

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение высшего образования

«ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

Факультет
вычислительной техники

Кафедра
«Вычислительная техника»

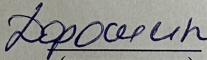
Направление подготовки: 09.03.01 «Информатика и вычислительная техника»
Профиль подготовки: «Программное обеспечение средств вычислительной техники и автоматизированных систем»

БАКАЛАВРСКАЯ РАБОТА
на тему
КОРПОРАТИВНЫЙ ДЕСКТОП-МЕССЕНДЖЕР

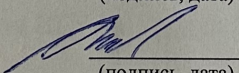
Студент


(подпись, дата) 30.05.2024 Педай Никита Денисович

Руководитель

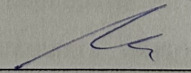

(подпись, дата) 30.05.2024 Дорошенко Иван Николаевич

Нормоконтролёр


(подпись, дата) 30.05.2024 Бычков Андрей Станиславович

Работа допущена к защите (протокол зас. кафедры от 27.05.24 г. № 11)

Заведующий кафедрой


(подпись) М.А. Митрохин
(фамилия, инициалы)

Работа защищена с отметкой отлично (протокол заседания ГЭК от

10.06.2024 № 17)

Секретарь ГЭК


(подпись) Е.И. Гурин
(фамилия, инициалы)

Пенза 2024

Федеральное государственное бюджетное образовательное учреждение высшего образования

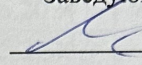
«ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

Факультет
вычислительной техники

Кафедра
«Вычислительная техника»

«Утверждаю»

Заведующий кафедрой

 М.А. Митрохин

«19» 04 2024г.

ЗАДАНИЕ НА БАКАЛАВРСКУЮ РАБОТУ

1. Студент Педай Никита Денисович гр. 20ВВП2 факультета вычислительной техники.

2. Тема работы «Корпоративный десктоп-мессенджер».

Тема утверждена приказом ПГУ № 405/0 от "19" 04 2024 г.

3. Руководитель бакалаврской работы доцент кафедры ВТ ПГУ Дорошенко И.Н.

4. Задание на работу (назначение разработки, исходные данные и т.п.)

Разработать клиентскую часть корпоративного мессенджера под управлением ОС Windows. Приложение должно предоставлять пользователям возможность обмениваться текстовыми сообщениями, файлами, добавлять новые контакты, удалять уже имеющиеся контакты, изменять настройки пользователя. Язык программирования C++, среда разработки Qt Creator.

5. Перечень подлежащих разработке вопросов.

Требуется произвести анализ предметных областей, выбрать среду разработки.

6. Календарный график выполнения работы

№ п.	Наименование этапов работы	Объем работы	Срок выполнения	Подпись руководителя
1	Анализ предметной области	50	26.04.2024-30.04.2024	Дорошенко
2	Выбор архитектуры. Приложения и технологий, применяемых при проектировании и разработке	10	01.05.2024-02.05.2024	Дорошенко
3	Разработка логики приложения	50	03.05.2024-07.05.2024	Дорошенко
4	Описание алгоритмов	40	08.05.2024-11.05.2024	Дорошенко
5	Разработка корпоративного десктоп-мессенджера	150	12.05.2024-29.05.2024	Дорошенко
6	Оформление выпускной работы	24	30.05.2024-31.05.2024	Дорошенко

Дата выдачи задания "19" 04 2024 г.

Руководитель бакалаврской работы Дорошенко И.Н. Дорошенко
(подпись, дата)

Задание к исполнению приняла студент НН Н.Д. Педай
(подпись, дата)

Работу к защите допустить
Декан факультета Л.Р. Фионова 04.06.2024
(подпись, дата)

Реферат

Бакалаврская работа содержит 132 л., 21 рис., 4 источн., 3 прил.
WINDOWS, БАЗА ДАННЫХ, QT CREATOR, SQLITE.

Цель работы – разработать клиентскую часть корпоративного мессенджера, позволяющую пользователям обмениваться различными видами сообщений.

В ходе разработки необходимо реализовать следующие возможности для пользователя: обмениваться текстовыми сообщениями, файлами, добавлять и удалять контакты, менять настройки пользователя.

В процессе разработки проводился анализ актуальности проекта, аналогов, основных платформ разработки ПО.

В результате проведенной работы создано приложение, соответствующее поставленной задаче.

					ПГУ 1.090301.35.001 ПЗ		
Изм	Лист	№ докум.	Подпись	Дата			
Разраб.	Педай Н.Д.			30.05.20	Литера	Лист	Листов
Пров.	Дорошенко И. Н.			30.05.20		2	
Н. Контр.	Бычков А.С.			30.05.20	ФВТ, каф. ВТ гр. 20ВВП2		
Утв.	Митрохин М.А.			30.05.20			
					Корпоративный десктоп-мессенджер		
					Пояснительная записка		

Содержание

Реферат	Ошибка! Закладка не определена.
Перечь сокращений и обозначений	3
Введение	4
1 Корпоративный мессенджер под управлением ОС Windows	6
1.1 Анализ предметной области	6
1.1.1 Обзор аналогов	7
1.1.2 Мессенджер Telegram	7
1.1.3 Мессенджер WhatsApp	8
1.1.4 Корпоративный мессенджер Slack	8
1.1.5 Корпоративный мессенджер Пачка	9
1.1.6 Постановка задачи	9
1.2 Выбор среды разработки	10
1.2.1 Qt Creator	10
1.2.2 Visual Studio Code	11
1.2.3 Eclipse	11
1.2.4 SQLite	12
1.3 Сравнение ОС	12
1.3.1 ОС Windows	12
1.3.2 ОС Linux	13
1.3.3 ОС MacOS	14
1.4 Архитектура приложения	15
1.5 Реализация приложения	17
1.5.1 Реализация аутентификации пользователя	17
1.5.2 Реализация списка контактов	24
1.5.3 Реализация добавления контактов	27
1.5.4 Реализация удаление контактов	30
1.5.5 Реализация поиска контакта	33
1.5.6 Реализация редактирования контактов	34

1.5.7 Реализация настроек пользователя	36
1.5.8 Реализация чата и отправки сообщений	39
1.5.9 Реализация отправки файлов	43
2 Безопасность жизнедеятельности	50
2.1 Потенциально опасные факторы при работе с ПЭВМ.....	50
2.2 Классификация помещения.....	51
2.3 Санитарно-гигиеническая характеристика	51
2.4 Расчет освещенности рабочего места	52
Заключение	54
Список использованных источников	55
Приложение А	56
Приложение А.1 – Диаграмма классов	Ошибка! Закладка не определена.
Приложение А.2 – Диаграмма прецедентов.....	57
Приложение А.3 – Диаграмма развертывания	58
Приложение А.4 – Диаграмма последовательности.....	59
Приложение Б	60
Приложение В.....	123

Перечь сокращений и обозначений

В данной работе были применены следующие сокращения и обозначения

БД – база данных

ОС – операционная система

ПК – персональный компьютер

API (Application Programming Interface) – программный интерфейс приложения

IDE (Integrated Development Enviroment) – Интегрированная среда разработки

MVC (Model-View-Controller) – «Модель-Представление-Контроллер» или «Модель-Вид-Контроллер»

UML (Unified Modeling Language) – унифицированный язык моделирования.

XML (Extensible Markup Language) – расширяемый язык разметки

					ПГУ 1.090301.35.001 ПЗ	Лист
						3
Изм.	Лист	№ докум.	Подпись	Дата		

Введение

В современном информационном обществе эффективная коммуникация является ключевым фактором успешной деятельности предприятий и организаций.

Существует множество инструментов, которые предназначены для обеспечения внутренней коммуникации между сотрудниками, такие как электронная почта, голосовая и видеосвязь, а также социальные сети. Но наиболее популярным и удобным способом обмена сообщениями являются мессенджеры. Наиболее популярными являются Telegram, WhatsApp, Discord и Slack. В связи с этим, мессенджеры становятся неотъемлемой частью рабочего процесса, обеспечивая быстрый и удобный обмен информацией между сотрудниками. Однако большинство компаний хотят максимально обезопасить свои данные, в связи с этим переходят на корпоративные мессенджеры.

Корпоративные мессенджеры предоставляют расширенные возможности для организации командной работы, обмена документами и интеграции с другими системами.

На российском рынке также представлены различные корпоративные мессенджеры, разработанные с учетом особенностей российских компаний и требований к безопасности данных. Эти приложения должны обеспечивать высокий уровень защиты данных, а также интеграцию других отечественных информационных систем.

Актуальность разработки корпоративного мессенджера обусловлена растущей потребностью компаний в эффективных инструментах для внутренней коммуникации.

В рамках работы будет проведен анализ существующих аналогов на рынке, выявлены их достоинства и недостатки, а также сформулированы требования к создаваемому мессенджеру. Затем будет разработана архитектура приложения, реализованы функциональные возможности и проведено тестирование разработанного мессенджера.

					ПГУ 1.090301.35.001 ПЗ	Лист
						4
Изм.	Лист	№ докум.	Подпись	Дата		

Результатом данной работы будет создание функционального и удобного в использовании корпоративного мессенджера под управлением ОС Windows, который поможет обеспечить конфиденциальность данных, улучшить коммуникацию между сотрудниками компании и повысить эффективность их работы.

					ПГУ 1.090301.35.001 ПЗ	Лист
						5
Изм.	Лист	№ докум.	Подпись	Дата		

1 Корпоративный мессенджер под управлением ОС Windows

1.1 Анализ предметной области

Начало разработки продукта всегда связано с анализом предметной области.

Необходимо определить кто будет пользоваться продуктом, выбрать технологии, которые позволят в будущем поддерживать и развивать продукт.

В данном разделе рассматриваются общие теоретические вопросы, связанные с проектированием и реализацией клиентской части корпоративного мессенджера. Изучение данных вопросов поможет определить основные проблемы и задачи для разработки продукта.

Предметная область разработки корпоративного мессенджера относится к области информационных технологий и включает в себя разработку программного обеспечения для внутренней коммуникации между сотрудниками компаний различных масштабов и отраслей [1].

На сегодняшний день мессенджеры являются эффективным инструментом для организации коммуникации и совместной работы сотрудников, работающих в разных офисах, городах и даже странах. Они позволяют сократить время ответа на вопросы, оперативно решать обнаруженные проблемы.

Ниже предоставлены следующие аспекты, которыми должен обладать современный мессенджер:

Функциональность и эффективная коммуникация: одним из главных аспектов является обеспечение разнообразных функций, позволяющих сотрудникам эффективно обмениваться информацией: обмениваться текстовыми сообщениями, голосовыми сообщениями, видеозвонками, передавать файлы различных форматов, а также создавать чаты для обсуждения проектов и задач.

Безопасность и конфиденциальность: важно обеспечить высокий уровень защиты данных и конфиденциальности сообщений с помощью надежных методов аутентификации, чтобы исключить несанкционированный доступ и предотвратить утечки данных.

Удобство использования: корпоративный мессенджер должен быть доступен на различных платформах, включая компьютеры. Интуитивно понятный интерфейс и простота использования играют важную роль, позволяя сотрудникам быстро освоить приложение и использовать его в повседневной работе.

Пользовательские настройки: важно предоставить пользователям возможность настройки приложения в соответствии с их предпочтениями и рабочими потребностями. Персонализация включает в себя выбор темы, оформления, настройку уровня конфиденциальности и другие параметры, которые помогут пользователю настроить мессенджер под себя.

1.1.1 Обзор аналогов

На сегодняшний день существует обширное количество мессенджеров [1]. Для анализа подобных приложений рассмотрим несколько популярных мессенджеров, таких как Telegram, WhatsApp, Slack и Пачка.

Каждое из них представляет собой разнообразный набор функций для обмена сообщениями, файлов, а также интеграцию с другими системами и сервисами. Проведение анализа аналогов поможет выявить особенности и возможности этих приложений, их преимущества и недостатки.

1.1.2 Мессенджер Telegram

Одним из наиболее используемых в последнее время мессенджеров является Telegram. Это multifunctional приложение, которое позволяет пользователям взаимодействовать друг с другом, передавать различные виды сообщений, а также файлы. Оно входит в число самых популярных мессенджеров в мире. Основными видимыми особенностями являются “секретные чаты”, которые не отправляются на сервер для хранения и работают по системе клиент - клиент, а также возможность установить таймер для автоматического удаления сообщений через заданный промежуток времени. Telegram имеет общедоступное API, с помощью которого разработчики могут интегрировать мессенджер в сторонние сервисы или веб сайты.

Таблица 1 – Достоинства и недостатки Telegram

Достоинства	Недостатки
Простота использования	Относительно нестабильная работа в условиях сетевых ограничений
Наличие мобильных приложений для различных платформ	Ограниченный выбор настроек прав доступа.
Неограниченное количество участников каналов	Ограничения в бесплатной версии

1.1.3 Мессенджер WhatsApp

Вторым известным мессенджером является WhatsApp, который предлагает широкий набор функций для коммуникации и совместной работы сотрудников, таких как аудиозвонки и видеосвязь, обмен файлами, кроссплатформенность. WhatsApp занимает лидирующие позиции среди мессенджеров в мире, с более чем 2 миллиардами установок.

Таблица 2 – Достоинства и недостатки WhatsApp

Достоинства	Недостатки
Полностью бесплатный функционал и простота использования	Ограничение по количеству участников в групповых чатах
Отсутствие рекламы	Недостаточные возможности для использования компаниями
Автоматическое сохранение медиа в галерею пользователя	

Рассмотрим корпоративные мессенджеры. Наиболее популярным из них является Slack.

1.1.4 Корпоративный мессенджер Slack

Это мессенджер, который широко используется в профессиональной среде для командной работы и общения. Он предоставляет инструменты для организации профессиональной коммуникации внутри организации и позволяет

создавать различные каналы для обсуждения проектов, задач и идей. Одной из ключевых особенностей Slack является его способность интегрироваться с другими сервисами и инструментами, такими как Google Drive, Trello, GitHub, Jira и многими другими. Это обеспечивает эффективный обмен файлами и информацией с другими приложениями внутри Slack.

Таблица 3 – Достоинства и недостатки Slack

Достоинства	Недостатки
Пользователь может настроить мессенджер под себя (оформление, уведомления, и другие параметры)	Платные тарифы с различными ограничениями
Кроссплатформенная совместимость	Ограниченные возможности по сравнению с другими мессенджерами

1.1.5 Корпоративный мессенджер Пачка

Из отечественных корпоративных мессенджеров можно хочу выделить – Пачку.

Он обладает очень простым и интуитивно понятным интерфейсом, что делает его доступным даже для пользователей с минимальным опытом работы с мессенджерами.

Таблица 4 – Достоинства и недостатки Пачки

Достоинства	Недостатки
Высокий уровень безопасности и конфиденциальности сообщений	Ограниченный функционал, в сравнении с популярными аналогами
Возможность общаться с другими пользователям анонимно, не делясь именем или номером телефона	Малая распространенность мессенджера

1.1.6 Постановка задачи

Задачей данного дипломного проекта является создание клиентской части корпоративного мессенджера под управлением ОС Windows. Приложение должно предоставлять пользователям возможность обмениваться текстовыми

сообщениями, файлами, добавлять новые контакты, удалять уже имеющиеся контакты, изменять настройки пользователя.

Используя данное приложение, пользователь сможет:

- регистрироваться;
- авторизоваться;
- просматривать чаты;
- обмениваться текстовыми сообщениями, файлами;
- добавлять или удалять контакты;
- редактировать контакты;
- редактировать настройки: atc, dns адреса сервера, качество видеофайла,

предпросмотр видео, а также обновление ПО, сброс приложение.

1.2 Выбор среды разработки

На данный момент в мире информационных технологий существует огромное количество различных средств разработки приложений на Windows.

На сегодняшний день самыми востребованными средами являются:

- Qt Creator;
- Eclipse;
- VS Code;

1.2.1 Qt Creator

Эта среда позволяет создавать приложения и программы на самых распространенных языках программирования. Рассмотрим некоторые из достоинств и недостатков Qt Creator.

Из достоинств можно отметить кроссплатформенность. Qt Creator обеспечивает возможность разработки приложений для различных операционных систем, включая Windows, macOS, Linux, iOS и Android, что позволяет разработчикам создавать приложения, совместимые с широким спектром платформ. Qt Creator тесно интегрирован с Qt фреймворком, предоставляя разработчикам инструменты для создания графических интерфейсов, обработки событий и многопоточного программирования. В Qt Creator предоставлен

широкий спектр инструментов, включая редактор кода с подсветкой синтаксиса, интегрированный отладчик, дизайнер интерфейсов, комплекты сборки и многое другое, что упрощает и ускоряет процесс разработки.

К недостаткам можно отнести проблемы с интеграцией сторонних библиотек и компонентов, в некоторых случаях может возникнуть конфликт версий. Qt Creator потребляет значительные ресурсы системы, особенно при работе с большими проектами, что периодически приводит к замедлению работы приложения на слабых компьютерах. Также следует отметить большой вес приложения и используемых библиотек.

1.2.2 Visual Studio Code

Visual Studio Code – IDE для редактирования кода, предоставляющая широкий набор инструментов для разработки. Рассмотрим некоторые достоинства и недостатки VS Code:

К достоинствам можно отнести легкость использования и интуитивно понятный интерфейс, по сравнению с аналогами, работу на компьютерах с недостаточно современными комплектующими, возможность интеграции с Git.

Из недостатков следует отметить меньшее количество инструментов для некоторых языков программирования, относительно меньшее сообщество, по сравнению с другими средами разработки.

1.2.3 Eclipse

Eclipse - среда разработки кроссплатформенных приложений с поддержкой многих языков программирования. Рассмотрим некоторые из достоинств и недостатков Eclipse.

К достоинствам можно отнести то, что IDE предоставляет множество инструментов для разработки, имеет большое сообщество, и обеспечивает хорошую поддержку, доступ к различным плагинам.

К недостаткам можно отнести специфичный интерфейс среды по сравнению с конкурентами, который будет вызывать сложности у новичков, упор IDE на язык java, а также большое количество компонентов для корректной и удобной работы.

Проанализировав достоинства и недостатки данных сред разработки, было принято решение разрабатывать мессенджер с помощью Qt Creator в связи с наличием документации, более удобным и интуитивно понятным интерфейсом, а также большим выбором подключаемых библиотек

1.2.4 SQLite

SQLite — это встраиваемая СУБД, которая позволяет хранить данные на устройстве, без использования сервера. Рассмотрим некоторые из достоинств и недостатков SQLite.

К достоинствам можно отнести то, что БД очень проста в интеграции и использовании, не требует отдельного сервера для работы, обладает высокой скоростью обработки запросов и операций и малым весом, очень надежна, является кроссплатформенной, а также поддерживает большинство языков программирования.

Из недостатков можно отметить то, что в SQLite отсутствует бесплатная поддержка, имеется ограничение многопоточности на запись.

Проанализировав достоинства и недостатки данной базы данных, а также сравнив с её аналогами, было принято решение разрабатывать мессенджер с помощью SQL Lite.

1.3 Сравнение ОС

Разработка корпоративного мессенджера будет происходить под ОС Windows, однако необходимо рассмотреть некоторые операционные системы, чтобы определить их достоинства и недостатки.

Выбор операционной системы играет ключевую роль при создании мессенджера, ведь это определяет доступность, безопасность и функциональность приложения для конечных пользователей. Рассмотрим достоинства и недостатки каждой из операционных систем.

1.3.1 ОС Windows

Достоинства ОС Windows:

					ПГУ 1.090301.35.001 ПЗ	Лист
						12
Изм.	Лист	№ докум.	Подпись	Дата		

– широкая совместимость. Windows поддерживает огромное количество программ и игр, что делает его привлекательным выбором для широкого сообщества пользователей;

– простота использования и интуитивно понятный интерфейс. ОС интуитивно понятна для большинства пользователей, что обеспечивает легкость в освоении и использовании операционной системы новичками;

– широкий выбор IDE. Операционная система совместима с множеством устройств и комплектующих, что позволяет пользователям выбирать среду разработки под свои потребности.

Недостатки ОС Windows:

– стоимость лицензий. Платная лицензия на Windows очень велика в цене, особенно для корпоративных клиентов;

– ограничения на производительность. Некоторые версии Windows требуют более современных комплектующих для оптимальной производительности;

– необходимость регулярных обновлений. Обновления Windows вызывают неудобства у пользователей и требуют перезагрузки системы, что часто приводит к потере времени и данных.

1.3.2 ОС Linux

Достоинства ОС Linux:

– безопасность. Linux известен своей высокой безопасностью благодаря регулярным обновлениям и большому сообществу разработчиков;

– функциональность. Хотя Linux менее распространен среди пользователей ПК приложений, он обладает множеством возможностей для разработки;

– удобство использования. В зависимости от дистрибутива, Linux имеет разные уровни сложности в использовании, для комфортного использования необходимы более глубокие знания у пользователей.

Недостатки ОС Linux:

– ограниченная поддержка программного обеспечения. Несмотря на то, что Linux поддерживает множество приложений, часть программ недоступны для этой платформы;

- фрагментация дистрибутивов. Linux имеет множество дистрибутивов с разными версиями ядра и библиотеками, что предоставляет усложненную разработку и обеспечение совместимости приложений с версиями Linux;
- сложность установки приложений. Установка приложений для неопытных пользователей будет непривычна и займет больше времени.

1.3.3 ОС MacOS

Достоинства ОС MacOS:

- интуитивно понятный интерфейс. MacOS предоставляет пользователю простой и легкий в использовании интерфейс. Это делает работу с устройствами Apple приятной и комфортной для пользователей любого уровня;
- высокий уровень безопасности. ОС известна своим высоким уровнем безопасности, что обеспечивает защиту персональных данных и конфиденциальной информации пользователей от угроз в сети и утечек;
- интеграция с устройствами Apple. Пользователи могут синхронизировать свои данные и работать с ними на разных устройствах Apple, таких как iPhone, iPad и MacBook;
- экосистема приложений. Пользователю предлагается большой выбор приложений и сервисов, что позволяет устанавливать приложения под свои цели.

Недостатки ОС MacOS:

- ограниченная аппаратная совместимость. ОС сильно ограничена выбором аппаратного оборудования, поскольку это система, работающая исключительно на устройствах Apple;
- ограниченная поддержка игр. MacOS не всегда поддерживает широкий спектр игр, доступных на других платформах, что ограничивает аудиторию для игровых приложений;
- высокая стоимость разработки. Разработка и обновление приложений для macOS очень дорого обходится из-за необходимости использования оборудования Apple и определенной платформы разработки Xcode;

					ПГУ 1.090301.35.001 ПЗ	Лист
						14
Изм.	Лист	№ докум.	Подпись	Дата		

– ограниченная пользовательская база. Пользовательская база macOS значительно меньше, чем у Windows, что снижает популярность приложений среди аудитории пользователей.

Проанализировав достоинства и недостатки данных операционных систем, был сделан выбор в пользу ОС Windows, по причине большого выбора IDE, документаций к ним, а также распространенности на рынке.

Также стоит отметить, что в скором времени планируется разработка мессенджера под другие ОС.

1.4 Архитектура приложения

Выбор хорошей архитектуры приложения невероятно важная вещь, поскольку правильно созданная архитектура при необходимости позволяет вносить изменения с минимальными затратами ресурсов.

При выборе архитектуры приложения было решено воспользоваться MVC архитектурой, её суть заключается в разделении данных и логики приложения на 3 уровня (Модель-Вид-Контроллер) [1].

Разработана архитектура будущего приложения (см. рисунок 1).

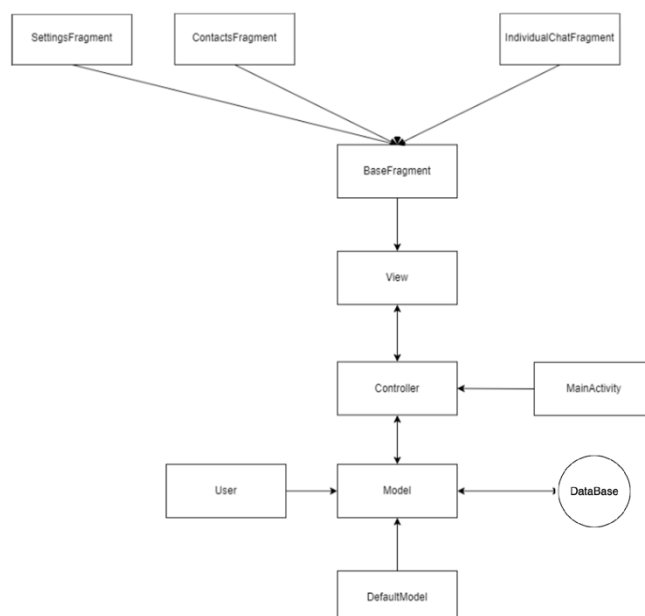


Рисунок 1 – MVC-архитектура приложения

Разработана UML – диаграмма классов будущего приложения (см. рисунок 2).

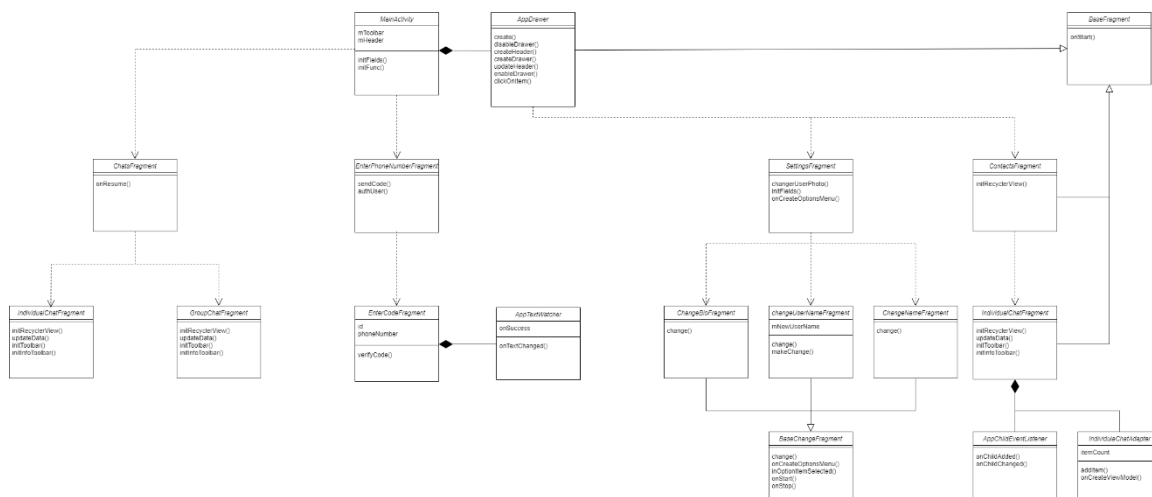


Рисунок 2 - UML – диаграмма

Рассмотрим данную диаграмму:

– AutentApp. Класс, который отвечает за ввод пользователем параметров, необходимых для входа в приложение (atc и dns адреса сервера, sip номер и пароль, далее происходит передача заполненных полей на сервер для последующего подтверждения;

– setInicialisation. Вспомогательный класс, отвечающий за процесс инициализации, обеспечивающий дальнейший вход в основное меню мессенджера;

– menu_Settings. Класс, создающий окно изменения настроек, который содержит несколько пунктов, таких как – изменение atc и dns адресов сервера, параметров просмотра медиа, а также обновления ПО и сброса приложения;

– getListOfContacts. Класс, отвечающий за загрузку контактов из БД в специальный список. Внутри используется IndividualChat, который создает с пользователем из контактов чат для общения;

– individualChat. Класс, который используется для загрузки сообщений из БД и для их отображения в ListWidget;

– addContactUser. Класс, использующийся для добавление нового контакта. Получает от пользователя имя нового контакта и номер телефона, затем отправляет запрос на сервер и сохраняет контакт в БД;

– delContact. Класс, который отвечает за удаление выбранного контакта из списка. Также осуществляется запрос на сервер, после успешного ответа происходит удаление выбранного контакта из базы данных;

– searchContact. Класс, отвечающий за поиск нужного контакта в общем списке. Если такой контакт существует, из списка исчезнут остальные контакты, а пользователю отобразится найденный контакт.

Для того, чтобы использовать приложение, пользователь должен авторизоваться, затем он сможет пользоваться всем функционалом приложения, который представлен на рисунке 3.

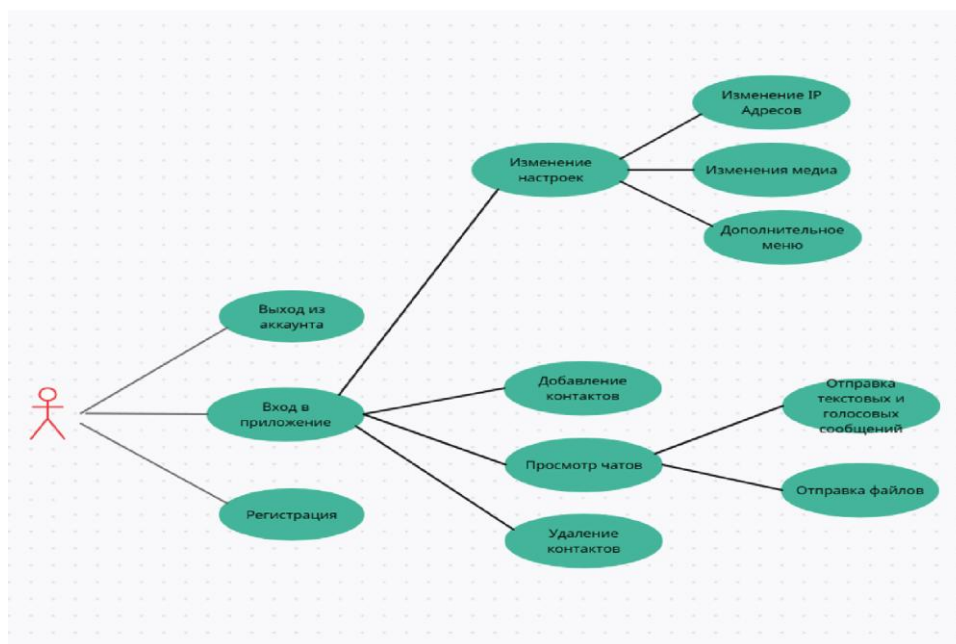


Рисунок 3 – UML-диаграмма вариантов использования

1.5 Реализация приложения

1.5.1 Реализация аутентификации пользователя

Была создана система аутентификации пользователей. Аутентификация использует метод settingsApp (), который принимает на вход:

- sip номер;
- atc и dns адреса сервера;
- пароль.

Пользователю необходимо ввести их (см. рисунок 4) в соответствующие поля. Для их реализации были использованы элементы LineEdit.

Также есть возможность автоматического заполнения данных. Для этого пользователь должен нажать соответствующую кнопку “Заполнить из файла”. После нажатия на которую, открывается диалоговое окно, в котором необходимо выбрать файл конфигурации. Поля заполнятся и произойдёт автоматический вход в приложение.

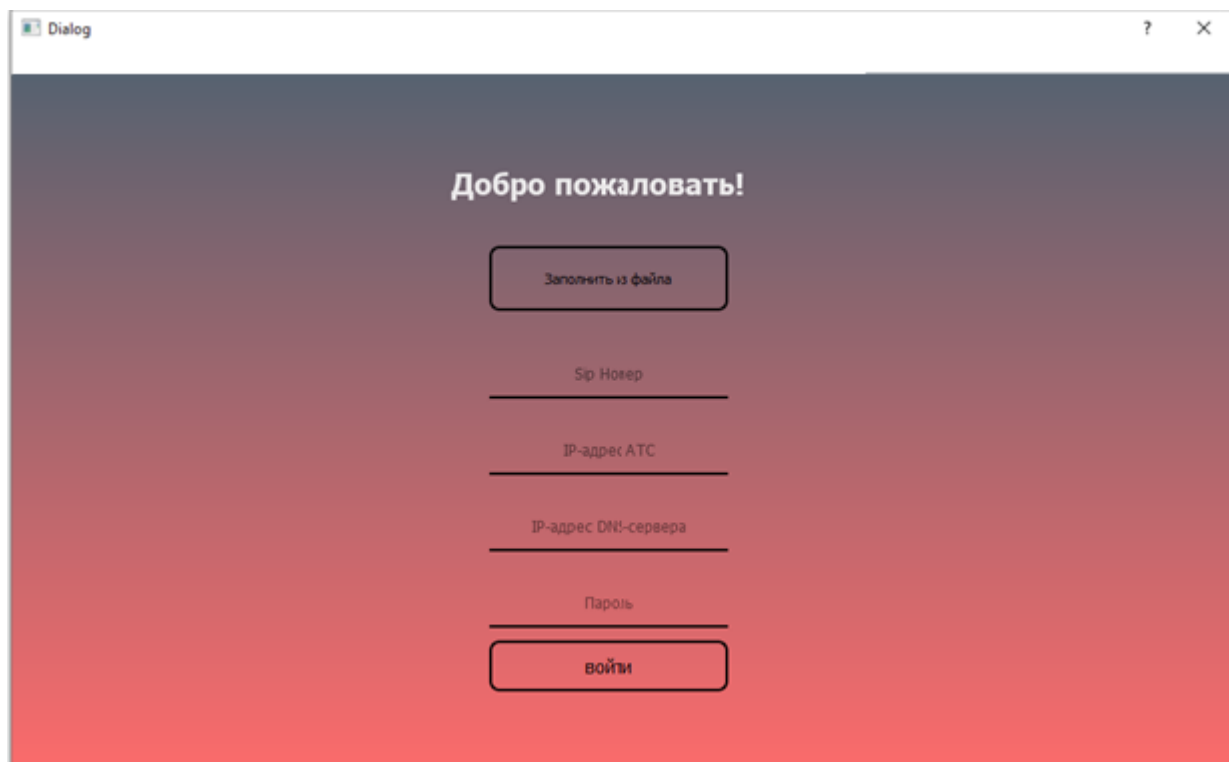


Рисунок 4 - Вход в приложение

Код фрагмента для авторизации пользователей представлен ниже.

```
const QString ID = "id";
const QString SIG = "signature";
const QString PHONE = "phone";
const QString SKZI = "ip_skzi";
const QString MON = "ip_mon";
const QString ATS = "ip_ats";
const QString DNS = "ip_dns";
const QString KS = "ks";
```

```

QJsonValue jsonId = json.value(ID);
QString id = jsonId.toString();

QJsonValue jsonSig = json.value(SIG);
QString sig = jsonSig.toString();


QJsonValue jsonPhone = json.value(PHONE);
QString phone = jsonPhone.toString();


QJsonValue jsonSkzi = json.value(SKZI);
QString skzi = jsonSkzi.toString();


QJsonValue jsonMon = json.value(MON);
QString mon = jsonMon.toString();


QJsonValue jsonAts = json.value(ATS);
QString ats = jsonAts.toString();


QJsonValue jsonDns = json.value(DNS);
QString dns = jsonDns.toString();


QJsonValue jsonKs = json.value(KS);
QString ks = jsonKs.toString();


QString proverka = id + sig + phone + skzi + mon + ats + dns;


SettingsApp::ins()->setID(id);
SettingsApp::ins()->setSignature(sig);
SettingsApp::ins()->setPhone(phone);
SettingsApp::ins()->setIpAts(ats);
SettingsApp::ins()->setIpDns(dns);

```

```

        qDebug() << "Settings get phone = " << SettingsApp::ins()-
>getPhone();

        qDebug() << "Settings get ip_ats = " << SettingsApp::ins()-
>getIpAts();

        qDebug() << "Settings get ip_dns = " << SettingsApp::ins()-
>getIpDns();

        int crc = crc32((unsigned char*)proverka.toUtf8().data(),
(unsigned int) proverka.toUtf8().length());

        if (ks.toInt(0, 16) == crc) {
            ui->Nomer_edit->setText(phone);
            ui->ATC_edit->setText(ats);
            ui->DNS_edit->setText(dns);

        } else {
            qDebug() << "сообщение об ошибке";
        }

void SettingsApp::setPhone(QString phone){
    mSettings->setValue( "phone", phone );

}

QString SettingsApp::getPhone() {
    return mSettings->value("phone", 0).toString();
}

void SettingsApp::setID(QString id){
    mSettings->setValue( "id", id );
}

QString SettingsApp::getID() {
    return mSettings->value("id", 0).toString();
}

void SettingsApp::setSignature(QString signature){

```

```

        mSettings->setValue( "signature", signature );
    }

QString SettingsApp::getSignature() {
    return mSettings->value("signature", 0).toString();
}

void SettingsApp::setIpAts(QString ip_ats){
    mSettings->setValue( "ip_ats", ip_ats );
}

QString SettingsApp::getIpAts() {
    return mSettings->value("ip_ats", 0).toString();
}

void SettingsApp::setIpDns(QString ip_dns){
    mSettings->setValue( "ip_dns", ip_dns );
}

void SettingsApp::setInitialization()
{
    mSettings->setValue("init", 1);
}

QString SettingsApp::getIpDns() {
    return mSettings->value("ip_dns", 0).toString();
}

int SettingsApp::getInitialization()
{
    return mSettings->value("init", 0).toInt();
}

```

Для того, чтобы пользователь не передал пустую строку, была создана проверка на заполненность поля, которая при обнаружении пустой строки выдаст

					ПГУ 1.090301.35.001 ПЗ	Лист
						21
Изм.	Лист	№ докум.	Подпись	Дата		

ошибку, и попросит пользователя убедиться в правильности заполнения (см. рисунок 5). Запрос на сервер не будет выполнен, пока одно из полей будет пустым.

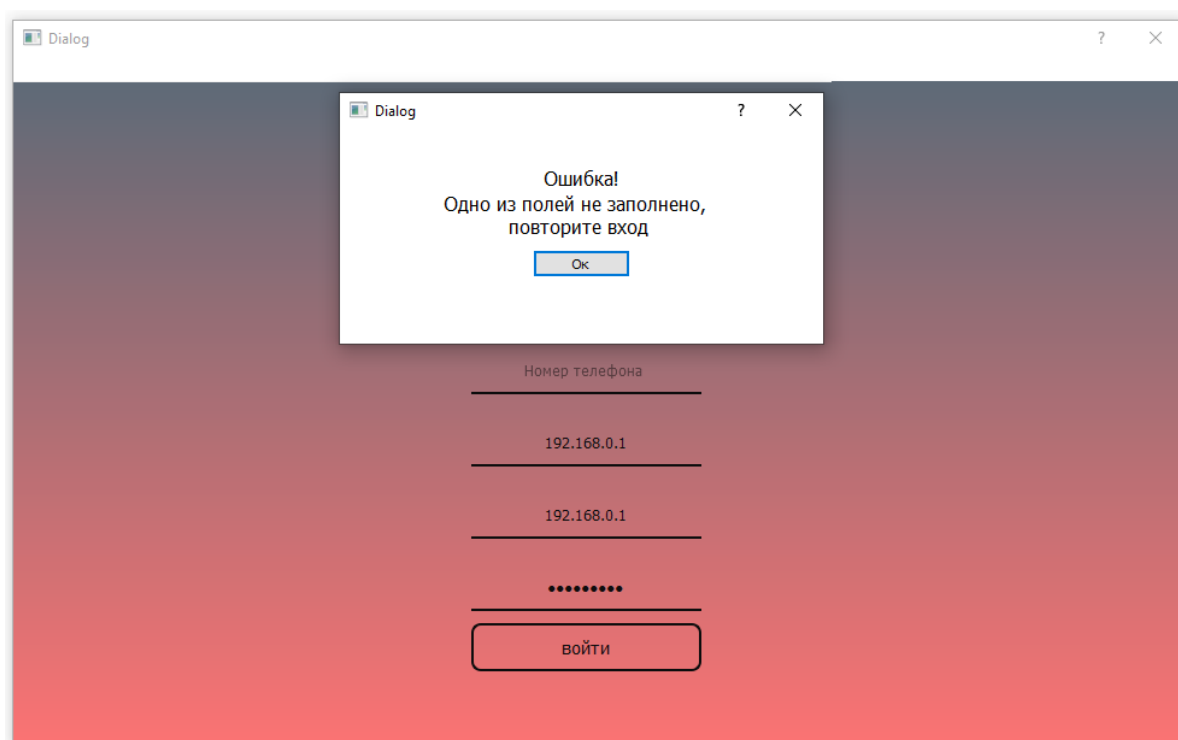


Рисунок 5 - Проверка на пустую строку

Код фрагмента для авторизации пользователей представлен ниже.

```
if(ui->Nomer_edit->text().isEmpty() || ui->ATC_edit->
text().isEmpty() || ui->DNS_edit->text().isEmpty())
{

    Triwindow window;
    window.setModal(true);
    window.exec();

}else{
    SettingsApp::ins()->setInitialization();

this->close();

SecondWindow window;
window.setModal(true);
```

```

window.exec();
}

```

После того как пользователь верно заполнит необходимые поля, произойдет отправка запроса на сервер, процесс проверки данных заполнения (см. рисунок 6), и, если данные корректны, пользователь перейдет в основной фрагмент приложения.

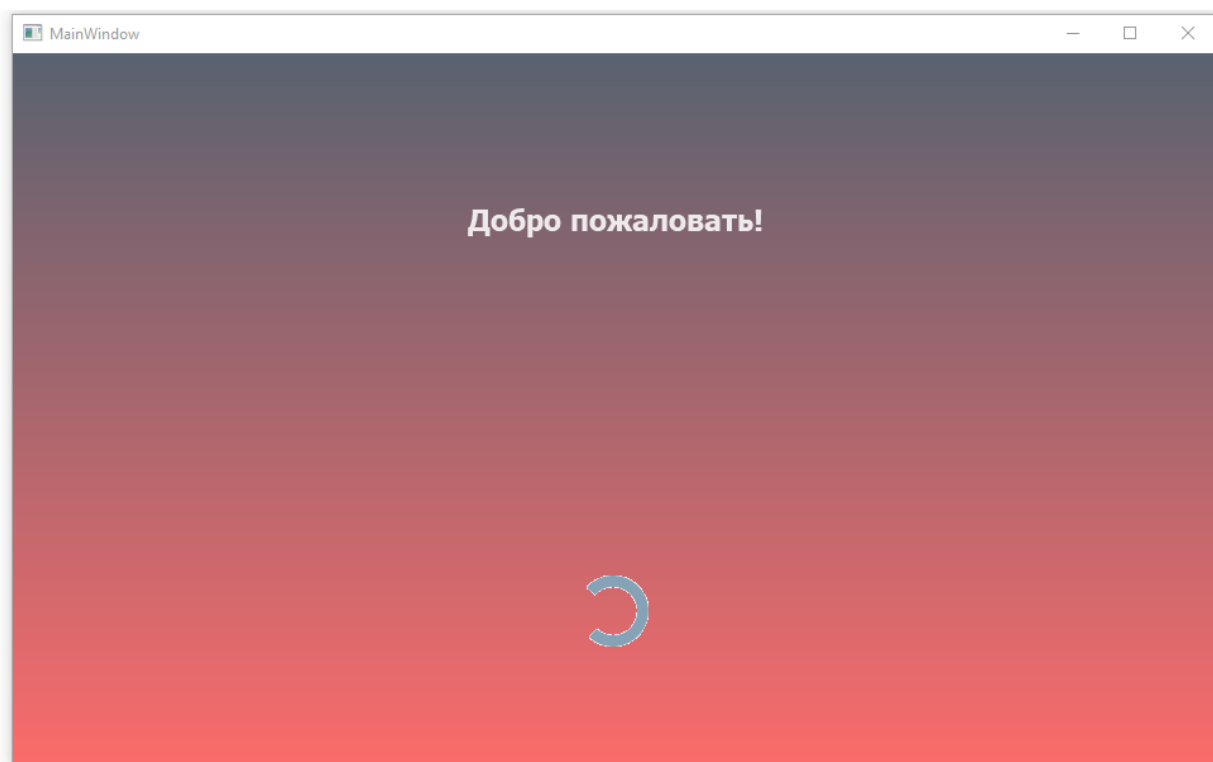


Рисунок 6 – Процесс инициализации

Код фрагмента для авторизации пользователей представлен ниже.

```

QMovie* movie = new QMovie("D:/Peday/loading1.gif"); // Load GIF

if (!movie->isValid()) // Проверка GIF
{
    qDebug() << "сообщение об ошибке";
}

// Play GIF

```

```

        QLabel* label = new QLabel(this);
        label->setGeometry(426,375,128,128);

        label->setMovie(movie);
        movie->start();
        if ( isInit == 0 ) {
            // Если пользователь уже вошел в систему, открываем
основной интерфейс
            this->close();
            MainWindow wew;
            wew.setModal(true);
            wew.exec();

        } else {
            // Если пользователь еще не вошел в систему, открываем окно
для входа
            this->close();
            SecondWindow sw;
            sw.setModal(true);
            sw.exec();
        }

```

1.5.2 Реализация списка контактов

После успешной инициализации, вызывается функция `GetListContacts`, которая отправляет запрос на сервер на получение списка контактов, и после успешного ответа отображает их.

За визуальное отображение контактов отвечает ui-элемент `ListWidget`, который отображает загруженный список контактов из БД. Данный элемент является универсальным и применяется в приложении также для отображения чатов (см. рисунок 7).



Рисунок 7 – Загрузка списка контактов

Код фрагмента загрузки списка контактов представлен ниже.

```

QUrl url ("http://getListOfContacts.php");

QNetworkRequest request(url); //our server with php-script
request.setRawHeader("Accept", "*/*");
request.setRawHeader("Cache-Control", "no-cache");
request.setRawHeader("Content-Type",
"application/json;charset=utf-8");

QJsonObject jb;
jb.insert("id", QJsonValue::fromVariant(SettingsApp::ins()-
>getID()));
jb.insert("signature", QJsonValue::fromVariant(SettingsApp::ins()-
>getSignature()));
QJsonDocument doc(jb);
QNetworkReply *reply = manager->post(request, doc.toJson());
ui->listWidget->clear();
if (reply->error()) {

```

```

        qDebug() << reply->errorString();
    } else {
        QString answer = reply->readAll();
        qDebug() << answer;
    }

    QJsonDocument doc = QJsonDocument::fromJson(answer.toUtf8());
    // Ищем данные по ключу
    QJsonValue value = doc.object().value("listOfContacts");
    // Если value это массив данных
    if ( value.type() == QJsonValue::Array ) {
        // Получаем массив
        QJsonArray array = value.toArray();
        spo_delete_all_contact();

        for( int i = 0; i < array.size(); i++ ) {
            sonValueRef ref = array[i];
            // Если ValueRef это объект
            if ( ref.type() == QJsonValue::Object ) {
                QJsonObject json = ref.toObject();

                QJsonValue jsonId = json.value("id");
                QString id = QString::number(jsonId.toInt());

                QJsonValue jsonName = json.value("name");
                QString name = jsonName.toString();

                QJsonValue jsonPhone = json.value("phone");
                QString phone = jsonPhone.toString();

                SpoContact contact;
                QByteArray ba = id.toUtf8();
                const char *c_str2 = ba.data();

```

```

contact.id_user = (char *)c_str2;
qDebug() << "id_user - " << c_str2;

QByteArray ba1 = name.toUtf8();
const char *c_str3 = ba1.data();
contact.full_name = (char *)c_str3;
qDebug() << "name - " << c_str3;

QByteArray ba2 = phone.toUtf8();
const char *c_str4 = ba2.data();
contact.sip_number = (char *)c_str4;
qDebug() << "sip_number - " << c_str4;

ui->listWidget->addItem(name); // отображение контактов

```

1.5.3 Реализация добавления контактов

Для добавления новых контактов, реализована функция AddContact. Пользователю необходимо нажать на соответствующую кнопку в левом нижнем углу приложения (см. рисунок 8, 9), ввести имя нового контакта и номер телефона, после чего будет отправлен запрос на сервер, и после успешного ответа добавленный контакт отобразится в списке контактов, а также в БД.



Рисунок 8 – Кнопка удаления контакта

Рисунок 9 – Окно добавление контакта

Код фрагмента для добавления контактов представлен ниже.

```

QUrl url ("http://addContact.php");

QNetworkRequest request(url); //our server with php-script

request.setRawHeader("Accept", "*/*");
request.setRawHeader("Cache-Control", "no-cache");
request.setRawHeader("Content-Type",
"application/json;charset=utf-8");

name = ui->Name_line->text();
nomer = ui->Nomer_line->text();

QJsonObject jб;

jб.insert("id", QJsonValue::fromVariant(SettingsApp::ins()-
>getID()));

jб.insert("signature", QJsonValue::fromVariant(SettingsApp::ins()-
>getSignature()));

jб.insert("name", QJsonValue::fromVariant(name);
jб.insert("nomer", QJsonValue::fromVariant(nomer);

QJsonDocument doc(jб);

QNetworkReply *reply = manager->post(request, doc.toJson());

void Add_contact::Dobavlenie(QNetworkReply* reply){
    /*
    if (reply->error()) {
        qDebug() << reply->errorString();
    } else {
        QString answer = reply->readAll();
        qDebug() << answer;
    }
    */

    //QString answer = reply->readAll();
    QString answer = "{\"id\": 32,\"name\":\"1\", \"phone\":\"123\"}";
    qDebug() << answer;

    QJsonDocument doc = QJsonDocument::fromJson(answer.toUtf8());

    // Ищем данные по ключу

```

```

if ( ref.type() == QJsonValue::Object ) {
    QJsonObject json = ref.toObject();

    QJsonValue jsonId = json.value("id");
    QString id = QString::number(jsonId.toInt());

    SpoContact contact;
    QByteArray ba = id.toUtf8();
    const char *c_str2 = ba.data();
    contact.id_user = (char *)c_str2;
    qDebug() << "id_user - " << c_str2;

    QByteArray ba1 = name.toUtf8();
    const char *c_str3 = ba1.data();
    contact.full_name = (char *)c_str3;
    qDebug() << "name - " << c_str3;

    QByteArray ba2 = nomer.toUtf8();
    const char *c_str4 = ba2.data();
    contact.sip_number = (char *)c_str4;
    qDebug() << "sip_number - " << c_str4;

```

1.5.4 Реализация удаление контактов

За удаление контакта отвечает функция `delete_contact`, которая вызывается после выбора нужного контакта из списка, и нажатия на соответствующую кнопку в правом верхнем углу приложения (см. рисунок 10, 11). Далее, в появившемся диалоговом окне, необходимо подтвердить удаление. При успешном ответа сервера, контакт удалится из базы данных и списка контактов `ListWidget`.



Рисунок 10 – Кнопка удаления контакта

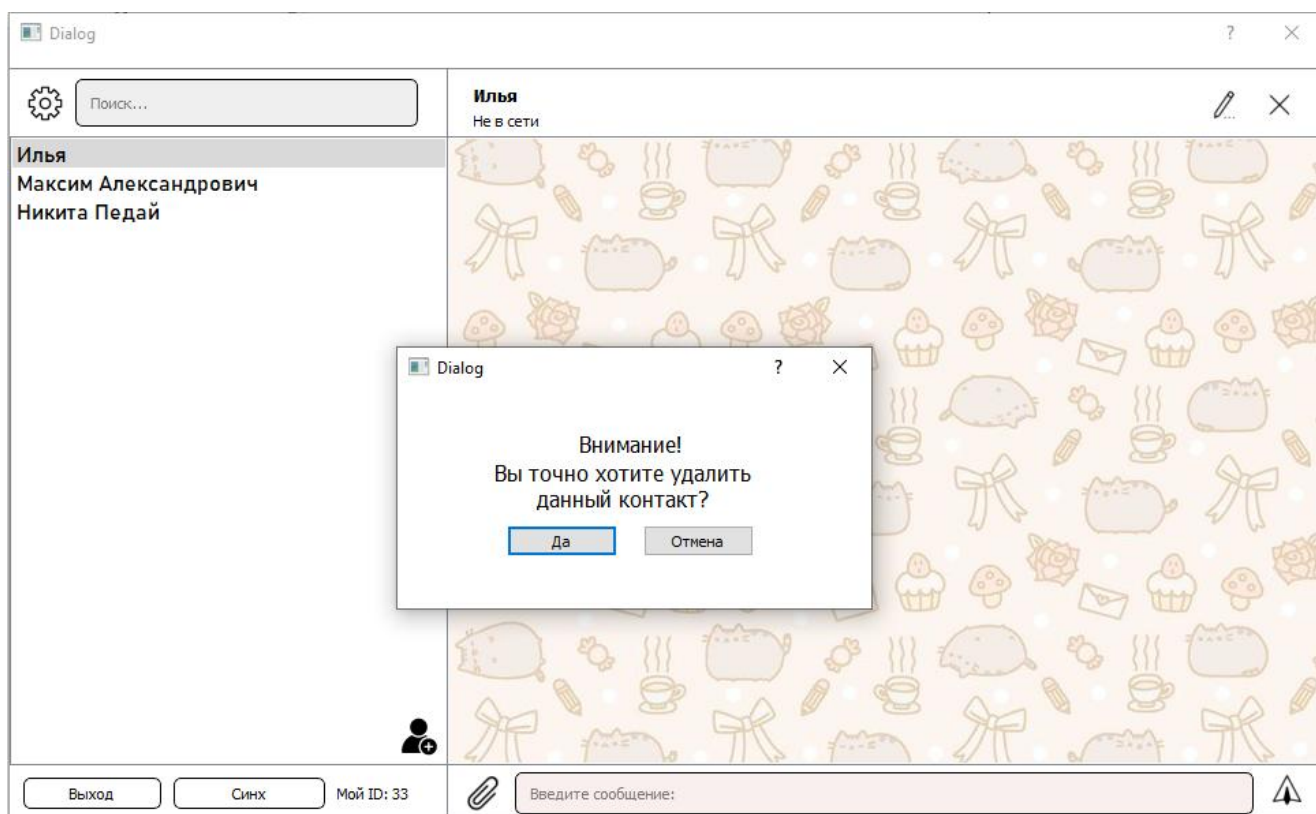


Рисунок 11 – Диалоговое окно подтверждения

Код фрагмента для удаления контакта представлен ниже.

```
manager = new QNetworkAccessManager(this);

connect(manager, &QNetworkAccessManager::finished, this,
&SecondWindow::func2);

QListWidgetItem *it = ui->listWidget->item(ui->listWidget-
>currentRow());

for (auto iter = spisok.begin(); iter != spisok.end(); iter++)
{
    if (it->text() == (((SpoContact *)*iter)->full_name))
    {
        spo_delete_contact((SpoContact *)*iter);

        QUrl url ("http://dropContact.php");
        QNetworkRequest request(url); //our server with php-
script

        request.setRawHeader("Accept", "*/*");
        request.setRawHeader("Cache-Control", "no-cache");
        request.setRawHeader("Content-Type",
"application/json; charset=utf-8");
        del_contact = (SpoContact *)*iter;

        QJsonObject jб;
        jб.insert("id",
QJsonValue::fromVariant(SettingsApp::ins()->getID()));
        jб.insert("signature",
QJsonValue::fromVariant(SettingsApp::ins()->getSignature()));
        jб.insert("id_user",
QJsonValue::fromVariant(((SpoContact *)*iter)->full_name));
        QJsonDocument doc(jб);
        QNetworkReply *reply = manager->post(request,
doc.toJson());
    }
}
```

					ПГУ 1.090301.35.001 ПЗ	Лист
						32
Изм.	Лист	№ докум.	Подпись	Дата		

```

        break;
    }
}
delete(it);

```

1.5.5 Реализация поиска контакта

При необходимости найти определенный контакт, пользователь может воспользоваться строкой поиска в левом верхнем углу программы (см. рисунок 12).



Рисунок 12 – Поисковая строка

Код фрагмента для поиска контакта представлен ниже.

```

bctbx_list_t * get_contact_search(const char * s) {
    char * buf;
    StorageResult clsres;

    if (!dbContext || dbContext->db == NULL) return NULL;

```

```

    buf = sqlite3_mprintf("SELECT * FROM spo_contact WHERE full_name
LIKE '%%%q%%' OR sip_number LIKE '%%%q%%' ORDER BY full_name ASC", s,
s);

    SPOLOGI("get_contacts request = %s", buf);

    clsres.result = NULL;

    sql_request(dbContext->db, buf, &clsres);

    sqlite3_free(buf);

    return clsres.result;
}

```

Пользователь увидит найденный контакт в списке пользователей, при условии, что он существует (см. рисунок 13).

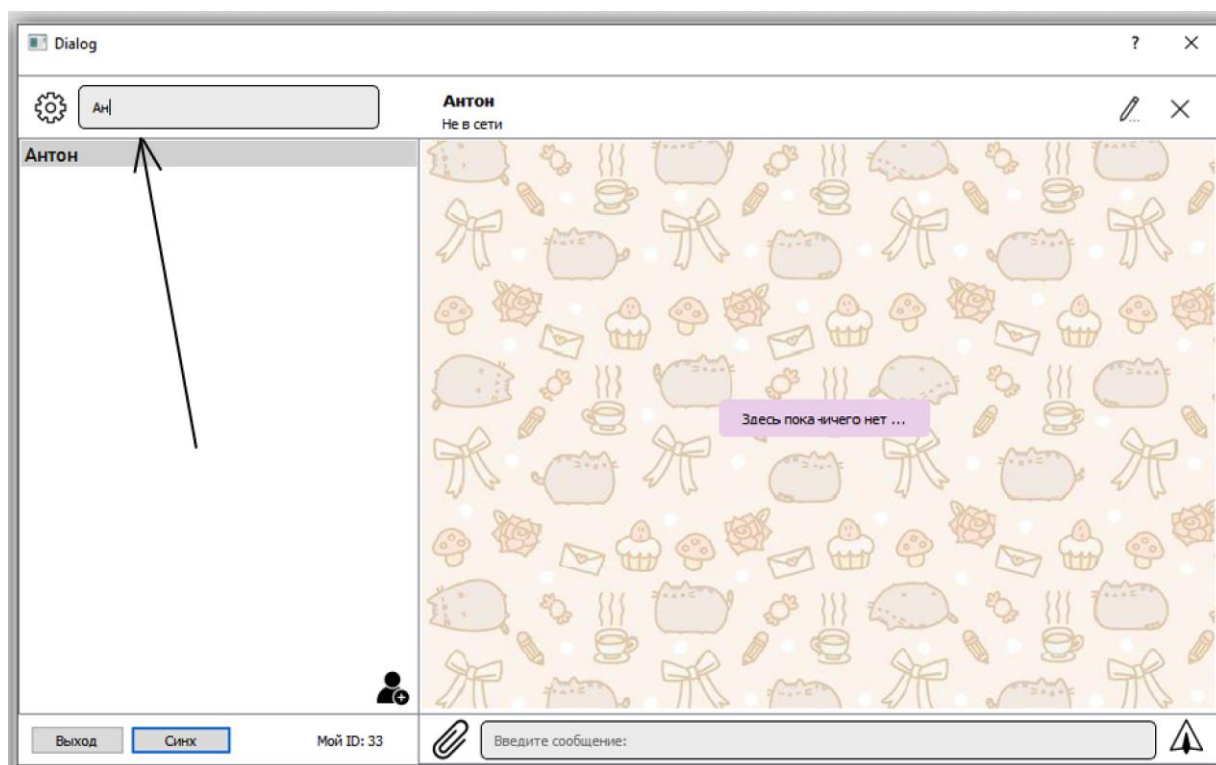


Рисунок 13 – Найденный контакт

1.5.6 Реализация редактирования контактов

Если пользователю необходимо изменить имя контакта, он должен выбрать необходимый ему контакт из списка, и нажать на соответствующую кнопку в виде карандаша в верхнем правом углу приложения (см. рисунок 14). После нажатия на

кнопку откроется окно, где пользователю необходимо будет ввести новое имя контакта. После успешного редактирования, пользователь увидит изменения в списке пользователей.

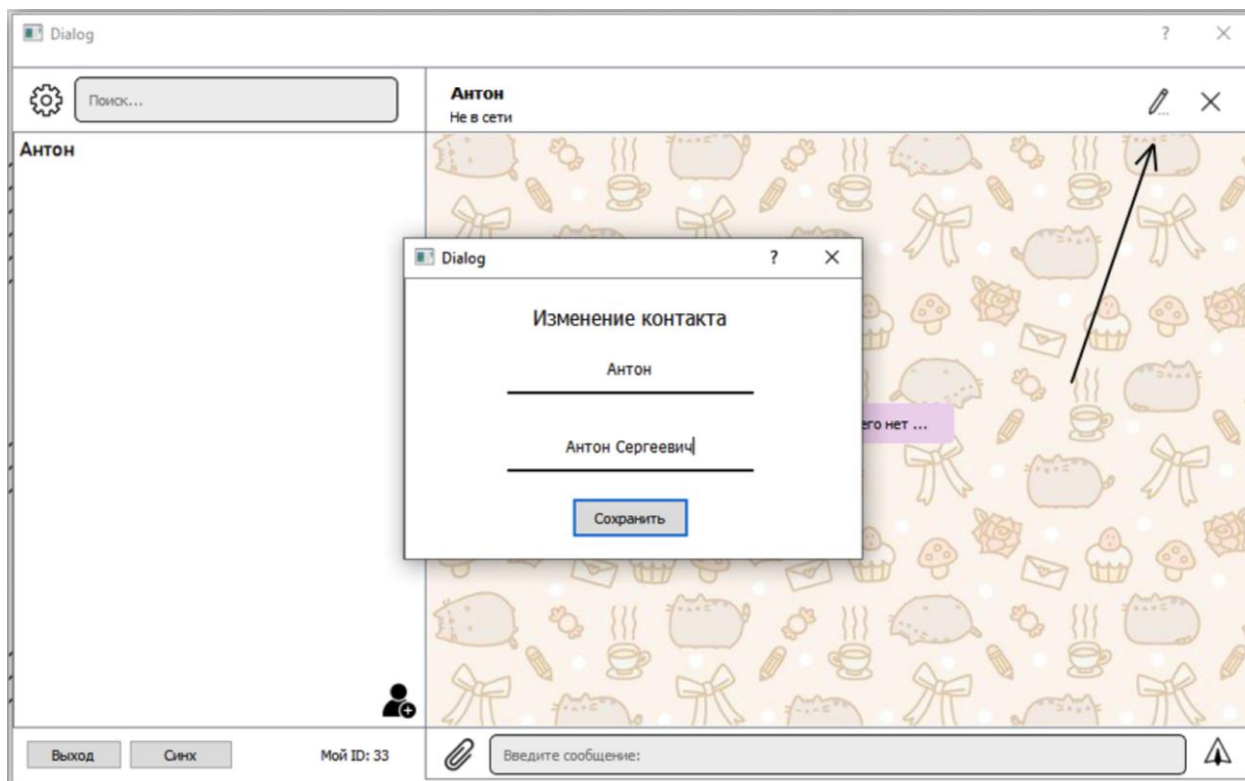


Рисунок 14 – Редактирование контакта

Код фрагмента для поиска контакта представлен ниже.

```
void update_contact(SpoContact * contact){
    char *buf;

    if (!dbContext || dbContext->db == NULL) return ;

    buf = sqlite3_mprintf("UPDATE spo_contact SET full_name=%Q,
        uri_photo=%Q WHERE (id = %u);",
        contact->full_name,
        contact->uri_photo,
        contact->id);

    qDebug("update_contact request = %s", buf);
    sql_request_generic(dbContext->db, buf);
    sqlite3_free(buf);
}
```

}

1.5.7 Реализация настроек пользователя

В окне настроек (см. рисунок 15), пользователь может изменить:

- atc и dns адреса сервера;
- выбрать предпочитаемое качество отображение видео;
- его предпросмотр в чате;
- произвести обновление ПО;
- выполнить сброс приложения.

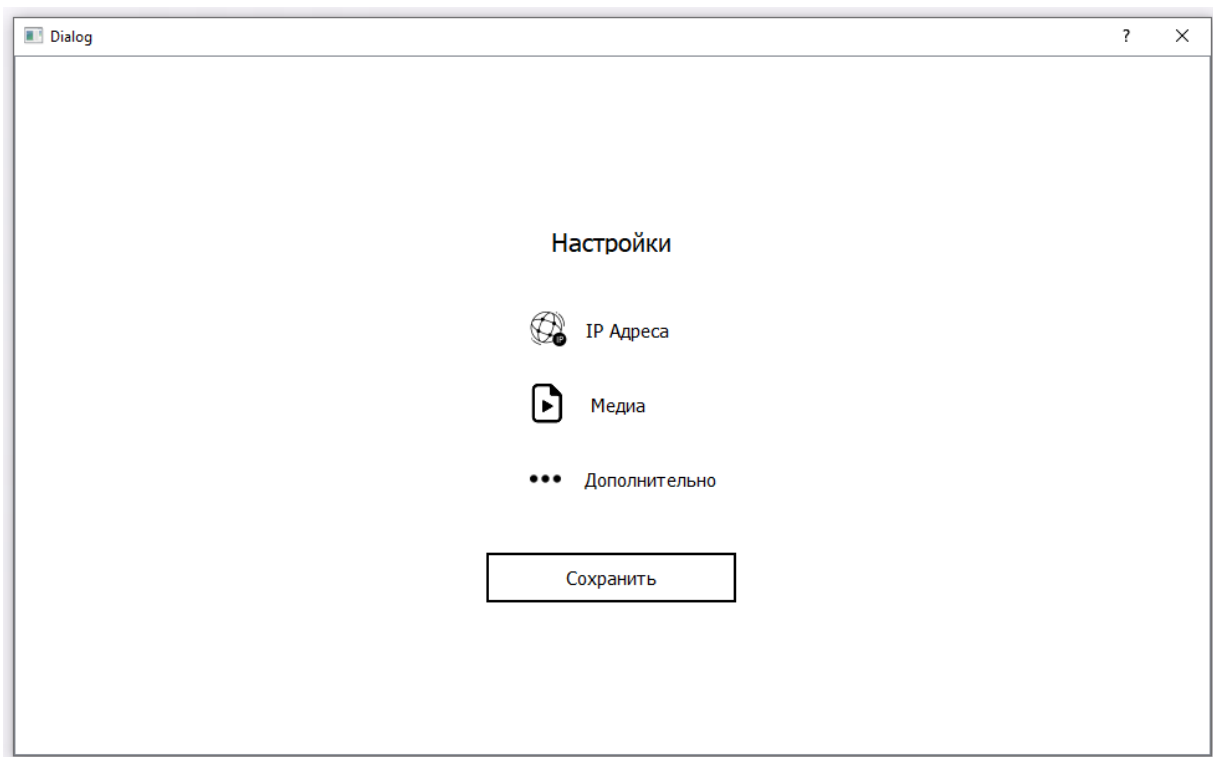


Рисунок 15 – Окно настроек

Для изменения адреса сервера отвечает метод `Adress_change`, который заменяет изначальные адреса на те, которые ввёл пользователь в соответствующие поля `LineEdit` (см. рисунок 16).

IP Адреса

АТС:

DNS:

Рисунок 16 - Окно ip адресов

Код фрагмента для удаления контакта представлен ниже.

```
SettingsApp::ins()->getIpAts() = ui->line_atc->text();
SettingsApp::ins()->getIpDns() = ui->line_dns->text();
QDebug() << SettingsApp::ins()->getIpAts();
QDebug() << SettingsApp::ins()->getIpDns();
```

Чтобы изменить качество просматриваемых видео, а также возможность включить или отключить предпросмотр видео в чате, пользователю необходимо перейти в раздел Медиа (см. рисунок 17).

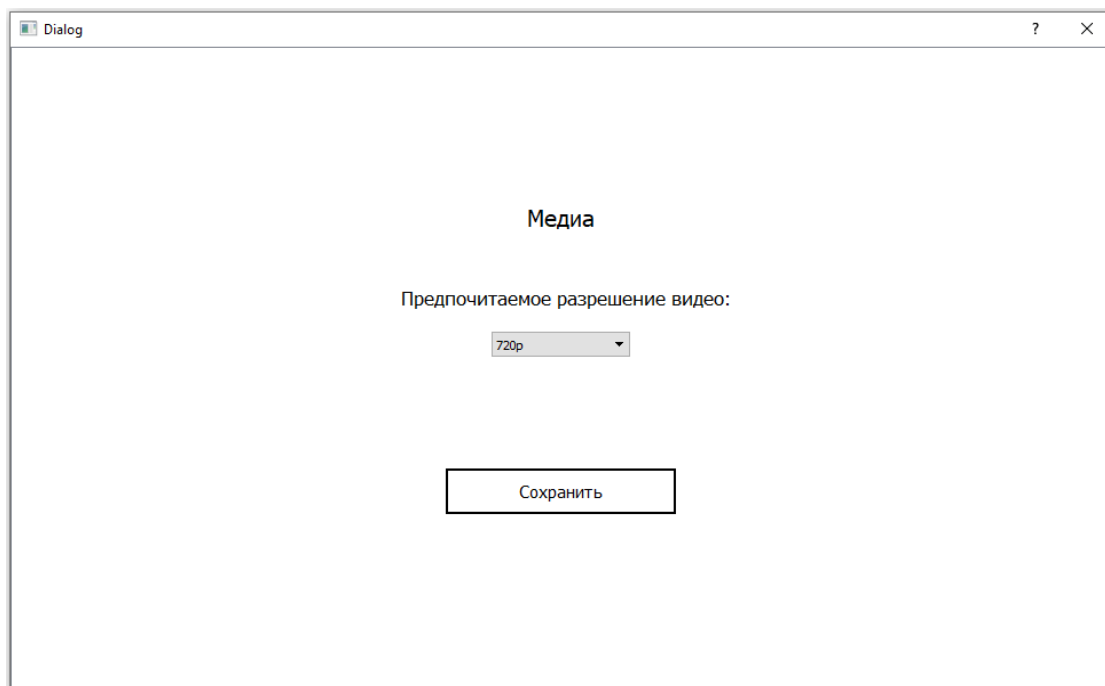


Рисунок 17 - Окно медиа

В разделе Дополнительно пользователь может проверить обновление ПО, а также сбросить приложение (см. рисунок 18).

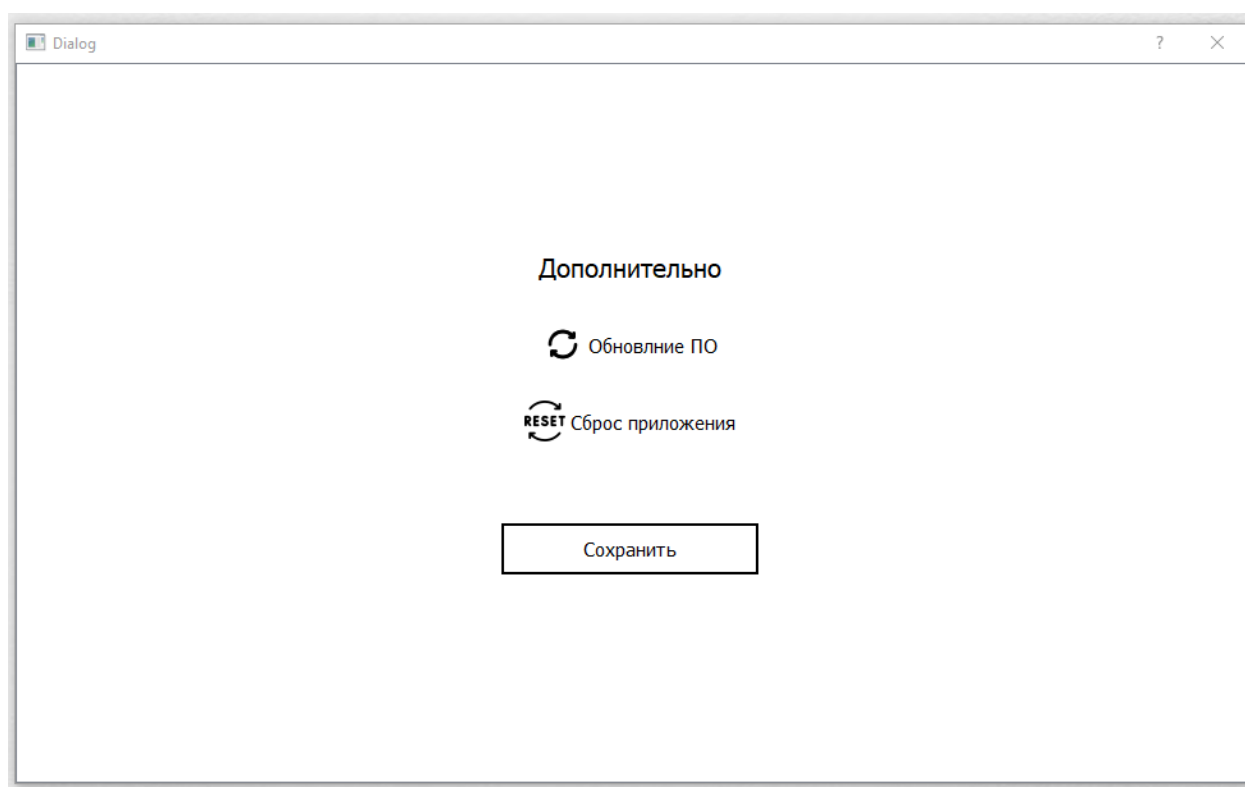


Рисунок 18 – Окно дополнительных параметров

При сбросе приложения удаляются все существующие контакты, чаты, а также настройки.

Код фрагмента для удаления контакта представлен ниже.

```
void delete_all_messages(unsigned long id_chat_room) {
    char *buf;

    if (!dbContext || dbContext->db == NULL) return;

    buf = sqlite3_mprintf("DELETE FROM spo_message WHERE
id_chat_room = %lu", id_chat_room);
    sql_request_generic(dbContext->db, buf);
    sqlite3_free(buf);
}

void delete_all_contact() {
    char *buf;

    if (!dbContext || dbContext->db == NULL) return ;

    buf = sqlite3_mprintf("DELETE FROM spo_contact");
    qDebug("delete_all_contact request = %s", buf);
    sql_request_generic(dbContext->db, buf);
    sqlite3_free(buf);
}

void spo_clear_settings(void) {
    delete_all_value();
}
```

1.5.8 Реализация чата и отправки сообщений

При нажатии на пользователя из списка контактов, открывается одиночный чат, в котором пользователь имеет возможность обмениваться текстовыми сообщениями, изображениями, файлами, очищать чат или удалить контакт.

Для отправки сообщения, пользователь должен нажать значок отправки сообщений в нижней правой части приложения (см. рисунок 19).

Функция IndividualChat определяет выбранный контакт из списка ListWidget, и подгружает чат из базы данных.

Для проверки новых сообщений, был выставлен флаг опроса сервера на интервал в 5 секунд. Это позволяет опрашивать сервер на поступление новых сообщений и подгружать их, как только они поступят.

Для удобства отображения сообщений используется функция SeeMassage. Он оптимизирован таким образом, чтобы загружать из БД определённое количество сообщений, которые будут видны пользователю. Другие сообщения загружаются из базы данных при прокрутке, которая реализована с помощью Vertical Scroll Bar.

Если в момент отправки сообщения одному из контактов отсутствует интернет – соединения, сообщения добавляются в специальную таблицу Message_err. При восстановлении сигнала, содержимое этой таблицы автоматически будет повторно отправлено пользователю.



Рисунок 19 - Окно диалога с контактом

Код фрагмента для поиска контакта представлен ниже.

```
manager = new QNetworkAccessManager(this); //создание менеджера
connect(manager, &QNetworkAccessManager::finished, this,
&res_message_thread::func);
while (1) {
    QUrl url ("http://getIncomingMessages.php");
    QNetworkRequest request(url); //our server with php-script
    request.setRawHeader("Accept", "*/*");
    request.setRawHeader("Cache-Control", "no-cache");
    request.setRawHeader("Content-Type",
"application/json; charset=utf-8");

    QJsonObject jb;
    jb.insert("id", QJsonValue::fromVariant(SettingsApp::ins()-
>getID()));
    jb.insert("signature",
QJsonValue::fromVariant(SettingsApp::ins()->getSignature()));
    QJsonDocument doc(jb);
    flag = 1;
    QNetworkReply *reply = manager->post(request, doc.toJson());
    while (flag) {

    }
    sleep (5);
}
QString answer = reply->readAll();
QJsonDocument doc = QJsonDocument::fromJson(answer.toUtf8());
QJsonValue ref = doc.object().value("messages");
if ( ref.type() == QJsonValue::Object ) {
    QJsonObject json = ref.toObject();

    QJsonValue jsonId = json.value("id");
```

```

QString id = QString::number(jsonId.toInt());

QJsonValue jsonText = json.value("text");
QString text = jsonText.toString();


SpoChatMessage messages;
messages.id = id.user;
messages.dir = 1;
messages.is_read = 0;
messages.type_content = 0;
messages.status = 1;
messages.id_file = 0;


QByteArray ab = id.toUtf8();
const char *c_str123 = ab.data();
messages.id_user = (char *)c_str123;
qDebug() << "id_user - " << c_str123;


QByteArray aa = text.toUtf8();
const char *c_str121 = aa.data();
messages.message = (char *)c_str121;
qDebug() << "message - " << c_str121;


messages.id_chat_room = NULL;
messages.date = NULL;
messages.date_send = NULL;
messages.id_ozu_file = NULL;
}

void create_chat_message_table(sqlite3* db) {

```

					ПГУ 1.090301.35.001 ПЗ	Лист
						42
Изм.	Лист	№ докум.	Подпись	Дата		

```

char* errmsg=NULL;
int ret;
unsigned int version;
int is_old_db = 0;
ret=sqlite3_exec(db,"CREATE TABLE IF NOT EXISTS spo_message ("
"id          INTEGER PRIMARY KEY AUTOINCREMENT,"
"dir          INTEGER,"
"is_read      INTEGER,"
"type_content INTEGER,"
"status       INTEGER,"
"id_file      INTEGER,"
"id_user      TEXT NOT NULL,"
"message      TEXT NOT NULL,"
"id_chat_room INTEGER,"
"date         INTEGER,"
"date_send    INTEGER"
");",
0, 0, &errmsg);

qDebug("DEBUG: create_chat_message_table ret = %d, errmsg = %s",
ret, errmsg);

if(ret != SQLITE_OK) {sqlite3_free(errmsg);
return ;
}

if (dbContext->is_new) {
version = SPO_DB_VER;
} else {
version = *(unsigned int *) (dbContext->mem->data+60);
}

```

1.5.9 Реализация отправки файлов

За отправку файлов отвечает функция `get_files`. Чтобы отправить файл, необходимо нажать на значок скрепки, и в открывшемся диалоговом окне

выбрать необходимый для отправки файла (см. рисунок 20). После выбора файла, необходимо нажать на кнопку отправить. Для удобства выбора файлов, реализован фильтр, который включает в себя сортировку:

- изображения;
- видео;
- документы.

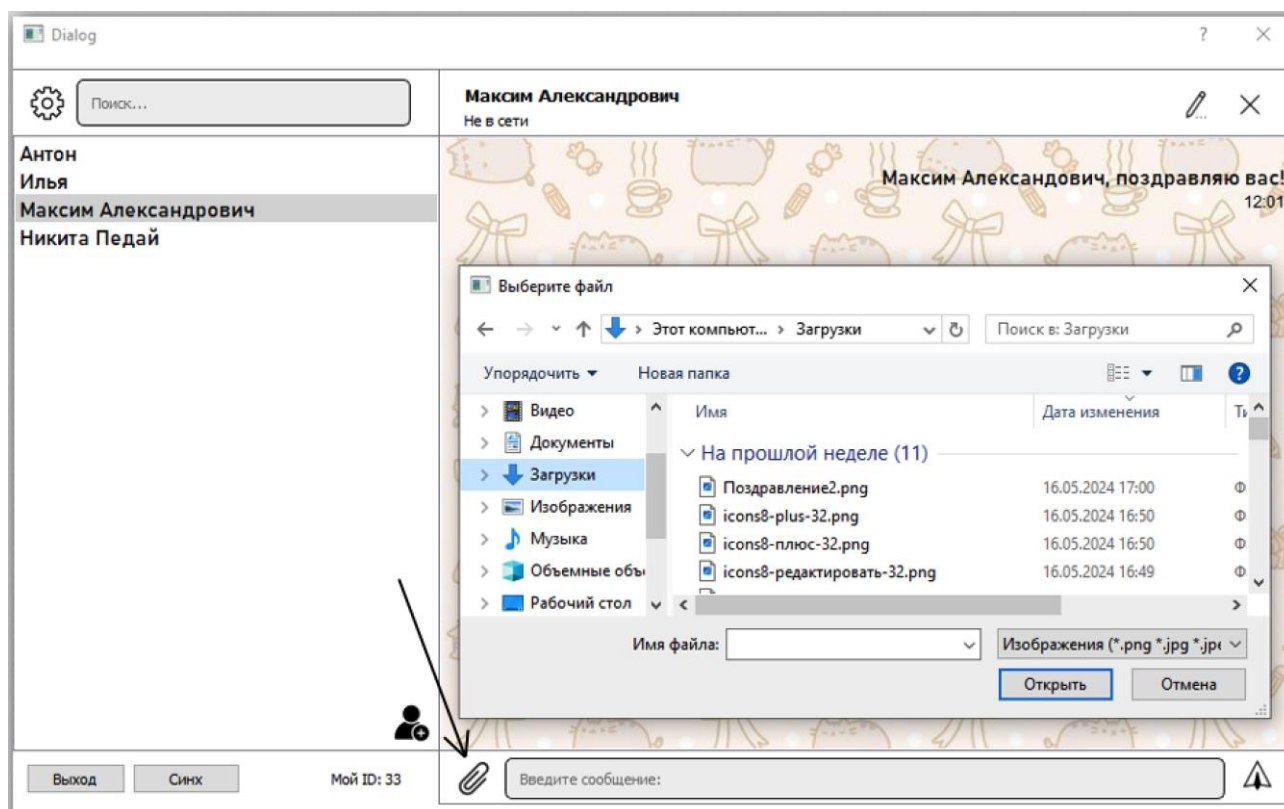


Рисунок 20 – Диалоговое окно выбора файла

Код фрагмента для удаления контакта представлен ниже.

```
QString fileName = QFileDialog::getOpenFileName(this, tr("Выберите файл"), "", tr("Изображения (*.png *.jpg *.jpeg)"));
```

```
if (!fileName.isEmpty()) {
    QPixmap pixmap(fileName);
    if (!pixmap.isNull()) {
        QWidget* itemWidget = new QWidget();
        QVBoxLayout* itemLayout = new QVBoxLayout(itemWidget);
```

```

        QLabel* imageLabel = new QLabel();
        imageLabel->setPixmap(pixmap);
        itemLayout->addWidget(imageLabel);
        itemLayout->setAlignment(Qt::AlignRight);

        QLabel* textLabel = new
QLabel(QFileInfo(fileName).fileName());

        itemLayout->addWidget(textLabel);
        textLabel->setAlignment(Qt::AlignCenter);

        QListWidgetItem* item = new QListWidgetItem();
        item->setSizeHint(itemWidget->sizeHint());
        ui->chats->addItem(item);
        ui->chats->setItemWidget(item, itemWidget);

        QTime currentTime = QTime::currentTime();

        // Форматирование времени в строку
        QString timeString = currentTime.toString("hh:mm");

        // Создание сообщения с временной меткой
        QString message = timeString;

        QListWidgetItem* lii = new QListWidgetItem(message);

        // Добавление сообщения в QListWidget
        ui->chats->addItem(lii);
        lii->setTextAlignment(Qt::AlignRight);
        lii->setFont(QFont("Buhnschrift", 9));

```

```

    } else {

        QMessageBox::warning(this, tr("Ошибка"), tr("Невозможно
загрузить изображение"));

    }

}

}

```

Функция `create_file_table` создаёт таблицу сообщений в базе данных. У файлов есть общая структура, которая состоит из id файла, статуса прочтения, пути расположения, полного имени, размера, а также время отправления (см. рисунок 21).

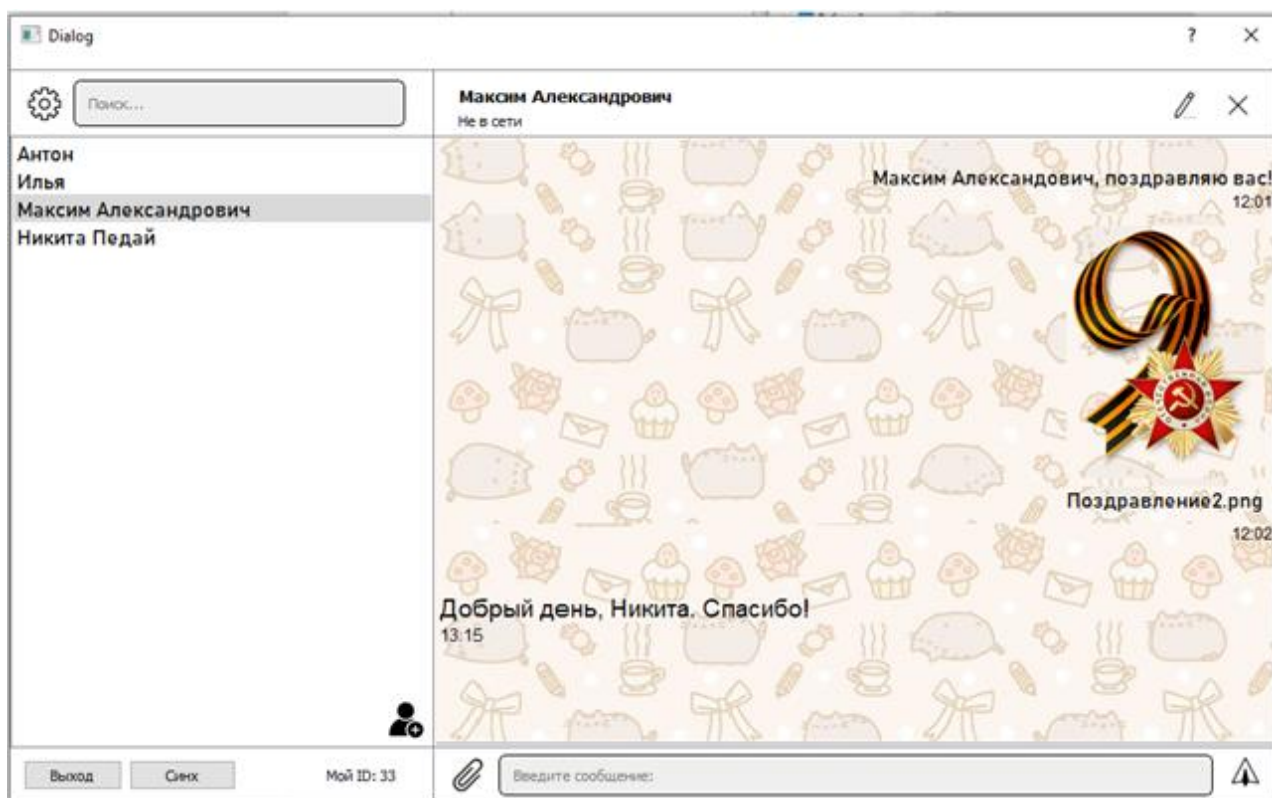


Рисунок 21 – Отправка файла

Код фрагмента для поиска контакта представлен ниже.

```

void create_file_table(sqlite3* db) {
    char* errmsg = NULL;
    int ret;
    ret=sqlite3_exec(db,"CREATE TABLE IF NOT EXISTS spo_file ("

```

```

        "id                INTEGER PRIMARY KEY AUTOINCREMENT,"
        "dir                INTEGER,"
        "id_from_server    TEXT,"
        "status            INTEGER,"
        "path              TEXT,"
        "url_download      TEXT,"
        "full_name         TEXT NOT NULL,"
        "size              INTEGER,"
        "date              INTEGER"

        ");",

        0,0,&errmsg);

    qDebug("DEBUG: create_file_table ret = %d, errmsg = %s", ret,
errmsg);

    if(ret != SQLITE_OK) {
        sqlite3_free(errmsg);
    }
}

static int callback(void *data, int argc, char **argv, char **colName)
{
    StorageResult *clsres = (StorageResult *)data;
    SpoFile * file = (SpoFile *) malloc(sizeof(SpoFile));

    file->id                = (unsigned long ) atol(argv[0]);
    file->dir                = (unsigned int) atoi(argv[1]);
    file->id_from_server= strdup(argv[2]);
    file->status            = (unsigned int) atoi(argv[3]);
    file->path              = strdup(argv[4]);
    file->url_download = strdup(argv[5]);
    file->name              = strdup(argv[6]);
    file->size              = (unsigned int) atoi(argv[7]);
    file->date              = strtoul(argv[8], NULL, 10);

```

```

        //clsres->result = bctbx_list_append(clsres->result, file);
        return 0;
    }

    bctbx_list_t * get_files_received(void) {
        char * buf;
        StorageResult clsres;

        if (!dbContext || dbContext->db == NULL) return NULL;

        buf = sqlite3_mprintf("SELECT * FROM spo_file WHERE dir = '1'
AND full_name NOT LIKE '%%.spx' ORDER BY date DESC");

        qDebug("get_files_received request = %s", buf);
        clsres.result = NULL;
        sql_request(dbContext->db, buf, &clsres);

        sqlite3_free(buf);

        return clsres.result;
    }

    QString fileName = QFileDialog::getOpenFileName(this, tr("Выберите
файл"), "", tr("Изображения (*.png *.jpg *.jpeg)"));

    if (!fileName.isEmpty()) {
        QPixmap pixmap(fileName);
        if (!pixmap.isNull()) {
            QWidget* itemWidget = new QWidget();
            QVBoxLayout* itemLayout = new QVBoxLayout(itemWidget);

            QLabel* imageLabel = new QLabel();

```

```

        imageLabel->setPixmap(pixmap);
        itemLayout->addWidget(imageLabel);
        itemLayout->setAlignment(Qt::AlignRight);

        QLabel* textLabel = new
QLabel(QFileInfo(fileName).fileName());
        itemLayout->addWidget(textLabel);
        textLabel->setAlignment(Qt::AlignCenter);

        QListWidgetItem* item = new QListWidgetItem();
        item->setSizeHint(itemWidget->sizeHint());
        ui->chats->addItem(item);
        ui->chats->setItemWidget(item, itemWidget);

        QTime currentTime = QTime::currentTime();

        // Форматирование времени в строку
        QString timeString = currentTime.toString("hh:mm");

        // Создание сообщения с временной меткой
        QString message = timeString;

        QListWidgetItem* lii = new QListWidgetItem(message);

        // Добавление сообщения в QListWidget
        ui->chats->addItem(lii);
        lii->setTextAlignment(Qt::AlignLeft);
        lii->setFont(QFont("Buhnschrift", 9));

        } else {

```

```

        QMessageBox::warning(this, tr("Ошибка"), tr("Невозможно
загрузить изображение"));
    }
}

```

2 Безопасность жизнедеятельности

На всех этапах жизненного цикла проекта, будь то его разработка, внедрение или эксплуатация, важно обеспечить безопасность человека. Чтобы создать безопасные и комфортные условия работы, необходимо правильно оценить опасные и вредные факторы, связанные с производственной средой, физической и умственной нагрузкой, а также нервно-эмоциональным напряжением [4]. Различные причины могут приводить к схожим изменениям в организме человека, поэтому важно принимать их во внимание и предпринимать соответствующие меры для обеспечения безопасности.

2.1 Потенциально опасные факторы при работе с ПЭВМ

Производственные факторы, которые представляют опасность и вредность, могут быть разделены на следующие группы по их природе: физические, химические, психофизиологические и биологические. При работе с компьютером (ПЭВМ) на программиста(пользователя) могут негативно влиять различные физические факторы, такие как изменение температуры воздуха (повышенная или пониженная), наличие избыточной пыли или газов в воздухе, избыточная или недостаточная влажность, высокий уровень шума, наличие ионизирующего излучения, риск поражения электрическим током, статическое электричество, плохое качество экрана дисплея и недостаточное освещение рабочего места. К химическим опасным факторам, которые могут постоянно воздействовать на программиста, относятся активные частицы, возникающие при ионизации воздуха в процессе работы на компьютере. В данном помещении отсутствуют биологические вредные производственные факторы [4]. К психологическим вредным факторам, которые могут оказывать воздействие на программиста в

					ПГУ 1.090301.35.001 ПЗ	Лист
						50
Изм.	Лист	№ докум.	Подпись	Дата		

течение рабочей смены, можно отнести нервно-эмоциональные перегрузки, умственное напряжение и перенапряжение зрительного анализатора.

2.2 Классификация помещения

Кабинет программиста, относящийся к категории В1-В4 (пожароопасность) I категории (выполняются легкие физические работы), должен соответствовать следующим требованиям: температура воздуха должна быть 22 °С (допустимый диапазон - 20-24 °С), оптимальная относительная влажность составляет 40-60% (допустимый предел - не более 75%), а скорость движения воздуха не должна превышать 0.1 м/с. Для обеспечения и автоматического поддержания оптимальных значений температуры, влажности, чистоты и скорости движения воздуха на рабочем месте, независимо от внешних условий, используется система водяного отопления в холодное время года и кондиционирование воздуха в теплое время года. Кондиционер, как часть вентиляционной системы, автоматически регулирует параметры воздушной среды в помещении. Владелец предприятия должен обеспечить следующие меры: поддержание работоспособности электрооборудования и его эксплуатацию в соответствии с требованиями нормативной технической документации, своевременное и качественное проведение профилактических работ, ремонта, модернизации и реконструкции энергетического оборудования, надежность работы электроустановок и безопасность их обслуживания, предотвращение использования технологий и методов работы, которые могут негативно влиять на окружающую среду, учет и анализ нарушений в работе электроустановок, несчастных случаев и принятие мер для устранения их причин, разработку должностных инструкций и производственных инструкций для электротехнического персонала, а также выполнение предписаний органов Госэнергонадзора.

2.3 Санитарно-гигиеническая характеристика

					ПГУ 1.090301.35.001 ПЗ	Лист
						51
Изм.	Лист	№ докум.	Подпись	Дата		

Ключевыми показателями, которые характеризуют микроклимат, являются следующие: температура воздуха, скорость движения воздуха и относительная влажность. Работа программиста предполагает соблюдение оптимальных требований и норм, включающих следующие значения: оптимальная температура воздуха составляет 22-24 °С, относительная влажность - 40-60%, а скорость движения воздуха должна быть 0,1 м/с. При работе с компьютером программист проводит продолжительное время в статическом положении и часто напрягает глаза. Так как работа за компьютером обычно происходит при искусственном освещении, необходимо обеспечить комфортные условия для сотрудников, чтобы уменьшить нагрузку на глаза. Отсутствие или недостаток освещения в помещении может негативно сказаться на здоровье глаз, снизить работоспособность, ухудшить кровообращение, вызвать сильное напряжение и доставить человеку дискомфорт. Это также может стать причиной сильных головных болей. После долгого рабочего дня возникают чувство усталости и слезоточивость глаз. При работе за компьютером следует соблюдать следующие требования: использовать лампы для освещения рабочего места; сочетать локальное и общее освещение; не позволять прямому и яркому попаданию солнечных лучей на монитор, так как это может повлиять на отображаемую картинку и вызвать искажения; не работать за компьютером в темных помещениях; помещения с компьютерами не должны находиться рядом с помещениями, где уровень шума и вибраций превышает нормальные значения; уровень шума на рабочем месте не должен превышать 50 дБА.

2.4 Расчет освещенности рабочего места

Правильный выбор и расчет освещенности рабочего места обеспечивает создание нормальных условий для зрения обслуживающего персонала, способствует повышению производительности труда. Освещение, соответствующее санитарным нормам, является главнейшим условием гигиены труда и культуры производства. При хорошем освещении устраняется напряжение зрения, ускоряется темп работы.

					ПГУ 1.090301.35.001 ПЗ	Лист
						52
Изм.	Лист	№ докум.	Подпись	Дата		

Для рационального освещения необходимо выполнение следующих условий:

- постоянная освещенность рабочих поверхностей;
- достаточная и равномерно распределенная яркость освещаемых рабочих поверхностей;
- отсутствие резких контрастов между яркостью рабочей поверхностью и окружающего пространства.

Для рационального освещения производственных помещений и рабочих мест большое значение имеет выбор цвета для окраски потолка, стен и производственного оборудования. Рассчитаем люминесцентное освещение кабинета, предназначенного для выполнения работ с размером объекта различения от 0,3 мин до 1 мин. Размеры помещений: A (длина)=4 м, B (ширина)=3 м, H (высота)=3 м. Площадь S помещения определяем по формуле:

$$S=A*B.$$

Подставив значения получим:

$$S=4*3=12 \text{ м}^2.$$

Освещение проектируется при помощи светильников ШОД минимальной освещенностью $E_{\text{мин}}=200$ лк, средней удельной мощностью 17- 23 Вт/м². Высота подвеса над рабочей поверхностью $H_p=2$ м.

Освещение выполняется лампами ЛБ-2*40, световой поток ламп $F_{\text{л}}=2480$ лм, длина 1.2 м, коэффициент запаса равен $k=1.5$. Определим показатель помещения F по формуле:

$$F=A*B/H_p(A+B).$$

Подставив значения получим:

$$F=4*3/2*(4+3)=0.86.$$

Затем, для $F=0,86$, коэффициентов отражения потолка $P_n=0,7$, стен $P_c=0,5$ и расчетной плоскости $P_p=0,3$ находим коэффициент использования светового потока $h=0,37$. формуле:

$$N=E_{\text{мин}}*S*K/F_{\text{л}}*h*Z*n,$$

где $n=2$ общее число ламп в светильнике.

					ПГУ 1.090301.35.001 ПЗ	Лист
						53
Изм.	Лист	№ докум.	Подпись	Дата		

Подставив значения получим:

$$N=200*12*1,5/2480*0,37*0.9*2=2 \text{ штуки.}$$

Общее количество ламп равно $n=2*2=4$.

Таким образом, выполнение перечисленных выше требований к организации рабочего места обеспечит комфортные условия умственного труда/

Заключение

В результате проделанной работы было успешно разработана клиентская часть корпоративного мессенджера, обладающая удобным и интуитивно понятным пользовательским интерфейсом. Данный проект представляет собой корпоративный мессенджер, который включает все необходимые функции и соответствует требованиям и поставленной задаче. Разработка данного приложения позволила значительно улучшить навыки создания клиентских приложений для ОС Windows, освоить среду разработки Qt Creator, а также расширить знания о работе с базами данных, API и интерфейсами.

В дальнейшем можно усовершенствовать разработанный мессенджер и добавить возможность отправки голосовых сообщений, реализацию видеоконференций, улучшить графический интерфейс. Так же можно улучшить окно настроек, добавив возможность редактировать профиль пользователя.

Список использованных источников

1. Корпоративный мессенджер под ОС Android. [Электронный ресурс] / режим доступа: <https://elib.pnzgu.ru/files/eb/doc/MfMjB7EOhVsQ.pdf>, свободный. (дата обращения 24.04.2024).
2. [Электронный ресурс] / режим доступа: <https://metanit.com/cpp/qt/> , свободный. (дата обращения 10.05.2024).
3. Бьерн Страуструп - Язык программирования C++ (дата обращения 28.04.2024).
4. Санитарно-гигиенические рекомендации для организации рабочего места пользователя персональным компьютером. [Электронный ресурс] / Режим доступа: <https://sakhgig.ru/main/info/664-sanitarno-gigienicheskie-rekomendacii-dlyaorganizacii-rabochego-mesta-polzovatelya-personalnym-kompyuterom.html>, свободный (дата обращения 25.05.2024).

Приложение А

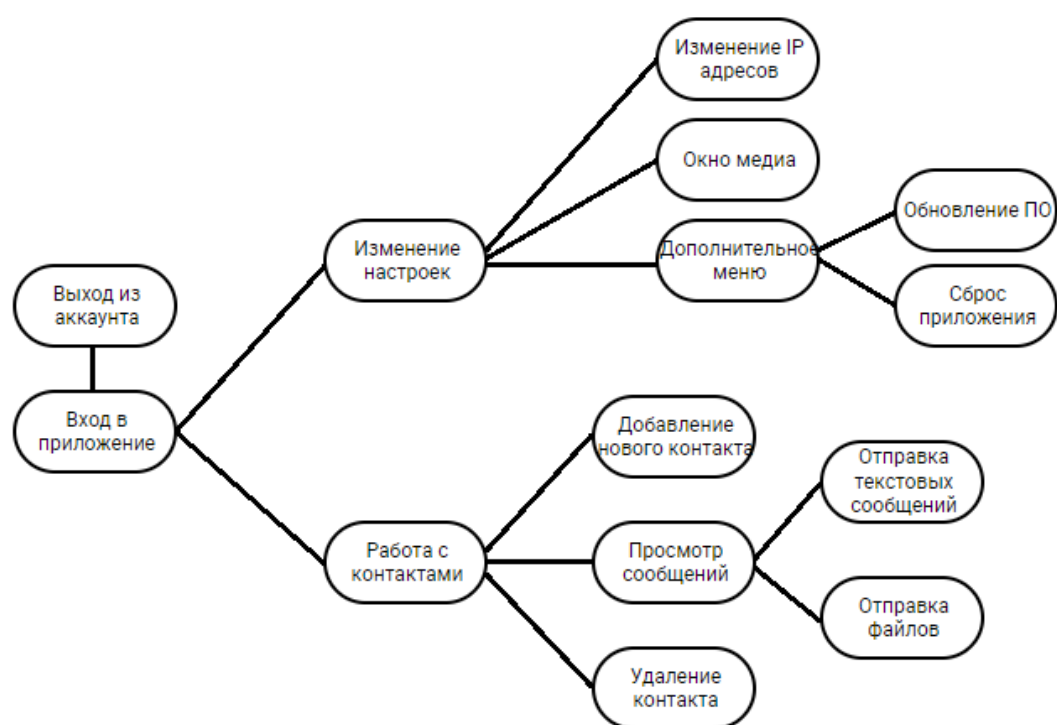
(Обязательное)

Корпоративный десктоп-мессенджер

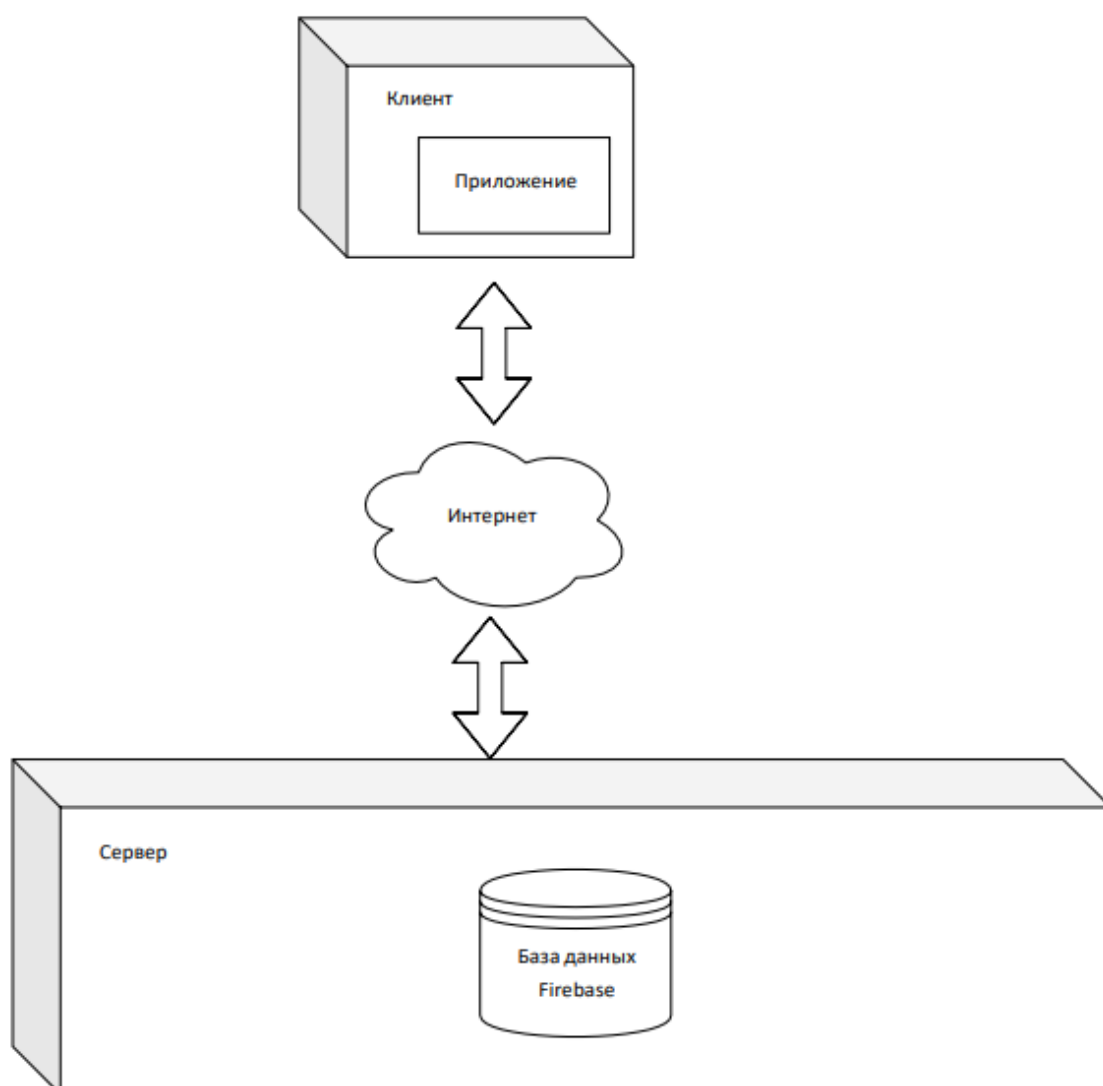
UML-диаграммы

					ПГУ 1.090301.35.001 ПЗ	Лист
						56
Изм.	Лист	№ докум.	Подпись	Дата		

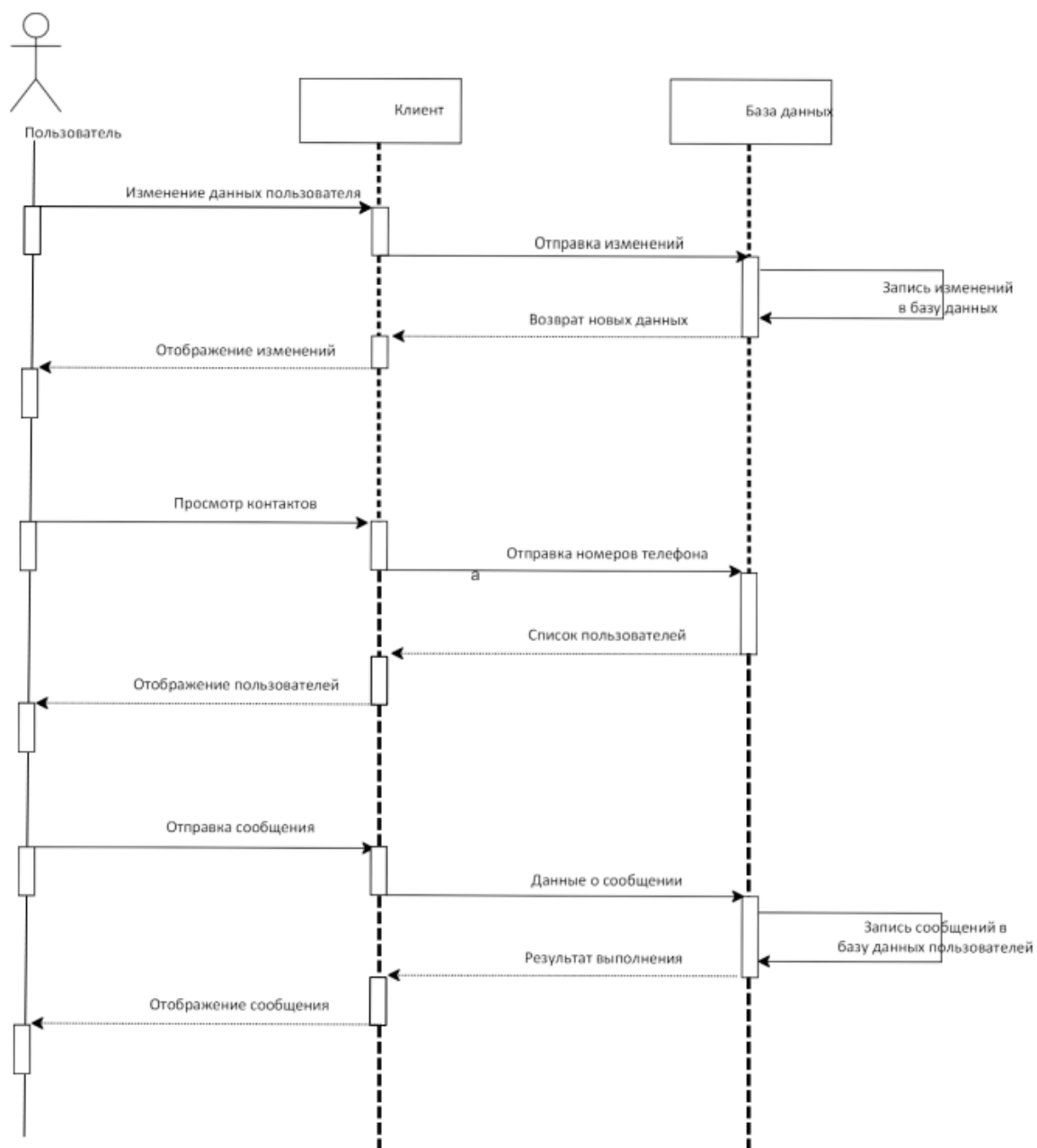
Приложение А.1 – Диаграмма прецедентов



Приложение А.2 – Диаграмма развертывания



Приложение А.3 – Диаграмма последовательности



Приложение Б

(Обязательное)

Корпоративный мессенджер под ОС Windows

Листинг программы

Файл mainwindow.h:

```
#ifndef MAINWINDOW_H
#define MAINWINDOW_H

#include <QtSql>
#include <QSqlDatabase>
#include <QSqlQuery>
#include <QSqlTableModel>
#include <QtNetwork/QNetworkAccessManager>
#include <QtNetwork/QHttpMultiPart>
#include <QtNetwork/QNetworkRequest>
#include <QtNetwork/QNetworkReply>
#include <QUrl>
#include <QFileDialog>
#include <QMessageBox>
#include <QPlainTextEdit>
#include <QFile>
#include <QDebug>
#include <QByteArray>
#include <QString>
#include <QtCore>
#include <QDialog>
#include <QDebug>
#include <QSettings>
#include <QJsonObject>
#include <QJsonDocument>
#include <QJsonArray>
#include <QJsonParseError>
#include <QDialog>

namespace Ui {
class MainWindow;
}

class MainWindow : public QDialog
```

```

{
    Q_OBJECT

    QJsonDocument doc;
    QString file_name;

public:
    explicit MainWindow(QWidget *parent = 0);
    ~MainWindow();
    void readFile();

private slots:

    void on_reg_clicked();

    void on_vhod_clicked();

    void on_add_clicked();

private:
    Ui::MainWindow *ui;
    QSqlDatabase db;

};

#endif // MAINWINDOW_H

```

Файлmainwindow.cpp:

```
#include "mainwindow.h"
#include "ui_mainwindow.h"
#include "secondwindow.h"
#include "triwindow.h"
#include "crc32.h"
#include "settingsapp.h"
#include "examplethreads.h"
#include "add_contact.h"
#include <QDialog>
#include <QString>
#include <QFile>
#include <QFileDialog>
#include <QMessageBox>
#include "dop_okno.h"
```

```
MainWindow::MainWindow(QWidget *parent) :
```

```
    QDialog(parent),
    ui(new Ui::MainWindow)
{
    ui->setupUi(this);
}
```

```
MainWindow::~MainWindow()
```

```
{
    delete ui;
}
```

					ПГУ 1.090301.35.001 ПЗ	Лист
						63
Изм.	Лист	№ докум.	Подпись	Дата		

```

void MainWindow::on_reg_clicked()
{
    const QString ID = "id";
    const QString SIG = "signature";
    const QString PHONE = "phone";
    const QString SKZI = "ip_skzi";
    const QString MON = "ip_mon";
    const QString ATS = "ip_ats";
    const QString DNS = "ip_dns";
    const QString KS = "ks";

    QString filter = "All File (*.*) ;; Text File (*.txt) ;; CFG File (*.CFG)";

    QString file_name = QFileDialog::getOpenFileName(this, "open a file", "D:/Peday", filter);
    QFile file(file_name);

    file.open(QFile::ReadOnly | QFile::Text);
    QTextStream in(&file);
    QString text = in.readAll();
    file.close();

    QJsonDocument doc = QJsonDocument::fromJson(text.toUtf8());
    QJsonObject json = doc.object();

    QJsonValue jsonId = json.value(ID);
    QString id = jsonId.toString();

    QJsonValue jsonSig = json.value(SIG);
    QString sig = jsonSig.toString();

    QJsonValue jsonPhone = json.value(PHONE);

```

```

QString phone = jsonPhone.toString();

QJsonValue jsonSkzi = json.value(SKZI);
QString skzi = jsonSkzi.toString();

QJsonValue jsonMon = json.value(MON);
QString mon = jsonMon.toString();

QJsonValue jsonAts = json.value(ATS);
QString ats = jsonAts.toString();

QJsonValue jsonDns = json.value(DNS);
QString dns = jsonDns.toString();

QJsonValue jsonKs = json.value(KS);
QString ks = jsonKs.toString();

QString proverka = id + sig + phone + skzi + mon + ats + dns;

SettingsApp::ins()->setID(id);
SettingsApp::ins()->setSignature(sig);
SettingsApp::ins()->setPhone(phone);
SettingsApp::ins()->setIpAts(ats);
SettingsApp::ins()->setIpDns(dns);

qDebug() << "Settings get phone = " << SettingsApp::ins()->getPhone();

qDebug() << "Settings get ip_ats = " << SettingsApp::ins()->getIpAts();

qDebug() << "Settings get ip_dns = " << SettingsApp::ins()->getIpDns();

int crc = crc32((unsigned char*)proverka.toUtf8().data(),

```

```

(unsigned int) proverka.toUtf8().length());

    if (ks.toInt(0, 16) == crc) {
        ui->Nomer_edit->setText(phone);
        ui->ATC_edit->setText(ats);
        ui->DNS_edit->setText(dns);

    } else {
        qDebug() << "сообщение об ошибке";
    }
}

void MainWindow::on_vhod_clicked()
{
    if(ui->Nomer_edit->text().isEmpty() || ui->ATC_edit->
    >text().isEmpty() || ui->DNS_edit->text().isEmpty())
    {

        Triwindow window;
        window.setModal(true);
        window.exec();

    }else{
        SettingsApp::ins()->setInitialization();

        this->close();
        SecondWindow window;
        window.setModal(true);
        window.exec();
    }
}

```

Файл settingsapp.h:

```
#ifndef SETTINGSAPP_H
#define SETTINGSAPP_H

#include <QSettings>

class SettingsApp
{
public:
    static SettingsApp* ins();
    QString getID();
    QString getPhone();
    QString getSignature();
    QString getIpSkzi();
    QString getIpAts();
    QString getIpDns();
    int getInitialization();

    void setID(QString id);
    void setPhone(QString phone);
    void setSignature(QString sig);
    void setIpSkzi(QString ip_skzi);
    void setIpAts(QString ip_ats);
    void setIpDns(QString ip_dns);
    void setInitialization();

private:
    SettingsApp();
    static SettingsApp* instance;
    QSettings* mSettings;
};

#endif // SETTINGSAPP_H
```

					ПГУ 1.090301.35.001 ПЗ	Лист
						67
Изм.	Лист	№ докум.	Подпись	Дата		

Файл settingsapp.cpp:

```
#include "settingsapp.h"
#include "secondwindow.h"
#include "add_contact.h"

SettingsApp* SettingsApp::instance;

SettingsApp::SettingsApp()
{
    mSettings = new QSettings("Hello", "Opp");
}

SettingsApp* SettingsApp::ins()
{
    if ( !instance ) {
        instance = new SettingsApp();
    }
    return instance;
}

void SettingsApp::setPhone(QString phone) {
    mSettings->setValue( "phone", phone );
}

QString SettingsApp::getPhone() {
    return mSettings->value("phone", 0).toString();
}
```

					ПГУ 1.090301.35.001 ПЗ	Лист
						68
Изм.	Лист	№ докум.	Подпись	Дата		

```

void SettingsApp::setID(QString id){
    mSettings->setValue( "id", id );
}

QString SettingsApp::getID() {
    return mSettings->value("id", 0).toString();
}

void SettingsApp::setSignature(QString signature){
    mSettings->setValue( "signature", signature );
}

QString SettingsApp::getSignature() {
    return mSettings->value("signature", 0).toString();
}

void SettingsApp::setIpAts(QString ip_ats){
    mSettings->setValue( "ip_ats", ip_ats );
}

QString SettingsApp::getIpAts() {
    return mSettings->value("ip_ats", 0).toString();
}

void SettingsApp::setIpDns(QString ip_dns){
    mSettings->setValue( "ip_dns", ip_dns );
}

void SettingsApp::setInitialization()
{
    mSettings->setValue("init", 1);
}

```

```
QString SettingsApp::getIpDns() {
    return mSettings->value("ip_dns", 0).toString();
}
```

```
int SettingsApp::getInitialization()
{
    return mSettings->value("init", 0).toInt();
}
```

Файл examplethreads.h:

```
#include "settingsapp.h"
#include "secondwindow.h"
#include "add_contact.h"

SettingsApp* SettingsApp::instance;
SettingsApp::SettingsApp()
{
    mSettings = new QSettings("Hello", "Opp");
}

SettingsApp* SettingsApp::ins()
{
    if ( !instance ) {
        instance = new SettingsApp();
    }
    return instance;
}

void SettingsApp::setPhone(QString phone) {
    mSettings->setValue( "phone", phone );
}
```

					ПГУ 1.090301.35.001 ПЗ	Лист
						70
Изм.	Лист	№ докум.	Подпись	Дата		

```

QString SettingsApp::getPhone() {
    return mSettings->value("phone", 0).toString();
}

void SettingsApp::setID(QString id){
    mSettings->setValue( "id", id );
}

QString SettingsApp::getID() {
    return mSettings->value("id", 0).toString();
}

void SettingsApp::setSignature(QString signature){
    mSettings->setValue( "signature", signature );
}

QString SettingsApp::getSignature() {
    return mSettings->value("signature", 0).toString();
}

void SettingsApp::setIpAts(QString ip_ats){
    mSettings->setValue( "ip_ats", ip_ats );
}

QString SettingsApp::getIpAts() {
    return mSettings->value("ip_ats", 0).toString();
}

void SettingsApp::setIpDns(QString ip_dns){
    mSettings->setValue( "ip_dns", ip_dns );
}

void SettingsApp::setInitialization()
{
    mSettings->setValue("init", 1);
}

```

```

}

QString SettingsApp::getIpDns() {
    return mSettings->value("ip_dns", 0).toString();
}

int SettingsApp::getInitialization()
{
    return mSettings->value("init", 0).toInt();
}

```

Файл examplethreads.cpp:

```

#include "examplethreads.h"
#include <QDebug>
#include "mainwindow.h"
#include "secondwindow.h"
#include "settingsapp.h"
#include "glav.h"
#include "Controller.h"
#include "add_contact.h"
#include "spo_db.h"
#include "secondwindow.h"

ExampleThreads::ExampleThreads(QString threadName) :
    name(threadName)
{

}

void ExampleThreads::run()
{
    //spo_save_contact(&contact);
}

```

					ПГУ 1.090301.35.001 ПЗ	Лист
						72
Изм.	Лист	№ докум.	Подпись	Дата		

```

spo_set_database_path("BazaDB.db");

sleep(1);

emit sendEndIn(SettingsApp::ins()->getInitialization());
}

```

Файл crc32.h:

```

#ifndef __CRC32_H
#define __CRC32_H

unsigned int crc32(unsigned char *buf, unsigned int len);

unsigned int KS_MOD32(unsigned short *pBuffer, unsigned int len,
unsigned int T, unsigned char isEnd);

#endif

```

Файл crc32.cpp:

```

#include "crc32.h"

static const unsigned int crc32table[256] = { 0x00000000, 0x77073096,
0xEE0E612C,
0x990951BA, 0x076DC419, 0x706AF48F, 0xE963A535, 0x9E6495A3,
0x0EDB8832,
0x79DCB8A4, 0xE0D5E91E, 0x97D2D988, 0x09B64C2B, 0x7EB17CBD,
0xE7B82D07,
0x90BF1D91, 0x1DB71064, 0x6AB020F2, 0xF3B97148, 0x84BE41DE,
0x1ADAD47D,
0x6DDDE4EB, 0xF4D4B551, 0x83D385C7, 0x136C9856, 0x646BA8C0,
0xFD62F97A,
0x8A65C9EC, 0x14015C4F, 0x63066CD9, 0xFA0F3D63, 0x8D080DF5,
0x3B6E20C8,
0x4C69105E, 0xD56041E4, 0xA2677172, 0x3C03E4D1, 0x4B04D447,
0xD20D85FD,
0xA50AB56B, 0x35B5A8FA, 0x42B2986C, 0xDBBBC9D6, 0xACBCF940,
0x32D86CE3,
0x45DF5C75, 0xDCD60DCF, 0xABD13D59, 0x26D930AC, 0x51DE003A,
0xC8D75180,
0xBFDD06116, 0x21B4F4B5, 0x56B3C423, 0xCFBA9599, 0xB8BDA50F,

```

					ПГУ 1.090301.35.001 ПЗ	Лист
						73
Изм.	Лист	№ докум.	Подпись	Дата		

0x2802B89E,
 0x5F058808, 0xC60CD9B2, 0xB10BE924, 0x2F6F7C87, 0x58684C11,
 0xC1611DAB,
 0xB6662D3D, 0x76DC4190, 0x01DB7106, 0x98D220BC, 0xEFD5102A,
 0x71B18589,
 0x06B6B51F, 0x9FBFE4A5, 0xE8B8D433, 0x7807C9A2, 0x0F00F934,
 0x9609A88E,
 0xE10E9818, 0x7F6A0DBB, 0x086D3D2D, 0x91646C97, 0xE6635C01,
 0x6B6B51F4,
 0x1C6C6162, 0x856530D8, 0xF262004E, 0x6C0695ED, 0x1B01A57B,
 0x8208F4C1,
 0xF50FC457, 0x65B0D9C6, 0x12B7E950, 0x8BBEB8EA, 0xFCB9887C,
 0x62DD1DDF,
 0x15DA2D49, 0x8CD37CF3, 0xFBD44C65, 0x4DB26158, 0x3AB551CE,
 0xA3BC0074,
 0xD4BB30E2, 0x4ADFA541, 0x3DD895D7, 0xA4D1C46D, 0xD3D6F4FB,
 0x4369E96A,
 0x346ED9FC, 0xAD678846, 0xDA60B8D0, 0x44042D73, 0x33031DE5,
 0xAA0A4C5F,
 0xDD0D7CC9, 0x5005713C, 0x270241AA, 0xBE0B1010, 0xC90C2086,
 0x5768B525,
 0x206F85B3, 0xB966D409, 0xCE61E49F, 0x5EDEF90E, 0x29D9C998,
 0xB0D09822,
 0xC7D7A8B4, 0x59B33D17, 0x2EB40D81, 0xB7BD5C3B, 0xC0BA6CAD,
 0xEDB88320,
 0x9ABFB3B6, 0x03B6E20C, 0x74B1D29A, 0xEAD54739, 0x9DD277AF,
 0x04DB2615,
 0x73DC1683, 0xE3630B12, 0x94643B84, 0x0D6D6A3E, 0x7A6A5AA8,
 0xE40ECF0B,
 0x9309FF9D, 0x0A00AE27, 0x7D079EB1, 0xF00F9344, 0x8708A3D2,
 0x1E01F268,
 0x6906C2FE, 0xF762575D, 0x806567CB, 0x196C3671, 0x6E6B06E7,
 0xFED41B76,
 0x89D32BE0, 0x10DA7A5A, 0x67DD4ACC, 0xF9B9DF6F, 0x8EBEEFF9,
 0x17B7BE43,
 0x60B08ED5, 0xD6D6A3E8, 0xA1D1937E, 0x38D8C2C4, 0x4FDDFF252,
 0xD1BB67F1,
 0xA6BC5767, 0x3FB506DD, 0x48B2364B, 0xD80D2BDA, 0xAF0A1B4C,

					ПГУ 1.090301.35.001 ПЗ	Лист
						74
Изм.	Лист	№ докум.	Подпись	Дата		

```

0x36034AF6,
    0x41047A60, 0xDF60EFC3, 0xA867DF55, 0x316E8EEF, 0x4669BE79,
0xCB61B38C,
    0xBC66831A, 0x256FD2A0, 0x5268E236, 0xCC0C7795, 0xBB0B4703,
0x220216B9,
    0x5505262F, 0xC5BA3BBE, 0xB2BD0B28, 0x2BB45A92, 0x5CB36A04,
0xC2D7FFA7,
    0xB5D0CF31, 0x2CD99E8B, 0x5BDEAE1D, 0x9B64C2B0, 0xEC63F226,
0x756AA39C,
    0x026D930A, 0x9C0906A9, 0xEB0E363F, 0x72076785, 0x05005713,
0x95BF4A82,
    0xE2B87A14, 0x7BB12BAE, 0x0CB61B38, 0x92D28E9B, 0xE5D5BE0D,
0x7CDCEFB7,
    0x0BDBDF21, 0x86D3D2D4, 0xF1D4E242, 0x68DD3BF8, 0x1FDA836E,
0x81BE16CD,
    0xF6B9265B, 0x6FB077E1, 0x18B74777, 0x88085AE6, 0xFF0F6A70,
0x66063BCA,
    0x11010B5C, 0x8F659EFF, 0xF862AE69, 0x616BFFD3, 0x166CCF45,
0xA00AE278,
    0xD70DD2EE, 0x4E048354, 0x3903B3C2, 0xA7672661, 0xD06016F7,
0x4969474D,
    0x3E6E77DB, 0xAED16A4A, 0xD9D65ADC, 0x40DF0B66, 0x37D83BF0,
0xA9BCAE53,
    0xDEBB9EC5, 0x47B2CF7F, 0x30B5FFE9, 0xBDBDF21C, 0xCABAC28A,
0x53B39330,
    0x24B4A3A6, 0xBAD03605, 0xCDD70693, 0x54DE5729, 0x23D967BF,
0xB3667A2E,
    0xC4614AB8, 0x5D681B02, 0x2A6F2B94, 0xB40BBE37, 0xC30C8EA1,
0x5A05DF1B,
    0x2D02EF8D };

```

```

unsigned int crc32(unsigned char *buf, unsigned int len) {
    unsigned int crc;
    crc = 0xFFFFFFFF;
    while (len--) {
        crc = crc32table[(crc ^ (unsigned char) * buf++) & 0x000000ff] ^
(crc >> 8);
    }
}

```

					ПГУ 1.090301.35.001 ПЗ	Лист
						75
Изм.	Лист	№ докум.	Подпись	Дата		

```

        return crc ^ 0xFFFFFFFF;
    }

    unsigned int KS_MOD32(unsigned short *pBuffer, unsigned int len,
        unsigned int T, unsigned char isEnd) {

        unsigned int t, sz = len;

        t = T;
        while (sz--)
            t = (t * 3) + (*pBuffer++);

        if (isEnd)
            return (t ^ 0xB432E6A8);
        else
            return t;
    }

```

Файл glav.h:

```

#ifndef GLAV_H
#define GLAV_H

#include <QDialog>
#include <QThread>
#include "QMainWindow"

namespace Ui {
class glav;
}

class glav : public QMainWindow
{
    Q_OBJECT

```

```

public:
    explicit glav(QWidget *parent = 0);
    ~glav();

public slots:
    void endIn(int isInit);

private:
    Ui::glav *ui;

};

#endif // GLAV_H

```

Файл glav.cpp:

```

#include "glav.h"
#include <QMovie>
#include <QDebug>
#include <QThread>
#include "settingsapp.h"
#include "examplethreads.h"
#include "secondwindow.h"
#include "mainmainwindow.h"
#include "ui_glav.h"
#include <cstring>
#include <time.h>
#include <stdio.h>
#include "spo_db.h"
#include "spmemvfs.h"
// #include <rillog.h>
#include <errno.h>
// #include <bthread.h>

```

					ПГУ 1.090301.35.001 ПЗ	Лист
						77
Изм.	Лист	№ докум.	Подпись	Дата		

```

#include "chat_file_table.h"
#include "chat_message_table.h"
#include "chat_room_table.h"
#include "contact_table.h"
#include "settings_table.h"
#include "spo_db_context.h"
#include "spo_db.h"

glav::glav(QWidget *parent) :
    QMainWindow(parent),
    ui(new Ui::glav)
{
    ui->setupUi(this);

    QMovie* movie = new QMovie("D:/Peday/loading1.gif"); // Load GIF

    if (!movie->isValid()) // Проверка GIF
    {
        qDebug() << "сообщение об ошибке";
    }

    // Play GIF
    QLabel* label = new QLabel(this);
    label->setGeometry(426,375,128,128);

    label->setMovie(movie);
    movie->start();
}

```

```

void glav::endIn(int isInit)
{
    if ( isInit == 0 ) {
        // Если пользователь уже вошел в систему, открываем основной
интерфейс
        this->close();
        MainWindow wew;
        wew.setModal(true);
        wew.exec();

    } else {
        // Если пользователь еще не вошел в систему, открываем окно
для входа
        this->close();
        SecondWindow sw;
        sw.setModal(true);
        sw.exec();
    }
}

glav::~~glav()
{
    delete ui;
}

```

Файл secondwindow.h:

```

#ifndef SECONDWINDOW_H
#define SECONDWINDOW_H

#include "res_message_thread.h"
#include "add_contact.h"
#include "change_contact.h"

```

```

#include <QDialog>
#include <QtSql>
#include "QLabel"
#include <QSqlDatabase>
#include <QSqlQuery>
#include <QSqlTableModel>
#include <QtNetwork/QNetworkAccessManager>
#include <QtNetwork/QHttpMultiPart>
#include <QtNetwork/QNetworkRequest>
#include <QtNetwork/QNetworkReply>
#include <QJsonArray>
#include <QJsonObject>
#include <QJsonValue>
#include <QListWidgetItem>
#include "QRegExp"

namespace Ui {
class SecondWindow;
}

class SecondWindow : public QDialog
{
    Q_OBJECT

public:
    explicit SecondWindow(QWidget *parent = 0);
    ~SecondWindow();
    QString text();
    QString open_label();
    QString who;

private slots:

```

```

void on_pushButton_ok_clicked();
void on_pushButton_get_clicked();
void on_pushButton_exit_clicked();
void func(QNetworkReply* reply);
void func2(QNetworkReply* reply);
void func3(QNetworkReply* reply);
//void func26(QNetworkReply* reply);
void on_add_contact_clicked();
void on_del_contact_clicked();
//void vibor_contacta_func();
void timerEvent();
void on_del_contact_2_clicked();
void on_del_contact_3_clicked();
void on_pushButton_exit_2_clicked();
void on_listWidget_itemClicked(QListWidgetItem *item);
void on_change_contact_button_clicked();

```

private:

```

QSqlDatabase db;
QSqlTableModel *model;
QString str2();
int numContacts();
QNetworkAccessManager *manager;
QNetworkRequest request;
std::list<void*> spisok;
Ui::SecondWindow *ui;
QString textMessage;
QString nomera_receivers;
QString idd;
QString idd1;
QString datee;
res_message_thread *SecondThread;

```

```

    QString dude;

    QTime m_time;

    QTimer m_timer;

    QElapsedTimer m_eTimer;

};

#endif // SECONDWINDOW_H

```

Файл secondwindow.cpp:

```

#include "secondwindow.h"
#include "ui_secondwindow.h"
#include "triwindow.h"
#include "mainwindow.h"
#include "glav.h"
#include "settingsapp.h"
#include "contact_table.h"
#include "spo_db.h"
#include "contact_table.h"
#include "add_contact.h"
#include "menu_settings.h"
#include "del_contact.h"
#include "dop_okno.h"
#include "res_message_thread.h"
#include "update_window.h"
#include "change_contact.h"

#include <list>
#include "QtSql/QtSqlDatabase"
#include "QtSql/QtSqlQuery"
#include <QDebug>
#include "QLabel"
#include <QTableWidget>
#include <QString>

```

```

#include <QFile>
#include <QFileDialog>
#include <QMessageBox>
#include <QNetworkAccessManager>
#include <QtSql>
#include <QSqlDatabase>
#include <QSqlQuery>
#include <QSqlTableModel>
#include <QtNetwork/QNetworkAccessManager>
#include <QtNetwork/QHttpMultiPart>
#include <QtNetwork/QNetworkRequest>
#include <QtNetwork/QNetworkReply>
#include <QUrl>
#include <QFileDialog>
#include <QMessageBox>
#include <QPlainTextEdit>
#include <QByteArray>
#include <QtCore>
#include <QDialog>
#include <QSettings>
#include <QLabel>
#include <QJsonObject>
#include <QJsonDocument>
#include <QJsonArray>
#include <QJsonValue>
#include <QJsonParseError>

void SecondWindow::func(QNetworkReply* reply){ //ответ сервера на
синхронизацию контактов

/*

    if (reply->error()) {

        qDebug() << reply->errorString();

```

```

    } else {
        QString answer = reply->readAll();
        qDebug() << answer;
    }
*/

//QString answer = reply->readAll();

QString answer =
"{\"listOfContacts\": [{\"id\":32,\"name\": \"Анатолий
Дмитриевич\", \"phone\": \"721\"}, {\"id\":34,\"name\": \"Максим
Александрович\", \"phone\": \"363\"}]}";

qDebug() << answer;

QJsonDocument doc = QJsonDocument::fromJson(answer.toUtf8());
// Ищем данные по ключу
QJsonValue value = doc.object().value("listOfContacts");
// Если value это массив данных
if ( value.type() == QJsonValue::Array ) {
    // Получаем массив
    QJsonArray array = value.toArray();
    spo_delete_all_contact();

    for( int i = 0; i < array.size(); i++ ) {
        QJsonValueRef ref = array[i];
        // Если ValueRef это объект
        if ( ref.type() == QJsonValue::Object ) {
            QJsonObject json = ref.toObject();

            QJsonValue jsonId = json.value("id");
            QString id = QString::number(jsonId.toInt());

            QJsonValue jsonName = json.value("name");
            QString name = jsonName.toString();

            QJsonValue jsonPhone = json.value("phone");

```

```

        QString phone = jsonPhone.toString();

        SpoContact contact;
        QByteArray ba = id.toUtf8();
        const char *c_str2 = ba.data();
        contact.id_user = (char *)c_str2;
        qDebug() << "id_user - " << c_str2;

        QByteArray ba1 = name.toUtf8();
        const char *c_str3 = ba1.data();
        contact.full_name = (char *)c_str3;
        qDebug() << "name - " << c_str3;

        QByteArray ba2 = phone.toUtf8();
        const char *c_str4 = ba2.data();
        contact.sip_number = (char *)c_str4;
        qDebug() << "sip_number - " << c_str4;

        contact.uri_photo = NULL;

        ui->listWidget->addItem(name); // отображение
КОНТАКТОВ

    }

}

}

}

SpoContact *del_contact;

void SecondWindow::func2(QNetworkReply* reply){ //ответ сервера на
удаление контакта

```

```

/*
    if (reply->error()) {
        qDebug() << reply->errorString();
    } else {
*/

        QString answer2 = "{\"error\": 200}";
        // QString answer2 = reply->readAll();
        qDebug() << answer2;
        spo_delete_contact(del_contact);

    }

void SecondWindow::func3(QNetworkReply* reply){ //ответ сервера на
отправку сообщения

    if (reply->error()) {
        qDebug() << reply->errorString();
    } else {

        QString answeerrr = "{\"id_msg\": 2, \"error\": \"200\"}";
        qDebug() << answeerrr;
        qDebug() << dude;

        SpoChatMessage messages;
        messages.id = NULL;
        messages.dir = 0;
        messages.is_read = NULL;
        messages.type_content = NULL;
        messages.status = 1;
        messages.id_file = 0;

        QByteArray ab = idd.toUtf8();

```

```

const char *c_str123 = ab.data();
messages.id_user = (char *)c_str123;

QByteArray aa = textMessage.toUtf8();
const char *c_str121 = aa.data();
messages.message = (char *)c_str121;
qDebug() << "message - " << c_str121;

messages.id_chat_room = NULL;
messages.date = NULL;

messages.date_send = NULL;

messages.id_ozu_file = NULL;

spo_save_chat_message(&messages);
//реализовать многострочную отправку сообщений
//изменить иконки отправки, настроек.
}

}

void SecondWindow::timerEvent()
{
    /*
    QDateTime datetime =
    QDateTime::fromMsecsSinceEpoch(m_eTimer.elapsed());
    int days = QDateTime::fromMsecsSinceEpoch(0).daysTo(datetime);
    if (days) {
        ui->label_date->setText(QString("%1d ").arg(days) +
m_time.toString("hh:mm:ss"));
    }
    */
}

```

```

        }else{
            ui->label_date->setText(m_time.toString("hh:mm:ss"));
        }
    */
}

SecondWindow::SecondWindow(QWidget *parent) :
    QDialog(parent),
    ui(new Ui::SecondWindow)
{
    ui->setupUi(this);

    manager = new QNetworkAccessManager(this); //создание менеджера

    connect(manager, &QNetworkAccessManager::finished, this,
    &SecondWindow::func);

    SecondThread = new res_message_thread("Hello");

    SecondThread->start();

    spisok = spo_get_contacts();

    for (auto iter = spisok.begin(); iter != spisok.end(); iter++)
    {
        ui->listWidget->addItem(((SpoContact *)*iter)-
        >full_name));
    }

    ui->label_9->setText(((SpoContact *)*spisok.begin())-
    >full_name));

    ui->label_id->setText(SettingsApp::ins()->getID());

```

```

        who = (((SpoContact *)*spisok.begin())->full_name);

    }

SecondWindow::~SecondWindow()
{
    delete ui;
}

void SecondWindow::on_pushButton_ok_clicked() //отправка сообщения по
кнопке OK
{
    manager = new QNetworkAccessManager(this); //создание менеджера
    connect(manager, &QNetworkAccessManager::finished, this,
    &SecondWindow::func3);

    QUrl url ("http://message.modelshik.ru/api/sendMessage.php");
    QNetworkRequest request(url); //our server with php-script
    request.setRawHeader("Accept", "*/*");
    request.setRawHeader("Cache-Control", "no-cache");
    request.setRawHeader("Content-Type",
    "application/json; charset=utf-8");

    QJsonObject jb;
    jb.insert("id", QJsonValue::fromVariant(SettingsApp::ins()-
    >getID()));
    jb.insert("signature", QJsonValue::fromVariant(SettingsApp::ins()-
    >getSignature()));
    textMessage = ui->lineEdit->text();
    idd = "3";
    jb.insert("message", QJsonValue::fromVariant(textMessage));
    jb.insert("id_user", QJsonValue::fromVariant(idd));

```

					ПГУ 1.090301.35.001 ПЗ	Лист
						89
Изм.	Лист	№ докум.	Подпись	Дата		

```

        jb.insert("date_send", QJsonValue::fromVariant(dude));
        QJsonDocument doc(jb);
        QNetworkReply *reply = manager->post(request, doc.toJson());

        QListWidgetItem* li = new QListWidgetItem("\n"+ ui->lineEdit-
>text() );
        ui->chats->addItem(li);
        li->setTextAlignment(Qt::AlignRight);
        QTime currentTime = QTime::currentTime();

        // Форматирование времени в строку
        QString timeString = currentTime.toString("hh:mm");

        // Создание сообщения с временной меткой
        QString message = timeString;

        QListWidgetItem* lo = new QListWidgetItem(message);

        // Добавление сообщения в QListWidget
        ui->chats->addItem(lo );
        lo->setTextAlignment(Qt::AlignRight);
        lo->setFont(QFont("Buhnschrift", 9));

        ui->lineEdit->clear();
    }

void SecondWindow::on_pushButton_get_clicked() //синхронизация списка
контактов
{
    QUrl url
("http://message.modelshik.ru/api/getListOfContacts.php");
    QNetworkRequest request(url); //our server with php-script
    request.setRawHeader("Accept", "*/*");
    request.setRawHeader("Cache-Control", "no-cache");

```

```

        request.setRawHeader("Content-Type",
"application/json;charset=utf-8");

        QJsonObject jб;

        jб.insert("id", QJsonValue::fromVariant(SettingsApp::ins()-
>getID()));

        jб.insert("signature", QJsonValue::fromVariant(SettingsApp::ins()-
>getSignature()));

        QJsonDocument doc(jб);

        QNetworkReply *reply = manager->post(request, doc.toJson());

        ui->listWidget->clear();
    }

void SecondWindow::on_pushButton_exit_clicked() //выход из приложения
{
    this->close();

    MainWindow vv;

    vv.setModal(true);

    vv.exec();
}

void SecondWindow::on_add_contact_clicked() //добавление контакта
{
    this->close();

    Add_contact ggg;

    ggg.setModal(true);

    ggg.exec();
}

void SecondWindow::on_del_contact_clicked() //удаление контакта
{
    Del_contact ii;

    ii.setModal(true);

    ii.exec()

    manager = new QNetworkAccessManager(this);

    connect(manager, &QNetworkAccessManager::finished, this,

```

```

&SecondWindow::func2);

    //SpoContact*select_contact = spisok.get(ui->listWidget-
>currentRow());

    QListWidgetItem *it = ui->listWidget->item(ui->listWidget-
>currentRow());

    for (auto iter = spisok.begin(); iter != spisok.end(); iter++)
    {
        if (it->text() == (((SpoContact *)*iter)->full_name))
        {
            spo_delete_contact((SpoContact *)*iter); //при
успешном ответе с сервера

            QUrl url
("http://message.modelshik.ru/api/dropContact.php");

            QNetworkRequest request(url); //our server with php-
script

            request.setRawHeader("Accept", "*/*");
            request.setRawHeader("Cache-Control", "no-cache");
            request.setRawHeader("Content-Type",
"application/json;charset=utf-8");

            del_contact = (SpoContact *)*iter;

            QJsonObject jб;
            jб.insert("id",
QJsonValue::fromVariant(SettingsApp::ins()->getID()));
            jб.insert("signature",
QJsonValue::fromVariant(SettingsApp::ins()->getSignature()));
            jб.insert("id_user",
QJsonValue::fromVariant(((SpoContact *)*iter)->full_name));

            QJsonDocument doc(jб);

            QNetworkReply *reply = manager->post(request,
doc.toJson());

```

```

        break;
    }

}

delete(it);

}

void SecondWindow::on_del_contact_2_clicked() // меню настроек
(открытие)
{
    this->close();
    menu_settings au;
    au.setModal(true);
    au.exec();
}

void SecondWindow::on_del_contact_3_clicked()
{

}

void SecondWindow::on_pushButton_exit_2_clicked() //Загрузка файлов
{
    QString fileName = QFileDialog::getOpenFileName(this, tr("Выберите
файл"), "", tr("Изображения (*.png *.jpg *.jpeg)"));
    if (!fileName.isEmpty()) {
        QPixmap pixmap(fileName);
        if (!pixmap.isNull()) {
            QWidget* itemWidget = new QWidget();
            QVBoxLayout* itemLayout = new QVBoxLayout(itemWidget);

            QLabel* imageLabel = new QLabel();

```

```

        imageLabel->setPixmap (pixmap);
        itemLayout->addWidget (imageLabel);
        itemLayout->setAlignment (Qt::AlignRight);

        QLabel* textLabel = new
QLabel (QFileInfo (fileName).fileName());
        itemLayout->addWidget (textLabel);
        textLabel->setAlignment (Qt::AlignCenter);
        QListWidgetItem* item = new QListWidgetItem();
        item->setSizeHint (itemWidget->sizeHint());
        ui->chats->addItem (item);
        ui->chats->setItemWidget (item, itemWidget);

        QTime currentTime = QTime::currentTime();

        // Форматирование времени в строку
        QString timeString = currentTime.toString("hh:mm");

        // Создание сообщения с временной меткой
        QString message = timeString;

        QListWidgetItem* lii = new QListWidgetItem(message);

        // Добавление сообщения в QListWidget
        ui->chats->addItem(lii);
        lii->setTextAlignment (Qt::AlignRight);
        lii->setFont (QFont ("Buhnschrift", 9));

    } else {

        QMessageBox::warning(this, tr("Ошибка"), tr("Невозможно
загрузить изображение"));
    }

```

```

    }
}

void SecondWindow::on_listWidget_itemClicked(QListWidgetItem *item)
{
    ui->label_9->setText(item->text());
    //QString idd = contact.id_user;
}

void SecondWindow::on_change_contact_button_clicked()
{
    Change_contact wi;
    wi.setModal(true);
    wi.exec();
}

```

Файл addcontact.h:

```

#ifndef ADD_CONTACT_H
#define ADD_CONTACT_H

#include "secondwindow.h"
#include "spo_db.h"
#include "contact_table.h"
#include <QDialog>
#include <QtSql>
#include <QSqlDatabase>
#include <QSqlQuery>
#include <QSqlTableModel>
#include <QtNetwork/QNetworkAccessManager>
#include <QtNetwork/QHttpMultiPart>
#include <QtNetwork/QNetworkRequest>
#include <QtNetwork/QNetworkReply>
#include <QJsonArray>
#include <QJsonObject>
#include <QJsonValue>

```

```

#include <QListWidgetItem>

namespace Ui {
class Add_contact;
}

class Add_contact : public QDialog
{
    Q_OBJECT

public:
    explicit Add_contact(QWidget *parent = 0);
    ~Add_contact();

private slots:
    void Dobavlenie(QNetworkReply* reply);

    void on_pushButton_clicked();

private:
    Ui::Add_contact *ui;
    std::list<void*> spisok;
    QSqlDatabase db;
    QSqlTableModel *model;
    QNetworkAccessManager *manager;
    QNetworkRequest request;
    QString name;
    QString nomer;
};

#endif // ADD_CONTACT_H

```

					ПГУ 1.090301.35.001 ПЗ	Лист
						96
Изм.	Лист	№ докум.	Подпись	Дата		

Файл addcontact.cpp:

```
#include "add_contact.h"
#include "ui_add_contact.h"
#include "secondwindow.h"
#include "spo_db.h"
#include "contact_table.h"
#include "settingsapp.h"
#include "mainmainwindow.h"

#include <QDialog>
#include <QtSql>
#include <QSqlDatabase>
#include <QSqlQuery>
#include <QSqlTableModel>
#include <QtNetwork/QNetworkAccessManager>
#include <QtNetwork/QHttpMultiPart>
#include <QtNetwork/QNetworkRequest>
#include <QtNetwork/QNetworkReply>
#include <QJsonArray>
#include <QJsonObject>
#include <QJsonValue>
#include <QListWidgetItem>

void Add_contact::Dobavlenie(QNetworkReply* reply){
    // if (reply->error()) {
    //     qDebug() << reply->errorString();
    //     } else {

        QString answeerrr = "{\"id\": 3, \"error\": \"200\"}";
        qDebug() << answeerrr;

        QJsonDocument doc =
```

					ПГУ 1.090301.35.001 ПЗ	Лист
						97
Изм.	Лист	№ докум.	Подпись	Дата		

```

QJsonDocument::fromJson(answerrr.toUtf8());

    QJsonValue jsonId = doc.object().value("id");
    QString id = QString::number(jsonId.toInt());

    SpoContact contact;

    QByteArray ba = id.toUtf8();
    const char *c_str2 = ba.data();
    contact.id_user = (char *)c_str2;

    QByteArray ag = name.toUtf8();
    const char *c_str1233 = ag.data();
    contact.full_name = (char *)c_str1233;

    QByteArray ba2 = nomer.toUtf8();
    const char *c_str4 = ba2.data();
    contact.sip_number = (char *)c_str4;

    contact.uri_photo = NULL;

    spo_save_contact(&contact);

    this->close();
    SecondWindow gg;
    gg.setModal(true);
    gg.exec();

    //    }
}

Add_contact::Add_contact(QWidget *parent) :
    QDialog(parent),
    ui(new Ui::Add_contact)

```

```

{
    ui->setupUi(this);

    manager = new QNetworkAccessManager(this); //создание менеджера

    connect(manager, &QNetworkAccessManager::finished, this,
    &Add_contact::Dobavlenie);
}

Add_contact::~~Add_contact()
{
    delete ui;
}

void Add_contact::on_pushButton_clicked()
{
    QUrl url ("http://message.modelshik.ru/api/addContact.php");
    QNetworkRequest request(url); //our server with php-script

    request.setRawHeader("Accept", "*/*");
    request.setRawHeader("Cache-Control", "no-cache");
    request.setRawHeader("Content-Type",
"application/json;charset=utf-8");

    name = ui->Name_line->text();
    nomer = ui->Nomer_line->text();

    QJsonObject jb;
    jb.insert("id", QJsonValue::fromVariant(SettingsApp::ins()-
>getID()));
    jb.insert("signature", QJsonValue::fromVariant(SettingsApp::ins()-
>getSignature()));
    jb.insert("name", QJsonValue::fromVariant(name));
    jb.insert("nomer", QJsonValue::fromVariant(nomer));
    QJsonDocument doc(jb);

```

```

        QNetworkReply *reply = manager->post(request, doc.toJson());
    }

```

Файл delcontact.h:

```

#ifndef DEL_CONTACT_H
#define DEL_CONTACT_H

#include "secondwindow.h"
#include "spo_db.h"
#include "contact_table.h"

#include <QDialog>
#include <QtSql>
#include <QSqlDatabase>
#include <QSqlQuery>
#include <QSqlTableModel>
#include <QtNetwork/QNetworkAccessManager>
#include <QtNetwork/QHttpMultiPart>
#include <QtNetwork/QNetworkRequest>
#include <QtNetwork/QNetworkReply>
#include <QJsonArray>
#include <QJsonObject>
#include <QJsonValue>
#include <QListWidgetItem>

namespace Ui {
class Del_contact;
}

class Del_contact : public QDialog
{
    Q_OBJECT

```

					ПГУ 1.090301.35.001 ПЗ	Лист
						100
Изм.	Лист	№ докум.	Подпись	Дата		

```

public:
    explicit Del_contact(QWidget *parent = 0);
    ~Del_contact();

private slots:
    void on_pushButton_3_clicked();
    void on_pushButton_2_clicked();
    // void on_vhod_clicked();
    void delete_contact(QNetworkReply* reply);

private:
    Ui::Del_contact *ui;
    QNetworkAccessManager *manager;
    QNetworkRequest request;
    std::list<void*> spisok;
};

#endif // DEL_CONTACT_H

```

Файл delcontact.cpp:

```

#include "del_contact.h"
#include "ui_del_contact.h"
#include "secondwindow.h"
#include "dop_okno.h"
#include "spo_db.h"
#include "contact_table.h"
#include "settingsapp.h"

#include <list>
#include "QtSql/QtSqlDatabase"
#include "QtSql/QtSqlQuery"
#include <QDebug>

```

					ПГУ 1.090301.35.001 ПЗ	Лист
						101
Изм.	Лист	№ докум.	Подпись	Дата		

```

#include <QList>
#include <QTableWidgetItem>
#include <QString>
#include <QFile>
#include <QFileDialog>
#include <QMessageBox>
#include <QNetworkAccessManager>
#include <QtSql>
#include <QSqlDatabase>
#include <QSqlQuery>
#include <QSqlTableModel>
#include <QtNetwork/QNetworkAccessManager>
#include <QtNetwork/QHttpMultiPart>
#include <QtNetwork/QNetworkRequest>
#include <QtNetwork/QNetworkReply>
#include <QUrl>
#include <QFileDialog>
#include <QMessageBox>
#include <QPlainTextEdit>
#include <QByteArray>
#include <QtCore>
#include <QDialog>
#include <QSettings>
#include <QLabel>
#include <QJsonObject>
#include <QJsonDocument>
#include <QJsonArray>
#include <QJsonValue>
#include <QJsonParseError>

//SpoContact *del_contact;

```

```

void Del_contact::delete_contact(QNetworkReply* reply){ //ответ
сервера на удаление контакта

/*
    if (reply->error()) {
        qDebug() << reply->errorString();
    } else {

        QString answer2 = "{\"error\": 200}";
        // QString answer2 = reply->readAll();
        qDebug() << answer2;
        spo_delete_contact(del_contact);

    }
*/

Del_contact::Del_contact(QWidget *parent) :
    QDialog(parent),
    ui(new Ui::Del_contact)
{
    ui->setupUi(this);
    manager = new QNetworkAccessManager(this); //создание менеджера
    connect(manager,&QNetworkAccessManager::finished, this,
    &Del_contact::delete_contact);

}

Del_contact::~Del_contact()
{
    delete ui;
}

void Del_contact::on_pushButton_3_clicked()

```

```

{
    this->close();
}

void Del_contact::on_pushButton_2_clicked()
{
    this->close();
}

```

Файл menu_settings.h:

```

#ifndef MENU_SETTINGS_H
#define MENU_SETTINGS_H

#include <QDialog>

namespace Ui {
class menu_settings;
}

class menu_settings : public QDialog
{
    Q_OBJECT

public:
    explicit menu_settings(QWidget *parent = 0);
    ~menu_settings();

private slots:
    void on_pushButton_clicked();

    void on_pushButton_2_clicked();

```

					ПГУ 1.090301.35.001 ПЗ	Лист
						104
Изм.	Лист	№ докум.	Подпись	Дата		

```

        void on_vhod_clicked();

        void on_pushButton_3_clicked();

private:
    Ui::menu_settings *ui;
};

#endif // MENU_SETTINGS_H

```

Файл menu_settings.cpp:

```

#include "menu_settings.h"
#include "ui_menu_settings.h"
#include "triwindow.h"
#include "mainwindow.h"
#include "glav.h"
#include "settingsapp.h"
#include "contact_table.h"
#include "spo_db.h"
#include "contact_table.h"
#include "add_contact.h"
#include "menu_settings.h"
#include "del_contact.h"
#include "secondwindow.h"
#include "ip_okno.h"
#include "media_okno.h"
#include "dop_okno.h"

menu_settings::menu_settings(QWidget *parent) :
    QDialog(parent),
    ui(new Ui::menu_settings)
{

```

```

        ui->setupUi(this);
    }

menu_settings::~menu_settings()
{
    delete ui;
}

void menu_settings::on_pushButton_clicked()
{
    this->close();
    ip_okno gg;
    gg.setModal(true);
    gg.exec();
}

void menu_settings::on_pushButton_2_clicked()
{
    this->close();
    media_okno ga;
    ga.setModal(true);
    ga.exec();
}

void menu_settings::on_vhod_clicked()
{
    // save settings
    this->close();
    SecondWindow a;
    a.setModal(true);
    a.exec();
}

```

```

void menu_settings::on_pushButton_3_clicked()
{
    this->close();
    Dop_okno ok;
    ok.setModal(true);
    ok.exec();
}

```

Файл ip_okno.h:

```

#ifndef IP_OKNO_H
#define IP_OKNO_H

#include <QDialog>

namespace Ui {
class ip_okno;
}

class ip_okno : public QDialog
{
    Q_OBJECT

public:
    explicit ip_okno(QWidget *parent = 0);
    ~ip_okno();

private slots:

    void on_vhod_clicked();

private:
    Ui::ip_okno *ui;

```

					ПГУ 1.090301.35.001 ПЗ	Лист
						107
Изм.	Лист	№ докум.	Подпись	Дата		

```
};
```

```
#endif // IP_OKNO_H
```

Файл ip_okno.cpp:

```
#include "ip_okno.h"
#include "ui_ip_okno.h"
#include "triwindow.h"
#include "mainwindow.h"
#include "glav.h"
#include "settingsapp.h"
#include "contact_table.h"
#include "spo_db.h"
#include "contact_table.h"
#include "add_contact.h"
#include "menu_settings.h"
#include "del_contact.h"
#include "secondwindow.h"
#include "menu_settings.h"
#include "dop_okno.h"
```

```
ip_okno::ip_okno(QWidget *parent) :
    QDialog(parent),
    ui(new Ui::ip_okno)
{
    ui->setupUi(this);
}
```

```
ip_okno::~ip_okno()
```

					ПГУ 1.090301.35.001 ПЗ	Лист
						108
Изм.	Лист	№ докум.	Подпись	Дата		

```

{
    delete ui;
}

void ip_okno::on_vhod_clicked()
{
    SettingsApp::ins()->getIpAts() = ui->line_atc->text();
    SettingsApp::ins()->getIpDns() = ui->line_dns->text();
    qDebug() << SettingsApp::ins()->getIpAts();
    qDebug() << SettingsApp::ins()->getIpDns();

    this->close();
    menu_settings g;
    g.setModal(true);
    g.exec();
}

```

Файл media_okno.h:

```

#ifndef MEDIA_OKNO_H
#define MEDIA_OKNO_H

#include <QDialog>

namespace Ui {
class media_okno;
}

class media_okno : public QDialog
{
    Q_OBJECT

public:

```

```

        explicit media_okno(QWidget *parent = 0);
        ~media_okno();

private slots:
    void on_vhod_clicked();

private:
    Ui::media_okno *ui;
};

#endif // MEDIA_OKNO_H

```

Файл media_okno.h:

```

#include "media_okno.h"
#include "ui_media_okno.h"
#include "triwindow.h"
#include "mainwindow.h"
#include "glav.h"
#include "settingsapp.h"
#include "contact_table.h"
#include "spo_db.h"
#include "contact_table.h"
#include "add_contact.h"
#include "menu_settings.h"
#include "del_contact.h"
#include "secondwindow.h"
#include "ip_okno.h"
#include "menu_settings.h"
#include "dop_okno.h"

```

```

media_okno::media_okno(QWidget *parent) :

```

					ПГУ 1.090301.35.001 ПЗ	Лист
						110
Изм.	Лист	№ докум.	Подпись	Дата		

```

        QDialog (parent),
        ui (new Ui::media_okno)
    {
        ui->setupUi (this);
    }

media_okno::~~media_okno()
{
    delete ui;
}

void media_okno::on_vhod_clicked()
{
    this->close();
    menu_settings g;
    g.setModal(true);
    g.exec();
}

```

Файл dop_okno.h:

```

#ifndef DOP_OKNO_H
#define DOP_OKNO_H

#include <QDialog>

namespace Ui {
class Dop_okno;
}

class Dop_okno : public QDialog
{
    Q_OBJECT

```

```

public:
    explicit Dop_okno(QWidget *parent = 0);
    ~Dop_okno();

private slots:

    void on_vhod_clicked();
    void on_pushButton_3_clicked();

private:
    Ui::Dop_okno *ui;
};

#endif // DOP_OKNO_H

```

Файл dop_okno.cpp:

```

#include "dop_okno.h"
#include "ui_dop_okno.h"
#include "triwindow.h"
#include "mainwindow.h"
#include "glav.h"
#include "settingsapp.h"
#include "contact_table.h"
#include "spo_db.h"
#include "contact_table.h"
#include "add_contact.h"
#include "menu_settings.h"
#include "del_contact.h"
#include "secondwindow.h"

Dop_okno::Dop_okno(QWidget *parent) :
    QDialog(parent),

```

					ПГУ 1.090301.35.001 ПЗ	Лист
						112
Изм.	Лист	№ докум.	Подпись	Дата		

```

        ui(new Ui::Dop_okno)
    {
        ui->setupUi(this);
    }

Dop_okno::~Dop_okno()
{
    delete ui;
}

void Dop_okno::on_vhod_clicked()
{
    this->close();
    menu_settings gp;
    gp.setModal(true);
    gp.exec();
}

```

Файл res_message_thread.h:

```

#ifndef RES_MESSAGE_THREAD_H
#define RES_MESSAGE_THREAD_H

#include <QThread>
#include <QtNetwork/QNetworkAccessManager>
#include <QtNetwork/QHttpMultiPart>
#include <QtNetwork/QNetworkRequest>
#include <QtNetwork/QNetworkReply>

class res_message_thread : public QThread
{
    Q_OBJECT

```

```

public:
    void run();
    explicit res_message_thread(QString threadName);

signals:
    void onResMes(int id_mes);

private:
    void func(QNetworkReply* reply);
    QString name1;    // Имя потока
    QNetworkAccessManager *manager;
    int flag;
};
#endif // RES_MESSAGE_THREAD_H

```

Файл res_message_thread.cpp:

```

#include "res_message_thread.h"
#include <QDebug>
#include "secondwindow.h"
#include "triwindow.h"
#include "mainwindow.h"
#include "glav.h"
#include "settingsapp.h"
#include "contact_table.h"
#include "spo_db.h"
#include "contact_table.h"
#include "add_contact.h"
#include "menu_settings.h"
#include "del_contact.h"
#include "dop_okno.h"
#include "chat_message_table.h"

```

```

void res_message_thread::func(QNetworkReply* reply){

    QString answer = reply->readAll();
    qDebug() << answer;
    QJsonDocument doc = QJsonDocument::fromJson(answer.toUtf8());
    QJsonValue ref = doc.object().value("messages");
    if ( ref.type() == QJsonValue::Object ) {
        QJsonObject json = ref.toObject();

        QJsonValue jsonId = json.value("id");
        QString id = QString::number(jsonId.toInt());

        QJsonValue jsonText = json.value("text");
        QString text = jsonText.toString();


        SpoChatMessage messages;
        messages.id = NULL;
        messages.dir = 1;
        messages.is_read = 0;
        messages.type_content = 0;
        messages.status = 1;
        messages.id_file = 0;


        QByteArray ab = id.toUtf8();
        const char *c_str123 = ab.data();
        messages.id_user = (char *)c_str123;
        qDebug() << "id_user - " << c_str123;


        QByteArray aa = text.toUtf8();
        const char *c_str121 = aa.data();
    }
}

```

```

        messages.message = (char *)c_str121;
        qDebug() << "message - " << c_str121;

        messages.id_chat_room = NULL;
        messages.date = NULL;
        messages.date_send = NULL;
        messages.id_ozu_file = NULL;

    }
}

res_message_thread::res_message_thread(QString threadName) :
    name1(threadName)
{
    manager = new QNetworkAccessManager(this); //создание менеджера
    connect(manager, &QNetworkAccessManager::finished, this,
        &res_message_thread::func);
}

void res_message_thread::run()
{
    /*
    while (1) {
        QUrl url
        ("http://message.modelshik.ru/api/getIncomingMessages.php");
        QNetworkRequest request(url); //our server with php-script
        */
        // request.setRawHeader("Accept", "*/*");

        request.setRawHeader("Cache-Control", "no-cache");
    }
}

```

```

        request.setRawHeader("Content-Type",
"application/json;charset=utf-8");

        QJsonObject jb;

        jb.insert("id", QJsonValue::fromVariant(SettingsApp::ins()-
>getID()));

        jb.insert("signature",
QJsonValue::fromVariant(SettingsApp::ins()->getSignature()));

        QJsonDocument doc(jb);

        flag = 1;

        QNetworkReply *reply = manager->post(request, doc.toJson());

        while (flag) {

            }

            sleep (5);

        }

    }
}

```

Файл triwindow.h:

```

#ifndef TRIWINDOW_H
#define TRIWINDOW_H

#include <QFileDialog>
#include <QDialog>

namespace Ui {
class Triwindow;
}

class Triwindow : public QDialog
{
    Q_OBJECT

```

					ПГУ 1.090301.35.001 ПЗ	Лист
						117
Изм.	Лист	№ докум.	Подпись	Дата		

```

public:
    explicit Triwindow(QWidget *parent = 0);
    ~Triwindow();

private slots:
    void on_voity_2_clicked();
    void on_zareg_clicked();
    void on_vhod_clicked();
    void on_pushButton_clicked();

private:
    Ui::Triwindow *ui;
};

#endif // TRIWINDOW_H

```

Файл triwindow.cpp:

```

#include "triwindow.h"
#include "ui_triwindow.h"
#include "mainwindow.h"
#include "add_contact.h"
#include "dop_okno.h"

```

```

Triwindow::Triwindow(QWidget *parent) :
    QDialog(parent),
    ui(new Ui::Triwindow)
{
    ui->setupUi(this);
}

```

					ПГУ 1.090301.35.001 ПЗ	Лист
						118
Изм.	Лист	№ докум.	Подпись	Дата		

```

}

Triwindow::~~Triwindow()
{
    delete ui;
}

void Triwindow::on_pushButton_clicked()
{
    this->close();
}

```

Файл update_window.h:

```

#ifndef UPDATE_WINDOW_H
#define UPDATE_WINDOW_H

#include <QDialog>
namespace Ui {
class update_window;
}

class update_window : public QDialog
{
    Q_OBJECT

public:
    explicit update_window(QWidget *parent = 0);
    ~update_window();

private:
    Ui::update_window *ui;
};

```

					ПГУ 1.090301.35.001 ПЗ	Лист
						119
Изм.	Лист	№ докум.	Подпись	Дата		

```
#endif // UPDATE_WINDOW_H
```

Файл update_window.cpp:

```
#include "update_window.h"
#include "ui_update_window.h"
#include "secondwindow.h"

update_window::update_window(QWidget *parent) :
    QDialog(parent),
    ui(new Ui::update_window)
{
    ui->setupUi(this);
    //ui->label_name->setText(scr);
}

update_window::~update_window()
{
    delete ui;
}
```

Файл change_contact.h:

```
#ifndef CHANGE_CONTACT_H
#define CHANGE_CONTACT_H

#include <QDialog>
#include "secondwindow.h"

namespace Ui {
class Change_contact;
}

class Change_contact : public QDialog
```

					ПГУ 1.090301.35.001 ПЗ	Лист
						120
Изм.	Лист	№ докум.	Подпись	Дата		

```

{

    Q_OBJECT

public:

    explicit Change_contact (QWidget *parent = 0);

    ~Change_contact ();

private:

    Ui::Change_contact *ui;

};

#endif // CHANGE_CONTACT_H

```

Файл change_contact.cpp:

```

#include "change_contact.h"
#include "ui_change_contact.h"
#include "secondwindow.h"
#include "mainwindow.h"
#include "settingsapp.h"

Change_contact::Change_contact (QWidget *parent) :
    QDialog (parent),
    ui (new Ui::Change_contact)
{
    ui->setupUi (this);

    ui->name_old->setText (S1);
}

Change_contact::~~Change_contact ()
{

```

					ПГУ 1.090301.35.001 ПЗ	Лист
						121
Изм.	Лист	№ докум.	Подпись	Дата		

```
delete ui;  
}
```

					ПГУ 1.090301.35.001 ПЗ	Лист
						122
Изм.	Лист	№ докум.	Подпись	Дата		

Приложение В

(Обязательное)

Корпоративный мессенджер под ОС Windows

Презентация

					ПГУ 1.090301.35.001 ПЗ	Лист
						123
Изм.	Лист	№ докум.	Подпись	Дата		

Тема ВКР: «Корпоративный мессенджер под ОС Windows»

Автор работы студент группы 20ВВП2 Педай Н.Д.
Научный руководитель доцент каф. ВТ ПГУ Дорошенко И.Н.

Пенза 2024

Цели и объекты работы

Целью работы является разработка приложений под управлением ОС Windows, позволяющее пользователям общаться в пределах одной компании

Объектом работы является приложение, обладающее следующими возможностями:

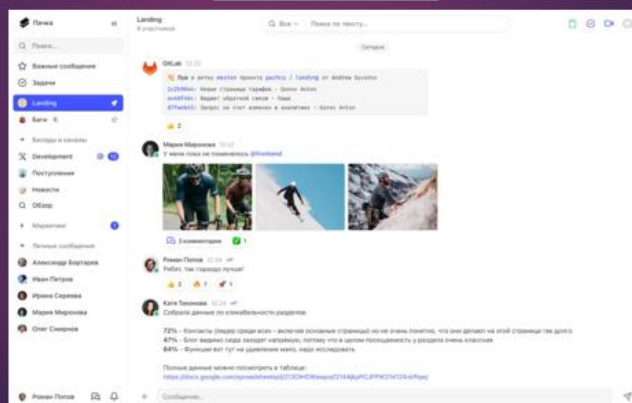
- Регистрация и авторизация пользователя
- Изменение настроек пользователя
- Отправка сообщений пользователю из списка контактов
- Отправка файлов
- Добавление и удаление контактов

Актуальность

В современном информационном обществе эффективная коммуникация является ключевым фактором успешной деятельности предприятий и организаций. В связи с этим большую популярность набирают корпоративные мессенджеры, которые помогают улучшить внутреннюю коммуникацию между сотрудниками.

Существует множество инструментов, которые предназначены для обеспечения внутренней коммуникации между сотрудниками, такие как электронная почта, голосовая и видеосвязь, а также социальные сети. Но наиболее популярным и удобным способом обмена сообщениями являются мессенджеры.

Аналоги



Изм.	Лист	№ докум.	Подпись	Дата

ПГУ 1.090301.35.001 ПЗ

Лист

125

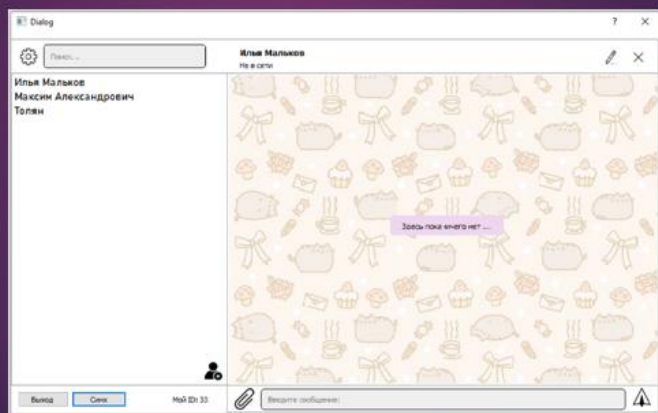
Выбор среды разработки



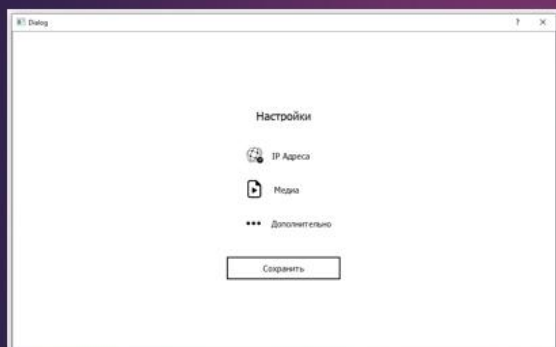
Диаграмма прецедентов



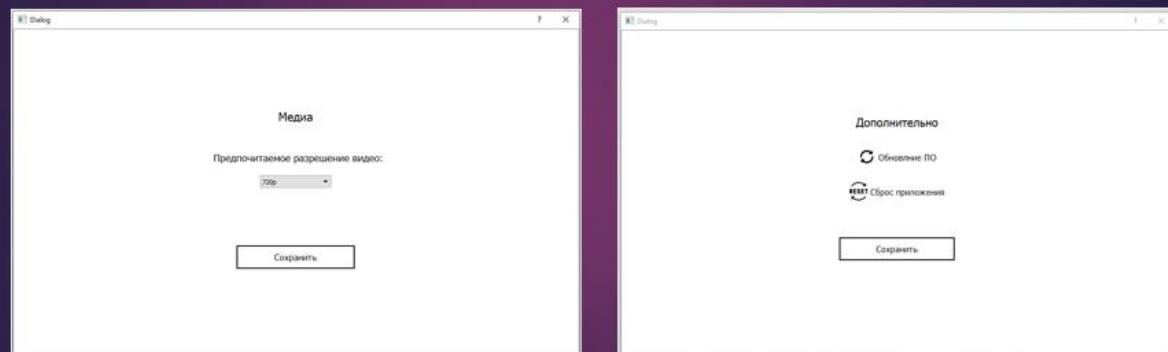
Демонстрация приложения Главное окно



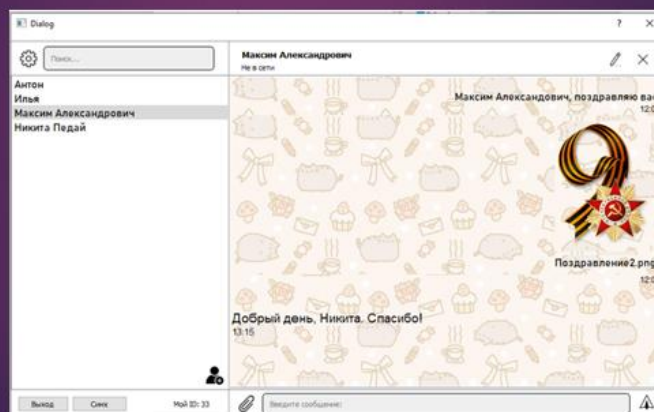
Демонстрация приложения Окно настроек



Демонстрация приложения Окно настроек



Демонстрация приложения Окно чата с контактом



Заключение

В результате выполнения данной выпускной квалификационной работы было успешно создана клиентская часть корпоративного мессенджера, обладающая удобным и интуитивно понятным пользовательским интерфейсом.

Разработка корпоративного мессенджера может повысить эффективность и оперативность работы в компании любой сферы деятельности.

Спасибо за внимание