интерфейса

В верхней части интерфейса расположена строка с раскрывающимся списком. Этот список предназначен для переключения между интерфейсами управления пользователями или ролями. В данном случае открыт интерфейс управления ролями. В конце данной строки находится кнопка «Создать новый элемент» она предназначена для создания нового пользователя или роли, в зависимости от открытого интерфейса. По нажатии на данную кнопку будет открыт диалог создания нового элемента, который приведён на рисунке 3.8.

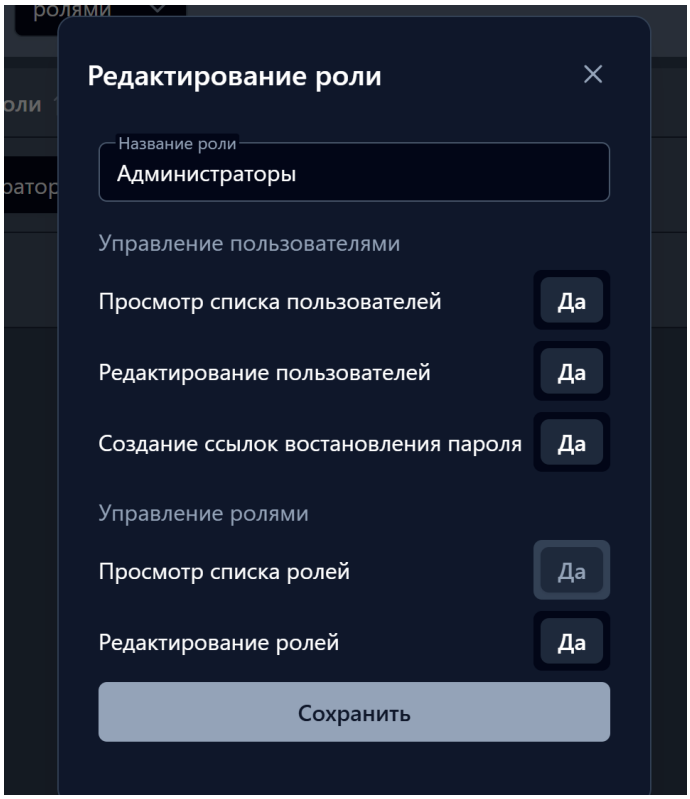

Рисунок 3.8 – Диалог создания роли

В центральной части раздела «Управление пользователями» интерфейса находится таблица, отображающая всех существующих в системе пользователей или ролей, в зависимости от выбранного варианта.

Для удобства пользователей предусмотрена возможность сортировки данных по некоторым столбцам. Такие столбцы отмечены иконкой, указывающей порядок сортировки, размещённой после названия столбца. Для сортировки по нужному

столбцу достаточно нажать на его заголовок и будет применена сортировка по возрастанию, а при повторном нажатии по убыванию.

В левой части таблицы находятся флажки, с помощью которых можно выбрать некоторые строки таблицы. При выборе хотя бы одной строки в нижней части таблицы появится кнопка «Удалить выбранные роли» или «Удалить выбранных пользователей». По нажатию на данную кнопу будет вызван диалог подтверждения удаления выбранных строк, приведённый на рисунке 3.9.

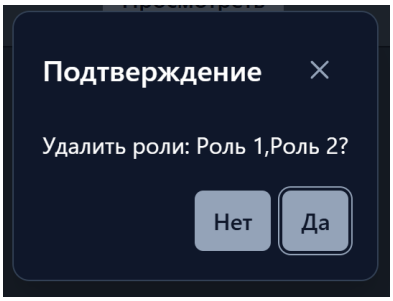

Рисунок 3.9 – Диалог подтверждения удаления выбранных строк

Также предусмотрена возможность редактирования строк. Для чего необходимо в конце строки нажать на иконку карандаша и отредактировать нужные поля, а затем нажать на иконку подтверждения в конце строки для сохранения введённых изменений. Изображение строки в режиме редактирования приведено на рисунке 3.10.

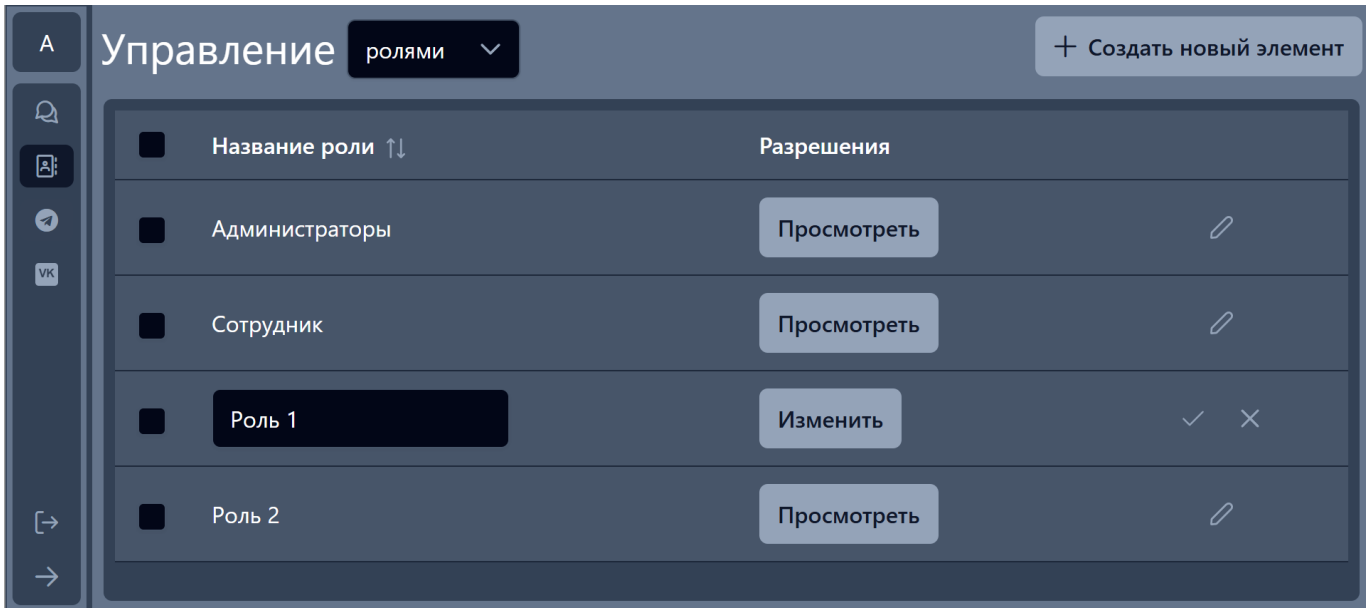

Рисунок 3.10 – Редактирование строки

Клиентский интерфейс в Teleagrm реализован с помощью бота. Стандартный диалог с ботом приведён на рисунке 3.11.

Рисунок 3.11 – Интерфейс бота в Telegram

Интерфейс бота Telegram выполнен следующим образом: Для начала работы с ботом пользователю необходимо отправить команду «/start». Важно отметить, что команда «/start» отправляется автоматически, как только пользователь нажимает на кнопку «Начать» или инициирует первый контакт с ботом. Это упрощает вход в систему, устраняя лишние шаги и потенциально экономя время пользователя.

После активации бота, чтобы вести диалог, пользователю достаточно просто отправлять сообщения, без использования каких-либо дополнительных команд, что обеспечивает простоту использования бота. После отправки сообщения боту они сразу перенаправляются в web-клиент системы.

Таким образом, взаимодействие с ботом интуитивно и напоминает привычную переписку, не требуя от пользователя запоминания специальных команд или дополнительных действий.

Также в случае возникновения ошибок бот автоматически отправит уведомления с объяснением проблемы. Такой подход минимизирует риск непонимания со стороны пользователя и снижает вероятность повторения ошибок, так как пользователь всегда будет иметь под рукой понятное сообщение с рекомендацией по устранению проблемы или инструкцией по следующему шагу.

Интерфейс клиента в сообществе ВКонтакте реализован аналогичным образом, за исключением того, что во ВКонтакте при написании первого сообщения в сообщество выводится форма, представленная на рисунке 3.12.

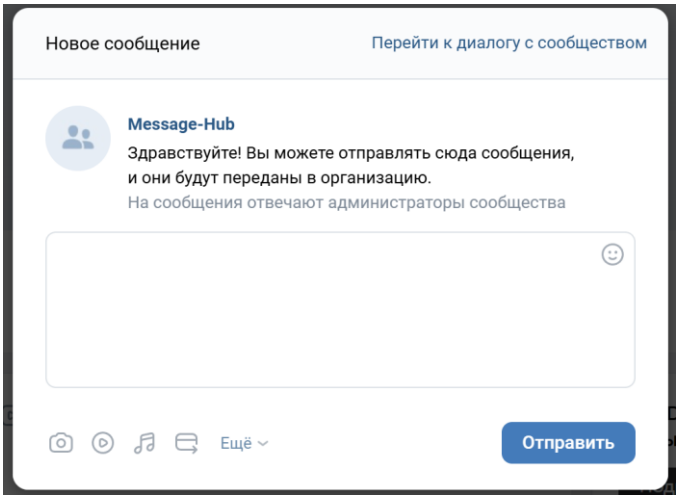

Рисунок 3.12 – Отправка первого сообщения в сообщество в ВКонтакте

4 БЕЗОПАСНОСТЬ И ЭКОЛОГИЧНОСТЬ

Проектирование и внедрение мультичата требует комплексного подхода не только к техническим и функциональным аспектам программного обеспечения, но и к вопросам обеспечения безопасности жизнедеятельности пользователей, экологичности, а также готовности к возможным чрезвычайным ситуациям. В данном разделе будут рассмотрены ключевые аспекты данных вопросов.

4.1 Безопасность

Поскольку мультичат представляет собой многофункциональную систему обмена сообщениями, активно используемую сотрудниками, работающими с клиентами, в течение всего рабочего дня, очень важно организовать условия труда, минимизирующие негативное воздействие работы за ПК на здоровье сотрудников.

В качестве основных факторов, способных нанести вред человеку выделяют:

– вредный, тяжелый, напряженный труд, связанный с деятельностью человека в производственной среде, обладающей опасными и вредными факторами;

– загрязнение воздуха, воды, почвы и продуктов питания вредными и опасными химическими веществами;

– воздействие на человека шума, вибрации, теплового, элекэлектромагнитного и ионизирующего излучения;

– социальная напряженность, конфликты и стрессы. [10]

Для снижения влияния этих факторов на сотрудников все элементы инфраструктуры, включая оборудование и рабочие помещения, должны соответствовать действующим санитарно-гигиеническим нормам и требованиям охраны труда.

4.1.1 Требования к ПК

Для обеспечения безопасности на рабочем месте, ПК должны соответствовать установленным нормам по уровню шума, вибрации, электромагнитного излучения. Нормативы по этим параметрам устанавливаются для конкретного рабочего места, а не для отдельных устройств или помещения в целом. Это означает, что при контроле

учитываются совокупные воздействия всех технических средств, находящихся в непосредственной близости к пользователю.

Согласно СанПиН 1.2.3685-21, были оформлены таблицы 4.1 и 4.2, в которых указаны допустимые значения уровней звукового давления в октавных полосах частот и уровня звука, а также предельно допустимые уровни электромагнитных полей. При необходимости, в СанПиН 1.2.3685-21 можно найти предельно допустимые уровни вибрации в пункте «Допустимые значения и уровни вибрации в помещениях общественных зданий».

Таблица 4.1 – Допустимые значения уровней звукового давления в октавных полосах частот и уровня звука

Уровни звукового давления в октавных полосах со среднегеометрическими частотами									Максимальные уровни звука в дБА
31,5 Гц	63 Гц	125 Гц	250 Гц	500 Гц	1000 Гц	2000 Гц	4000 Гц	8000 Гц	
79 дБ	63 дБ	52 дБ	45 дБ	39 дБ	35 дБ	32 дБ	30 дБ	28 дБ	55

Таблица 4.2 – Предельно допустимые уровни электромагнитных полей

Диапазон частот	30-300 кГц	0,3-3 МГц	3-30 МГц	30-300 МГц	0,3-300 ГГц
Нормируемый параметр	Напряженность электрического поля В/м				Плотность потока энергии мкВт/см^2
Предельно-допустимые уровни	25	15	10	3	10 25 для случаев облучения от антенн

Важно отметить, что для учёта данных требований можно размещать вне основных рабочих зон оборудование, создающее повышенный уровень шума и вибрации на рабочем месте, например, серверы, принтеры и источники бесперебойного питания.

Конструкция ПК должна обеспечивать возможность фронтального обзора экрана пользователем. Цветовое оформление корпуса должно быть выполнено в спокойных и мягких тонах с диффузным рассеиванием света. Поверхности корпуса, клавиатуры и других устройств должны быть матовыми, с коэффициентом отражения от 0,4 до 0,6, и не содержать блестящих элементов, способных создавать блики. Также

конструкция мониторов ПК должна предусматривать возможность регулировки яркости и контрастности изображения.

Требования к техническим характеристикам мониторов ПК устанавливаются ГОСТ Р 50948–2001. В соответствии с ним к экранам мониторов предъявляются следующие требования:

– Яркость белого фрагмента экрана должна быть не менее 35 кд/м².

– Неравномерность яркости экрана не должна превышать 20 %.

– Контрастность в монохромном режиме должна составлять не менее 3:1.

– Временная нестабильность изображения, то есть непреднамеренное изменение яркости на экране со временем, не должна фиксироваться.

– Пространственная нестабильность изображения, проявляющаяся в непреднамеренных изменениях положения фрагментов изображения на экране, должна быть минимальной и не создавать заметных искажений.

На практике, ООО «ИнфоКом» в основном использует облачную инфраструктуру, которая позволяет переместить большую часть инфраструктуры в центр обработки данных, тем самым оставив в офисе только потребительские ПК и принтер. В следствии чего в офисе не присутствует какого-либо оборудования способного нарушать указанные нормы.

4.1.2 Требования к помещениям для работы с ПК

При оборудовании рабочих мест с ПК необходимо соблюдать минимальные нормы площади. Так для ПК с мониторами на базе электронно-лучевых трубок минимальная площадь на одного пользователя должна составлять не менее 6 м^2. При использовании мониторов на базе плоских экранов, например, жидкокристаллических или плазменных, допустимая площадь может быть уменьшена до 4,5 м². Однако рекомендуется выделять не менее 10 м² на каждого пользователя для обеспечения комфортных условий труда.

Внутренняя отделка помещений, предназначенных для работы с ПК, должна выполняться с использованием диффузно отражающих материалов, коэффициенты отражения, которых должны соответствовать следующим значениям: потолок – (0,7

– 0,8), стены – (0,5 – 0,6), пол – (0,3 – 0,5). Соблюдение этих требований необходимо для предотвращения бликов и создания благоприятных условий освещённости.

Помещения, в которых размещаются рабочие места с ПК, должны быть оборудованы защитным заземлением или занулением согласно техническим требованиям эксплуатации и ГОСТ Р 50571.5.54-2013. При этом не допускается размещение рабочих мест вблизи источников электромагнитных помех – силовых кабелей, вводных устройств, трансформаторов высокого напряжения и технологического оборудования, способного негативно влиять на работу ПК.

ООО «ИнфоКом» арендует офис в бизнес центре с планировкой, представленной на рисунке 4.1.

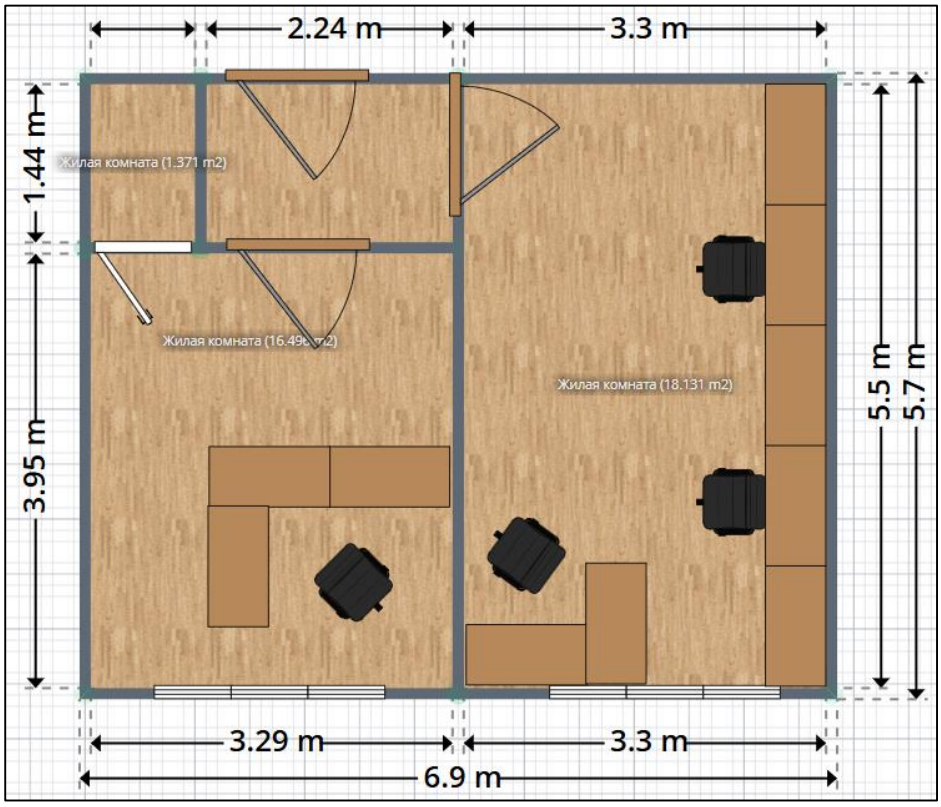

Рисунок 4.1 – Планировка офиса ООО «ИнфоКом»

На рисунке видно, что офис рассчитан на четыре рабочих места, с учётом того, что в организации используются мониторы на базе плоских экранов минимальная площадь помещения без учёта проходов должна составлять 18 м^2, что, как видно из рисунка, выполняется.

В помещении потолок окрашен в белый цвет, а стены – в синий, что позволяет предотвратить появление бликов. Распределительные электрощиты в здании размещены на лестничной клетке, вдалеке от арендуемого помещения, что соответствует требованиям.

4.1.3 Требования к освещению на рабочих местах, оборудованных ПК

Помещения, предназначенные для работы с ПК, должны быть оборудованы как естественным, так и искусственным освещением. Использование помещений без естественного освещения допускается лишь при наличии обоснования и положительного санитарно-эпидемиологического заключения. Освещение должно соответствовать нормативам, установленным СП 52.13330.2016.

Окна в помещении рекомендуется ориентировать на север или северо-восток, обеспечивая при этом возможность регулирования светового потока при помощи жалюзи, штор и козырьков. Рабочие столы с мониторами следует располагать боком к окнам, чтобы естественный свет падал преимущественно слева.

Искусственное освещение должно быть равномерным, предпочтительно – комбинированным, с использованием общего и местного освещения. Светильники должны обеспечивать освещенность поверхности стола на уровне (300 – 500) лк, при этом освещенность экрана не должна превышать 300 лк. Следует избегать бликов, прямая и отражённая блесткость должны быть ограничены. Яркость светящихся объектов в поле зрения не должна превышать 200 кд/м², а бликов на экране – 40 кд/м². Показатели ослепленности не должны превышать 20. Яркость светильников в угловой зоне излучения (50° – 90°) не должна быть выше 200 кд/м², а защитный угол – не менее 40°. Светильники местного освещения должны иметь непросвечивающий отражатель с тем же минимальным защитным углом. Соотношение яркости между рабочими поверхностями должно составлять (3:1 – 5:1), между рабочими и фоновыми – не более 10:1.

В качестве основного источника искусственного освещения рекомендуется использовать люминесцентные лампы, допускается применение металлогалогенных

ламп и ламп накаливания в местных светильниках. Общее освещение следует размещать сбоку от рабочих мест, параллельно линии зрения пользователя, либо локально над передним краем стола. Коэффициент запаса для осветительных установок необходимо принимать равным 1,4. Коэффициент пульсации должен составлять не более 5 %. Также важно проводить чистку окон и светильников не реже двух раз в год и своевременно заменять перегоревшие лампы.

В помещениях ООО «ИнфоКом» используется, как естественное, так и искусственное освещение. Окна в помещении ориентированы в западном направлении. Все остальные требования к освещению выполняются.

4.1.4 Требования к микроклимату, содержанию аэроионов и вредных химических веществ в воздухе на рабочих местах, оборудованных ПК

В помещениях, где работа с ПК является основной необходимо обеспечивать оптимальные параметры микроклимата, в соответствии с категориями работ Iа и Iб, установленные СанПиН 1.2.3685-21 в пункте «Допустимые величины параметров микроклимата на рабочих местах в помещениях». В остальных случаях допустимо поддержание микроклимата на уровнях, предусмотренных указанным стандартом. Оптимальные значения температуры воздуха, температуры поверхностей, относительной влажности и скорости движения воздуха зависят от времени года и категории работ и приведены в таблице 4.3.

Таблица 4.3 – Фрагмент допустимых величин параметров микроклимата на рабочих местах в помещениях из СанПиН 1.2.3685-21

Период года	Категория работ по уровню энергозатрат, Вт	Температура воздуха, °С	Температура поверхностей, °С	Относительная влажность воздуха, %	Скорость движения воздуха, м/с
Холодный	Iа (до 139)	22 – 24	21 – 25	40 – 60	0,1
	Iб (140 – 174)	21 – 23	20 – 24	40 – 60	0,1
Тёплый	Iа (до 139)	23 – 25	22 – 26	40 – 60	0,1
	Iб (140 – 174)	22 – 24	21 – 25	40 – 60	0,1

Кроме параметров микроклимата, важным показателем является содержание аэроионов. Их уровни в помещениях с ПК должны соответствовать требованиям Сан-

ПиН 1.2.3685-21. Также необходимо контролировать содержание вредных химических веществ в воздухе помещений. Концентрации загрязняющих веществ не должны превышать предельно допустимых значений, установленных для атмосферного воздуха в соответствии с СанПиН 1.2.3685-21.

В помещениях ООО «ИнфоКом» данные требования к микроклимату, содержанию аэроионов и вредных химических веществ в воздухе на рабочих местах соблюдаются и важно отметить, что, так как офис арендуется в бизнес центе, данные требования периодически проверяются администрацией бизнес центра.

4.1.5 Требования к организации и оборудованию рабочих мест с ПК

Рабочие места с ПК должны быть организованы с соблюдением эргономических, санитарных и технических требований. Расстояние между тыльной стороной одного видеомонитора и экраном другого должно быть не менее 2 м, между боковыми сторонами – не менее 1,2 м. Для работ, требующих высокой концентрации внимания, рекомендуется использовать перегородки высотой (1,5 – 2) м.

Экран видеомонитора должен находиться от глаз пользователя на расстоянии (600 – 700) мм, но не ближе 500 мм с учетом размеров алфавитно-цифровых знаков и символов.

Рабочий стол должен обеспечивать удобное размещение оборудования, иметь коэффициент отражения поверхности (0,5 – 0,7), модульные размеры шириной (800 – 1400) мм и глубиной (800 – 1000) мм. Высота регулируемой столешницы должна регулироваться в диапазоне от 680 до 800 мм, а нерегулируемой – составлять 725 мм. Обязательно наличие пространства для ног: не менее 600 мм по высоте, 500 мм по ширине, 450 мм по глубине у колен и на уровне вытянутых ног – не менее 650 мм.

Кресло должно быть подъемно-поворотным, с независимой и надёжной регулировкой по высоте, углу наклона сиденья и спинки. Поверхность кресла – полумягкая, нескользящая, с воздухопроницаемым покрытием.

Кресло должно обеспечивать:

– ширину и глубину поверхности сиденья не менее 400 мм;

– поверхность сиденья с закругленным передним краем;

– регулировку высоты поверхности сиденья в пределах (400 – 550) мм и углов наклона вперед до 15° и назад до 5°;

– высоту опорной поверхности спинки (300 ± 20) мм, ширину – не менее 380 мм и радиус кривизны горизонтальной плоскости – 400 мм;

– угол наклона спинки в вертикальной плоскости в пределах ± 30°;

– регулировку расстояния спинки от переднего края сиденья в пределах (260 – 400) мм;

– стационарные или съемные подлокотники длиной не менее 250 мм и шириной – (50 –70) мм;

– регулировку подлокотников по высоте над сиденьем в пределах (230 ± 30) мм и внутреннего расстояния между подлокотниками в пределах (350 – 500) мм.

Рабочее место должно быть оборудовано подставкой для ног шириной не менее 300 мм, глубиной не менее 400 мм, с возможностью регулировки по высоте до 150 мм и углу наклона до 20°, с рифленой поверхностью и бортиком 10 мм по переднему краю.

Клавиатура должна размещается на расстоянии (100 – 300) мм от переднего края стола или на специальной регулируемой поверхности, отделённой от основной.

В ходе рассмотрения оборудования рабочих мест в ООО «ИнфоКом» было установлено, что практически все представленные требования выполняются, за исключением, того, что глубина столешницы рабочего стола составляет 700 мм.

4.1.6 Нормирование работы за ПК и перерывов

Для нормирования работы за ПК и перерывов применяется оценка тяжести и напряжённости труда сотрудников, использующих ПК, которая проводится по утверждённым методикам с учётом условий труда и функционального состояния работников. Для профессий с высокой степенью ответственности и нагрузки такая оценка осуществляется специализированными организациями.

Трудовая деятельность с ПК делится на три группы:

– «А» – считывание информации с экрана;

– «Б» – ввод данных;

– «В» – творческая работа в диалоговом режиме.

Если в течение смены выполняются различные виды работ, основным считается тот, который занимает не менее 50 % времени.

Нагрузки для каждой группы ограничиваются:

– для группы «А» – до 60.000 знаков;

– группы «Б» – до 40.000 знаков;

– группы «В» – до 6 часов за смену.

В зависимости от категории работы и объёма нагрузки устанавливается суммарное время регламентированных перерывов, приведённых в таблице 4.4.

Таблица 4.4 – Суммарное время регламентированных перерывов в зависимости от продолжительности работы, вида и категории трудовой деятельности с ПК

Категория работы с ПК	Уровень нагрузки за рабочую смену при видах работ с ПК			Суммарное время регламентированных перерывов, мин	
	группа А, количество знаков	группа Б, количество знаков	группа В, ч	при 8-часовой смене	при 12-часовой смене
I	до 20.000	до 15.000	до 2	50	80
II	до 40.000	до 30.000	до 4	70	110
III	до 60.000	до 40.000	до 6	90	140

Для профилактики утомления рекомендуется чередовать работу с ПК и без неё. При появлении дискомфорта, даже при соблюдении всех норм, следует ограничить время работы индивидуально.

Если невозможно сменить вид деятельности, необходимо проводить 15-минутные перерывы через каждые (45 – 60) минут. Непрерывная работа с экраном не должна превышать 1 часа. В ночные смены продолжительность всех перерывов должна быть увеличена на 30 %.

В период отдыха следует выполнять упражнения для снятия зрительного, нервно-эмоционального и мышечного напряжения. Пользователям с высокой психоэмоциональной нагрузкой рекомендуется психологическая разгрузка в специально оборудованных помещениях.

В ООО «ИнфоКом» используется CRM-система Битрикс-24, в которой организация использует функционал учёта времени работы сотрудников, посредствам которого и выполняются требования к нормированию работы и перерывам.

4.1.7 Требования к графическому интерфейсу программного обеспечения

Графический интерфейс разрабатываемого программного обеспечения должен быть спроектирован с учётом требований безопасности, эргономики и зрительного комфорта. Основным нормативным документом, регулирующим параметры графических интерфейсов, выступает ГОСТ Р 52872-2019.

В первую очередь, структура интерфейса должны быть логически упорядочена и представлена понятной для пользователя форме, например, рабочее окно приложения должно быть разделено на отдельные функциональные зоны: панель навигации, список чатов, зону переписки и строку ввода сообщений.

Цветовое оформление интерфейса должно обеспечивать контраст между текстом и фоном не менее 3:1 для чёткого восприятия информации при различных условиях освещённости. Также интерфейс должен исключать использование мигающих элементов и резких цветовых перепадов, чтобы соответствовать требованиям по предотвращению зрительного раздражения, изложенным в ГОСТ Р 52872-2019.

Шрифты, применяемые в интерфейсе, должны иметь стандартную беззасечную форму, а их размер обеспечивать высоту символов на экране не менее 3,2 мм, чтобы гарантировать читаемость. Межстрочные и межбуквенные интервалы должны быть подобраны таким образом, чтобы информация воспринималась быстро и без дополнительного напряжения.

Интерфейс мультичата должен обладать масштабируемостью, позволяющей изменять размер текста и элементов управления под индивидуальные потребности пользователей. Также должна быть реализована адаптивность интерфейса к различным типам устройств – от настольных ПК до мобильных телефонов и планшетов. При этом должна сохраняется функциональность и логика отображения интерфейсных компонентов, независимо от размера и ориентации экрана.

Визуальная стабильность интерфейса мультичата также имеет важное значение для обеспечения безопасности труда. Элементы интерфейса не должны перемещаться самопроизвольно, перекрываться или мерцать, чтобы снизить утомляемость пользователей и уменьшить вероятность ошибок при выполнении операций.

Интерфейс мультичата в целом соответствует требованиям: структура логична, элементы интерфейса устойчивы, применены читаемые шрифты и реализована масштабируемость.

4.2 Экологичность

При эксплуатации мультичата необходимо уделять особое внимание вопросам минимизации негативного воздействия на окружающую среду. Несмотря на нематериальную природу программного обеспечения, его разработка, внедрение и сопровождение связаны с использованием вычислительной и офисной техники, мебели и расходных материалов, которые в дальнейшем становятся отходами, подлежащими утилизации.

Согласно 14 статье федерального закона N 89 «Об отходах производства и потребления» организации обязаны осуществлять классификацию отходов на следующие классы:

– «I класс» – чрезвычайно опасные отходы;

– «II класс» – высокоопасные отходы;

– «III класс» – умерено опасные отходы;

– «IV класс» – малоопасные отходы;

– «V класс» – практически неопасные отходы.

Важно отметить, что в случае образования отходов первого и второго класса на их владельцев федеральный закон возлагает обязанность по их утилизации.

В рамках реализации принципа экологической ответственности ООО «ИнфоКом» выстраивает системный подход к обращению с оргтехникой. Так по мере износа, компьютерная и офисная техника выводится из эксплуатации и передаётся специализированным компаниям, имеющим лицензии на обращение с отходами I – IV

классов опасности, что позволяет предотвратить попадание в окружающую среду тяжёлых металлов, кислот и других опасных компонентов, содержащихся в электронике.

Офисная мебель, подлежащая списанию, также проходит оценку на возможность повторного использования или ремонта. Непригодная для эксплуатации мебель утилизируется через специализированные пункты приёма как отходы V класса опасности.

Также важно отметить что ООО «ИнфоКом» в рамках своей деятельности уже использует ЭДО, применение которого позволяет существенно сократить объёмы печатной документации за счёт перевода внутреннего и внешнего документооборота в электронный формат, что в свою очередь, снижает потребление бумаги, картриджей, электроэнергии и уменьшает объёмы хранения бумажных архивов.

Однако несмотря на широкое использование ЭДО, часть процессов в организации по-прежнему требует применения бумажных носителей – как в силу требований законодательства, так и из-за привычек пользователей. В связи с этим особое внимание уделяется рациональному использованию бумаги: применяется двухсторонняя печать, исключается лишнее копирование, оптимизируются бизнес-процессы. Отработанные бумажные материалы сортируются, собираются в специальные контейнеры и передаются на переработку организациям, имеющим лицензию на их переработку.

До недавнего времени в организации использовались люминесцентные лампы, содержащие ртуть. Поскольку такие лампы относятся к I классу опасности, их сбор и утилизация осуществлялись в соответствии с действующими нормами. В настоящее время организация поэтапно отказывается от использования люминесцентных ламп, заменяя их на светодиодные светильники, которые не содержат токсичных веществ и обладают значительно более длительным сроком службы.

Разработка мультичата осуществлялась с учётом принципов энергоэффективности. Программные решения были ориентированы на минимизацию потребления ресурсов за счёт использования оптимизированного кода и эффективных алгоритмов,

что позволяет существенно снижать нагрузку на серверное оборудование и вычислительные ресурсы.

Важно отметить, что использование мультичата влияет на повседневную деятельность пользователей. Перенос взаимодействий между компанией и её клиентами в электронный формат сокращает объём бумажной документации и уменьшает количество личных визитов в офис, что сопровождается снижением транспортной активности, что в совокупности способствуют сокращению выбросов углекислого газа и снижению общего воздействия на окружающую среду.

4.3 Чрезвычайные ситуации

В процессе эксплуатации мультичата, как и любой другой информационной системы, важно учитывать потенциальные риски, связанные с чрезвычайными ситуациями, особенно в офисной среде. Одной из наиболее вероятных чрезвычайных ситуаций в офисной среде является пожар, вызванный коротким замыканием, перегрузкой электросети, неисправностью оборудования или неосторожным обращением с огнём.

В целях предупреждения пожаров организация обязана соблюдать требования пожарной безопасности, описанные в федеральном законе N 123 «Технический регламент о требованиях пожарной безопасности».

Во исполнение данного закона в бизнес центре, где ООО «ИнфоКом» арендует офис, все помещения оснащены автоматической системой пожарной сигнализации, средствами оповещения, а также средствами первичного пожаротушения. Внутренняя отделка помещений выполнена из трудногорючих материалов. Обеспечивается легкодоступность, не загромождённсть эвакуационных путей и размещение планов эвакуации людей при пожаре на видных местах.

Также согласно постановлению правительства от 16.09.2020 N 1479 «Об утверждении Правил противопожарного режима в Российской Федерации» руководитель организации обязан:

– утвердить инструкцию о мерах пожарной безопасности;

– контролировать исправность средств пожаротушения не реже раза в год;

– обеспечивать своевременное прохождение сотрудниками программам противопожарного инструктажа;

– проводить практические тренировки по эвакуации раз в полгода;

– обеспечивать проверку здания на соответствие нормативами пожарной безопасности не реже раза в год;

– организовывать раз в 5 лет проведение эксплуатационных испытаний сооружений предназначенных для эвакуации людей;

– обеспечивать наличие знаков пожарной безопасности, обозначающих пути эвакуации и эвакуационные выходы.

В силу того, что ООО «ИнфоКом» арендует офис в бизнес центе, большая часть указанных обязательств выполняется администрацией бизнес цента.

В случае возникновения признаков возгорания, таких как запах гари, дым или открытое пламя сотрудникам организации необходимо:

– немедленно сообщить об этом по телефону в пожарную;

– приступить к эвакуации людей по установленным маршрутам, избегая использования лифтов и не возвращаясь за личными вещами;

– при отсутствии непосредственной угрозы жизни и здоровью принять меры по тушению пожара с помощью имеющихся средств первичного пожаротушения, например, огнетушителей, пожарных кранов.

После завершения эвакуации ответственный сотрудник должен проконтролировать численность эвакуированных, при необходимости повторно уведомить экстренные службы и обеспечить координацию дальнейших действий до прибытия пожарных подразделений.

ЗАКЛЮЧЕНИЕ

В результате выполнения бакалаврской работы был спроектирован и реализован мультичат, обеспечивающей централизованную обработку клиентских обращений, поступающих из Telegram и ВКонтакте.

Перед началом проектирования программного обеспечения был проведён предпроектный анализ. В частности, был выполнен анализ деятельности компании ООО «ИнфоКом», а также изучены существующие аналоги разрабатываемой системы. Такой подход позволил определить ключевые потребности организации и сформулировать основные требования к функциональности будущего решения.

Также в рамках предпроектного анализа, были рассмотрены API сторонних сервисов, включая Telegram Bot API и API ВКонтакте, по результатам которого были определены способы интеграции с этими платформами.

На основе проведённого анализа было сформулировано техническое задание. Далее в процессе проектирования программного обеспечения было принято решение реализовать систему в виде набора микросервисов. В частности, были выделены следующие микросервисы: микросервис мгновенного обмена сообщениями, Telegram бот, бот ВКонтакте, бэкенд и фронтенд web-клиента. Для каждого микросервиса были описаны его функции, а также способы взаимодействия с другими компонентами системы.

В результате проектирования была получена UML-диаграмма вариантов использования программного обеспечения, а также были построены UML-диаграммы диаграммы компонентов и последовательностей, описывающие работу системы.

Далее был выполнен выбор средств реализации программного обеспечения. Бэкенд системы было решено реализовать с использованием языка программирования Python и web-фреймворка FastAPI. Для реализации фронтенда был использован JavaScript в связке с Vue.js, PrimeVue, Pinia и tailwindcss. В качестве реляционной базы данных был выбран PostgreSQL, в качестве резидентной базы данных – Redis, а

также S3 хранилище MinIO. Для развёртывания компонентов системы было решено использовать Docker и обратный прокси-сервера traefik.

В ходе проектирования базы данных было проведено инфологическое, логическое и физическое проектирование базы данных, по результатам которого были выделены 11 сущностей и 15 связей между ними.

Далее была осуществлена разработка пользовательского интерфейса. Для её осуществления, предварительно был проведён анализ пользовательских интерфейсов аналогичных решений, таких как Callibri и Jivo. На основе этого анализа было сформулировано представление о ключевых требованиях к удобству, функциональности и визуальному оформлению программного обеспечения. В результате проделанной работы был разработан собственный интерфейс, ориентированный на минимализм, интуитивную понятность и адаптацию к привычным моделям взаимодействия пользователей.

После завершения разработки программное обеспечение было успешно развёрнуто на сервере, что подтвердило его работоспособность, стабильность и соответствие требованиям, предъявленным на этапе проектирования.

БИБЛИОГРАФИЧЕСКИЕ ССЫЛКИ

1 Telegram Bot API [Электронный ресурс]. URL: https://core.telegram.org/bots/api (дата обращения: 09.04.2025).

2 Форматы запросов [Электронный ресурс]. URL: https://dev.vk.com/ru/api/api-requests (дата обращения: 09.04.2025).

3 Introduction to microservices [Электронный ресурс]. URL: https://cloud.google.com/architecture/microservices-architecture-introduction (дата обращения: 09.04.2025).

4 Отличие SOA от микросервисной архитектуры [Электронный ресурс]. URL: https://microarch.ru/blog/soa-vs-msa (дата обращения: 09.04.2025).

5 A pattern language for microservices [Электронный ресурс]. URL: https://microservices.io/patterns/ (дата обращения: 09.04.2025).

6 Фешина Е. В. Базы данных: учебник. Изд-во КубГАУ, 2020. С. 38.

7 Илюшечкин В. М. Основы использования и проектирования баз данных: учебник для вузов. Изд-во Юрайт, 2025. С. 180–187.

8 Нестеров С. А. Базы данных: учебник и практикум для вузов. Изд-во Юрайт, 2023. С. 55–57.

9 Стружкин Н. П. Базы данных: проектирование: учебник для вузов. Изд-во Юрайт, 2025. С. 395–396.

10 Резчиков, Е. А. Безопасность жизнедеятельности: учебник для вузов. Изд-во Юрайт, 2025. С. 16–17.

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1 Белов, С. В. Безопасность жизнедеятельности и защита окружающей среды (техносферная безопасность): учебник для среднего профессионального образования / С. В. Белов. – 6-е изд., перераб. и доп. – Москва: Изд-во Юрайт, 2025. – 638 с.

2 Беляков, Г. И. Пожарная безопасность, безопасность в чрезвычайных ситуациях и оказание первой помощи: учебник для вузов / Г. И. Беляков. – 5-е изд., перераб. и доп. – Москва: Изд-во Юрайт, 2025. – 529 с.

3 Илюшечкин, В. М. Основы использования и проектирования баз данных: учебник для вузов / В. М. Илюшечкин. – Москва: Изд-во Юрайт, 2025. – 213 с.

4 Микросервисная и монолитная архитектуры [Электронный ресурс]. – Режим доступа: https://iqdev.digital/blog/perehod-na-mikroservisnuyu-arhitekturu-perejti-nelzya-ostatsya. – 09.04.2025.

5 Микросервисы, скалы и гигантские приложения [Электронный ресурс]. – Режим доступа: https://cloud.vk.com/blog/mikroservisy-skaly-i-gigantskie-prilozheniya. – 09.04.2025.

6 Нестеров, С. А. Базы данных: учебник и практикум для вузов. / С. А. Нестеров. – Москва: Изд-во Юрайт, 2023. – 259 с.

7 Отличие SOA от микросервисной архитектуры [Электронный ресурс]. – Режим доступа: https://microarch.ru/blog/soa-vs-msa. – 09.04.2025.

8 Резчиков, Е. А. Безопасность жизнедеятельности: учебник для вузов / Е. А. Резчиков, А. В. Рязанцева. – 4-е изд., перераб. и доп. – Москва: Изд-во Юрайт, 2025. – 638 с.

9 Стандарт организации: Оформление выпускных, квалификационных и курсовых работ (проектов). – Благовещенск: Амурский государственный университет, 2018. – 75с.

10 Стружкин, Н. П. Базы данных: проектирование: учебник для вузов. / Н. П. Стружкин, В. В. Годин. – Москва: Изд-во Юрайт, 2025. – 477 с.

11 Фешина, Е. В. Базы данных: учебник. / Е. В. Фешина, В. В. Ткаченко. – Краснодар: Изд-во КубГАУ, 2020. – 172 с.

12 Форматы запросов [Электронный ресурс]. – Режим доступа: https://dev.vk.com/ru/api/api-requests. – 09.04.2025.

13 Шнырёв, С. Л. Базы данных: учебное пособие. / С. Л. Шнырёв. – Москва: Изд-во НИЯУ МИФИ, 2011. – 224 с.

14 A pattern language for microservices [Электронный ресурс]. – Режим до-ступа: https://microservices.io/patterns/. – 09.04.2025.

15 aiogram [Электронный ресурс]. – Режим доступа: https://aiogram.dev/. – 09.04.2025.

16 Alembic’s documentation [Электронный ресурс]. – Режим доступа: https://alembic.sqlalchemy.org/en/latest/. – 09.04.2025.

17 Docker [Электронный ресурс]. – Режим доступа: https://www.docker.com/. – 09.04.2025.

18 FastAPI framework documentation [Электронный ресурс]. – Режим до-ступа: https://fastapi.tiangolo.com/. – 09.04.2025.

19 HTTPX documentation [Электронный ресурс]. – Режим доступа: https://www.python-httpx.org/. – 09.04.2025.

20 Introduction to microservices [Электронный ресурс]. – Режим доступа: https://cloud.google.com/architecture/microservices-architecture-introduction. – 09.04.2025.

21 Microservices [Электронный ресурс]. – Режим доступа: https://aws.amazon.com/ru/microservices/. – 09.04.2025.

22 Pinia [Электронный ресурс]. – Режим доступа: https://pinia.vuejs.org/. – 09.04.2025.

23 PostgreSQL 16.3 documentation [Электронный ресурс]. – Режим досту-па: https://www.postgresql.org/docs/16/index.html. – 09.04.2025.

24 PrimeVue [Электронный ресурс]. – Режим доступа: https://primevue.org/. – 09.04.2025.

25 Pydantic documentation [Электронный ресурс]. – Режим доступа: https://docs.pydantic.dev/latest/. – 09.04.2025.

26 Pytest documentation [Электронный ресурс]. – Режим доступа: https://docs.pytest.org/en/7.1.x/contents.html. – 09.04.2025.

27 Python 3.12.4 documentation [Электронный ресурс]. – Режим доступа: https://docs.python.org/3/. – 09.04.2025.

28 Redis [Электронный ресурс]. – Режим доступа: https://redis.io/. – 09.04.2025.

29 RESTful API Design Guide by Google [Электронный ресурс]. – Режим доступа: https://cloud.google.com/apis/design. – 09.04.2025.

30 SQLAlchemy documentation [Электронный ресурс]. – Режим доступа: https://www.sqlalchemy.org/. – 09.04.2025.

31 tailwindcss [Электронный ресурс]. – Режим доступа: https://tailwindcss.com/. – 09.04.2025.

32 Telegram Bot API documentation [Электронный ресурс]. – Режим досту-па: https://core.telegram.org/bots/api/. – 09.04.2025.

33 traefik [Электронный ресурс]. – Режим доступа: https://traefik.io/traefik/. – 09.04.2025.

34 Vite [Электронный ресурс]. – Режим доступа: https://vite.dev/. – 09.04.2025.

35 VK API documentation [Электронный ресурс]. – Режим доступа: https://dev.vk.com/en/guide/. – 09.04.2025.

36 Vue.js [Электронный ресурс]. – Режим доступа: https://vuejs.org/. – 09.04.2025.

ПРИЛОЖЕНИЕ А

Документооборот предприятия

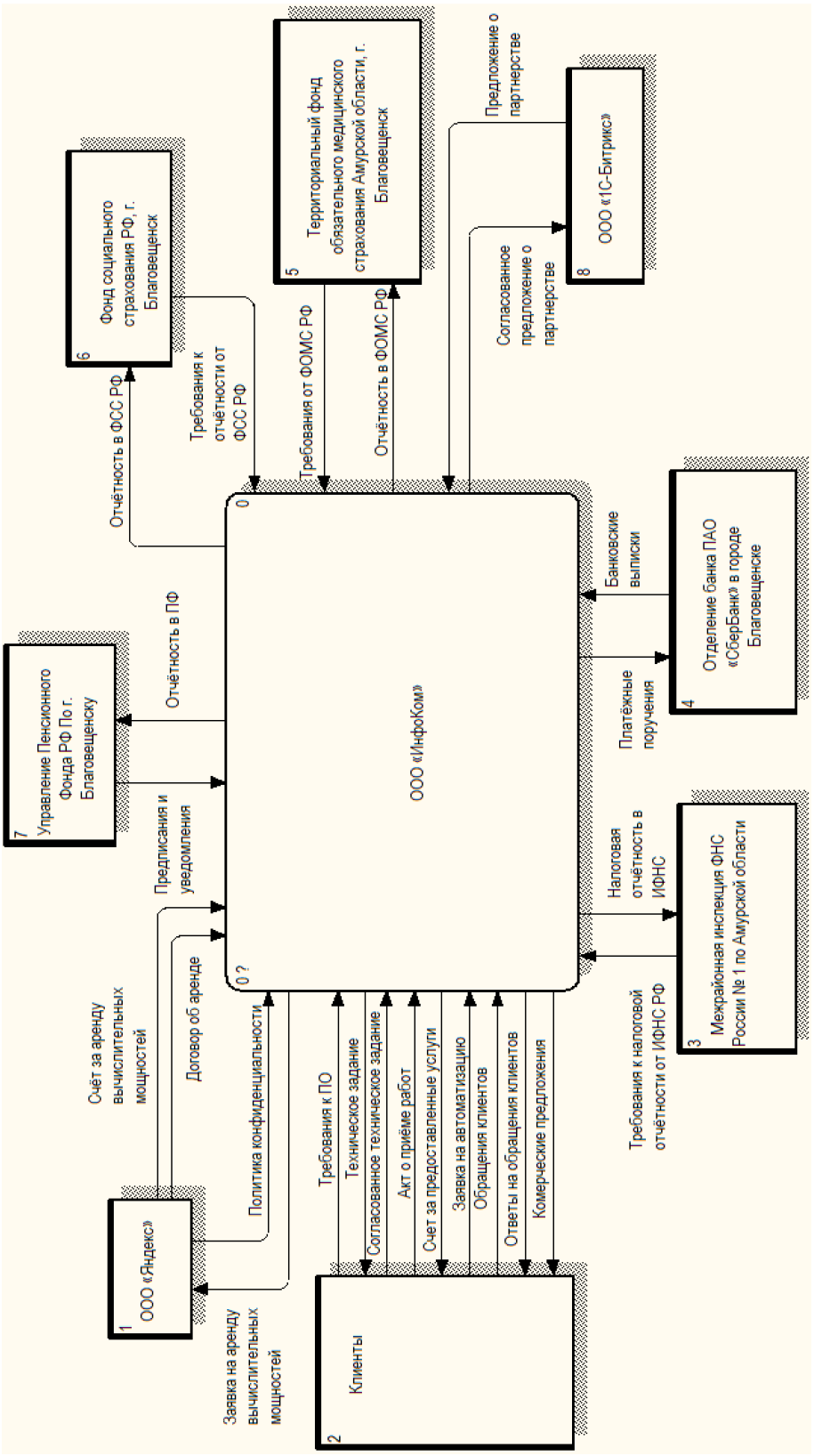

Рисунок А.1 – Диаграмма потока данных внешнего документооборота предприятия

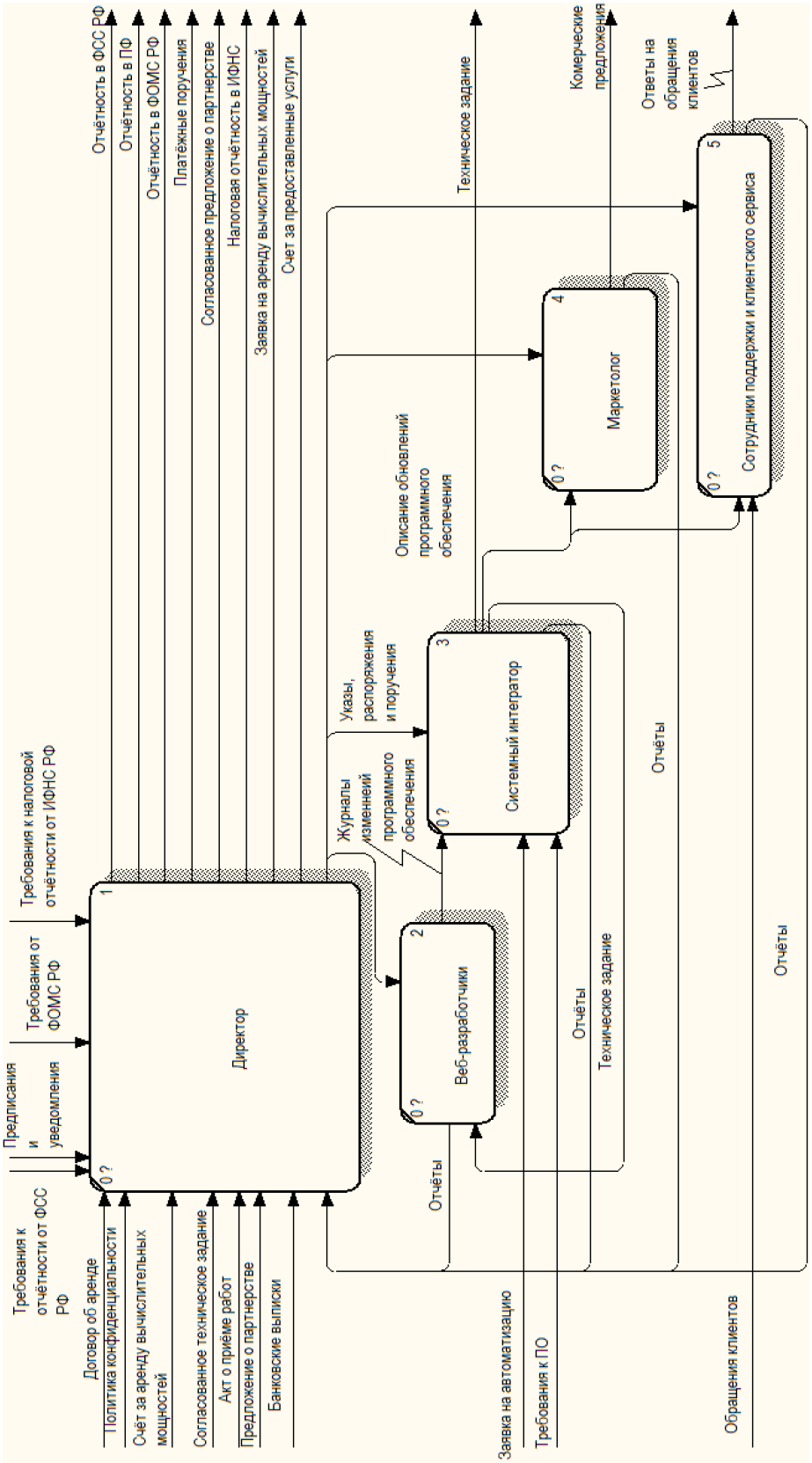

Рисунок А.2 – Диаграмма потока данных внутреннего документооборота предприятия

ПРИЛОЖЕНИЕ Б

Техническое задание

1 Введение

1.1 Наименованием программы

Наименование программы «MessageHub».

1.2 Краткая характеристика области применения программы

Система предназначена для обеспечения двустороннего взаимодействия между клиентами организации и ее сотрудниками, посредствам обращения клиентов в организацию через Telegram и ВКонтакте с последующей их обработкой через web-интерфейс данной системы, сотрудниками организации.

Данная система может быть использована, например, в службах поддержки клиентов, в системах онлайн-торговли.

2 Основание для разработки

2.1. Основание для проведения разработки

Данный проект носит учебный характер.

2.2. Наименование и условное обозначение темы разработки

Наименование темы разработки – «MessageHub».

Условное обозначение разработки – «MessageHub».

3 Назначение разработки

3.1 Функциональное назначение программы

Функциональное назначение разрабатываемого программного обеспечения заключается в создании комплексной системы управления деловыми коммуникациями организации, объединяющей в едином web-интерфейсе обращения клиентов из различных каналов взаимодействия с организацией. В рамках первой итерации предусмотрена интеграция с мессенджером Telegram и социальной сетью ВКонтакте.

3.2 Эксплуатационное назначение программы

Данная система предназначена для использования в службе поддержки клиентов организации.

4 Требования к программе

4.1 Требования к функциональным характеристикам

4.1.1 Требования к составу выполняемых функций

Программа должна обеспечивать возможность выполнения следующих функций:

– получение сообщений отправленных, клиентами организации, чат-боту в Telegram или в сообщество в ВКонтакте;

– отображение полученных сообщений в web-интерфейсе системы;

– отправку ответных сообщений клиентам организации от имени чат-бота в Telegram или сообщества в ВКонтакте через web-интерфейс системы.

4.1.2 Требования к организации входных данных

Со стороны клиентов организации входные данные должны быть организованны в виде сообщений, отправляемых в чат-бота Telegram или в сообщество в ВКонтакте.

Со стороны сотрудников организации входные данные должны быть организованны в виде сообщений, отправляемых в web-интерфейс программы.

4.1.3. Требования к организации выходных данных

Со стороны клиентов организации выходные данные должны быть организованны в виде сообщений, отправляемых от имени чат-бота в Telegram или сообщества в ВКонтакте.

Со стороны сотрудников организации выходные данные должны быть организованны в виде чатов с клиентами организации, представленных в web-интерфейсе системы.

4.1.4. Требования к временным характеристикам

Рекомендованное время пересылки недолжно превышать 1 секунды, максимально допустимым считается 2 секунды, без учёта обработки файлов.

4.2 Требования к надёжности

4.2.1. Требования к обеспечению надежного функционирования программы

Для надёжного функционирования системы должны обеспечивается следующие организационно-технические мероприятия:

– организация бесперебойного питания технических средств;

– организация бесперебойного доступа к сети «Интернет»;

– организация бесперебойного доступа к Telegram;

– организация бесперебойного доступа к ВКонтакте;

– наличие доменного имени;

– наличие статического IP-адреса.

4.2.2. Время восстановления после отказа

Восстановление системы после отказа, вызванного отсутствием электропитания, не должно превышать времени загрузки и запуска служб операционной системы.

Восстановление системы после отказа, вызванного отсутствием доступа к сети «Интернет», не должно превышать времени восстановления доступа к сети.

Восстановление системы после отказа, вызванного отсутствием доступа к Telegram или ВКонтакте, не должно превышать времени восстановления доступа к нему.

4.3. Условия эксплуатации

4.3.1. Климатические условия эксплуатации

Климатические условия эксплуатации определяются, требованиями к климатическим условиям эксплуатации используемых технических средств.

4.3.2. Требования к видам обслуживания

Требования к видам обслуживания включают в себя требованиями к видам обслуживания используемых технических средств.

4.3.3. Требования к численности и квалификации персонала

Для развёртывания и поддержания работоспособности достаточно одного системного администратора.

В задачи системного администратора входят:

– поддержание работоспособности технических средств;

– развёртывание системы и необходимого программного обеспечения.

Для корректного обслуживания клиентов организации требуется как минимум один сотрудник, который изучил руководство пользователя.

Требования к квалификации клиентов организации ограничиваются умением пользоваться Telegram или ВКонтакте.

4.4. Требования к составу и параметрам технических средств

Состав технических средств ограничивается сервером и клиентскими компьютерами.

К серверному компьютеру предъявляются следующие минимальные требования:

– процессор x86 с двумя ядрами и тактовой частотой, не менее 3.5 ГГц или 2 vCPU;

– оперативная память объёмом, не менее 2 Гб;

– жёсткий диск объёмом не менее 20 Гб;

– выход в интернет, со скоростью от 1000Мбит/с.

Требования к клиентским устройствам определяются используемым web-браузером на основе Chromium начиная с версии 44.0.2403.157.

4.5. Требования к информационной и программной совместимости

4.5.1. Требования к информационным структурам и методам решения

Пользовательский интерфейс должен быть реализован на основе web-технологий.

Взаимодействие клиентских устройств с серверной частью должно осуществляется посредствам REST API в формате JSON.

4.5.2. Требования к исходным кодам и языкам программирования

Для написания серверной части системы предписывается использовать язык программирования Python.

Для написания клиентской части системы предписывается использовать язык программирования JavaScript с CSS и HTML.

4.5.3. Требования к программным средствам, используемым программой

Все используемые программе средства, используемые программой должны обладать возможностью помещения в контейнер Docker.

При использовании реляционной базы данных предписывается использовать СУБД PostgreSQL.

В случае необходимости реализации кэша необходимо использовать Redis.

Также для обеспечения работы, возможно, использование обратного прокси сервера.

4.6 Требование к маркировке и упаковке

Требования к упаковке отсутствуют так как система передаётся по сети «Интернет», через репозиторий GitHub.

4.7 Требования к транспортированию и хранению

Отсутствуют, в силу распространения по сети «Интернет», через репозиторий GitHub.

4.8 Специальные требования

4.8.1 Требования к эргономике и технической эстетике

Эргономические и эстетические требования включают удобство взаимодействия пользователей с интерфейсом, простоту восприятия информации и лёгкость навигации.

Интерфейс должен быть интуитивно понятен и удобен для всех категорий пользователей, независимо от их уровня подготовки. Кнопки, функции и меню должны располагаться логично и быть легко доступными.

Цветовая палитра должна быть гармоничной и спокойной, чтобы обеспечить комфортное взаимодействие на протяжении длительного времени. Шрифты, значки и кнопки должны быть единообразными. Элементы управления должны быть достаточно крупными для удобного нажатия, при этом расстояние между ними должно исключать случайные нажатия.

С точки зрения эстетики, интерфейс должен быть визуально привлекательным, гармонично сочетая минимализм и современные тенденции в дизайне. Он должен не только привлекать внимание пользователя, но и создавать ощущение легкости и простоты. Каждый элемент интерфейса должен быть продуманным и эргономичным, чтобы не отвлекать от основного функционала.

5 Требования к программной документации

Требования к документации устанавливаются следующими документами:

– ГОСТ 19.101-77 Виды программных документов;

– ГОСТ 19.102-77 Стадии разработки;

– ГОСТ 19.201-78 Техническое задание. Требования к содержанию и оформлению;

– ГОСТ 19.401-78 Текст программы;

– ГОСТ 19.402 -78 Описание программы;

– ГОСТ 19.404-79 Пояснительная записка;

– ГОСТ 19.502-78 Описание применения;

– ГОСТ 19.503-79 Руководство системному программисту;

– ГОСТ 19.504-79 Руководство программиста;

– ГОСТ 19.505-79 Руководство оператору;

– ГОСТ 19.701-90 Схемы алгоритмов, программ данных и систем.

6 Стадии и этапы разработки

Стадии и этапы разработки приведены в таблице Б.1.

Таблица Б.1 – Стадии и этапы разработки

Стадия разработки	Этапы разработки	Сроки	Содержание работ
Техническое задание	Определение требований к системе.	27.03.2025 – 28.03.2025	Проведение собеседования с заказчиком с целью выявления требований.
	Разработка технического задания.	31.03.2025 – 05.04.2025	Составление технического задания.
	Согласование и утверждение технического задания.	06.04.2025	Техническое задание должно быть изучено заказчиком, после чего должны быть устранены все недочёты и подписано техническое задание.
Эскизный проект	Разработка эскизного проекта	07.04.2025 – 20.04.2025	Декомпозиция задачи, определение форматов данных и интерфейсов.
Технический проект	Разработка технического проекта	21.04.2025 – 27.04.2025	Детальное описание методов и алгоритмов.
Рабочий проект	Разработка рабочего проекта	28.04.2025 – 18.05.2025	Реализация программы.
	Разработка программной документации	19.05.2025 – 25.05.2025	Разработка программных документов в соответствии с требованиями.
Внедрение	Тестирование программного средства	26.05.2025 – 01.06.2025	Тестирование программного средства.

7 Порядок контроля и приёмки

Приёмочные испытания программы должны проводиться согласно разработанной исполнителем и согласованной заказчиком «Программы и методики испытаний». Ход проведения приемо-сдаточных испытаний документируется в протоколе испытаний. На основании протокола испытаний подписывается акт приемки-сдачи программы в эксплуатацию.

ПРИЛОЖЕНИЕ В

Диаграмма вариантов использования

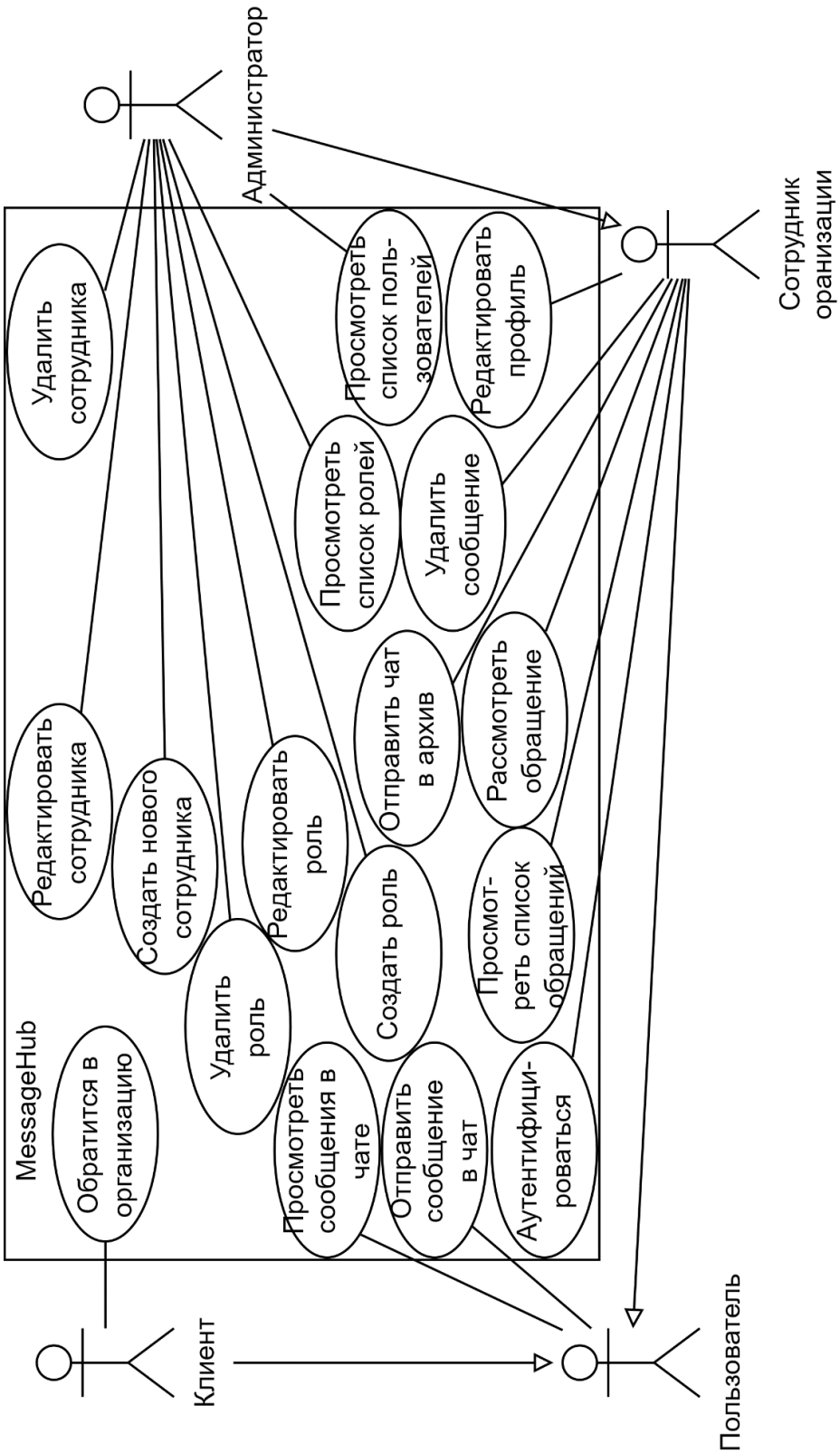

Рисунок В.1 – UML-диаграмма вариантов использования проектируемого программного обеспечения

ПРИЛОЖЕНИЕ Г

Диаграмма компонентов

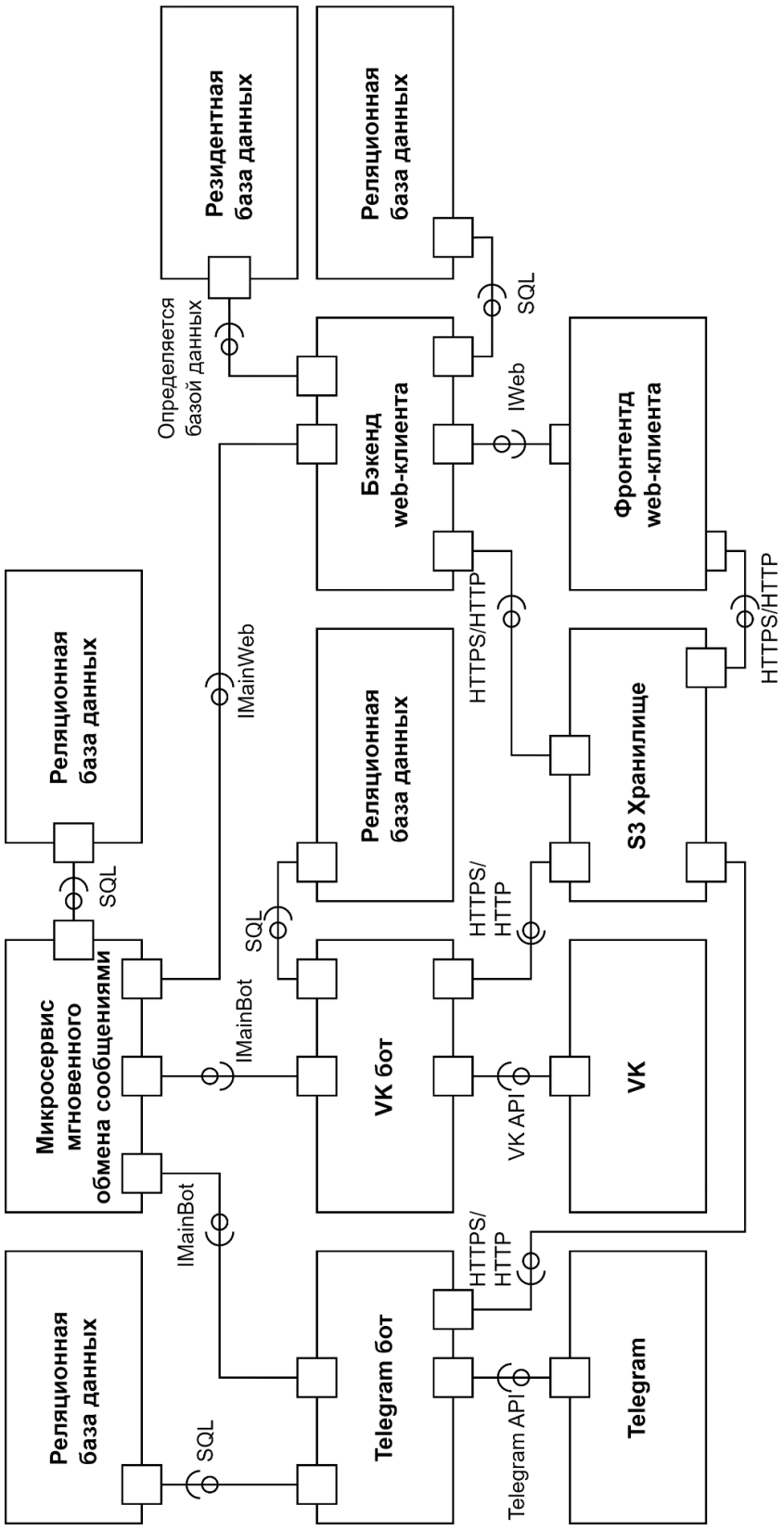

Рисунок Г.1 – UML-диаграмма компонентов программы

ПРИЛОЖЕНИЕ Д

Диаграммы последовательностей

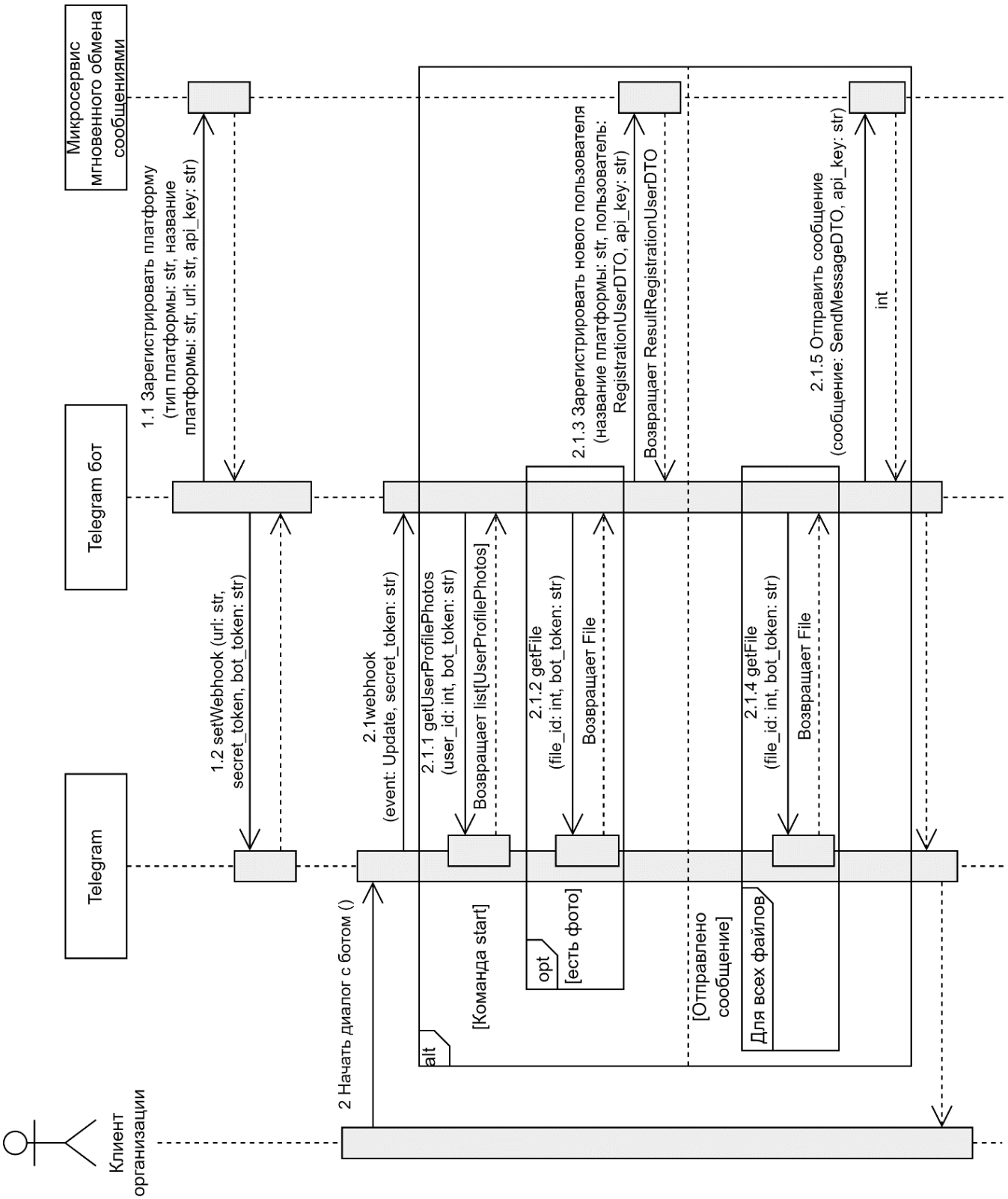

Рисунок Д.1 – UML-диаграмма последовательности взаимодействия пользователя с разрабатываемым программным обеспечением посредствам Telegram бота часть 1

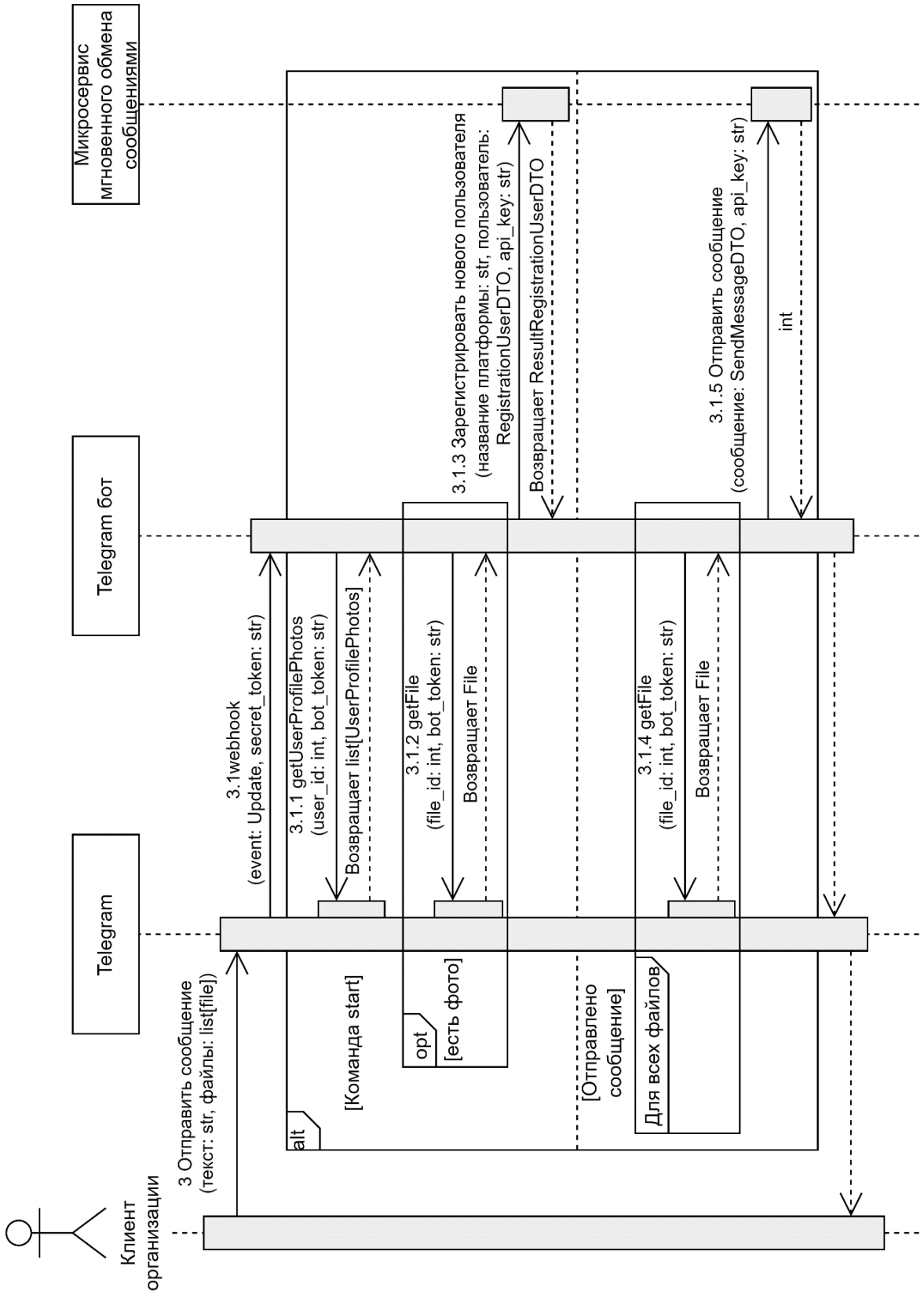

Рисунок Д.2 – UML-диаграмма последовательности взаимодействия пользователя с разрабатываемым программным обеспечением посредствам Telegram бота часть 2

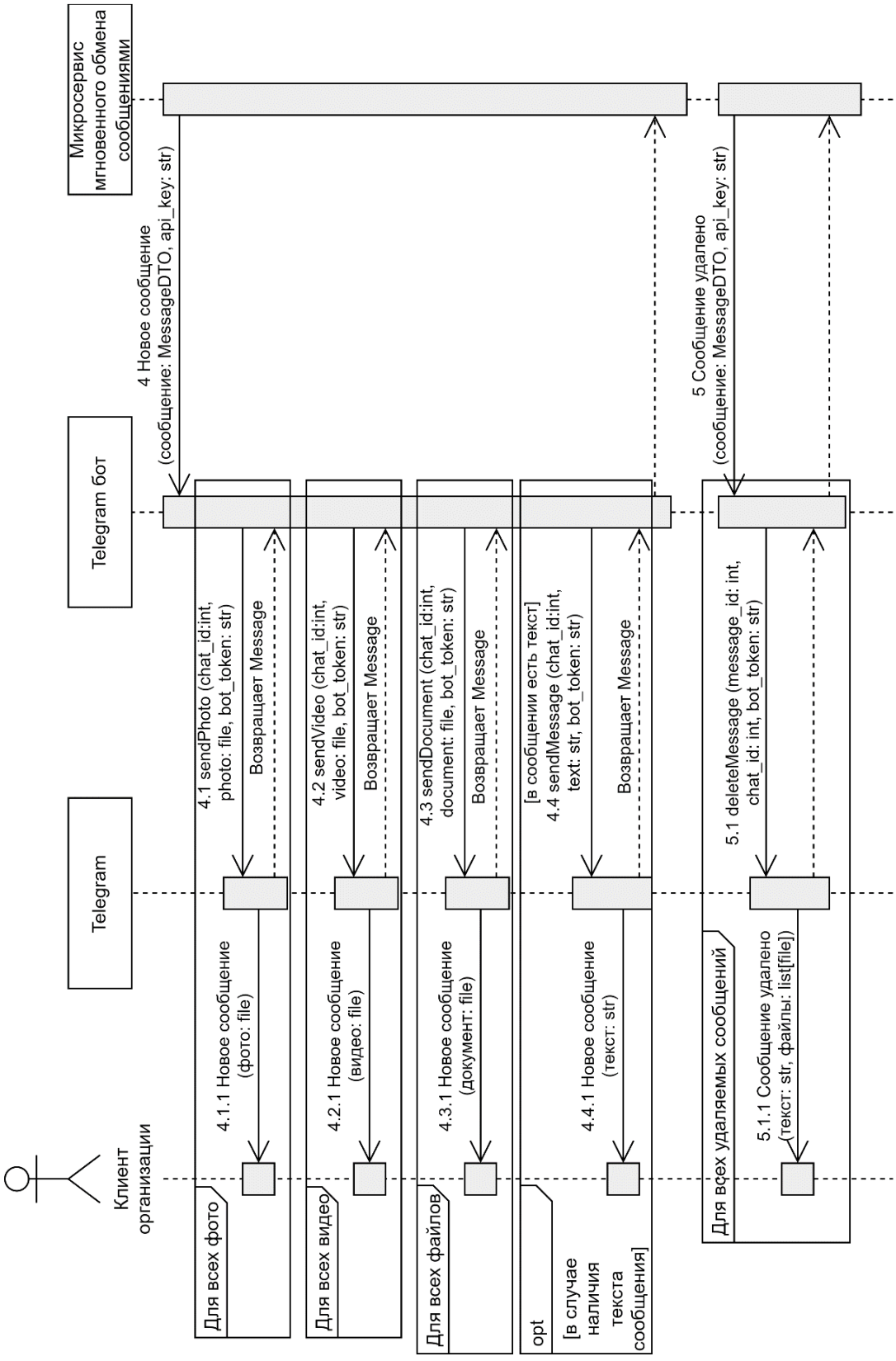

Рисунок Д.3 – UML-диаграмма последовательности взаимодействия пользователя с разрабатываемым программным обеспечением посредствам Telegram бота часть 3

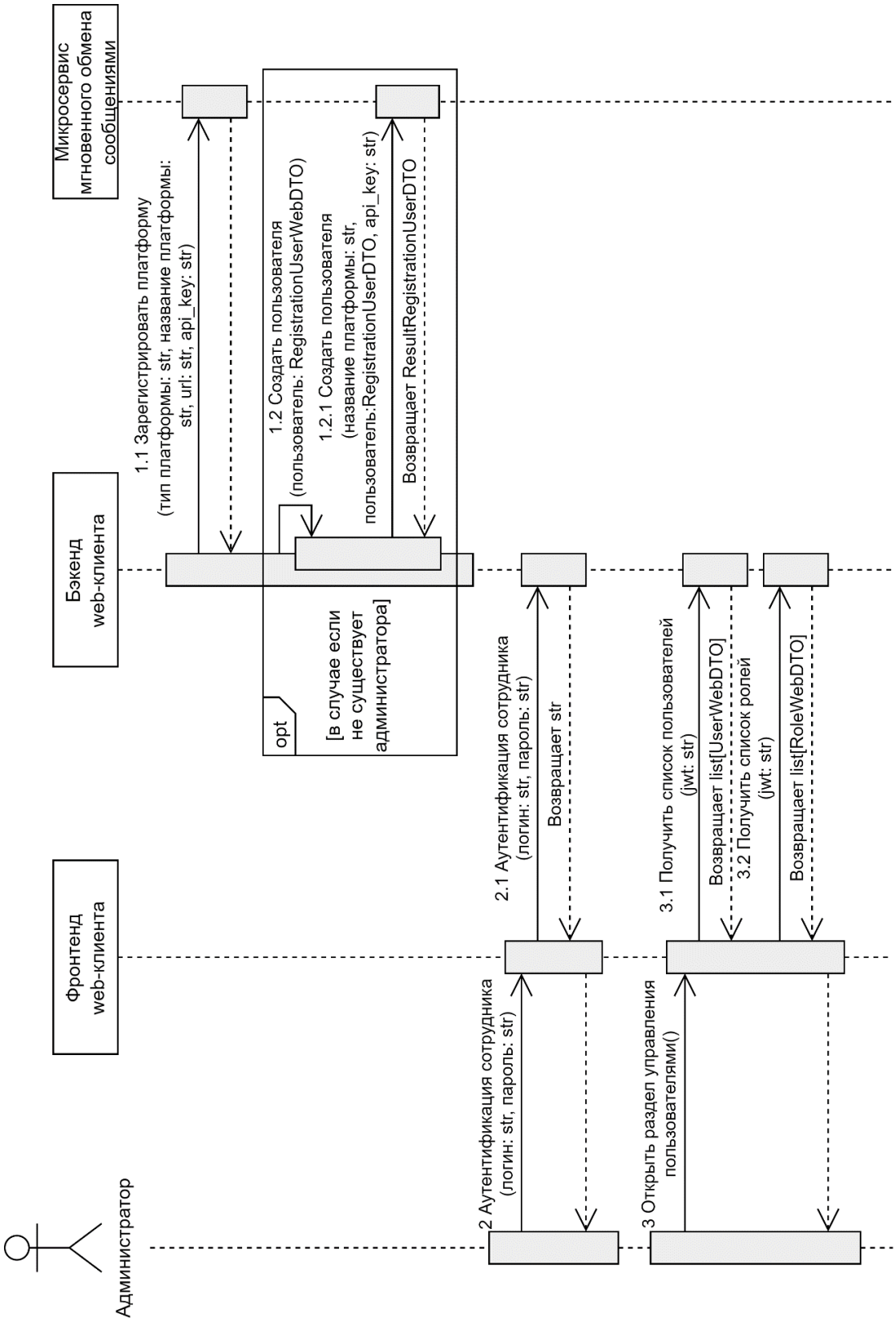

Рисунок Д.4 – UML-диаграмма последовательности взаимодействия администратора с разрабатываемым программным обеспечением часть 1

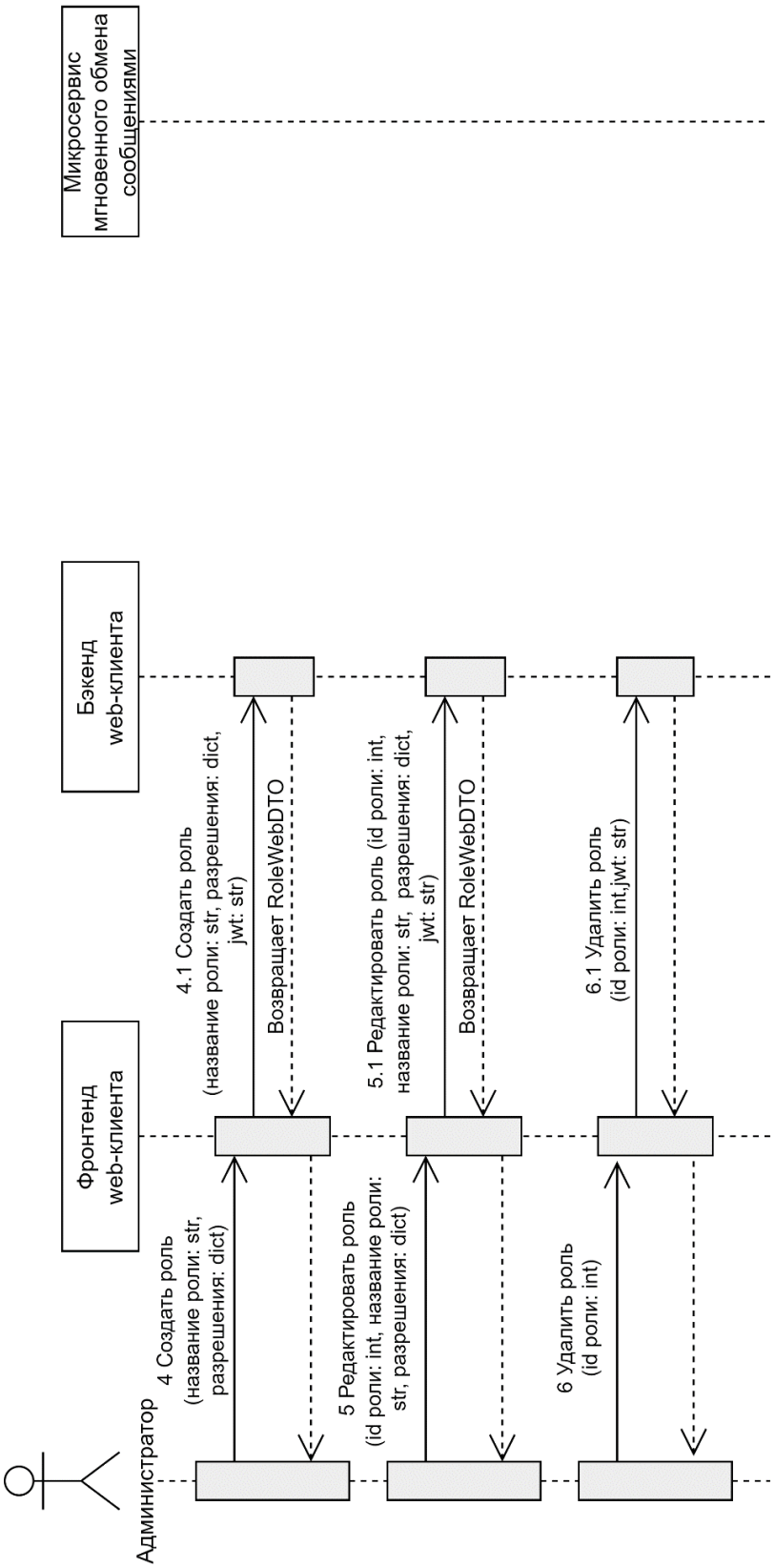

Рисунок Д.5 – UML-диаграмма последовательности взаимодействия администратора с разрабатываемым программным обеспечением часть 2

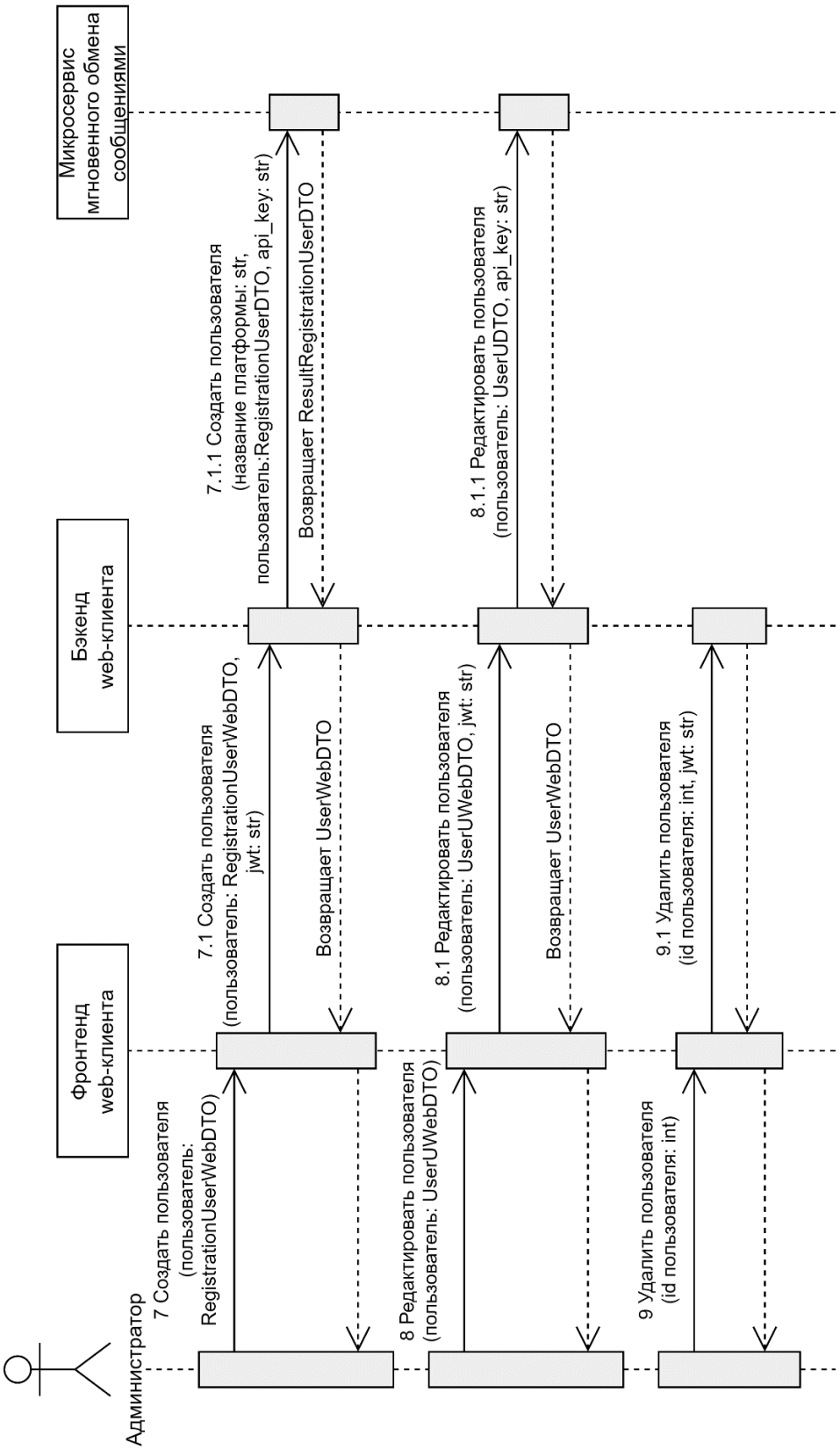

Рисунок Д.6 – UML-диаграмма последовательности взаимодействия администратора с разрабатываемым программным обеспечением часть 3

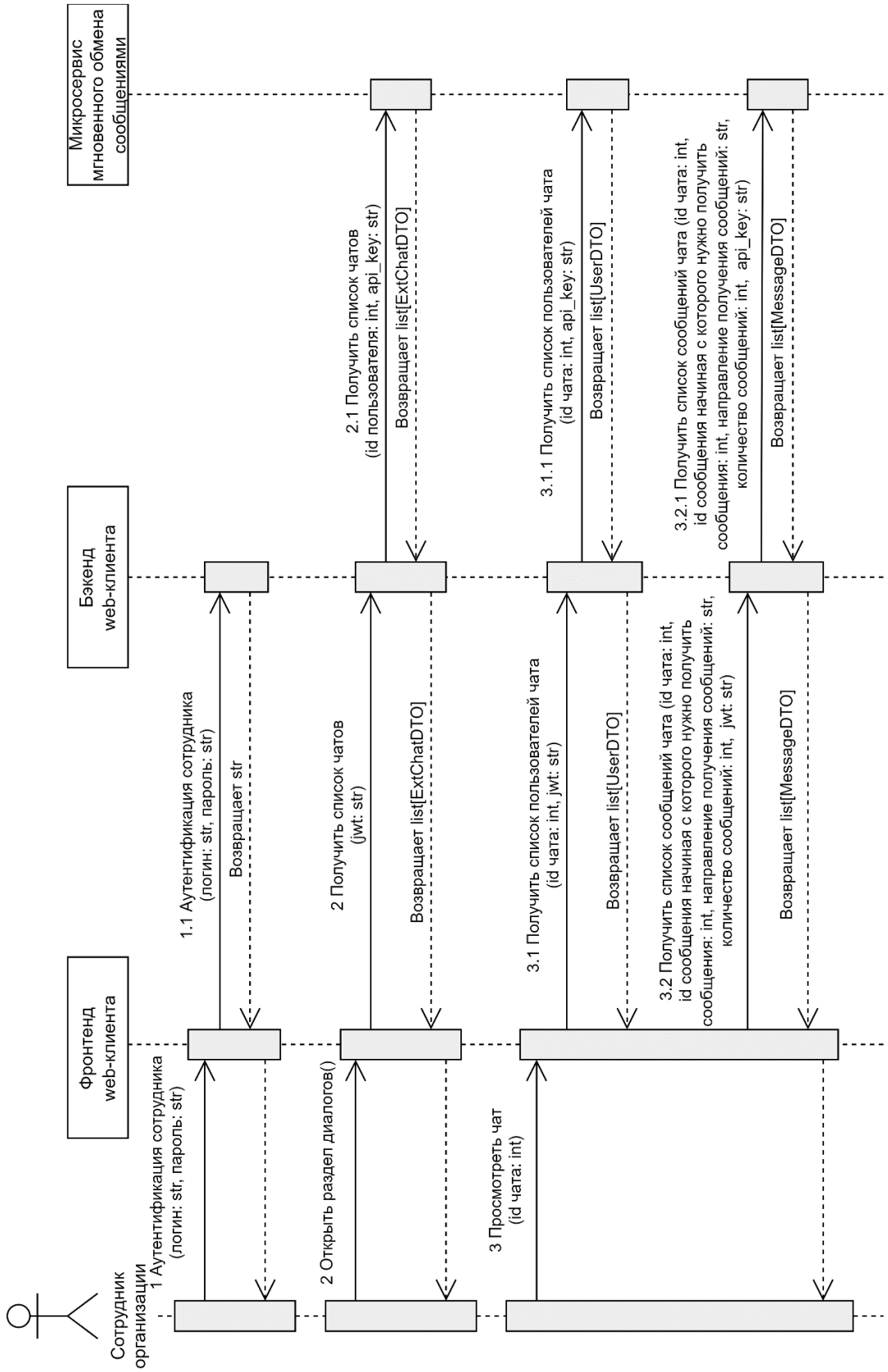

Рисунок Д.7 – UML-диаграмма последовательности взаимодействия сотрудника с разрабатываемым программным обеспечением часть 1

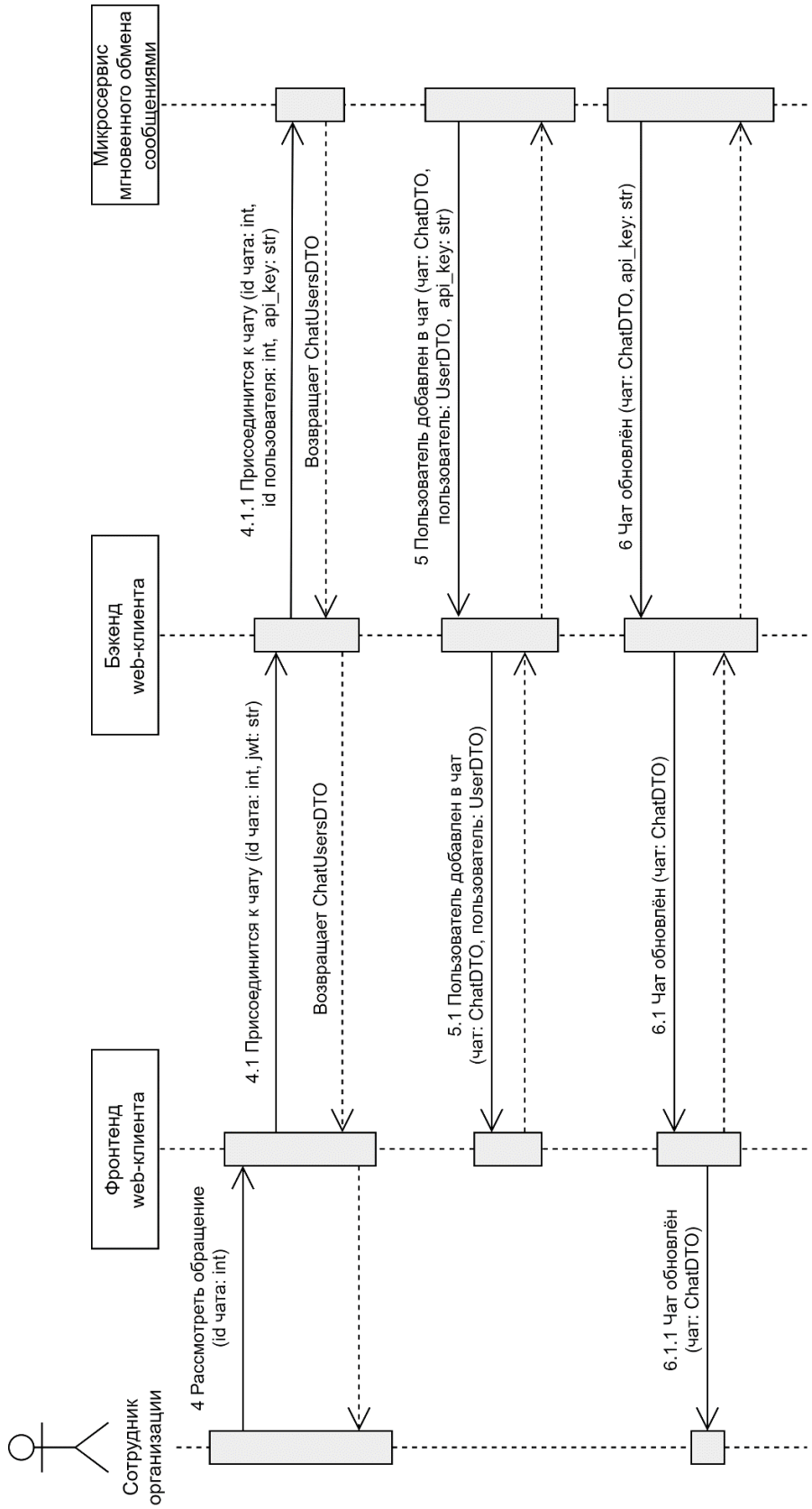

Рисунок Д.8 – UML-диаграмма последовательности взаимодействия сотрудника с разрабатываемым программным обеспечением часть 2

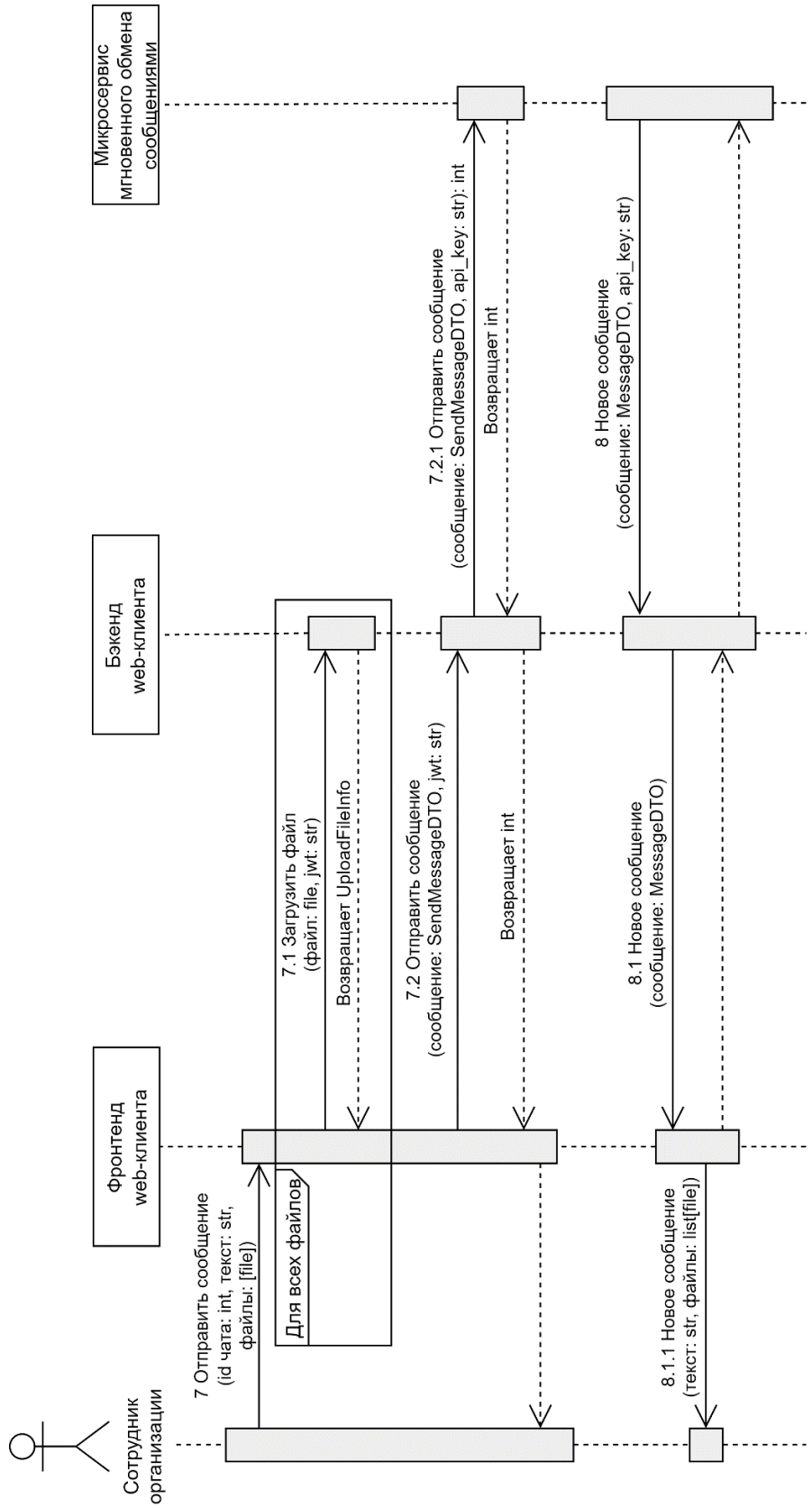

Рисунок Д.9 – UML-диаграмма последовательности взаимодействия сотрудника с разрабатываемым программным обеспечением часть 3

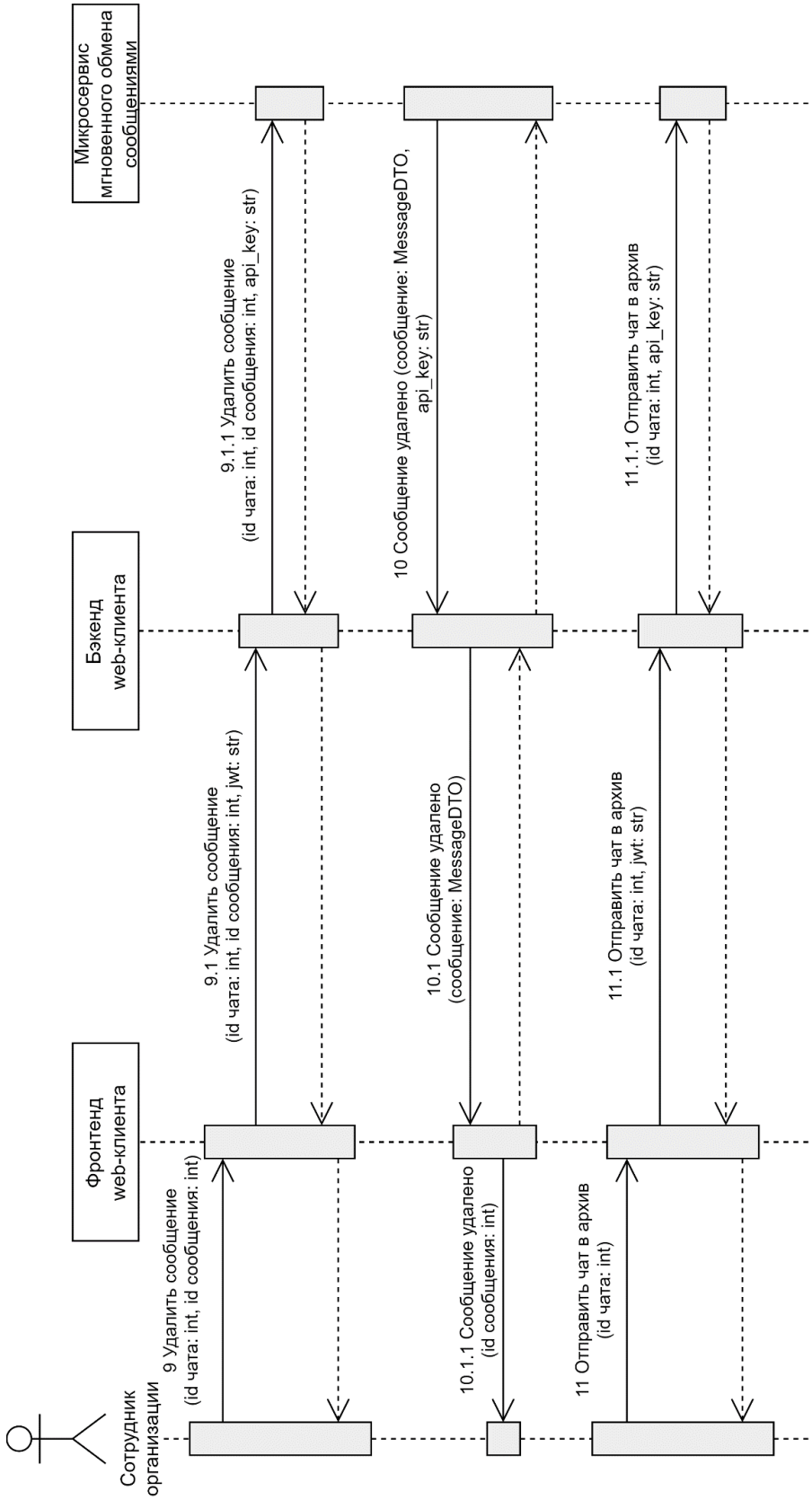

Рисунок Д.10 – UML-диаграмма последовательности взаимодействия сотрудника с разрабатываемым программным обеспечением часть 4

ПРИЛОЖЕНИЕ Е

Используемые структуры данных

Таблица Е.1 – Структура данных RegistrationUserDTO

Поле	Тип	Описание
name	String	Имя пользователя, ограниченное 256 символами.
icon_url	String	url фото профиля пользователя. Опциональное поле. Строка ограничена 256 символами.

Таблица Е.2 – Структура данных ResultRegistrationUserDTO

Поле	Тип	Описание
user_id	Integer	Идентификатор зарегистрированного пользователя.
chat_id	Integer	Идентификатор чата, в который был добавлен пользователь. Данное возвращается только для платформ типа «web».

Таблица Е.3 – Структура данных RegistrationUserWebDTO

Поле	Тип	Описание
name	String	Имя создаваемого пользователя. Строка ограничена 256 символами.
email	String	Адрес электронной почты создаваемого пользователя.
password	String	Пароль создаваемого пользователя.
icon_url	String	url фото профиля пользователя. Опциональное поле. Строка ограничена 256 символами.

Таблица Е.4 – Структура данных UserDTO

Поле	Тип	Описание
id	Integer	Идентификатор пользователя.
platform_id	Integer	Идентификатор платформы к которой относится данный пользователь.
name	String	Имя пользователя, ограниченное 256 символами.
icon_url	String	url фото профиля пользователя. Опциональное поле. Строка ограничена 256 символами.

Таблица Е.5 – Структура данных UserWebDTO

Поле	Тип	Описание
id	Integer	Идентификатор пользователя.
name	String	Имя пользователя, ограниченное 256 символами.
email	String	Адрес электронной почты. Ограничено 256 символами.
hashed_password	String	Хэш-сумма пароля. Длинна строки ограничена 1024 символами.

Продолжение таблицы Е.5

Поле	Тип	Описание
is_active	Boolean	Флаг активности пользователя. В случае false указывает, что пользователь удалён.
is_root	Boolean	Флаг суперпользователя. Пользователи с данным флагом игнорируют ограничения ролей.
icon_url	String	url фото профиля пользователя. Опциональное поле. Строка ограничена 256 символами.
role_id	Integer	Идентификатор роли пользователя. Опциональное поле.
settings	JSON	Настройки пользователя.

Таблица Е.6 – Структура данных UserUDTO

Поле	Тип	Описание
id	Integer	Идентификатор пользователя.
name	String	Имя пользователя, ограниченное 256 символами. Опциональное поле.
icon_url	String	url фото профиля пользователя. Опциональное поле. Строка ограничена 256 символами.

Таблица Е.7 – Структура данных UserUWebDTO

Поле	Тип	Описание
id	Integer	Идентификатор пользователя.
name	String	Имя пользователя, ограниченное 256 символами. Опциональное поле.
email	String	Адрес электронной почты. Ограничено 256 символами. Опциональное поле.
password	String	Хэш-сумма пароля. Длинна строки ограничена 1024 символами. Опциональное поле.
is_active	Boolean	Флаг активности пользователя. В случае false указывает, что пользователь удалён. Опциональное поле.
is_root	Boolean	Флаг суперпользователя. Пользователи с данным флагом игнорируют ограничения ролей. Опциональное поле.
icon_url	String	url фото профиля пользователя. Опциональное поле. Строка ограничена 256 символами.
role_id	Integer	Идентификатор роли пользователя. Опциональное поле.
settings	JSON	Настройки пользователя. Опциональное поле.

Таблица Е.8 – Структура данных RoleWebDTO

Поле	Тип	Описание
id	Integer	Идентификатор роли.
name	String	Название роли, ограниченное 256 символами.

Продолжение таблицы Е.8

Поле	Тип	Описание
permissions	JSON	Разрешения роли.

Таблица Е.9 – Структура данных ChatDTO

Поле	Тип	Описание
id	Integer	Идентификатор чата.
name	String	Название чата, ограниченное 256 символами.
is_waiting_answer	Boolean	Флаг указывающий, что чат находится в состоянии «Входящий»
is_archive	Boolean	Флаг указывающий, что чат находится в архиве.
platform_id	Integer	Идентификатор платформы к которой относится данный чат.
icon_url	String	url иконки чата. Опциональное поле. Строка ограничена 256 символами.
last_message_send_at	TimeStamp	Время отправки последнего сообщения в чат.

Таблица Е.10 – Структура данных ExtChatDTO

Поле	Тип	Описание
id	Integer	Идентификатор чата.
name	String	Название чата, ограниченное 256 символами.
is_waiting_answer	Boolean	Флаг указывающий, что чат находится в состоянии «Входящий»
is_archive	Boolean	Флаг указывающий, что чат находится в архиве.
platform_id	Integer	Идентификатор платформы к которой относится данный чат.
icon_url	String	url иконки чата. Опциональное поле. Строка ограничена 256 символами.
last_message_send_at	TimeStamp	Время отправки последнего сообщения в чат. Опциональное поле.
last_read_message_id	Integer	Идентификатор последнего прочитанного сообщения в чате указанным пользователем. Опциональное поле.
user_in_chat	Boolean	Флаг указывающий присутствие указанного пользователя в данном чате. Опциональное поле.
count_unredeble_messgaes	Integer	Количество непрочитанных сообщений указанным пользователем. Опциональное поле.
platform_name	String	Название платформы к которой относится чат. Опциональное поле.

Таблица Е.11 – Структура данных ChatUsersDTO

Поле	Тип	Описание
user_id	Integer	Идентификатор пользователя.
chat_id	Integer	Идентификатор чата.
user_in_chat	Boolean	Флаг указывающий находится ли пользователь в чате.
last_read_ message_id	Integer	Идентификатор последнего прочитанного сообщения в чате пользователем. Опциональное поле.
user_in_chat	Boolean	Флаг указывающий присутствие указанного пользователя в данном чате.

Таблица Е.12 – Структура данных MessageDTO

Поле	Тип	Описание
id	Integer	Идентификатор сообщения.
chat_id	Integer	Идентификатор чата в который отправлено сообщение.
sender_id	Integer	Идентификатор пользователя отправившего сообщение.
send_at	TimeStamp	Время отправки сообщения.
text	String	Текст сообщения длинной в 4096 символов. Опциональное поле.
attachments	JSON	Описание прикреплённых к сообщению файлов. Содержит массивы «images», «videos», «files» с изображениями, видео и файлами соответственно. Сведения об изображении хранятся в структуре Image, видео в Video, а файлы в File.
is_hide	Boolean	Флаг удаления сообщения.
delete_at	TimeStamp	Время удаления сообщения. Опциональное поле.

Таблица Е.13 – Структура данных Image

Поле	Тип	Описание
name	String	Название изображения.
url	String	url по которому можно скачать изображение.

Таблица Е.14 – Структура данных Video

Поле	Тип	Описание
name	String	Название видео.
url	String	url по которому можно скачать видео.
miniature	Image	Превью видео.

Таблица Е.15 – Структура данных File

Поле	Тип	Описание
name	String	Название файла.
url	String	url по которому можно скачать файл.

Таблица Е.16 – Структура данных UploadFileInfo

Поле	Тип	Описание
url	String	url по которому можно скачать загруженный файл.

Таблица Е.17 – Структура данных SendMessageDTO

Поле	Тип	Описание
chat_id	Integer	Идентификатор чата в который отправлено сообщение.
sender_id	Integer	Идентификатор пользователя отправившего сообщение.
text	String	Текст сообщения длинной в 4096 символов. Опциональное поле.
attachments	JSON	Описание прикреплённых к сообщению файлов. Содержит массивы «images», «videos», «files» с изображениями, видео и файлами соответственно. Сведения об изображении хранятся в структуре Image, видео в Video, а файлы в File.

Таблица Е.18 – Фрагмент структуры данных «Update»

Поле	Тип	Описание
update_id	Integer	Уникальный идентификатор обновления, которые начинаются с некоторого положительного числа и увеличиваются.
message	Message	Новое присланное сообщение. Необязательное поле, указываемое в случае отправки боту нового сообщения.
edited_message	Message	Новая версия сообщения, которая известна боту и была отредактирована. Необязательное поле, указываемое в случае если сообщение было изменено.

Таблица Е.19 – Фрагмент структуры данных «Message»

Поле	Тип	Описание
message_id	Integer	Уникальный идентификатор сообщения.
from	User	Опционально. Отправитель. Может быть пустым в каналах.
date	Integer	Дата отправки сообщения (Unix time).
chat	Chat	Диалог, в котором было отправлено сообщение.

Продолжение таблицы Е.19

Поле	Тип	Описание
text	String	Опционально. Для текстовых сообщений: текст сообщения, 0-4096 символов.
entities	Массив из MessageEntity	Опционально. Для текстовых сообщений: особые сущности в тексте сообщения.
photo	Массив из PhotoSize	Опционально. Список фотографий различного размера.
video	Video	Опционально. Видео отправленное в сообщении.
document	Document	Опционально. Информация о файле.

ПРИЛОЖЕНИЕ Ж

Описание API микросервисов

1 Описание API микросервиса мгновенного обмена сообщениями

1.1 Метод «Зарегистрировать платформу»

Описание: Метод предназначен для регистрации новой платформы в системе микросервиса мгновенного обмена сообщениями, например, Telegram бота. Используется при инициализации платформы, чтобы установить связь между внешним интерфейсом и системой сообщений.

Параметры: приведены в таблице Ж.1.

Таблица Ж.1 – Параметры метода «Зарегистрировать платформу»

Параметр	Тип	Описание
Тип платформы	str	Строка, принимающая значения «bot» или «web». Определяет используемый микросервисом тип интерфейса.
Название платформы	str	Уникальное название платформы в системе. Используется как идентификатор.
url	str	URL-адрес, на котором подключаемый микросервис будет ожидать события от микросервиса мгновенного обмена сообщенями.
api_key	str	Ключ API для аутентификации.

Содержание: В ходе выполнения метода осуществляется проверка существования в базе данных платформы с указанным названием «Название платформы». Если такая платформа уже существует, осуществляется обновление сведений о данной платформе иначе, создаётся новая запись с переданными данными. После выполнения данных действий обработка запроса завершается.

Результат: нет.

1.2 Метод «Зарегистрировать нового пользователя»

Описание: Метод предназначен для регистрации новых пользователей в микросервиса мгновенного обмена сообщениями. В случае регистрации пользователя для платформы типа «bot» автоматически осуществляется создание нового чата, в который добавляется создаваемый пользователь, а также состояние созданного чата будет установлено в состояние «Входящий».

Параметры: приведены в таблице Ж.2.

Таблица Ж.2 – Параметры метода «Зарегистрировать нового пользователя»

Параметр	Тип	Описание
Название платформы	str	Название платформы к которой будет прикреплён созданный пользователь.
Пользователь	Registration-UserDTO	Информация о регистрируемом пользователе.
api_key	str	Ключ API для аутентификации.

Содержание: Входе выполнения метода осуществляется создание нового пользователя в базе данных микросервиса мгновенного обмена сообщениями. Если пользователь прикрепляется к платформе типа «bot», то осуществляется создание нового чата с состоянием «Входящий» и оповещение всех платформ о изменении состояния чата.

Результат: представлен в формате ResultRegistrationUserDTO.

1.3 Метод «Отправить сообщение»

Описание: Метод отправляет сообщение в указанный чат.

Параметры: приведены в таблице Ж.3.

Таблица Ж.3 – Параметры метода «Отправить сообщение»

Параметр	Тип	Описание
Сообщение	SendMes-sageDTO	Сведения об отправляемом сообщении.
api_key	str	Ключ API для аутентификации.

Содержание: В ходе выполнения метода осуществляется извлечение сведений о пользователе и чате в который отправляется сообщение. Если чат находится в состоянии «Архив», его статус меняется на «Входящий», если сообщение отправлено клиентом, или «Активный», если сообщение отправлено сотрудником организации. Затем определяется список платформ участников чата, и сообщение отправляется этим платформам в фоновом режиме, путём вызова метода «Новое сообщение».

Результат: целое число, соответствующее идентификатору отправленного сообщения.

1.4 Метод «Удалить сообщение»

Описание: Метод удаляет указанное сообщение в указанном чате.

Параметры: приведены в таблице Ж.4.

Таблица Ж.4 – Параметры метода «Удалить сообщение»

Параметр	Тип	Описание
id чата	int	Идентификатор чата в котором необходимо удалить сообщение.
id сообщения	int	Идентификатор сообщения, которое необходимо удалить.
api_key	str	Ключ API для аутентификации.

Содержание: В ходе выполнения метода проверяется наличие чата и сообщения указанных в параметрах. В случае успешной проверки, сообщение помечается как удалённое. После этого определяется список платформ участников чата, и для этих платформ вызывается метод «Сообщение удалено».

Результат: нет.

1.5 Метод «Редактировать пользователя»

Описание: Метод обновляет сведения о пользователя в системе.

Параметры: приведены в таблице Ж.5.

Таблица Ж.5 – Параметры метода «Редактировать пользователя»

Параметр	Тип	Описание
Пользователь	UserUDTO	Объект с обновлённой информацией о пользователе.
api_key	str	Ключ API для аутентификации.

Содержание: В ходе выполнения метода из параметра «Пользователь» извлекается идентификатор пользователя, сведения о котором нужно обновить, а затем в базе данных обновляются сведения об указанном пользователе.

Результат: нет.

1.6 Метод «Получить список чатов»

Описание: Метод предназначен для получения списка чатов, относительно указанного пользователя.

Параметры: приведены в таблице Ж.6.

Таблица Ж.6 – Параметры метода «Получить список чатов»

Параметр	Тип	Описание
id пользователя	int	Идентификатор пользователя для которого необходимо получить список чатов.
api_key	str	Ключ API для аутентификации.

Содержание: В процессе выполнения данного метода микросервис мгновенного обмена сообщениями извлечёт из своей базы данных список чатов в которых состоит пользователь, а также список чатов в которых не состоит пользователь. После чего завершит выполнение метода путём возращения списка чатов в формате ExtChatDTO.

Результат: список чатов для указанного пользователя в формате ExtChatDTO.

1.7 Метод «Получить список пользователей чата»

Описание: Метод предназначен для получения списка пользователей указанного чата.

Параметры: приведены в таблице Ж.7.

Таблица Ж.7 – Параметры метода «Получить список пользователей чата»

Параметр	Тип	Описание
id чата	int	Идентификатор чата, для которого нужно получить список пользователей.
api_key	str	Ключ API для аутентификации.

Содержание: В процессе выполнения данного метода микросервис мгновенного обмена сообщениями извлечёт из своей базы данных список пользователей чата с идентификатором «id чата», который возвращает в формате UserDTO в результате выполнения метода.

Результат: список пользователей чата в формате UserDTO.

1.8 Метод «Получить список сообщений чата»

Описание: Метод предназначен для получения списка сообщений чата.

Параметры: приведены в таблице Ж.8.

Таблица Ж.8 – Параметры метода «Получить список сообщений чата»

Параметр	Тип	Описание
id чата	int	Идентификатор чата, для которого необходимо получить сообщения.
id сообщения начиная с которого нужно получить сообщения	int	Идентификатор сообщения относительно которого нужно получать сообщения. Если идентификатор сообщения неизвестен, то он устанавливается в -1 и означает что нужно получить последние сообщения в чате.
Направление получения сообщений	str	Параметр, принимающий значения «up» или «down», которые указывают в какую «сторону» от сообщения с указанными идентификатором нужно получить сообщения, имеется в виду получение сообщений, присланных раньше или позже указанного сообщения.
Количество сообщений	int	Параметр, указывающий наибольшее количество сообщений, содержащихся в результате выполнения метода
api_key	str	Ключ API для аутентификации.

Содержание: В процессе выполнения данного метода микросервис мгновенного обмена сообщениями извлечёт из своей базы данных список сообщений чата, в соответствии с заданными параметрам, который он вернёт в формате MessageDTO в результате выполнения метода.

Результат: список сообщений чата в формате MessageDTO.

1.9 Метод «Присоединится к чату»

Описание: Метод предназначен для добавления указанного пользователя в указанный чат.

Параметры: приведены в таблице Ж.9.

Таблица Ж.9 – Параметры метода «Присоединится к чату»

Параметр	Тип	Описание
id пользователя	int	Идентификатор пользователя, которого необходимо добавить в чат.
id чата	int	Идентификатор чата, в который нужно добавить указанного пользователя.
api_key	str	Ключ API для аутентификации.

Содержание: В ходе выполнения метода микросервис мгновенного обмена сообщениями добавит пользователя в указанный чат и вернёт сведения о добавлении пользователя в чат в формате ChatUsersDTO.

Параллельно в фоновом режиме микросервис мгновенного обмена сообщениями вызовет метод «Пользователь добавлен в чат» на всех платформах с типом интерфейса «web».

Результат: сведения о добавлении пользователя в чат в формате ChatUsersDTO.

1.10 Метод «Удалить сообщение»

Описание: Метод предназначен для сообщения в рамках указанного чата.

Параметры: приведены в таблице Ж.10.

Таблица Ж.10 – Параметры метода «Удалить сообщение»

Параметр	Тип	Описание
id чата	int	Идентификатор чата в котором нужно удалить сообщение.
id сообщения	int	Идентификатор сообщения, которое нужно удалить.
api_key	str	Ключ API для аутентификации.

Содержание: В процессе выполнения данного метода микросервис мгновенного обмена сообщениями помечает указанное сообщение как удалённое в своей базе данных. Затем вызывается метод «Сообщение удалено» для всех платформ.

Результат: нет.

1.11 Метод «Отправить чат в архив»

Описание: Метод предназначен для перевода указанного чата в состояние «Архив».

Параметры: приведены в таблице Ж.11.

Таблица Ж.11 – Параметры метода «Отправить чат в архив»

Параметр	Тип	Описание
id чата	int	Идентификатор чата, который необходимо переместить в архив.
api_key	str	Ключ API для аутентификации.

Содержание: В процессе выполнения данного метода микросервис мгновенного обмена сообщениями изменяет состояние указанного чата на «Архив» и удаляет из него всех сотрудников. После чего вызывает метод «Чат обновлён» всех платформ с типом интерфейса «web», информируя их факте изменения чата чата.

Результат: нет.

2 Описание API Telegram бота

2.1 Метод «WebHook»

Описание: Метод предназначен для обработки событий, приходящих из Telegram.

Параметры: Структура Update Telegram bot API.

Содержание: В случае если в структуре Update передана команда «/start» осуществляет регистрацию нового пользователя в микросервисе мгновенного обмена сообщениями. Для чего сначала пытается получить список фотографий профиля пользователя путём вызова метода «getUserProfilePhotos» Telegram bot API и в случае наличия фотографий загружает последнюю фотографию в S3 хранилище системы, путём вызова метода «getFile» Telegram bot API. После чего вызывает метод «Зарегистрировать нового пользователя» микросервиса мгновенного обмена сообщениями. Затем получив результат его выполнения сохраняет в своей базе данных сведения о созданном пользователе.

В случае если структуре Update передаётся сообщение пользователя Telegram, проверяется наличие файлов в нём. Если файлы присутствуют они сначала получатся с помощью метода «getFile» Telegram bot API, а затем загружаются в S3 хранилище системы. После загрузки файлов вызывается метод «Отправить сообщение» микросервиса мгновенного обмена сообщениями и сохраняются в базе данных сведения об отправленном сообщении.

Результат: нет.

2.2 Метод «Новое сообщение»

Описание: Метод предназначен для обработки события отправки сообщения в чат, от микросервиса мгновенного обмена сообщениями.

Параметры: приведены в таблице Ж.12.

Таблица Ж.12 – Параметры метода «Новое сообщение»

Параметр	Тип	Описание
сообщение	MessageDTO	Отправленное сообщение в чат.
api_key	str	Ключ API для аутентификации.

Содержание: В ходе выполнения метода из параметра «сообщение» извлекается идентификатор чата в который было отправлено сообщение. Затем по данному идентификатору из базы данных извлекается идентификаторы чатов Telegram в которые нужно отправить сообщение. В случае наличия в сообщении файлов осуществляется их скачивание из S3 хранилища и последующая отправка одним их методов: «sendPhoto», «sendVideo» или «sendDocument» Telegram bot API. Отправка текста осуществляется методом «sendMessage».

Результат: нет.

2.3 Метод «Удалить сообщение»

Описание: Метод предназначен для обработки события удаления сообщения, от микросервиса мгновенного обмена сообщениями.

Параметры: приведены в таблице Ж.13.

Таблица Ж.13 – Параметры метода «Удалить сообщение»

Параметр	Тип	Описание
сообщение	MessageDTO	Удаляемое сообщение.
api_key	str	Ключ API для аутентификации.

Содержание: В ходе выполнения метода из параметра «сообщение» извлекается его идентификатор по которому из базы данных извлекаются пары «id чата» и

«id сообщения» отправленных в Telegram и для каждой такой пары вызывается метод «deleteMessage» Telegram bot API.

Результат: нет.

3 Описание API бэкенда web-клиента

3.1 Метод «Аутентификация сотрудника»

Описание: Метод предназначен для JWT токена аутентификации пользователя, испытуемого при дальнейших обращениях к бэкенду web-клиента.

Параметры: приведены в таблице Ж.14.

Таблица Ж.14 – Параметры метода «Аутентификация сотрудника»

Параметр	Тип	Описание
Логин	str	Логин аутентифицируемого пользователя.
Пароль	str	Пароль аутентифицируемого пользователя.

Содержание: Метод извлекает из базы данных сведения о пользователе по переданному логину. Затем сравнивает хеш сумму сохранённого пароля с хеш суммой переданного пароля. Если они совпадают, генерируется JWT токен. Этот токен возвращается клиенту и далее используется для авторизации при обращении к защищённым методам API.

Результат: строка с JWT токеном.

3.2 Метод «Получить список пользователей»

Описание: Метод предназначен для получения списка пользователей системы, отображаемых в панели управления.

Параметры: «jwt» – JWT токен аутентификации пользователя.

Содержание: В начале выполнения метода проверяется правильность преданного JWT токена и права пользователя которому он принадлежит. В случае если токен валиден и пользователю достаточно прав. Из базы данных бэкенда web-клиента извлекается список пользователей системы в формате UserWebDTO.

Результат: список пользователей в формате UserWebDTO.

3.3 Метод «Получить список ролей»

Описание: Метод предназначен для получения списка ролей системы, отображаемых в панели управления.

Параметры: «jwt» – JWT токен аутентификации пользователя.

Содержание: В начале выполнения метода проверяется правильность переданного JWT токена и права пользователя. В случае если токен валиден и пользователю достаточно прав. Из базы данных бэкенда web-клиента извлекается список ролей системы в формате RoleWebDTO.

Результат: список пользователей в формате RoleWebDTO.

3.4 Метод «Создать роль»

Описание: Метод предназначен для создания новой роли в системе.

Параметры: приведены в таблице Ж.15.

Таблица Ж.15 – Параметры метода «Создать роль»

Параметр	Тип	Описание
Название роли	str	Название создаваемой роли.
Разрешения	str	JSON-объект с набором разрешений для создаваемой роли.
jwt	str	JWT токен аутентифицированного пользователя.

Содержание: Метод проверяет валидность переданного JWT токена. Если токен действителен и пользователь обладает соответствующими правами, система сохраняет в базе данных новую роль с указанным названием и разрешениями. Далее она возвращается в виде объекта RoleWebDTO.

Результат: созданная роль в формате RoleWebDTO.

3.5 Метод «Редактировать роль»

Описание: Метод предназначен для редактирования уже созданной в системе роли.

Параметры: приведены в таблице Ж.16.

Таблица Ж.16 – Параметры метода «Редактировать роль»

Параметр	Тип	Описание
id роли	int	Идентификатор редактируемой роли.
Название роли	str	Новое название роли.
Разрешения	str	Новый набор разрешений роли в формате JSON.
jwt	str	JWT токен аутентифицированного пользователя.

Содержание: Метод проверяет JWT токен и наличие прав у пользователя на редактирование ролей. После чего выполняет обновление соответствующей роли в базе данных. Обновлённая роль возвращается в формате RoleWebDTO.

Результат: обновлённая роль в формате RoleWebDTO.

3.6 Метод «Удалить роль»

Описание: Метод предназначен для удаления существующей роли из системы.

Параметры: приведены в таблице Ж.17.

Таблица Ж.17 – Параметры метода «Удалить роль»

Параметр	Тип	Описание
id роли	int	Идентификатор удаляемой роли.
jwt	str	JWT токен аутентифицированного пользователя.

Содержание: Метод проверяет переданный JWT токен и наличие у пользователя прав на удаление ролей. После чего из базы данных удаляется роль с указанным идентификатором.

Результат: нет.

3.7 Метод «Создать пользователя»

Описание: Метод предназначен для создания пользователя через web-клиент.

Параметры: приведены в таблице Ж.18.

Таблица Ж.18 – Параметры метода «Создать пользователя»

Параметр	Тип	Описание
Пользователь	Registration-UserWeb-DTO	Объект, содержащий сведения о создаваемом пользователе.
jwt	str	JWT токен аутентифицированного пользователя.

Содержание: В ходе выполнения метода выполняется проверка JWT токена и наличие у пользователя прав на создание новых пользователей. Затем данные о создаваемом пользователе преобразуются из формата RegistrationUserWebDTO в формат RegistrationUserDTO, который используется микросервисом мгновенного обмена сообщениями. После чего будет вызван метод «Создать пользователя» микросервиса мгновенного обмена сообщениями. В результате выполнения которого будет получен идентификатор созданного пользователя. Этот идентификатор будет объединён с информацией, содержащейся в параметре «Пользователь», и сохранён в базе данных бэкенда web-клиента. В завершении выполнения метода будет возвращён объект созданного пользователя в формате UserWebDTO.

Результат: созданный пользователь в формате UserWebDTO.

3.8 Метод «Редактировать пользователя»

Описание: Метод предназначен для редактирования сведений о пользователе через web-клиент.

Параметры: приведены в таблице Ж.19.

Таблица Ж.19 – Параметры метода «Создать пользователя»

Параметр	Тип	Описание
Пользователь	UserUWeb-DTO	Объект с обновлённой информацией о пользователе.
jwt	str	JWT токен аутентифицированного пользователя.

Содержание: В ходе выполнения метода осуществляется валидация JWT токена, после чего бэкенд web-клиента преобразовывает параметр «пользователь» из формата UserUWebDTO в формат UserUDTO. Далее вызывается метод «Редактировать пользователя» микросервиса мгновенного обмена сообщениями. После успешного выполнения, которого бэкенд web-клиента обновляет информацию о пользователе в собственной базе данных. В завершение выполнения метода возвращается объект пользователя в формате UserWebDTO.

Результат: обновлённый объект пользователя в формате UserWebDTO.

3.9 Метод «Удалить пользователя»

Описание: Метод предназначен для удаления пользователя через web-клиент.

Параметры: приведены в таблице Ж.20.

Таблица Ж.20 – Параметры метода «Удалить пользователя»

Параметр	Тип	Описание
id пользователя	int	Идентификатор удаляемого пользователя.
jwt	str	JWT токен аутентифицированного пользователя.

Содержание: В ходе выполнения метода выполняется валидация токена аутентификации, после которой backend web-клиента проверяет наличие пользователя с заданным идентификатором и в случае, если пользователь существует, он помечается в базе данных как удалённый. Важно заметить, что удаление не затрагивает микросервис мгновенного обмена сообщениями, чтобы обеспечить сохранность истории переписки и корректное отображение старых сообщений.

Результат: нет.

3.10 Метод «Получить список чатов»

Описание: Метод предназначен для удаления пользователя через web-клиент.

Параметры: «jwt» – JWT токен аутентификации пользователя.

Содержание: В ходе выполнения метода выполняется валидация токена аутентификации, после которой backend web-клиента извлекает из переданного JWT токена идентификатор пользователя. Далее, используя этот идентификатор, бэкенд вызывает метод «Получить список чатов» микросервиса мгновенного обмена сообщениями в результате выполнения которого будет получен список чатов для указанного пользовали в формате ExtChatDTO.

Результат: список чатов для указанного пользователя в формате ExtChatDTO.

3.11 Метод «Получить список пользователей чата»

Описание: Метод предназначен для получения списка пользователей определённого чата. Используется web-интерфейсом для отображения сообщений чата.

Параметры: приведены в таблице Ж.21.

Таблица Ж.21 – Параметры метода «Получить список пользователей чата»

Параметр	Тип	Описание
id чата	int	Идентификатор чата, для которого необходимо получить список пользователей.
jwt	str	JWT токен аутентифицированного пользователя.

Содержание: В ходе выполнения метода осуществляется валидация JWT токена, после чего бэкенд web-клиента вызывает метод «Получить список пользователей чата» микросервиса мгновенного обмена сообщениями. В результате успешного выполнения, которого бэкенд web-клиента получает список пользователи в формате UserDTO. В завершение выполнения метода возвращается список пользователей чата в формате UserDTO.

Результат: список пользователи указанного чата в формате UserDTO.

3.12 Метод «Получить список сообщений чата»

Описание: Метод предназначен для получения списка сообщений чата. Используется web-клиентом для отображения сообщений чата.

Параметры: приведены в таблице Ж.22.

Таблица Ж.22 – Параметры метода «Получить список сообщений чата»

Параметр	Тип	Описание
id чата	int	Идентификатор чата, для которого необходимо получить сообщения.
id сообщения начиная с которого нужно получить сообщения	int	Идентификатор сообщения относительно которого нужно получать сообщения. Если идентификатор сообщения неизвестен, то он устанавливается в -1 и означает что нужно получить последние сообщения в чате.
Направление получения сообщений	str	Параметр, принимающий значения «up» или «down», которые указывают в какую «сторону» от сообщения с указанными идентификатором нужно получить сообщения, имеется в виду получение сообщений, присланных раньше или позже указанного сообщения.
Количество сообщений	int	Параметр, указывающий наибольшее количество сообщений, содержащихся в результате выполнения метода
jwt	str	JWT токен аутентифицированного пользователя.

Содержание: В ходе выполнения метода осуществляется валидация JWT токена, после чего бэкенд web-клиента вызывает метод «Получить список сообщений» микросервиса мгновенного обмена сообщениями. В результате успешного выполнения метода бэкенд web-клиента получает список сообщений в формате MessageDTO и возвращает его в результате выполнения метода.

Результат: список сообщений чата в формате MessageDTO.

3.13 Метод «Присоединится к чату»

Описание: Метод предназначен для добавления пользователя в указанный чат через web-клиент.

Параметры: приведены в таблице Ж.23.

Таблица Ж.23 – Параметры метода «Присоединится к чату»

Параметр	Тип	Описание
id чата	int	Идентификатор чата к которому необходимо присоединится.
jwt	str	JWT токен аутентифицированного пользователя.

Содержание: В ходе выполнения метода осуществляется валидация JWT токена, после чего бэкенд web-клиента извлекает идентификатор пользователя из JWT и вызывает метод «Присоединиться к чату» микросервиса мгновенного обмена сообщениями. В результате выполняя данного метода микросервис мгновенного обмена сообщениями вернёт сведения о добавлении пользователя в чат в формате ChatUsersDTO, которые бэкенд web-клиента вернёт в качестве результата выполнения метода.

Результат: сведения о добавлении пользователя в чат в формате ChatUsersDTO.

3.14 Метод «Пользователь добавлен в чат»

Описание: Метод предназначен для обработки события добавления пользователя в чат, от микросервиса мгновенного обмена сообщениями.

Параметры: приведены в таблице Ж.24.

Таблица Ж.24 – Параметры метода «Пользователь добавлен в чат»

Параметр	Тип	Описание
Чат	ChatDTO	Чат в который был добавлен пользователь.
Пользователь	UserDTO	Пользователь который был добавлен в чат.
api_key	str	Ключ API для аутентификации.

Содержание: В ходе выполнения метода переданные в параметры «Чат» и «Пользователь» будут преданы всем активным фронтендам web-клиента, путём вызова метода «Пользователь добавлен в чат», с целью оповестить их о добавлении нового пользователя в чат, для корректного отображения сообщений в чате.

Результат: нет.

3.15 Метод «Чат обновлён»

Описание: Метод предназначен для обработки события обновления чата, от микросервиса мгновенного обмена сообщениями, например, в следствии изменения статуса чата.

Параметры: приведены в таблице Ж.25.

Таблица Ж.25 – Параметры метода «Чат обновлён»

Параметр	Тип	Описание
Чат	ChatDTO	Чат сведения о котором были обновлены.
api_key	str	Ключ API для аутентификации.

Содержание: В ходе выполнения метода переданный в параметр «Чат» будет предан всем активным фронтендам web-клиента, путём вызова метода «Чат обновлён», с целью оповестить их об изменении сведений о чате.

Результат: нет.

3.16 Метод «Загрузить файл»

Описание: Метод предназначен для загрузки файла в S3 хранилище системы через web-клиент.

Параметры: приведены в таблице Ж.26.

Таблица Ж.26 – Параметры метода «Загрузить файл»

Параметр	Тип	Описание
Файл	file	Файл, который необходимо загрузить в S3 хранилище. Предаётся в multipart/form-data.
jwt	str	JWT токен аутентифицированного пользователя.

Содержание: В ходе выполнения метода осуществляется валидация JWT токена, после чего файл, переданный через параметр «file», загружается в S3 хранилище. По завершении загрузки которого формируется объект UploadFileDTO, содержащий информацию о загруженном файле. Этот объект возвращается фронтенду для дальнейшего использования при формировании сообщения.

Результат: сведения о загруженном файле в формате UploadFileDTO.

3.17 Метод «Отправить сообщение»

Описание: Метод предназначен для отправки сообщения через web-клиент.

Параметры: приведены в таблице Ж.27.

Таблица Ж.27 – Параметры метода «Отправить сообщение»

Параметр	Тип	Описание
Сообщение	SendMessageDTO	Объект, содержащий сведения об отправляемом сообщении.
jwt	str	JWT токен аутентифицированного пользователя.

Содержание: В ходе выполнения метода осуществляется валидация JWT токена, после чего бэкенд web-клиента вызывает метод «Отправить сообщение» микросервиса мгновенного обмена сообщениями, в результате выполнения которого он вернёт идентификатор отправленного сообщения. Этот идентификатор бэкенд web-клиента вернёт в качестве результата выполнения метода.

Результат: идентификатор отправленного сообщения.

3.18 Метод «Новое сообщение»

Описание: Метод предназначен для обработки события отправки нового сообщения от микросервисом мгновенного обмена сообщениями.

Параметры: приведены в таблице Ж.28.

Таблица Ж.28 – Параметры метода «Новое сообщение»

Параметр	Тип	Описание
Сообщение	MessageDTO	Объект, содержащий отправленное сообщение.
api_key	str	Ключ API для аутентификации.

Содержание: В ходе выполнения метода осуществляется проверка API-ключа микросервиса. После успешной аутентификации бэкенд web-клиента вызывает метод «Новое сообщение» для каждого подключённого фронтенда, передавая объект MessageDTO как параметр. Это позволяет в реальном времени отобразить новое сообщение у всех пользователей.

Результат: нет.

3.19 Метод «Удалить сообщение»

Описание: Метод предназначен для удаления сообщения в чате через web-клиент.

Параметры: приведены в таблице Ж.29.

Таблица Ж.29 – Параметры метода «Удалить сообщение»

Параметр	Тип	Описание
id чата	int	Идентификатор чата в котором нужно удалить сообщение.
id сообщения	int	Идентификатор сообщения, которое нужно удалить.
jwt	str	JWT токен аутентифицированного пользователя.

Содержание: В ходе выполнения метода осуществляется валидация JWT токена пользователя, по которому извлекается идентификатор пользователя, инициировавшего удаление. После чего бэкенд web-клиента вызывает метод «Удалить сообщение» микросервиса мгновенного обмена сообщениями. Который пометит сообщение как удалённое и инициирует распространение события об удалении.

Результат: нет.

3.20 Метод «Сообщение удалено»

Описание: Метод предназначен для обработки события удаления сообщения от микросервисоа мгновенного обмена сообщениями.

Параметры: приведены в таблице Ж.30.

Таблица Ж.30 – Параметры метода «Сообщение удалено»

Параметр	Тип	Описание
Сообщение	MessageDTO	Объект, содержащий удалённое сообщение.
api_key	str	Ключ API для аутентификации.

Содержание: В ходе выполнения метода осуществляется проверка API-ключа микросервиса. После успешной аутентификации бэкенд web-клиента вызывает метод «Сообщение удалено» для каждого подключённого фронтенда, передавая объект MessageDTO как параметр. Это позволяет в реальном времени отобразить удаление сообщения у всех пользователей.

Результат: нет.

3.21 Метод «Отправить чат в архив»

Описание: Метод предназначен для перевода указанного чата в состояние «Архив», с целью скрытия его из основного интерфейса пользователя.

Параметры: приведены в таблице Ж.31.

Таблица Ж.31 – Параметры метода «Отправить чат в архив»

Параметр	Тип	Описание
id чата	int	Идентификатор чата, который необходимо переместить в архив.
jwt	str	JWT токен аутентифицированного пользователя.

Содержание: В ходе выполнения метода происходит валидация JWT токена. Далее бэкенд web-клиента вызывает метод «Отправить чат в архив» микросервиса мгновенного обмена сообщениями, который выставми статус указанного чата в состояние «Архив».

Результат: нет.

ПРИЛОЖЕНИЕ И

Сущности базы данных и связи между ними

Таблица И.1 – Атрибуты сущности «Платформа»

Наименование атрибута	Описание атрибута	Тип данных	Диапазон значений	Пример
Код платформы	Идентификатор платформы. Ключевой атрибут	Числовой	Больше 0	1
Тип платформы	Тип платформы	Перечисление	–	web или bot
Название платформы	Название платформы	Текстовый	–	Telegram
URL	Ссылка на конвертирующий микросервис	Текстовый	–	https://exaample.com

Таблица И.2 – Атрибуты сущности «Пользователь»

Наименование атрибута	Описание атрибута	Тип данных	Диапазон значений	Пример
Код пользователя	Идентификатор пользователя. Ключевой атрибут	Числовой	Больше 0	1
Имя	Имя пользователя	Текстовый	–	Коля
URL фото профиля	URL по которому можно получить фото профиля	Текстовый	–	https://example.com/2d4.png

Таблица И.3 – Атрибуты сущности «Пользователь Telegram»

Наименование атрибута	Описание атрибута	Тип данных	Диапазон значений	Пример
Код пользователя	Идентификатор пользователя. Ключевой атрибут	Числовой	Больше 0	1
Код пользователя Telegram	Идентификатор пользователя в Telegram	Числовой	–	1

Таблица И.4 – Атрибуты сущности «Пользователь VK»

Наименование атрибута	Описание атрибута	Тип данных	Диапазон значений	Пример
Код пользователя	Идентификатор пользователя. Ключевой атрибут	Числовой	Больше 0	1
Код пользователя VK	Идентификатор пользователя в VK	Числовой	–	1

Таблица И.5 – Атрибуты сущности «Пользователь Web»

Наименование атрибута	Описание атрибута	Тип данных	Диапазон значений	Пример
Код пользователя	Идентификатор пользователя. Ключевой атрибут	Числовой	Больше 0	1
Адрес электронной почты	Адрес электронной почты пользователя	Текстовый	–	example@expmple.ru
Хэш пароля	Хэш-сумма пароля пользователя	Текстовый	–	$2b$12$8Q4…
Активный пользователь	Флаг указывающий на активность пользователя (при false свидетельствует о том что пользователь удалён)	Логический	–	True
Суперпользователь	Флаг указывающий, что данный пользователь является суперпользователем	Логический	–	True
Настройки	Список настроек	JSON	–	{“example”: true}

Таблица И.6 – Атрибуты сущности «Роль»

Наименование атрибута	Описание атрибута	Тип данных	Диапазон значений	Пример
Код роли	Идентификатор роли. Ключевой атрибут	Числовой	Больше 0	1
Название роли	Название роли	Текстовый	–	Роль 1
Разрешения	Разрешения роли	JSON	–	{“user.list”: true}

Таблица И.7 – Атрибуты сущности «Чат»

Наименование атрибута	Описание атрибута	Тип данных	Диапазон значений	Пример
Код чата	Идентификатор чата. Ключевой атрибут	Числовой	Больше 0	1
Название	Название чата	Текстовый	–	Чат
URL иконки чата	URL-адрес по которому можно получить иконку чата	Текстовый	–	https://example.com/1.png
Входящий	Флаг указывающий на состояние чата «Входящий»	Логический	–	True
Архивный	Флаг указывающий на состояние чата «Архив»	Логический	–	True
Время отправки последнего сообщения в чате	Время в которое было отправлено последнее сообщение в чате	Дата время	–	2025-04-06T00:49:27.919

Таблица И.8 – Атрибуты сущности «Участники чата»

Наименование атрибута	Описание атрибута	Тип данных	Диапазон значений	Пример
Код пользователя	Идентификатор пользователя. Ключевой атрибут	Числовой	Больше 0	1
Код чата	Идентификатор чата. Ключевой атрибут	Числовой	Больше 0	1

Продолжение таблицы И.8

Наименование атрибута	Описание атрибута	Тип данных	Диапазон значений	Пример
Пользователь в чате	Флаг указывающий на присутствие пользователя в чате	Логический	–	True

Таблица И.9 – Атрибуты сущности «Сообщение»

Наименование атрибута	Описание атрибута	Тип данных	Диапазон значений	Пример
Код сообщения	Идентификатор сообщения. Ключевой атрибут	Числовой	Больше 0	1
Время отправки	Время отправки сообщения	Дата время	–	2024-06-25T23:58:18.149Z
Текст	Текст сообщения	Текстовый	–	Привет.
Вложения	Сведения о прикреплённых файлах	JSON	–	{“files”:…}
Удалено	Флаг указывающий, что сообщение удалено	Логический	–	True
Время удаления	Время удаления	Дата время	–	2024-06-25T23:58:18.149Z

Таблица И.10 – Атрибуты сущности «Сообщение Telegram»

Наименование атрибута	Описание атрибута	Тип данных	Диапазон значений	Пример
Код сообщения Telergram	Идентификатор сообщения в Telegram. Ключевой атрибут	Числовой	Больше 0	1
Код чата Telergram	Код чата Telergram. Ключевой атрибут	Числовой	Больше 0	1

Таблица И.11 – Атрибуты сущности «Сообщение VK»

Наименование атрибута	Описание атрибута	Тип данных	Диапазон значений	Пример
Код сообщения VK	Идентификатор сообщения в VK. Ключевой атрибут	Числовой	Больше 0	1
Код чата VK	Код чата VK. Ключевой атрибут	Числовой	Больше 0	1

Таблица И.12 – Таблица связей сущностей

Название 1 сущности	Название 2 сущности	Название связи	Тип связи	Описание
Платформа	Пользователь	Относится/ Относится	1:N	К одной платформе могут относится несколько пользователей. А один пользователь относится к одной платформе.
Платформа	Чат	Относится/ Относится	1:N	К одной платформе могут относится несколько чатов. А один чат относится к одной платформе.
Пользователь	Пользователь Telegram	Являться/ Является	1:1	Пользователь может являться пользователем Telegram. Пользователь Telegram является пользователем.
Пользователь	Пользователь VK	Являться/ Является	1:1	Пользователь может являться пользователем VK. Пользователь VK является пользователем.
Пользователь Telegram	Чат	Состоять/ Состоять	N:1	Один пользователь Telegram может состоять в одном чате. В одном чате могут состоять несколько пользователей Telegram.
Пользователь VK	Чат	Состоять/ Состоять	N:1	Один пользователь VK может состоять в одном чате. В одном чате могут состоять несколько пользователей VK.
Пользователь	Пользователь Web	Являться/ Является	1:1	Пользователь может являться пользователем Web. Пользователь Web является пользователем.
Пользователь Web	Роль	Назначена/ Назначена	N:1	Одному пользователю Web может быть назначена только одна роль. Одна роль может быть назначена нескольким пользователям Web.

Продолжение таблицы И.12

Название 1 сущности	Название 2 сущности	Название связи	Тип связи	Описание
Пользователь	Участники чата	Быть/Есть	1:N	Один пользователь может быть участником нескольких чатов. Участник чата есть пользователь.
Чат	Участники чата	Имеет/ Относится	1:N	Один чат имеет несколько участников. Один участник чата относится к одному чату.
Участники чата	Сообщение	Прочитать/ Прочитано	N:1	Один участник чата может прочитать последним одно сообщение в чате. Одно сообщение может быть прочитано последним в чате несколькими пользователями.
Чат	Сообщение	Отправлено/ Отправлено	1:N	В один чат может быть отправлено несколько сообщений. Одно сообщение может быть отправлено только в один чат.
Сообщение	Сообщение VK	Представлено/ Относится	1:N	Одно сообщение может быть представлено как несколько сообщений VK. Одно сообщение VK может относится только к одному сообщению.
Сообщение	Сообщение Telegram	Представлено/ Относится	1:N	Одно сообщение может быть представлено как несколько сообщений Telegram. Одно сообщение Telegram может относится только к одному сообщению.
Сообщение	Пользователь	Отправлено/Отправить	N:1	Одно сообщение обязательно должно быть отправлено одними пользователем. Один пользователь может отправить несколько сообщений.

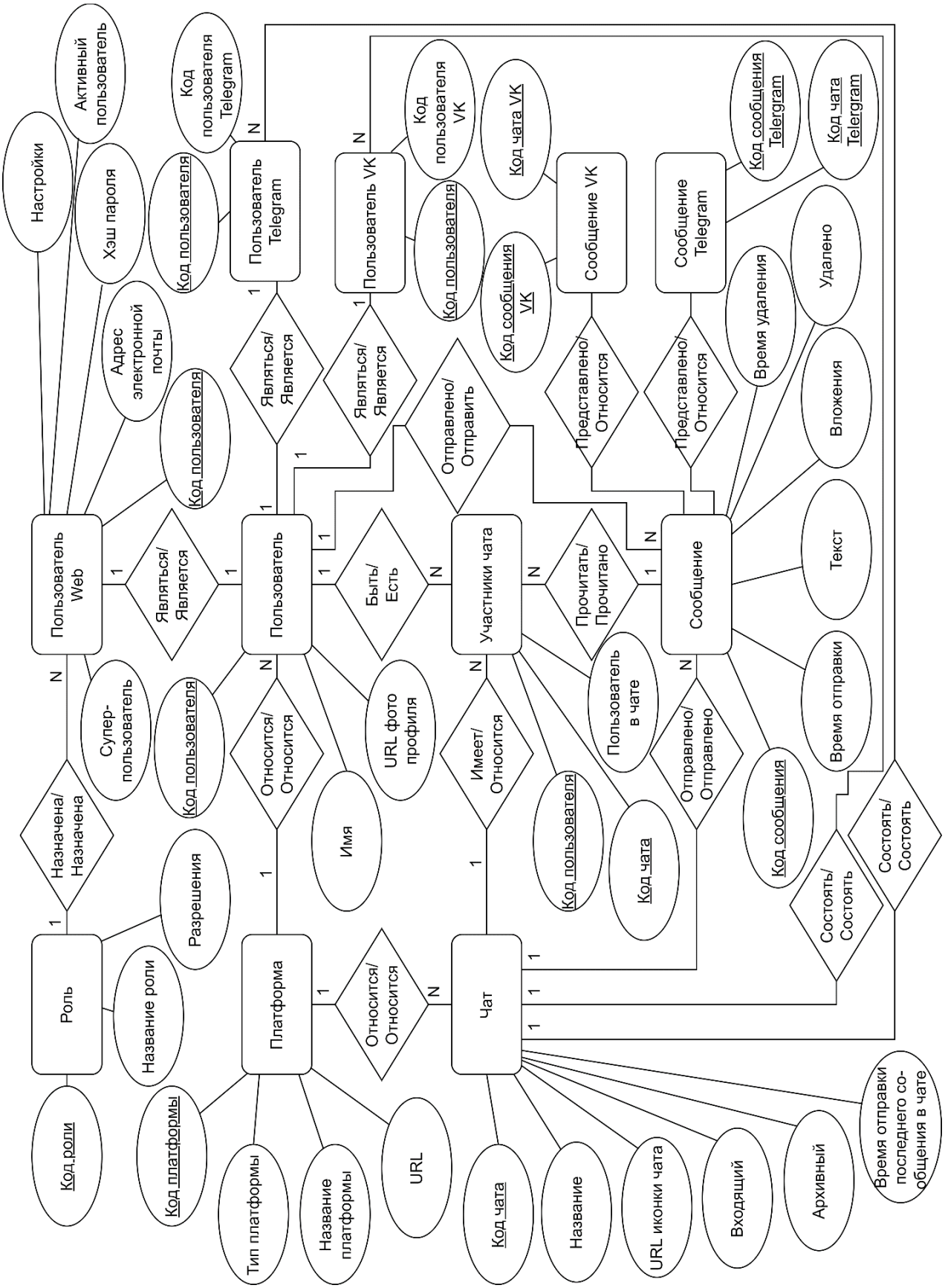

Рисунок И.1 – ER-диаграмма базы данных в нотации Чена

ПРИЛОЖЕНИЕ К

Логическая и физическая модели базы данных

Таблица К.1 – Преобразование связей между сущностями

Связь	Кратность	Класс принадлежности сущности 1	Класс принадлежности сущности 2	Результат преобразования
Платформа – Пользователь	1:N	Не обязательный	Обязательный	Первичный ключ сущности «Платформа» должен быть скопирован в сущность «Пользователь». Получаются отношения: «Платформа» и «Пользователь».
Платформа – Чат	1:N	Не обязательный	Обязательный	Первичный ключ сущности «Платформа» должен быть скопирован в сущность «Чат». Получаются отношения: «Платформа» и «Чат».
Пользователь – Пользователь Telegram	1:1	Не обязательный	Обязательный	Первичный ключ сущности «Пользователь» должен быть скопирован в сущность «Пользователь Telegram». Получаются отношения: «Пользователь» и «Пользователь Telegram».
Пользователь – Пользователь VK	1:1	Не обязательный	Обязательный	Первичный ключ сущности «Пользователь» должен быть скопирован в сущность «Пользователь VK». Получаются отношения: «Пользователь» и «Пользователь VK».
Пользователь Telegram – Чат	N:1	Обязательный	Не обязательный	Первичный ключ сущности «Чат» должен быть скопирован в сущность «Пользователь Telegram». Получаются отношения: «Пользователь Telegram» и «Чат».
Пользователь VK – Чат	N:1	Обязательный	Не обязательный	Первичный ключ сущности «Чат» должен быть скопирован в сущность «ПользовательVK». Получаются отношения: «ПользовательVK» и «Чат».
Пользователь – Пользователь Web	1:1	Не обязательный	Обязательный	Первичный ключ сущности «Пользователь» должен быть скопирован в сущность «Пользователь Web». Получаются отношения: «Пользователь» и «Пользователь Web».

Продолжение таблицы К.1

Связь	Кратность	Класс принадлежности сущности 1	Класс принадлежности сущности 2	Результат преобразования
Пользователь Web – Роль	N:1	Не обязательный	Не обязательный	Получаются отношения: «Пользователь Web», «Роль» и «Роли сотрудников». Отношение «Роли сотрудников» получается в результате преобразования связи и содержит первичные ключи первых двух отношений.
Пользователь – Участники чата	1:N	Не обязательный	Обязательный	Первичный ключ сущности «Пользователь» должен быть скопирован в сущность «Участники чата». Получаются отношения: «Пользователь» и «Участники чата».
Чат – Участники чата	1:N	Не обязательный	Обязательный	Первичный ключ сущности «Чат» должен быть скопирован в сущность «Участники чата». Получаются отношения: «Чат» и «Участники чата».
Участники чата – Сообщение	N:1	Обязательный	Не обязательный	Первичный ключ сущности «Сообщение» должен быть скопирован в сущность «Участники чата». Получаются отношения: «Участники чата» и «Сообщение».
Чат – Сообщение	1:N	Не обязательный	Обязательный	Первичный ключ сущности «Чат» должен быть скопирован в сущность «Сообщение». Получаются отношения: «Чат» и «Сообщение».
Сообщение – Сообщение VK	1:N	Не обязательный	Обязательный	Первичный ключ сущности «Сообщение» должен быть скопирован в сущность «Сообщение VK». Получаются отношения: «Сообщение» и «Сообщение VK».
Сообщение – Сообщение Telegram	1:N	Не обязательный	Обязательный	Первичный ключ сущности «Сообщение» должен быть скопирован в сущность «Сообщение Telegram». Получаются отношения: «Сообщение» и «Сообщение Telegram».
Сообщение – Пользователь	N:1	Обязательный	Не обязательный	Первичный ключ сущности «Пользователь» должен быть скопирован в сущность «Сообщение». Получаются отношения: «Сообщение» и «Пользователь».

Отношение «Платформа»

Код платформы	Тип платформы	Название платформы	URL

Отношение «Пользователь»

Код пользователя	Имя	URL фото профиля	Код платформы

Отношение «Пользователь Telegram»

Код пользователя	Код пользователя Telegram	Код чата

Отношение «Пользователь VK»

Код пользователя	Код пользователя VK	Код чата

Отношение «Пользователь Web»

Код пользователя	Адрес электронной почты	Хэш пароля	Активный пользователь	Супер-пользователь	Настройки	Код роли

Отношение «Роли сотрудников»

Код пользователя	Код роли

Отношение «Роль»

Код роли	Название роли	Разрешения

Отношение «Чат»

Код чата	Название	URL иконки чата	Входящий	Архивный	Время отправки последнего со-общения в чате	Код платформы

Отношение «Участники чата»

Код пользователя	Код чата	Пользователь в чате	Код последнего прочитанного сообщения

Отношение «Сообщение»

Код сообщения	Время отправки	Текст	Вложения	Удалено	Время удаления	Код чата	Код пользователя

Отношение «Сообщение VK»

Код сообщения VK	Код чата VK	Код сообщения

Отношение «Сообщение Telegram»

Код сообщения Telegram	Код чата Telegram	Код сообщения

Рисунок К.1 – Набор отношений, полученных после преобразования связей сущностей

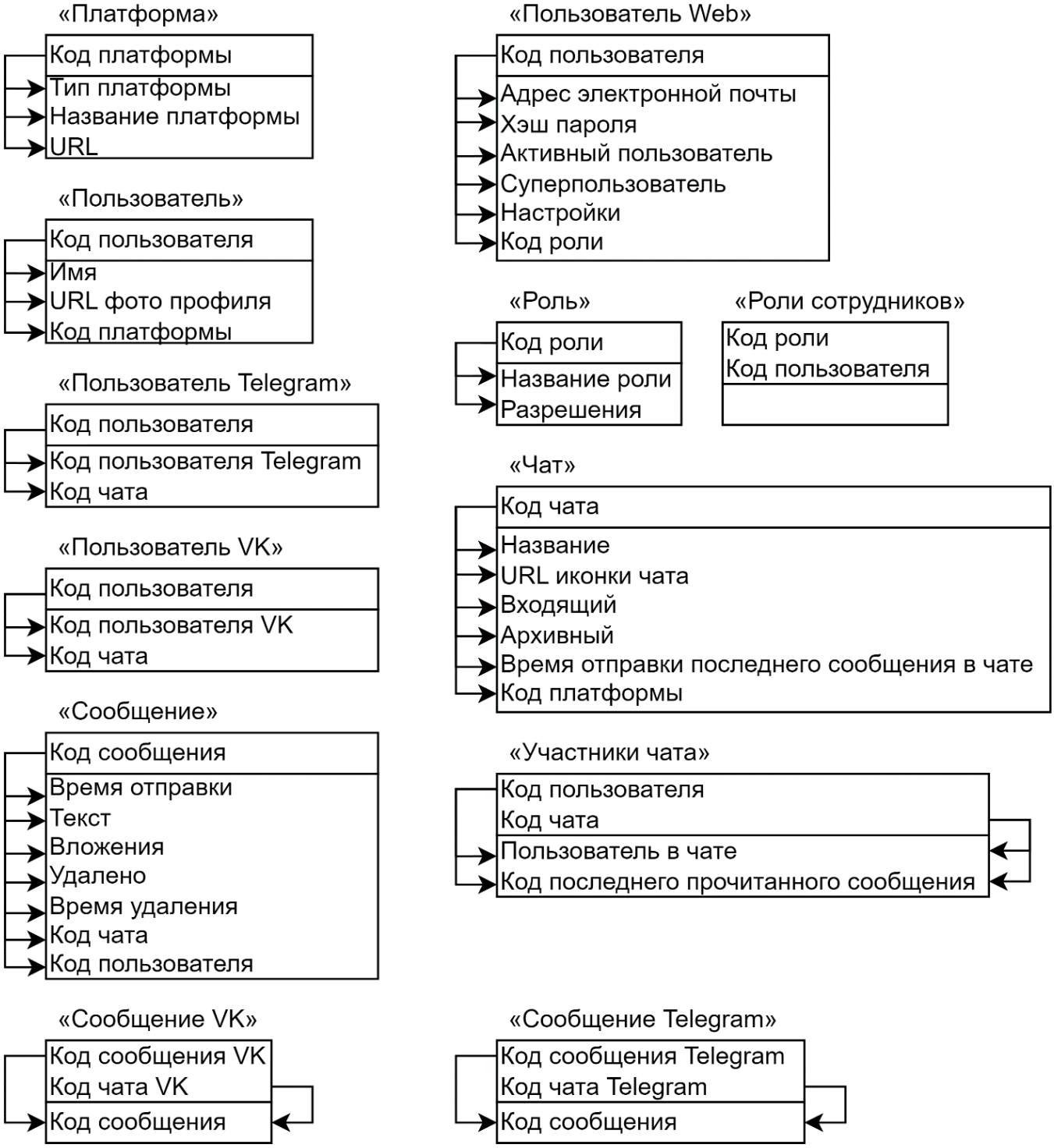

Рисунок К.2 – Зависимости атрибутов отношений базы данных

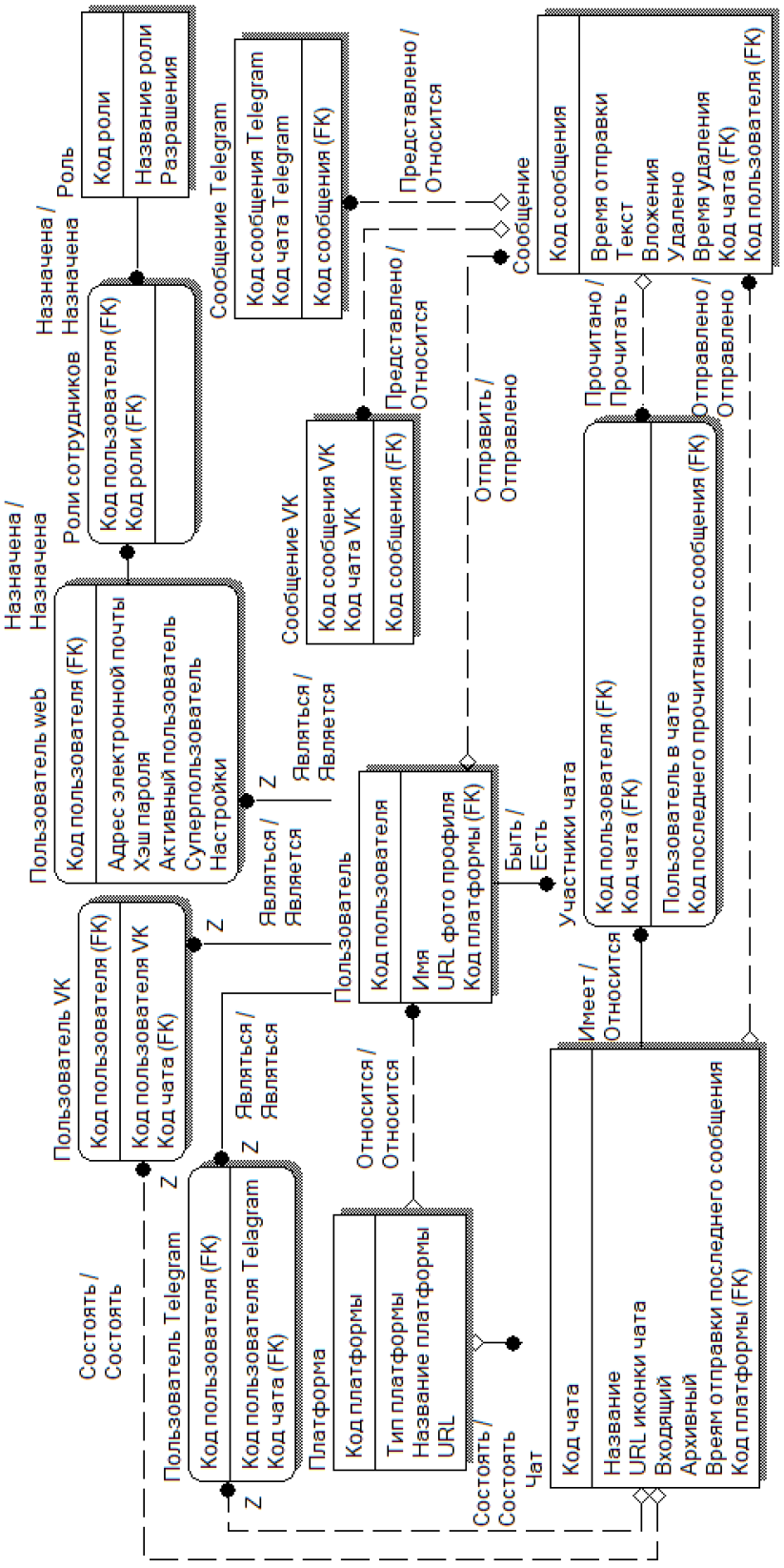

Рисунок К.3 – Диаграмма базы данных на этапе логического проектирования в нотации IDEF1X

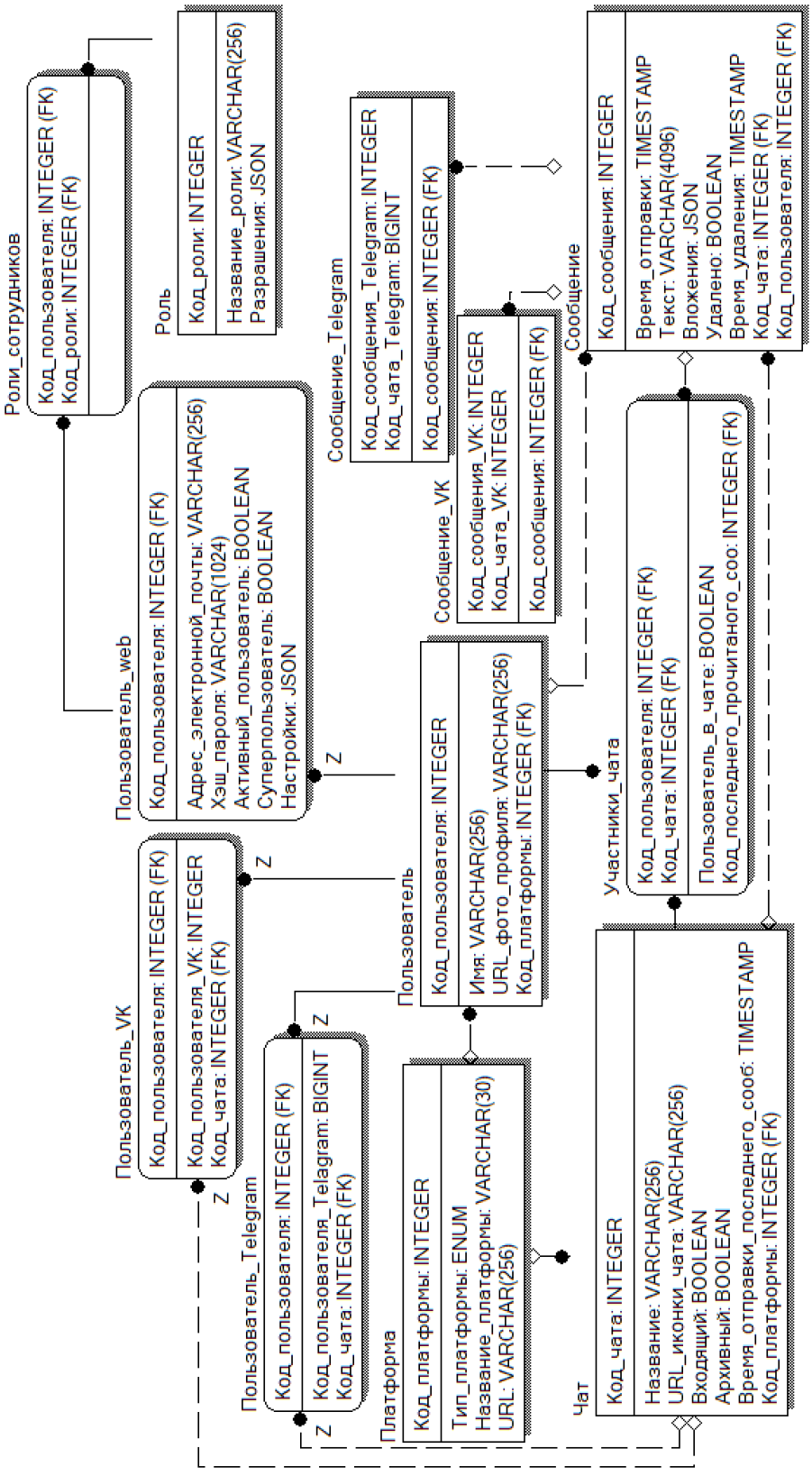

Рисунок К.4 – Диаграмма базы данных на этапе физического проектирования в нотации IDEF1X