

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»

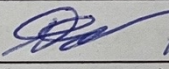
Профиль подготовки: «Программное обеспечение средств вычислительной
техники и автоматизированных систем»

БАКАЛАВРСКАЯ РАБОТА

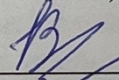
на тему

**ПРОГРАММНОЕ СРЕДСТВО ДЛЯ ПЕРЕДАЧИ РЕЧЕВОЙ И
ФАЙЛОВОЙ ИНФОРМАЦИИ**

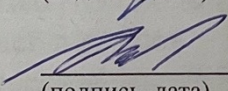
Студент

 17.06.24 Шадрин Данила Алексеевич
(подпись, дата)

Руководитель

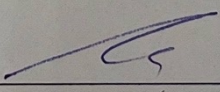
 17.06.24 Егоров Валерий Юрьевич
(подпись, дата)

Нормоконтролёр

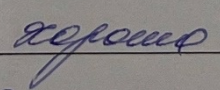
 17.06.24 Бычков Андрей Станиславович
(подпись, дата)

Работа допущена к защите (протокол заседания. кафедры от 27.05.2024г. № 11)

Заведующий кафедрой


(подпись)

М.А. Митрохин
(фамилия, инициалы)

Работа защищена с отметкой  (протокол заседания ГЭК от

24.06.2024г. № 33)

Секретарь ГЭК


(подпись)

Гурин Е.И.
(фамилия, инициалы)

Пенза 2024

Факультет
вычислительной техники

Кафедра
«Вычислительная техника»

«Утверждаю»
Заведующий кафедрой
М.А. Митрохин
« ____ » ____ 20 ____ г.

ЗАДАНИЕ НА БАКАЛАВРСКУЮ РАБОТУ

1. Студент Шадрин Данила Алексеевич гр.20ВВП2 факультета вычислительной техники

2. Тема работы: «Программное средство для передачи речевой и файловой информации»

Тема утверждена приказом ПГУ № 405/о от 19 апреля 2024 г.

3. Руководитель бакалаврской работы к.т.н. каф. ВТ ПГУ Егоров В.Ю.

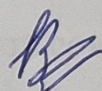
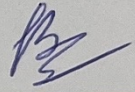
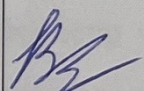
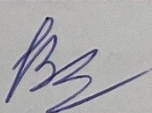
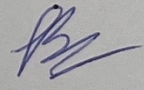
4. Задание на работу (назначение разработки, исходные данные и т.п.)

Разработать программное средство для передачи речевой и файловой информации. Приложение должно позволять пользователям передавать медиа данные при помощи общедоступных интернет-ресурсов IP-телефонии. Так же приложение должно реализовывать защиту передаваемых данных. Приложение должно быть спроектировано с использованием популярного паттерна проектирования. Интерфейс программы должен обладать понятным графическим интерфейсом.

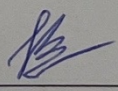
5. Перечень подлежащих разработке вопросов

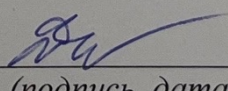
Требуется произвести анализ предметной области: изучить существующие аналоги, изучить способы проектирования и разработки приложения. Также необходимо изучить технологии для разработки приложения, такие как протокол установления сеансов SIP, библиотека PJSIP и т.д.

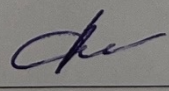
6. Календарный график выполнения работы

№ п/п	Наименование этапов работы	Объем работы	Срок выполнения	Подпись руководителя
1.	Изучение предметной области	50	26.04.2024- 30.04.2024	
2.	Анализ требований	10	01.05.2024- 02.05.2024	
3.	Выбор технологий, применяемых при проектировании и разработке	50	03.05.2024- 07.05.2024	
4.	Описание алгоритмов	40	08.05.2024- 11.05.2024	
5.	Разработка приложения «VoIPPhone»	150	12.05.2024- 29.05.2024	
6.	Оформление пояснительной записки	24	30.05.2024- 31.05.2024	

Дата выдачи задания 19 апреля 2024 г.

Руководитель бакалаврской работы  Егоров В.Ю.
(подпись, дата)

Задание к исполнению принял студент  Шадрин Д.А.
(подпись, дата)

Работу к защите допустить
Декан факультета  20.06.2024 Фионова Л.Р.

Реферат

Бакалаврская работа содержит 93 л., 29 рис., 1 табл., 4 источн., 3 прил.

КЛИЕНТСКОЕ ПРИЛОЖЕНИЕ, VISUAL STUDIO, MVP, C++, .NET, PJSIP

Цель работы – разработать программное средство, позволяющее передавать речевую и файловую информацию.

Разработанное приложение должно позволять пользователям авторизовываться при помощи учетной записи, созданной у любого провайдера, предоставляющего IP-телефонию. Также приложение должно позволять совершать голосовые вызовы по зашифрованному каналу связи, отправлять и принимать зашифрованные файлы и текстовые сообщения.

Результатом проделанной работы является приложение, который обладает простым и понятным интерфейсом и позволяет обмениваться зашифрованными медиа данными при помощи общедоступного сервиса, предоставляющего услуги IP-телефонии. При разработке было использовано большое количество технологий, к которым относятся: язык программирования C++, платформа .NET, библиотека PJSIP и среда разработки Visual Studio.

ПГУ 1.090301.39.001 ПЗ

Изм.	Лист	№ докум.	Подпись	Дата
Разраб.		Шадрин Д.А.		17.06.24
Провер.		Егоров В.Ю.		17.06.24
Н. Контр.		Бычков А.С.		17.06.24
Утверд.		Митрохин М.А.		17.06.24

Программное средство для
передачи речевой и
файловой информации
Пояснительная записка

Лит.	Лист	Листов
	2	
ФВТ, каф. ВТ гр. 20ВВП2		

Содержание

Перечень сокращений и обозначений	7
Введение	8
1 Анализ предметной области	9
1.1 Анализ темы	9
1.2 Анализ аналогов приложений	10
1.3 Постановка задачи	16
2 Выбор технологий и средств разработки приложения	19
2.1 Язык программирования C++	19
2.2 Платформа .Net	20
2.3 Среда разработки Visual Studio	20
2.4 Протокол SIP	21
2.5 Библиотека PJSIP	22
3 Реализация приложения	23
3.1 Описание организации передачи зашифрованных данных	23
3.2 Описание работы приложения	24
3.3 Архитектура приложения	26
3.4 Программная реализация приложения	27
3.5 Основные алгоритмы взаимодействия пользователя и приложения	35
4 Тестирование приложения	40
4.1 Авторизация пользователя, заранее зарегистрированного на общедоступном интернет-ресурсе IP-телефонии	40
4.2 Добавление нового контакта в контактную книжку	42
4.3 Вызов клиента SIP и завершение вызова	44
4.4 Отправка сообщений клиенту	46
4.5 Отправка файлов клиенту	49
5 Безопасность жизнедеятельности	50
5.1 Вред здоровью при работе с ПК	50

5.2 Снижение вреда для здоровья при работе с ПК	50
5.3 Техника безопасности при работе с компьютером	51
Заключение	52
Список использованных источников	53
Приложение А – Uml-диаграммы	54
Приложение А.1 – Диаграмма последовательностей	55
Приложение А.2 – Диаграмма развертывания	56
Приложение А.3 – Диаграмма компонентов	57
Приложение А.4 – Диаграмма вариантов использования	58
Приложение Б – Листинг	59
Приложение В – Презентация	90

Перечень сокращений и обозначений

В настоящей работе применяют следующие сокращения и обозначения

ГОСТ – государственный стандарт

ПК – персональный компьютер

API (Application Programming Interface) – программный интерфейс приложения

CLR (Common Language Runtime) – общезыковая исполняющая среда, являющаяся частью Microsoft .NET Framework

GUI (Graphical User Interface) – графический пользовательский интерфейс

ICE (Interactive Connectivity Establishment) – протокол для обеспечения связи между участниками сеанса связи через NAT

IDE (Integrated Development Environment) – интегрированная среда разработки

IP (Internet Protocol) – межсетевой протокол

MVP (Model-View-Presenter) – шаблон проектирования, используемый при разработке программного обеспечения

NAT (Network Address Translation) – технология трансляции сетевых адресов

RTP (Real-time Transport Protocol) – транспортный протокол реального времени

SDP (Session Description Protocol) – протокол описания сеансов

SIP (Session Initiation Protocol) – иницирующий протокол сеансов

STUN (Simple Traversal of UDP through Network Address Translators) – простой обход UDP через сетевые преобразователи адресов

TURN (Traversal Using Relays around NAT) – обход с использованием ретрансляторов вокруг NAT

UDP (User Datagram Protocol) – протокол пользовательских датаграмм

UML (Unified Modeling Language) – язык графического описания

					<i>ПГУ 1.090301.39.001 ПЗ</i>	Лист
						5
Изм.	Лист	№ докум.	Подпись	Дата		

Введение

В современном мире информационные технологии играют всё более важную роль в повседневной жизни людей. Одной из наиболее востребованных областей информационных технологий является передача информации, включая речевую и файловую. В условиях быстрого развития интернета и его доступности для широких слоёв населения, всё большее значение приобретают общедоступные интернет-ресурсы IP-телефонии.

Актуальность данной темы обусловлена растущей популярностью IP-телефонии в бизнес-сфере благодаря своей простоте, надежности, обилием функций и возможности общаться в любое время и из любой точки мира.

Подводя итог, хочется отметить, что разработка программного средства является востребованной задачей, так как позволяет облегчить процесс коммуникации.

					<i>ПГУ 1.090301.39.001 ПЗ</i>	<i>Лист</i>
						6
<i>Изм.</i>	<i>Лист</i>	<i>№ докум.</i>	<i>Подпись</i>	<i>Дата</i>		

1 Анализ предметной области

1.1 Анализ темы

Каждый год многие компании переходят от традиционных телефонных сетей к IP-телефонии. Это не удивительно, так как в отличие от традиционной телефонии, где используется отдельная сеть телефонных линий, IP-телефония работает поверх существующей инфраструктуры интернета. Это делает её более гибкой и экономичной, поскольку не требует прокладки новых линий связи или модернизации уже существующих.

Подключение к IP-телефонии обычно занимает меньше времени, чем подключение традиционной телефонии. Это связано с тем, что процесс подключения включает в себя всего несколько этапов: выбор провайдера IP-телефонии, заключение договора, настройка оборудования и тестирование соединения.

После подключения к IP-телефонии пользователь получает доступ к широкому спектру услуг, таких как голосовая связь, видеосвязь, конференцсвязь, переадресация вызовов, автоответчик и чат. Пользователь также может использовать различные устройства для использования услуг, такие как стационарные телефоны, мобильные телефоны, компьютеры и планшеты.

Стоит отметить и недостатки связи через протоколы IP. Во-первых, IP-телефония не работает без подключения к интернету, так как все звонки, сообщения и файлы проходят через сервер, где в дальнейшем пересылаются нужному клиенту. Во-вторых, необходимо уделять особое внимание мерам безопасности, поскольку данные, передаваемые через интернет, могут быть перехвачены и использованы злоумышленниками в корыстных целях.

В целом развитие данного направления становится все более популярным среди компаний и организаций, стремящихся оптимизировать свои расходы на связь и повысить эффективность работы своих сотрудников. Поэтому приложение для передачи речевой и файловой информации будет пользоваться спросом.

1.2 Анализ аналогов приложений

Приложения для IP-телефонии — это специализированные программы, которые позволяют совершать и принимать звонки, отправлять и принимать мгновенные сообщения, а также обмениваться файлами. На рынке существует множество приложений, которые реализованы на различных платформах, таких как Windows, macOS, Linux, Android и IOS, каждое из них использует свой подход для решения задач.

Для более корректной реализации приложения, необходимо проанализировать аналоги, выявить их слабые и сильные стороны, а также определить основные требования к разрабатываемому приложению.

В этом пункте будут рассмотрены три самых популярных приложения для IP-телефонии, а именно:

- SIP-софтфон «PhonerLite»;
- автономный софтфон «Bria»;
- SIP-телефон «X-Lite».

1.2.1 SIP-софтфон «PhonerLite»

Интерфейс SIP-софтфона «PhonerLite» представлен на рисунке 1.

Основные особенности данного приложения – это запись разговоров, синхронизации с Google и звонки по ссылкам в браузере.

SIP-софтфон «PhonerLite» имеет большое количество различных функций, предоставляемые пользователю. Ниже представлены основные из них:

- управление контактами. У пользователя есть возможность, добавлять, редактировать, и удалять контакты;
- журнал вызовов. Пользователь может просматривать историю вызовов, включая время и продолжительность звонка;
- синхронизация контактов. Софтфон предоставляет возможность синхронизации контактов между устройствами через облачное хранилище;
- блокировка вызовов. В приложении реализована функция, которая может блокировать нежелательные контакты;

– интеграция с другими приложениями. Приложение поддерживает интеграцию с популярными мессенджерами для обмена контактами.

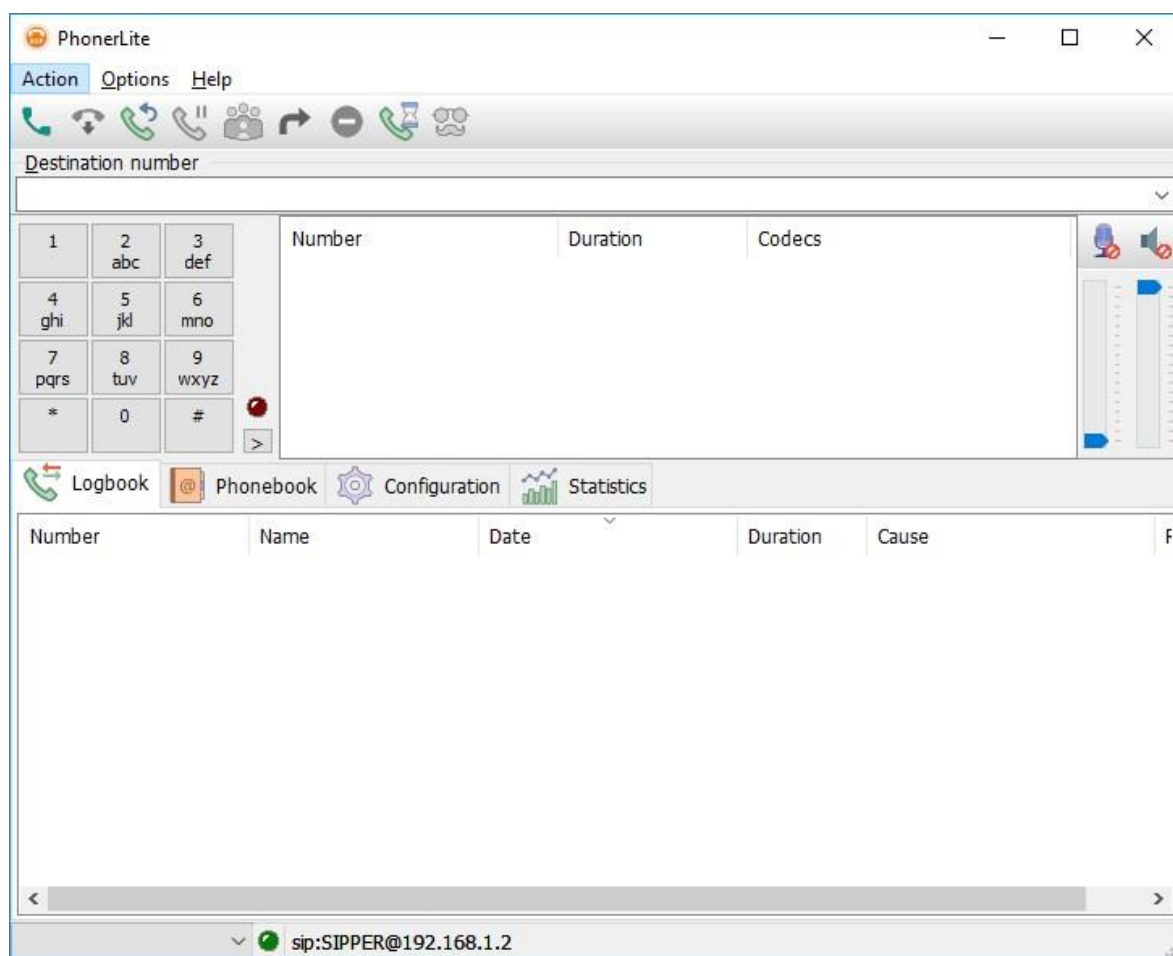


Рисунок 1 - Главная страница SIP-софтфона

Основной недостаток данного софтфона – это посредственная защита данных.

1.2.2 Автономный софтфон «Bria»

Интерфейс автономного софтфона «Bria» представлен на рисунках 2 и 3.

Главные достоинства «Bria» – это переадресация, перевод звонка и возможность использования группового чата.

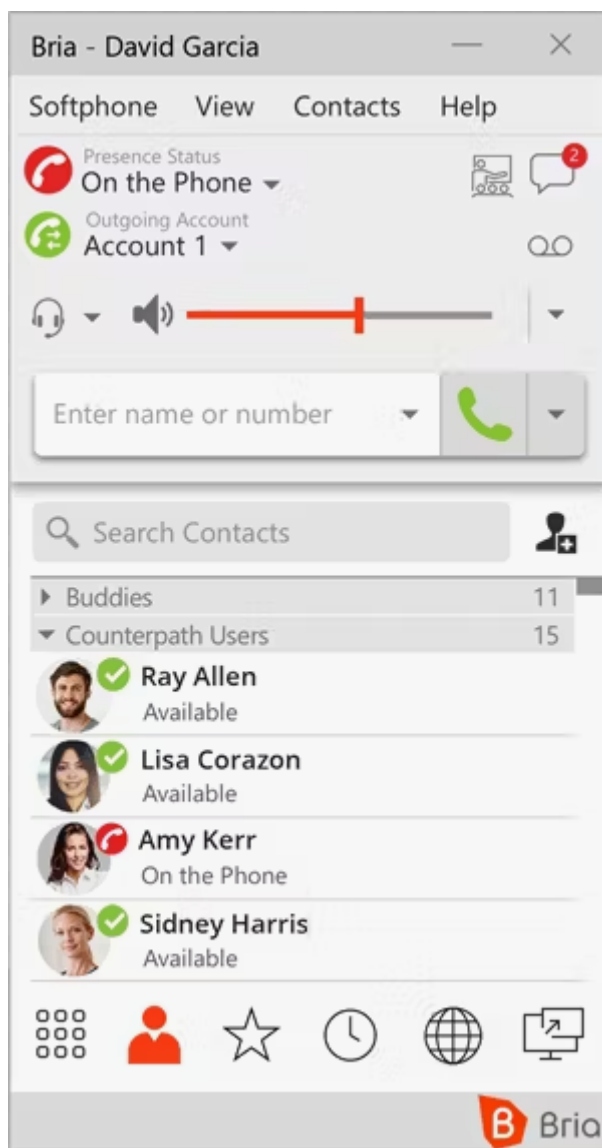


Рисунок 2 - Главная страница онлайн-библиотеки

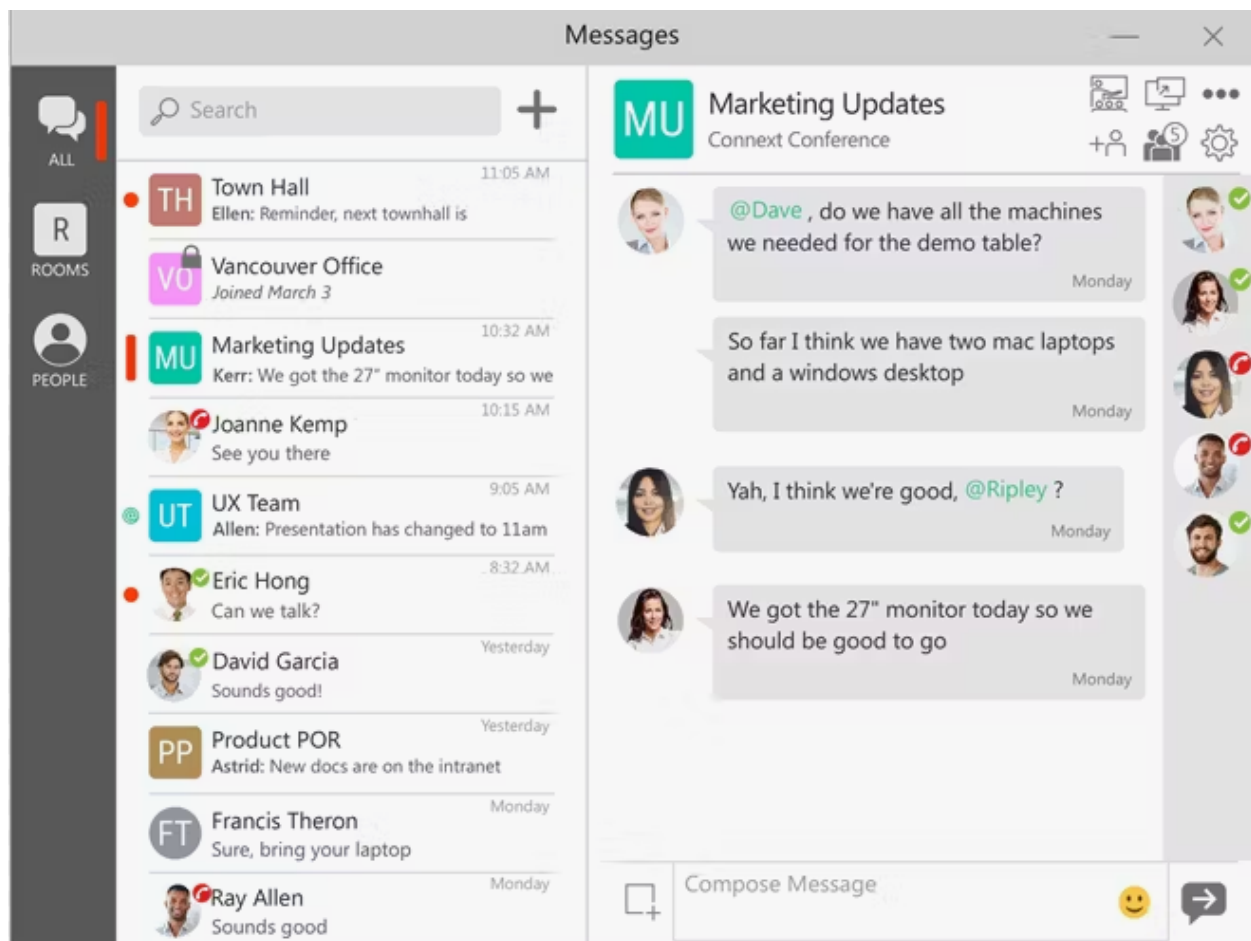


Рисунок 3 - Страница с библиотекой клиента

Софтфон «Bria» предоставляет большое количество различных возможностей пользователю. Основные из них представлены ниже:

- безопасность. «Bria» обеспечивает высокий уровень безопасности данных;
- интеграция с популярными сервисами. Софтфон поддерживает интеграцию с различными платформами, такими как Microsoft Teams, Cisco, Webex и другими;
- удобный интерфейс. Интерфейс данного приложения понятный и удобный;
- масштабируемость. Система может быть адаптирована под нужды компаний любого размера;
- поддержка различных устройств. «Bria» возможно использовать на любых платформах, таких как компьютеры и телефоны.

Изм.	Лист	№ докум.	Подпись	Дата

ПГУ 1.090301.39.001 ПЗ

Лист

11

Основной недостаток атомарного софтбокса – отсутствие бесплатной версии программы.

1.2.3 SIP-телефон «X-Lite».

Интерфейс SIP-телефон «X-Lite» представлены на рисунке 4.

Основное преимущество данного сервиса перед конкурентами – это возможность использования на любых платформах и использование основных функций совершенно бесплатно.



Рисунок 4 - Главная страница сервиса

SIP-телефон «X-Lite» оказывает большой выбор функций для коммуникации. Основные возможности перечислены ниже:

- поддержка различных платформ. Приложение позволяет совершать звонки с любых платформ, таких как Windows, Linux, IOS и Android;
- совместимость с другими приложениями. «X-Lite» позволяет совершать вызовы на другие SIP-телефоны;
- конференцсвязь. SIP-телефон предоставляет возможность создать конференцию для трех пользователей, где можно общаться;

Изм.	Лист	№ докум.	Подпись	Дата

Главные недостатки «X-Lite» – отсутствие какой-либо защиты данных, и неудобный интерфейс.

1.2.4 Вывод

Исходя из анализа аналогов следует отметить, что все они являются хорошими и конкурентноспособными приложениями для совершения SIP-звонков. У каждого есть свои плюсы и минусы, а также особые функции, которые выделяются среди других похожих приложений.

Результаты анализа существующих аналогов можно представить в таблице 1, показывающие минусы и плюсы рассматриваемых SIP-телефонов.

Таблица 1 – Достоинства и недостатки рассмотренных приложений

Функционал	«PhonerLite»	«Bria»	«X-Lite»
Удобный и понятный интерфейс	+	+	-
Шифрование данных	+	-	-
Бесплатное использование	+-	-	+
Возможность передачи файлов	-	-	-
Возможность добавление контактов	+	+	+
Чат для общения	+	+	+

Проанализировав плюсы и минусы конкурентов, было принято решение разработать собственное приложение для совершения IP-звонков, которое будет называться VoIPhone. Данное приложение будет соответствовать основным требованиям, которым предлагает пользователь к IP-софтфонам, а именно: удобный интерфейс; все основные функции, необходимые для комфортного использования; возможность авторизованным пользователям совершать звонки; возможность добавления контактов в контактную книжку; возможность использования чата, для общения с контактами.

Помимо базового функционала, будут реализованы возможности, которые отсутствуют у аналогов и будут привлекательны конечному пользователю. А

именно будет реализована возможность передачи файлов, и дополнительная защита данных путем шифрования алгоритмом ГОСТ 28147-89 «Магма».

Конкуренция среди IP-телефонов слишком высока, поэтому разрабатываться будет десктопное приложение со своими особенностями, которых нет у аналогов.

Преимуществами «VoIPhone» перед другими IP-софтфонами будет то, что пользователю будет доступен функционал IP-софтфона, в котором имеется возможность передачи зашифрованных файлов, текстовых сообщений, а также возможность устанавливать голосовую связь по зашифрованному каналу.

1.3 Постановка задачи

Приложение «VoIPhone» должно реализовывать следующий функционал:

- авторизация пользователя, заранее зарегистрированного на общедоступном интернет-ресурсе IP-телефонии;
- добавление нового контакта в контактную книжку;
- вызов клиента по зашифрованному каналу связи;
- отправка зашифрованных сообщений клиенту;
- отправка зашифрованных файлов клиенту;
- обработка входящих звонков;
- обработка входящих сообщений;
- обработка входящих файлов.

Минимальные системные требования:

- операционная система: Windows (10, 11);
- минимальная скорость интернета: 2 Мбит/сек;
- минимальный объем оперативной памяти: 2 ГБ (для Windows 11 требуется 4 ГБ оперативной памяти);
- минимальный объем свободной памяти на жестком диске: 40 МБ.

Данное приложение имеет функционал, который разделен на две категории, а именно:

- для неавторизованного пользователя;
- для авторизованного пользователя.

Диаграмма вариантов использования приложения для неавторизованного пользователя представлена на рисунке 5.

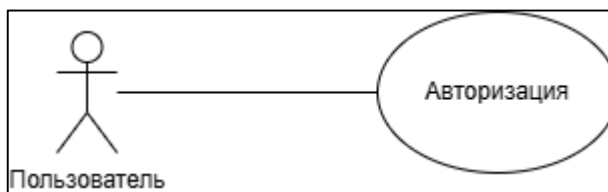


Рисунок 5 – UML диаграмма вариантов использования неавторизованного пользователя

Неавторизованный пользователь имеет возможность только авторизоваться. Авторизованному пользователю будет доступен расширенный функционал работы программы. UML диаграмма представлена на рисунке 6.

Функциональные возможности, доступные авторизованному пользователю:

- принятие входящих звонков;
- принятие входящих сообщений;
- просмотр профиля;
- принятие входящих файлов;
- добавление контактов в контактную книжку;
- совершение исходящих вызовов;
- отправка исходящих сообщений;
- отправка исходящих файлов;
- возможность смены аккаунта.

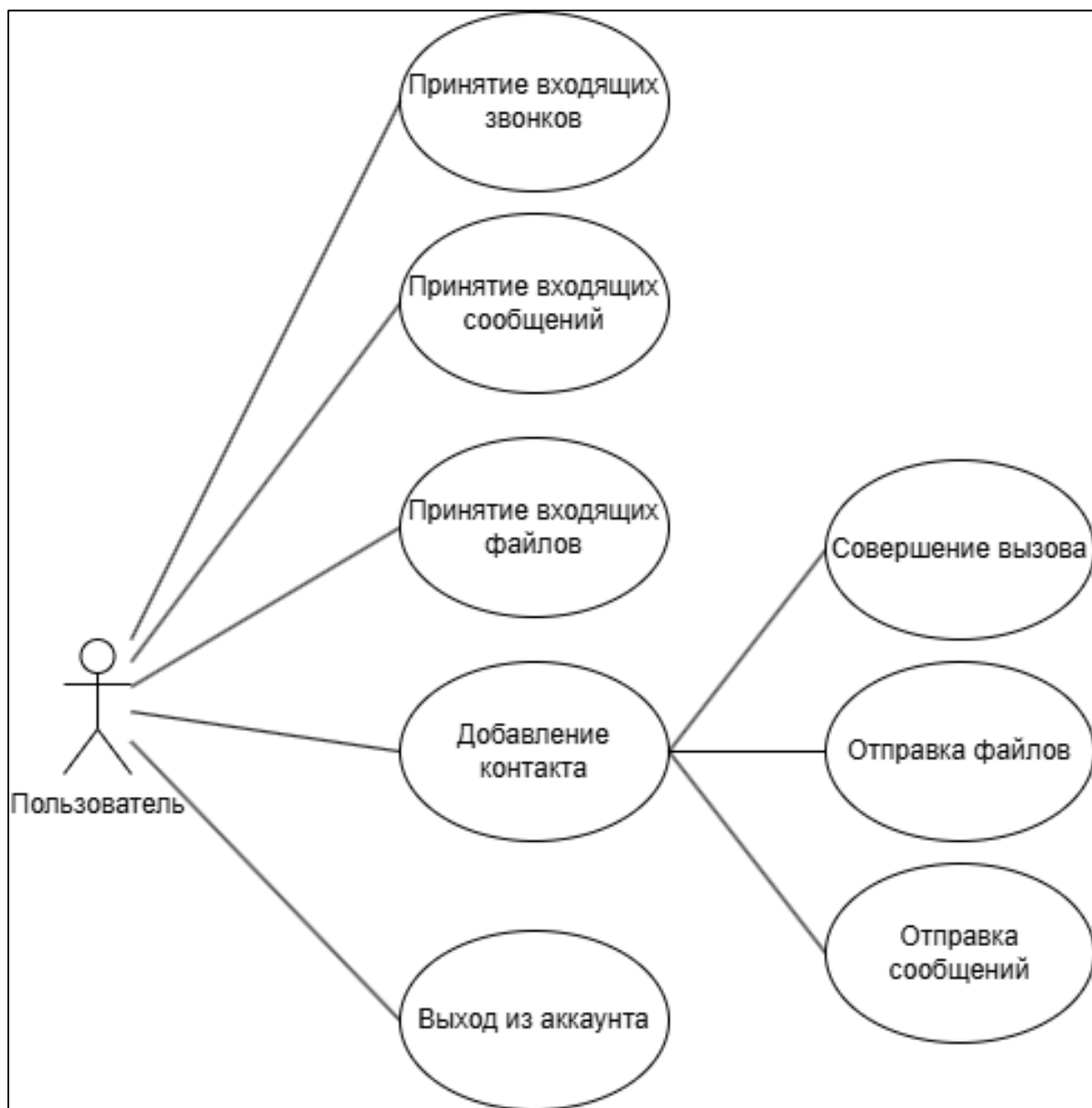


Рисунок 6 – UML диаграмма вариантов использования авторизованного пользователя

2 Выбор технологий и средств разработки приложения

В данном разделе будут рассмотрены технологии и средства разработки, которые были использованы для реализации программы SIP клиент «VoIPhone».

2.1 Язык программирования C++

Язык программирования C++ – является мощным и универсальным языком программирования, который широко используется для разработки различных типов приложений, включая системное программирование, разработку игр, научные вычисления, мобильные приложения и многое другое [1]. Он объединяет в себе возможности низкоуровневого языка C и объектно-ориентированного программирования, что делает его гибким и эффективным инструментом для создания сложных программ. Этот язык является универсальным и его используют для разработки десктопных приложений под различные операционные системы, написания игр, системного программирования, нейросетей и мобильных приложений.

Данный язык программирования является одним из наиболее распространенных так, как позволяет разработчикам создавать сложные программы с высокой производительностью и эффективностью.

C++ остается популярным выбором для разработчиков, работающих в областях, где требуется близкое взаимодействие с аппаратным обеспечением, высокая производительность и эффективное управление ресурсами. Благодаря своей гибкости, C++ также широко используется в различных сферах разработки программного обеспечения, включая создание крупных проектов, библиотек, фреймворков и многих других приложений.

Среди преимуществ C++ можно выделить возможность работы на различных платформах, высокую скорость выполнения программ, возможность оптимизации кода для конкретных задач, обширные библиотеки и фреймворки для разработки, а также поддержку различных парадигм программирования, включая процедурное, объектно-ориентированное и обобщенное программирование.

Компания Microsoft активно развивает и обновляет язык C++, с каждой новой версией появляется много различного функционала.

Таким образом, выбор C++ и WinForms для реализации проекта обусловлен необходимостью интеграции с существующим кодом на C++, удобством разработки GUI на платформе Windows, поддержкой Windows-специфичных функций, простотой и быстротой разработки, а также богатыми возможностями и гибкостью при создании приложений.

2.2 Платформа .Net

Платформа .Net является набором технологий, разработанных компанией Microsoft для создания и выполнения приложений на различных устройствах и операционных системах.

Данная платформа включает в себя набор библиотек и инструментов для удобной и быстрой разработки различного программного обеспечения любого уровня сложности.

Стоит отметить, что в .Net есть общезыковая среда исполнения CommonLanguageRuntime (CLR), которая упрощает разработку приложений и компонентов. Эта среда обеспечивает выполнение определенного функционала, такого как сборка мусора, обработка исключений, безопасное использование памяти и отладка программы.

2.3 Среда разработки Visual Studio

VisualStudio – это интегрированная среда разработки (IDE), разработанная компанией Microsoft. Данная IDE является мощным инструментом разработки, отладки и тестирования программ, написанных на языках программирования, разработанных компанией Microsoft, а именно C, C++, C#, F#, Visual Basic.NET. Также есть возможность разрабатывать на Python, JavaScript и других языках. Используется для разработки мобильных, десктопных, веб-приложений, а также игр.

Данной приложением было выбрано в качестве основной среды разработки

из-за того, что оно очень хорошо интегрировано с платформой .NET и языком программирования C++, в VisualStudio удобно тестировать и отлаживать свое приложение, устанавливать необходимые пакеты с помощью удобного инструмента NuGet, также есть возможность использовать систему контроля версий.

Главный конкурент VisualStudio – это Rider. Данное IDE от компании JetBrains является очень хорошим и конкурентно способным, но был выбран продукт от компании Microsoft, так как он является бесплатным.

2.4 Протокол SIP

Протокол установления сеансов (SIP) – это протокол прикладного уровня, который описывает способ для осуществления голосовых и видео-звонков, а также для отправки моментальных сообщений. Он регулирует взаимодействие между приложениями для установления соединения между ними, даже если они находятся на значительном расстоянии друг от друга. Так же этот стандарт определяет способ создания каналов связи и согласования протоколов для обмена информации между клиентами.

Так как SIP сам по себе не может передать голос или сообщение он работает совместно с другими протоколами прикладного уровня:

- SDP (Session Description Protocol) – для обмена данными;
- RTP (Real-time Transport Protocol) – для осуществления голосовой связи в реальном времени;
- TLS (Transport Layer Security) – для шифрования передаваемых данных.

Передача информации производится при помощи протокола IP.

Так же у протокола SIP есть и аналог — это протокол H.323. Он выполняет все тоже самое, что и протокол установления сеансов, но работает на уровне битовых полей, что в идеальных условиях реализации позволяет экономить сетевой трафик [2]. Однако в современных условиях быстрого распространения широкополосного интернета это преимущество уже не выглядит столь значимым.

2.5 Библиотека PJSIP

PJSIP – это бесплатная библиотека с открытым исходным кодом, которая позволяет работать со стеком протоколов обеспечивающих передачу медиа данных: SIP, SDP, RTP, STUN, TURN и ICE. Так же реализовывает обход NAT функциональности и прекрасно работает с различными брандмауэрами. Библиотека имеет множество различных модулей, которые объединены в одни высокоуровневый API [3].

PJSIP компактен и многофункционален. Он поддерживает аудио, видео, сведения о присутствии, мгновенный обмен сообщениями и имеет обширную документацию.

Так же стоит отметить, что библиотека доступна на любых платформах начиная от портативных телефонов и заканчивая высоконагруженными серверами IP-телефонии.

					<i>ПГУ 1.090301.39.001 ПЗ</i>	Лист
						20
Изм.	Лист	№ докум.	Подпись	Дата		

3 Реализация приложения

В этом разделе описан процесс разработки приложения «VoIPhone». Будут рассмотрены основные компоненты приложения и функционал. Цель этого раздела заключается в предоставлении всей необходимой информации о создании приложения «VoIPhone».

3.1 Описание организации передачи зашифрованных данных

В приложении для обмена речевой, файловой и текстовой информацией используются API-функции сторонней библиотеки. Однако она не поддерживает шифрование данных по алгоритму ГОСТ 28147-89 «Магма».

В связи с этим было принято решение модифицировать исходный код библиотеки, добавив собственную реализацию алгоритма шифрования. Для безопасного обмена медиа данными необходимо предварительно обменяться ключами безопасности любым доступным способом.

Согласованный ключ следует записать в текстовый файл рядом с исполняемым файлом приложения. После этого пользователи смогут передавать друг-другу данные без каких-либо искажений.

Топология системы и распределение ее компонентов по физическим узлам представлены на рисунке 7.

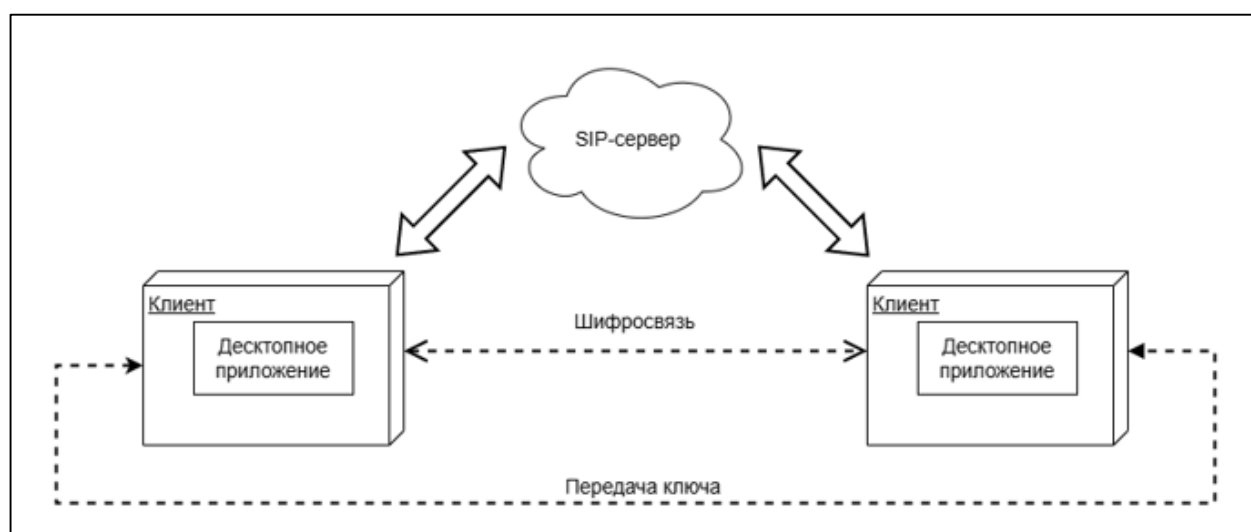


Рисунок 7 – UML-диаграмма разворачивания системы

3.2 Описание работы приложения

При разработке приложения «VoIPhone» использовалась библиотека для работы со стеком протоколов SIP «PJSIP».

На рисунке 8 приведена диаграмма последовательности принципа работы приложения «VoIPhone».

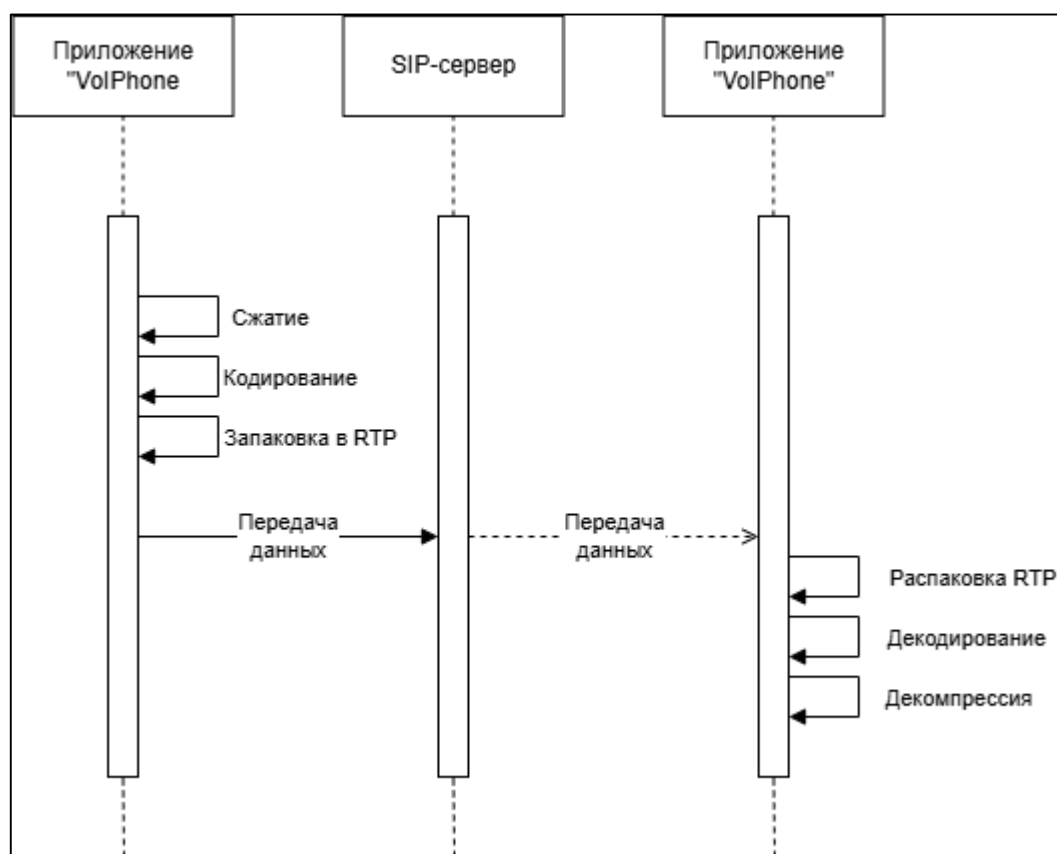


Рисунок 8 – UML-диаграмма последовательности работы приложения

Алгоритм работы приложения с медиа данными заключается в следующем: пользователь инициирует начало передачи данных посредством выбора соответствующей кнопки на интерфейсе. Следующим шагом приложение запрашивает доступ к порту 5060 для передачи информации. Далее сформируется SDP-объект, содержащий информацию о параметрах передаваемой информации, версии протоколов, а также используемых кодеков.

Для передачи пакетов необходимо осуществить некоторые действия:

- сжатие. Под данным действием понимается применение кодека для

потока. Это необходимо, поскольку никакие сети не смогут передать исходные медиа данные ввиду их тяжеловесности. Аудиопоток – последовательность цифровых отсчетов (samples), которые описывают изменение звукового сигнала во времени. Ввиду того, что поток аудио может быть очень большой в приложение встроена поддержка аудиокодеков Speex, G.723.1 и др. Их задача заключается в уменьшении размера исходного аудиофайла;

- кодирование. После того как кодек сжал исходные данные, следует кодирование для обеспечения безопасности. Кодирование происходит при помощи симметричного блочного алгоритма ГОСТ 28147-89 «Магма» в режиме гаммирования;

- запаковка потока в пакет RTP. Подготовка данных к их передаче по сети. Протокол RTP предназначен для доставки данных в реальном времени. При этом определяется тип поля данных, производится нумерация потоков, присвоение временных меток для синхронизации дорожек и мониторинг доставки;

- передача по сети по протоколу UDP, обеспечивает минимальный интервал между пакетами.

При получении пакета с сервера необходимо выполнить действия в обратном порядке:

- Распаковка RPT пакета. Синхронизация, получение трафика и передача его в декодер.

- Декодирование. Расшифровка информации путем обратного кодирования алгоритмом «Магма».

- Декомпрессия. Получение чистых данных, готовых для передачи пользователю.

- Обновление интерфейса.

Во время передачи данных есть вероятность потери пакетов. Для этого в приложении используется Jitter Buffer, он позволяет получать данные с небольшой задержкой, что позволяет выиграть дополнительное время для повторного запроса данных.

3.3 Архитектура приложения

Архитектура MVP была выбрана в качестве паттерна проектирования приложения.

MVP (Model-View-Presenter) – это популярный шаблон проектирования, который используется для разделения компонентов приложения на три основные части: модель (Model), представление (View) и презентера (Presenter) [4]. Каждая из этих частей выполняет свою определенную функцию, из-за чего упрощается разработка, поддержка и масштабируемость приложения.

Модель – это компонент, который отвечает за работу с данными приложения. Модель представляет собой структуру данных, бизнес-логику и методы доступа к данным. Она не зависит от представления или контроллера и обеспечивает абстракцию данных, что позволяет легко изменять данные без влияния на другие части приложения.

Представление – реализует отображение данных модели и маршрутизацию пользовательских команд или событий презентеру.

Презентер – это связующее звено между моделью и представлением. Он обрабатывает пользовательские запросы, взаимодействует с моделью для получения или изменения данных и передает данные в представление для отображения пользователю.

Диаграмма компонентов представлена на рисунке 9.

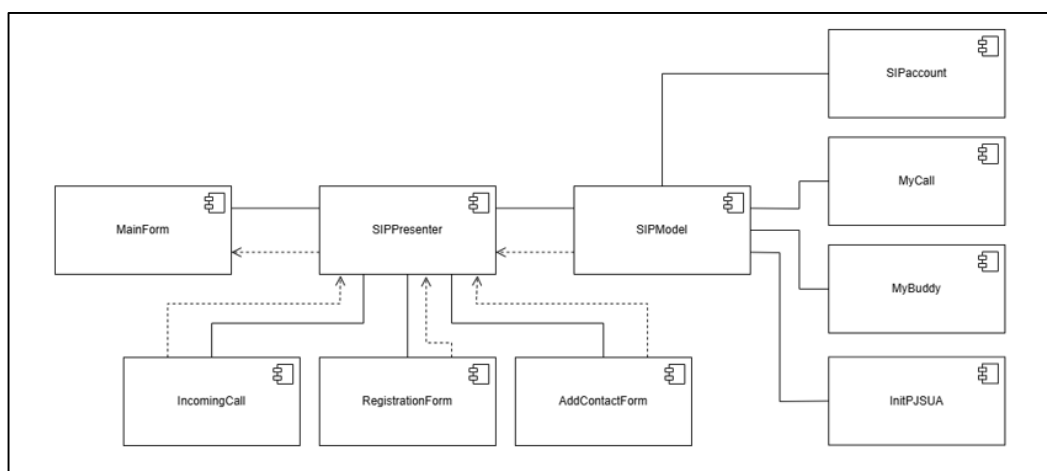


Рисунок 9 – Диаграмма компонентов

Клиентская часть приложения разработана на WinForms и для взаимодействия с сервером, предоставляющий IP-телефонию, используется библиотека PJSIP. Обмен данными между клиентами происходит при помощи функций, которые реализованы в библиотеке.

3.4 Программная реализация приложения

Данное приложение содержит три основных класса:

- MainForm;
- SIPPresenter;
- SIPModel.

MainForm – главное окно приложения, которое отображает все обновления модели.

SIPPresenter – класс реализующий презентер, содержит методы для взаимодействия с моделью и отображением. Функционал этих методов заключается в обработки событий от представления и вызове обновления модели.

SIPModel – класс реализующий модель, который содержит набор классов для взаимодействия с библиотекой. Функционал этих классов заключается в обработке входящих и исходящих звонков, сообщений, файлов и регистрации аккаунта.

3.4.1 MainForm

Этот класс является представлением и отвечает за отображение изменений модели и передачу событий презентеру. Для этого реализованы следующие методы, к которым привязаны события презентера:

- loginRegButton_Click();
- makeCallButton_Click();
- sendTextButton_Click();
- sendFileButton_Click().

Метод loginRegButton_Click() уведомляет презентер о нажатии кнопки регистрации. Реализация кода приведена в листинге 1.

Листинг 1. loginRegButton_Click():

```
event System::Action^ LoginRegButtonClicked;
```

```

System::Void loginRegButton_Click(System::Object^ sender,
System::EventArgs^ e) {
    LoginRegButtonClicked();
}

```

Метод `makeCallButton_Click()` () уведомляет презентер о нажатии кнопки совершения голосового вызова. Реализация кода приведена в листинге 2.

Листинг 2. `makeCallButton_Click()`:

```

event System::Action^ makeCall;
System::Void makeCallButton_Click(System::Object^ sender,
System::EventArgs^ e) {
    makeCall();
}

```

Метод `sendTextButton_Click()` уведомляет презентер о нажатии кнопки отправки мгновенного сообщения. Реализация кода приведена в листинге 3.

Листинг 3. `sendTextButton_Click()`:

```

event System::Action^ sendMsg;
System::Void sendTextButton_Click(System::Object^ sender,
System::EventArgs^ e) {
    sendMsg();
}

```

Метод `sendFileButton_Click()` уведомляет презентер о нажатии кнопки отправки файла клиенту. Реализация кода приведена в листинге 4.

Листинг 4. `sendFileButton_Click()`:

```

event System::Action^ sendFile;
System::Void sendFileButton_Click(System::Object^ sender,
System::EventArgs^ e) {
    sendFile();
}

```

3.4.2 SIPModel

Данный компонент содержит множество классов, которые используются в приложении. В нем содержатся следующие классы:

- `InitPJSUA`. Данный класс содержит в себе свойства, которые необходимы для инициализации библиотеки;
- `MyCall`. Данный класс предоставляет методы для обработки входящих и исходящих вызовов, а также для отправки файлов;
- `MyBuddy`. Этот класс переопределяет метод класса родителя, который предоставляет информацию о состоянии отправки сообщения клиенту;

– SIPAccount. Класс содержит поля и переопределенные методы родительского класса, которые необходимы для отслеживания изменений состояния аккаунта. В нем содержатся такие поля, как массив звонков и ссылка на класс презентер, а также методы, которые информируют о входящем звонке, входящих сообщениях.

Будут рассмотрены основные методы данного класса, а именно:

- loginAccount();
- makeCallSs();
- sendMessage();
- makeCallNs().

Метод loginAccount() реализует авторизацию на сервере SIP, в качестве передаваемых параметров принимается массив, который хранит в себе информацию о пользователе, а также ссылка на презентер. В случае успешной регистрации функция возвращает значение true. Реализация кода приведена в листинге 5.

Листинг 5. loginAccount():

```
bool
VoIPhone::SIPModel::loginAccount(System::Collections::ArrayList^
destination, SIPPresenter^ pres)
{
    acc = new SIPaccount(pres);
    System::Collections::Generic::List<System::String^>^
userData = gcnew
System::Collections::Generic::List<System::String^>();
    for each (auto item in destination)
    {
        userData->Add(dynamic_cast<System::String^>(item));
    }
    uData userDataNString;
    userDataNString.sipServ = (const
char*)(System::Runtime::InteropServices::Marshal::StringToHGlobalAns
i(userData[0])).ToPointer();
    userDataNString.sipProxy = (const
char*)(System::Runtime::InteropServices::Marshal::StringToHGlobalAns
i(userData[1])).ToPointer();
    userDataNString.sipDomain = (const
char*)(System::Runtime::InteropServices::Marshal::StringToHGlobalAns
i(userData[2])).ToPointer();
    userDataNString.sipUsername = (const
char*)(System::Runtime::InteropServices::Marshal::StringToHGlobalAns
i(userData[3])).ToPointer();
```

```

        userDataNString.sipPassword = (const
char*) (System::Runtime::InteropServices::Marshal::StringToHGlobalAns
i(userData[4])).ToPointer();
        pj::AccountConfig acc_cfg;
        acc_cfg.idUri = "sip:" +
userDataNString.sipUsername + "@" + userDataNString.sipServ;
        acc_cfg.regConfig.registrarUri = "sip:" +
userDataNString.sipServ;
        acc_cfg.sipConfig.proxies.push_back("sip:" +
userDataNString.sipProxy);
        acc_cfg.sipConfig.authCreds.push_back(pj::AuthCredInfo("Dige
ts", "*", userDataNString.sipUsername, PJSIP_CRED_DATA_PLAIN_PASSWD,
userDataNString.sipPassword));
        try {
            acc->create(acc_cfg, true);
            this->buddyCreate();
        }
        catch (pj::Error& err) {
            System::Console::WriteLine("Авторизация аккаунта
завершилась с ошибкой(Create) \r\n Error #" + gcnew
System::String(err.info().data()));
            return false;
        }
        if (!acc->isValid()) {
            System::Console::WriteLine("Error in valid account");
            return false;
        }
        else {
            try {
                pj::PresenceStatus ps;
                ps.status = PJSUA_BUDDY_STATUS_ONLINE;
                // Optional, set the activity and some note
                ps.activity = PJRPID_ACTIVITY_BUSY;
                ps.note = "On the phone";
                acc->setOnlineStatus(ps);
            }
            catch (pj::Error& err) {
                System::Console::WriteLine("Инициализация
библиотеки завершилась с ошибкой(PresenceStatus) \r\n Error #" +
gcnew System::String(err.info().data()));
                return false;
            }
        }
        return true;
    }
}

```

Метод makeCallSs() выполняет голосовой вызов удаленного клиента, в качестве передаваемого параметра принимается строка в которой хранится адрес назначения. Реализация кода приведена в листинге 6.

Листинг 6. makeCallSs():

					ПГУ 1.090301.39.001 ПЗ	Лист
						28
Изм.	Лист	№ докум.	Подпись	Дата		

```

void VoIPhone::SIPModel::makeCalls(System::String^ destination)
{
    int startIndex = destination->IndexOf("\r\n");
    String^ resultString = destination->Substring(startIndex +
2);
    const char* chAddress = (const
char*)(System::Runtime::InteropServices::Marshal::StringToHGlobalAns
i(resultString)).ToPointer();
    std::string userName = chAddress;
    System::Runtime::InteropServices::Marshal::FreeHGlobal(IntPt
r((void*)chAddress));

    pj::Call* call = new MyCall(*acc);
    acc->calls.push_back(call);
    pj::CallOpParam prm(true);
    prm.opt.audioCount = 1;
    prm.opt.videoCount = 0;

    std::thread callTh([this, userName, call, prm] {
        CallThread(userName, call, prm);
    });
    callTh.detach();
}
void VoIPhone::SIPModel::CallThread(std::string callAddress,
pj::Call* call, pj::CallOpParam prm)
{
    pj_thread_desc desc;
    pj_thread_t* thread = 0;
    pj_thread_register("callIs", desc, &thread);

    try {
        call->makeCall(callAddress, prm);
    }
    catch (pj::Error& err) {
        std::cout << "Error with call " + err.info() <<
std::endl;
        acc->calls.pop_back();
        return;
    }
    _callActive = true;
}

```

Метод `sendMessage()` реализует отправку мгновенных сообщений, в качестве передаваемых параметров принимает строку с сообщением и адрес назначения. Реализация кода приведена в листинге 7.

Листинг 7. `sendMessage()`:

```

void VoIPhone::SIPModel::sendMessage(System::String^ msg,
System::String^ fromUri) const
{
    int          startIndex          = fromUri->IndexOf("\r\n");

```

					ПГУ 1.090301.39.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		29

```

String^      resultString  = fromUri->Substring(startIndex +
2);
    const char* chAddress    = (const
char*) (System::Runtime::InteropServices::Marshal::StringToHGlobalAns
i(resultString)).ToPointer();
    const char* chMsg        = (const
char*) (System::Runtime::InteropServices::Marshal::StringToHGlobalAns
i(msg)).ToPointer();
    std::string userName     = chAddress;
    pj::BuddyConfig cfg;
    cfg.uri = userName;
    pj::Buddy* buddy = new pj::Buddy();
    buddy->create(*acc, cfg);
    pj::SendInstantMessageParam simp;
    simp.content = chMsg;
    System::Runtime::InteropServices::Marshal::FreeHGlobal(IntPt
r((void*)chAddress));
    System::Runtime::InteropServices::Marshal::FreeHGlobal(IntPt
r((void*)chMsg));
    try {
        buddy->sendInstantMessage(simp);
    }
    catch (pj::Error& err) {
        std::cout << err.info() << std::endl;
    }
    delete buddy;
}

```

Метод makeCallNs() реализует отправку файлов, в качестве передаваемого параметра принимается строка в которой хранится адрес. Реализация кода приведена в листинге 8.

Листинг 8. SendData():

```

VoIPhone::callData VoIPhone::SIPModel::makeCallNs(std::string
destination)
{
    VoIPhone::callData callData = {};
    pj::Call* call = new MyCall(*acc);
    acc->calls.push_back(call);
    pj::CallOpParam prm(true);
    prm.opt.audioCount = 1;
    prm.opt.videoCount = 0;
    std::string userName = destination;
    call->makeCall(destination, prm);
    callData.call = call;
    callData.prm = (pj::CallOpParam*)&prm;
    return callData;
}

```

3.4.3 SIPPresenter

Данный компонент содержит ссылки на класс представления и модели и выполняет обработку событий, переданных с представления.

Будут рассмотрены основные методы данного класса, а именно:

- OnLoginRegButtonClicked();
- OnSendMessageButtonClicked();
- OnMakeCallButtonClicked();
- OnSendFileButtonClicked();

Метод OnLoginRegButtonClicked() реализует обращение к модели и обновление представления. В случае успешной работы модели обновиться главный интерфейс программы. Реализация кода приведена в листинге 9.

Листинг 9. OnLoginRegButtonClicked():

```
void VoIPhone::SIPPresenter::OnLoginRegButtonClicked() {
    if (view->_accRegFlag == ACC_NOT_REGISTERED) {
        RegistrationForm^ regForm = gcnew RegistrationForm();
        regForm->ShowDialog();
        ArrayList^ accountInfo = regForm->getData();
        if (model->loginAccount(accountInfo, this)) {
            view->loginRegButton->Text = L"Выйти";
            view->loginAccountInfo->Text = L"SIP ID: " +
accountInfo[3]->ToString() + L"@ " + accountInfo[2]->ToString();
            view->_accRegFlag = ACC_IS_REGISTERED;
        }
        else
            System::Windows::Forms::MessageBox::Show("Введен
неправильный логин или пароль!");
    }
    else if (view->_accRegFlag == ACC_IS_REGISTERED) {
        if (MessageBox::Show(
            L"Вы действительно хотите выйти?",
            L"Выход",
            MessageBoxButtons::YesNo,
            MessageBoxIcon::Exclamation
        ) == System::Windows::Forms::DialogResult::Yes) {
            OnExitButtonClicked();
        }
    }
}
```

Метод OnSendMessageButtonClicked() реализует обращение к модели с целью отправки мгновенного сообщения. Реализация кода приведена в листинге 10.

Листинг 10. OnSendMessageButtonClicked():

```
void VoIPhone::SIPPresenter::OnSendMessageButtonClicked()
{
    if (view->_accRegFlag == ACC_IS_REGISTERED) {
        model->sendMessage(view->sendMessageTextBox->Text-
>ToString(), view->contactConInfoLabel->Text->ToString());
        view->sendMessageTextBox->Clear();
    }
    else if (view->_accRegFlag == ACC_NOT_REGISTERED) {
        MessageBox::Show("Для начала войдите в аккаунт");
    }
}
```

Метод OnMakeCallButtonClicked() реализует обращение к модели с целью совершения голосового вызова. В случае успешного вызова будет изменен главный интерфейс программы. Реализация кода приведена в листинге 11.

Листинг 11. OnMakeCallButtonClicked():

```
void VoIPhone::SIPPresenter::OnMakeCallButtonClicked()
{
    if (view->_accRegFlag == ACC_IS_REGISTERED) {
        if (view->_inActiveCall == ACC_CALL_NO_ACTIVE) {
            if (view->contactConInfoLabel->Text != "") {
                model->makeCallSs(view->contactConInfoLabel->Text-
>ToString());
                view->_inActiveCall = ACC_CALL_ACTIVE;
                view->makeCallButton->Text = "Завершить";
            }
            else
                System::Windows::Forms::MessageBox::Show("Вы не
выбрали кому звонить");
        }
        else if (view->_inActiveCall == ACC_CALL_ACTIVE) {
            if (view->chatRichTextBox->Text != "") {
                view->chatRichTextBox->SelectedText = view-
>chatRichTextBox->Text + System::Environment::NewLine + "Соединение
разорвано...";
            }
            model->hangupCalls();
            view->_inActiveCall = ACC_CALL_NO_ACTIVE;
            view->makeCallButton->Text = "Вызов";
        }
    }
    else if (view->_accRegFlag == ACC_NOT_REGISTERED)
        System::Windows::Forms::MessageBox::Show("Для начала войдите
в аккаунт");
}
```

Метод OnSendFileButtonClicked реализует обращение к модели с целью отправки файла клиенту. Для этого функция открывает проводник для того, чтобы выбрать файл и передает название файла в функцию, реализованную в модели. Реализация кода приведена в листинге 11.

Листинг 11. OnSendFileButtonClicked():

```
void VoIPhone::SIPPresenter::OnSendFileButtonClicked()
{
    OpenFileDialog^ openDlg = gcnew OpenFileDialog();
    openDlg->Title = "Выбор файла для отправки";
    openDlg->Filter = "Image
Files (*.BMP;*.JPG;*.GIF)|*.BMP;*.JPG;*.GIF|All files (*.*)|*.*";
    if (DialogResult::OK == openDlg->ShowDialog())
    {
        String^ fileName = openDlg->FileName;
        model->makeCallNs(view->contactConInfoLabel->Text-
>ToString(), fileName);
    }
}
```

3.5 Основные алгоритмы взаимодействия пользователя и приложения

UML диаграммы деятельности разработаны для наглядного отображения последовательности действий между пользователем и приложением в процессе взаимодействия.

Для начала будет рассмотрен алгоритм авторизация аккаунта, заранее созданного на общедоступном интернет-ресурсе IP-телефонии. Для того, чтобы пользователь имел возможности использовать все возможности приложения, предлагаемых приложением ему необходимо выполнить авторизацию.

При попытке авторизации с неправильными учетными данными пользователю будет выведено сообщение об ошибке. В случае успешной авторизации пользователь получит доступ ко всем функциям приложения.

Последовательность описанных действий представлены на рисунке 10.

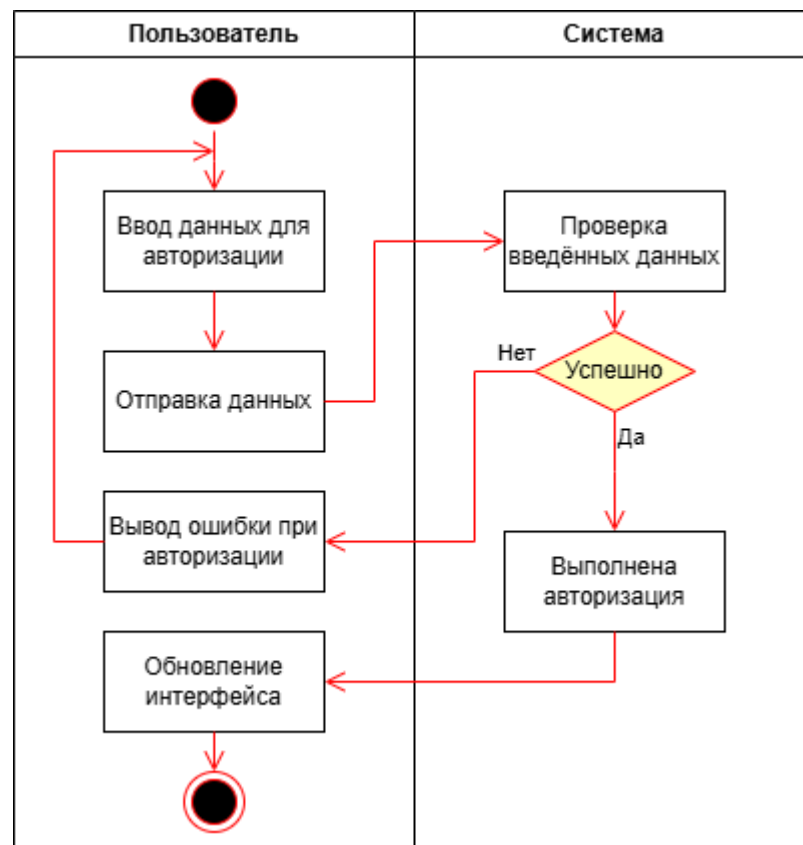


Рисунок 10 – UML-диаграмма деятельности авторизации

Далее рассмотрен алгоритм добавления контакта в контактную книжку. Чтобы пользователю добавить контакт, ему необходимо нажать на кнопку «добавить контакт». Далее передается управление презентеру, который в свою очередь обращается к модели для проверки авторизации пользователя. Если пользователь авторизован, тогда откроется модальное окно, в которое следует записать данные контакта. Далее идет проверка правильности введенного адреса, если пользователя с таким адресом не существует, то презентер выводит предупреждение пользователю о том, что пользователя с таким адресом не существует.

После удачной проверки адреса пользователя, презентер формирует данные, полученные с модального окна, и отправляет их в главное окно для дальнейшего использования.

Последовательность описанных действий представлены на рисунке 11.

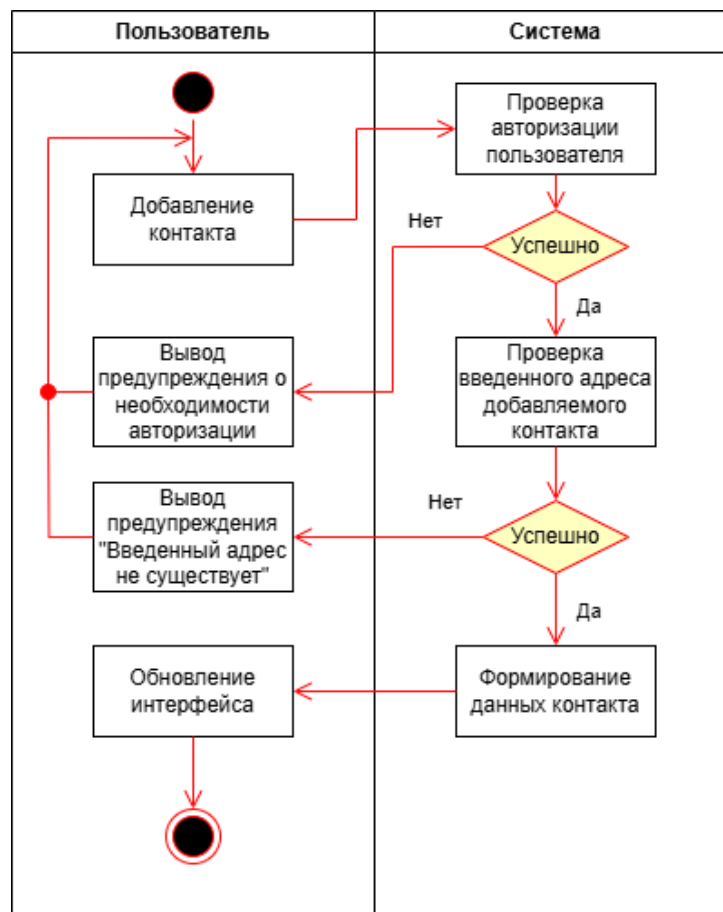


Рисунок 11 – UML-диаграмма деятельности добавление контакта в контактную книжку

На рисунке 12 представлена UML диаграмма деятельности совершения вызова контакта, добавленного ранее. Данный алгоритм работает следующий образом, пользователь выбирает контакт из контактной книжки и нажимает кнопку вызов. Далее управление передается презентеру, который проверяет не вышел ли пользователь из аккаунта и выбран ли контакт. Если какая-либо проверка завершилась неудачей, то презентер выведет сообщение об ошибке. В ином случае управление предается модели, которая вызовет соответствующие функции вызова. В случае успешного вызова модель, вернет положительный результат презентеру, который обновит главное окно. Если функция вызова завершиться с ошибкой, модель сообщит это презентеру, который в свою очередь сообщит об этом пользователю.

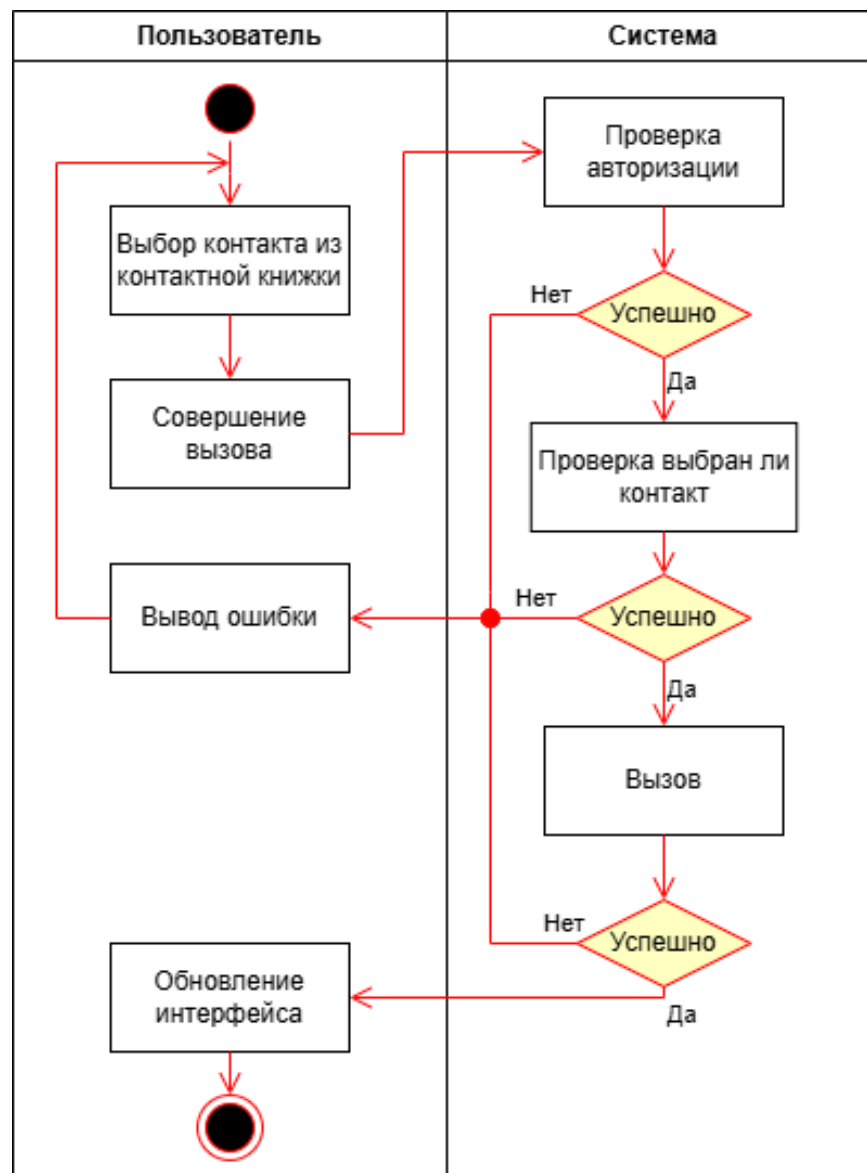


Рисунок 12 – UML-диаграмма деятельности совершения вызова

Далее рассмотрена возможность отправки файлов пользователю добавленного в список контактов. Для отправки файла нужно выбрать контакт из контактной книжки и нажать на кнопку «файл» рядом с полем для ввода текстового сообщения. При этом система проверит авторизацию пользователя в приложении. Если пользователь не авторизован, ему выведется предупреждение. Если же пользователь авторизован и предупреждения не было, тогда пользователю откроется окно выбора файла. После выбора файла модель вызовет функцию отправки файла. В случае успешной передачи файла модель сообщит об этом презентер, а тот в свою очередь обновит чат на главном окне приложения. Если же

модель не смогла отправить файл выбранному контакту, она уведомит об этом презентер, который выведет соответствующее сообщение в чат главного окна приложения.

Последовательность действий по отправке файла контакту, выбранного из контактной книжки представлены на рисунке 13.

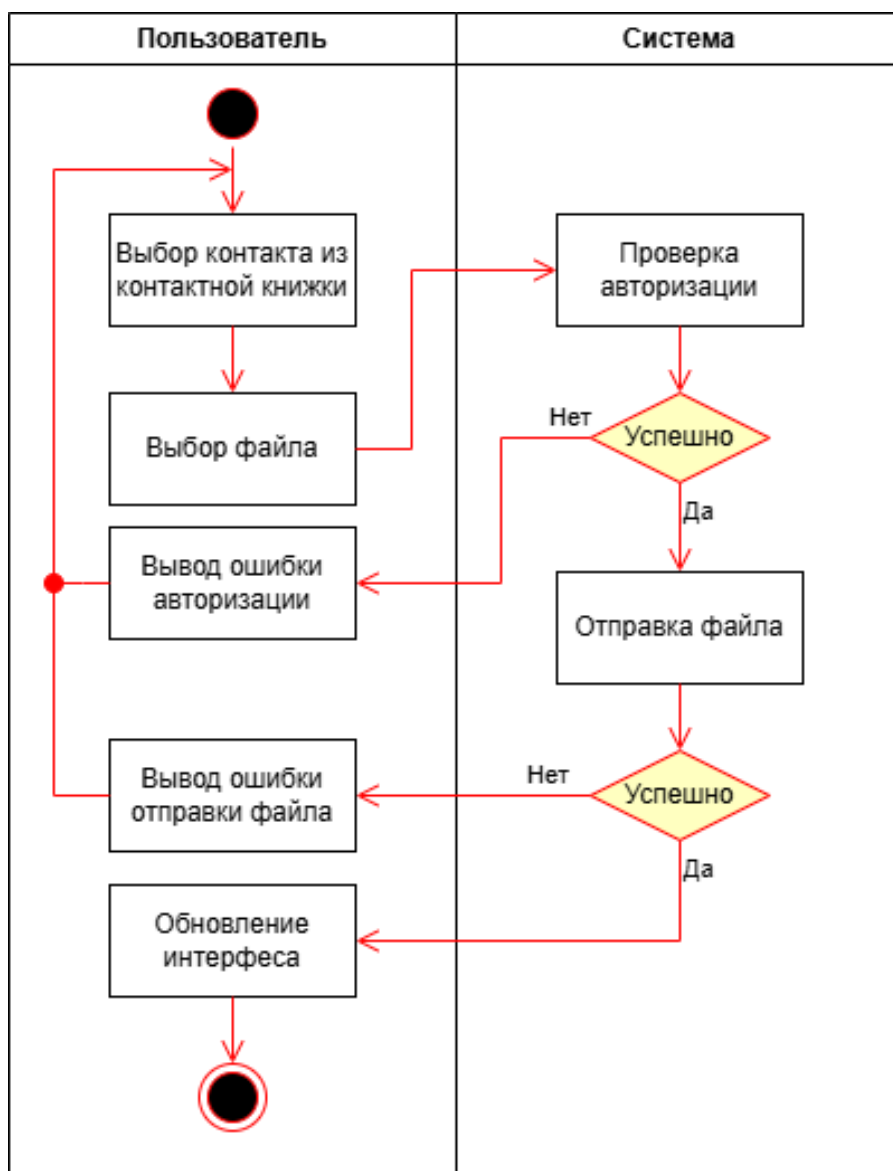


Рисунок 13 – UML-диаграмма отправки файла

4 Тестирование приложения

Для проверки работоспособности программного средства «VoIPhone» необходимо провести его тестирование.

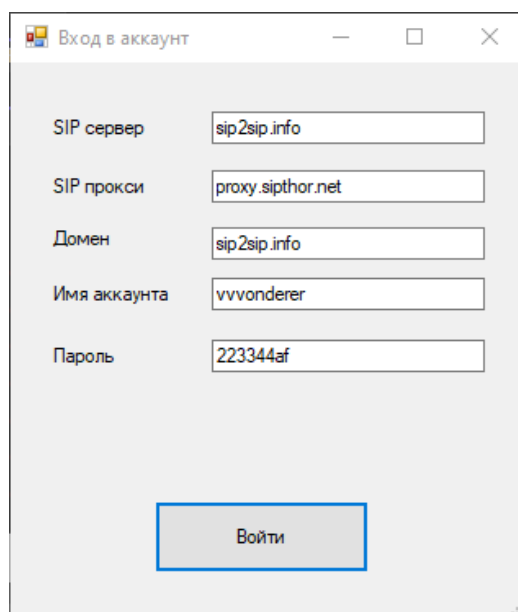
Тестирование будет состоять из следующих действий:

- авторизация пользователя, заранее зарегистрированного на общедоступном интернет-ресурсе IP-телефонии;
- добавление нового контакта в контактную книжку;
- вызов клиента SIP и завершение вызова;
- отправка сообщений клиенту;
- отправка файлов клиенту.

4.1 Авторизация пользователя, заранее зарегистрированного на общедоступном интернет-ресурсе IP-телефонии

Для тестирования авторизации необходимо перейти на соответствующее окно нажав на кнопку «Войти» и ввести SIP сервер, SIP прокси, домен, логин (имя аккаунта) и пароль.

На рисунке 14 представлен ввод данных при авторизации.



Вход в аккаунт

SIP сервер: sip2sip.info

SIP прокси: proxy.sipthor.net

Домен: sip2sip.info

Имя аккаунта: vvvonderer

Пароль: 223344af

Войти

Рисунок 14 – Авторизация пользователя

После успешной авторизации пользователь попадет на главную страницу приложения. Показано на рисунке 15.

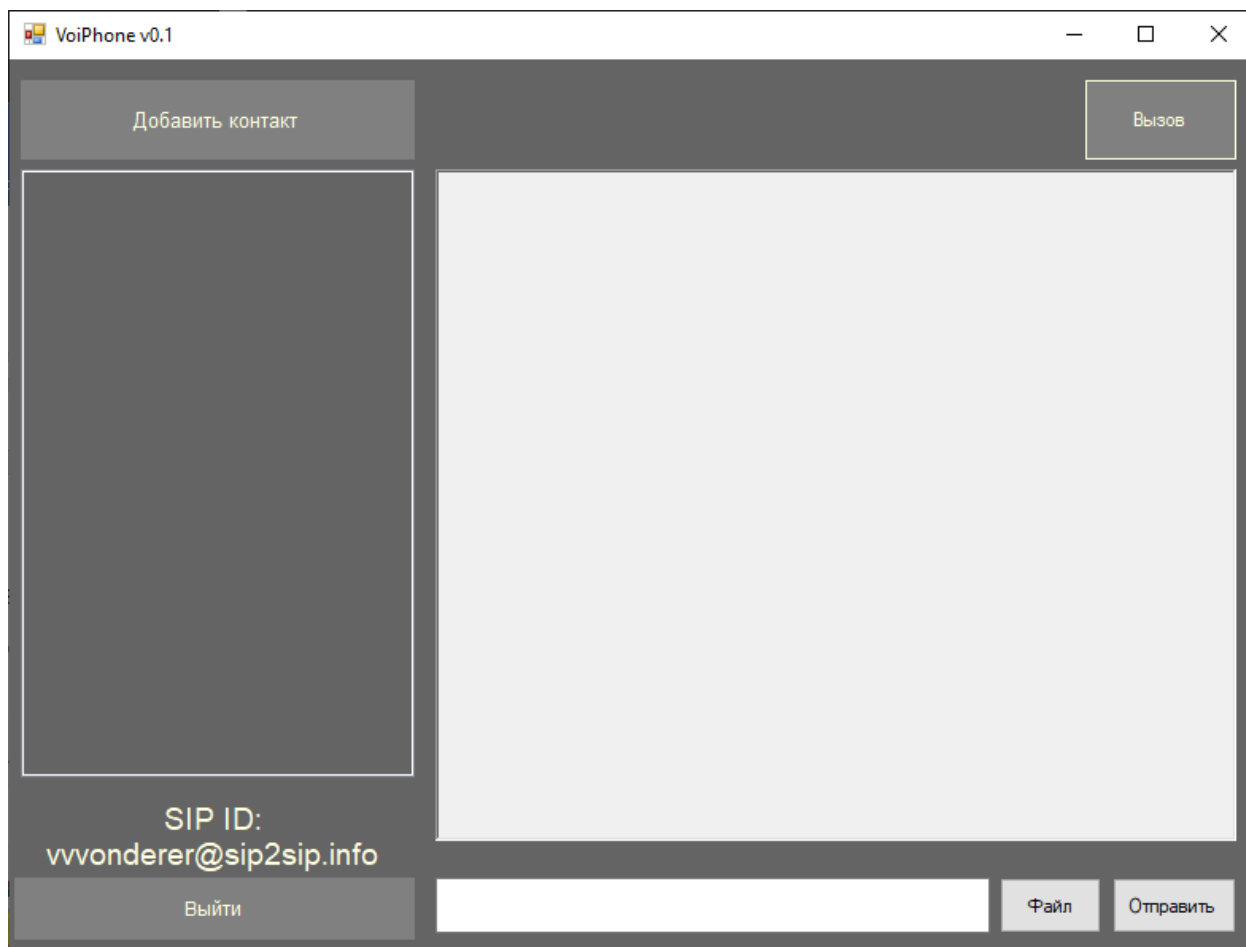


Рисунок 15 – Главное окно приложения после успешной авторизации

Если ввести неверные данные, то программа выведет предупреждение. На рисунке 16 это отражено.

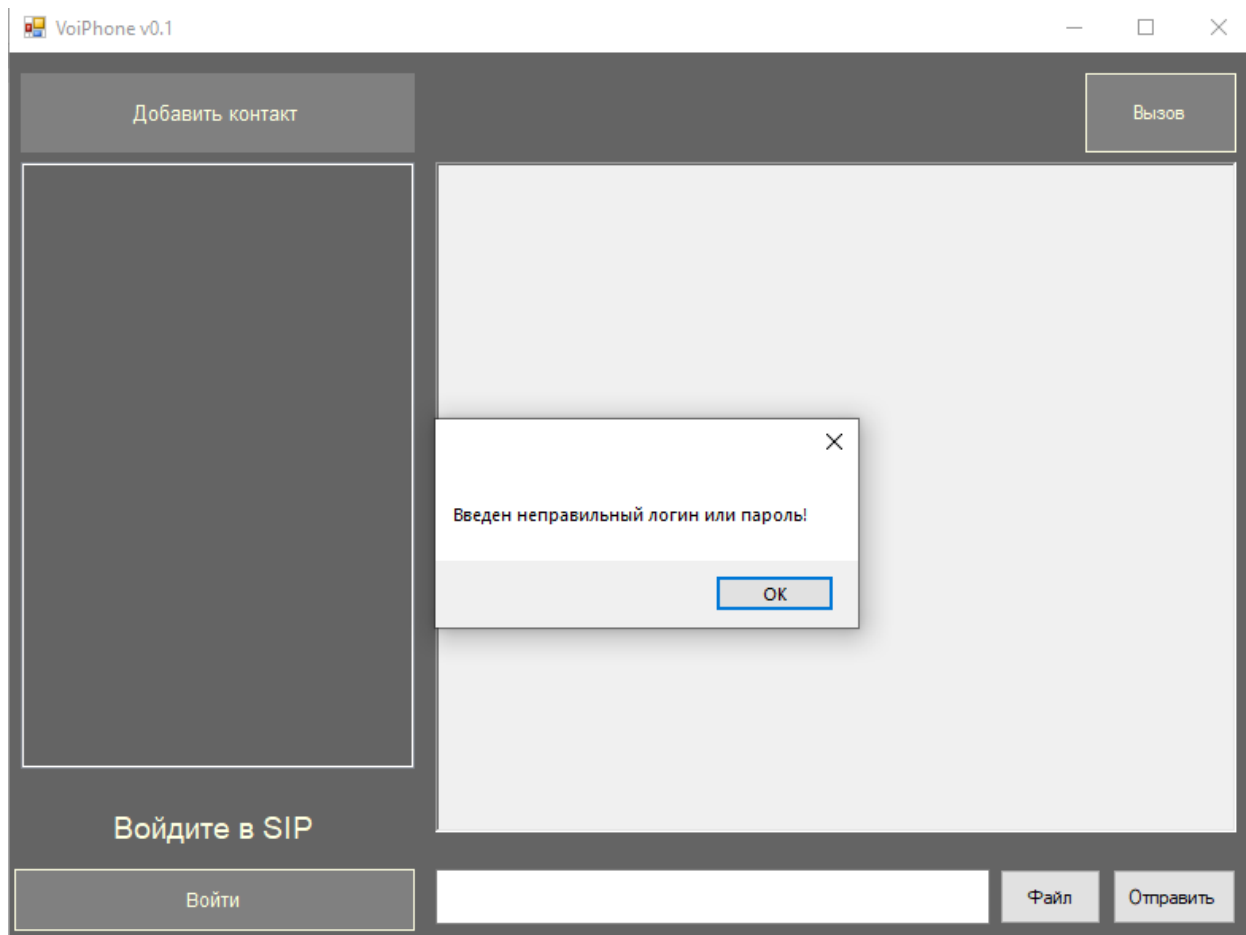


Рисунок 16 – Вывод предупреждения при неверном вводе данных

4.2 Добавление нового контакта в контактную книжку

Для того, чтобы протестировать функционал добавление нового контакта в контактную книжку, необходимо нажать на кнопку добавить контакт после чего откроется модальное окно. Пример добавления контакта представлен на рисунке 17.

Рисунок 17 – Добавление контакта

Изм.	Лист	№ докум.	Подпись	Дата

После добавления контакта, обновится пользовательский интерфейс, где отобразится новый контакт, который представлен на рисунке 18.

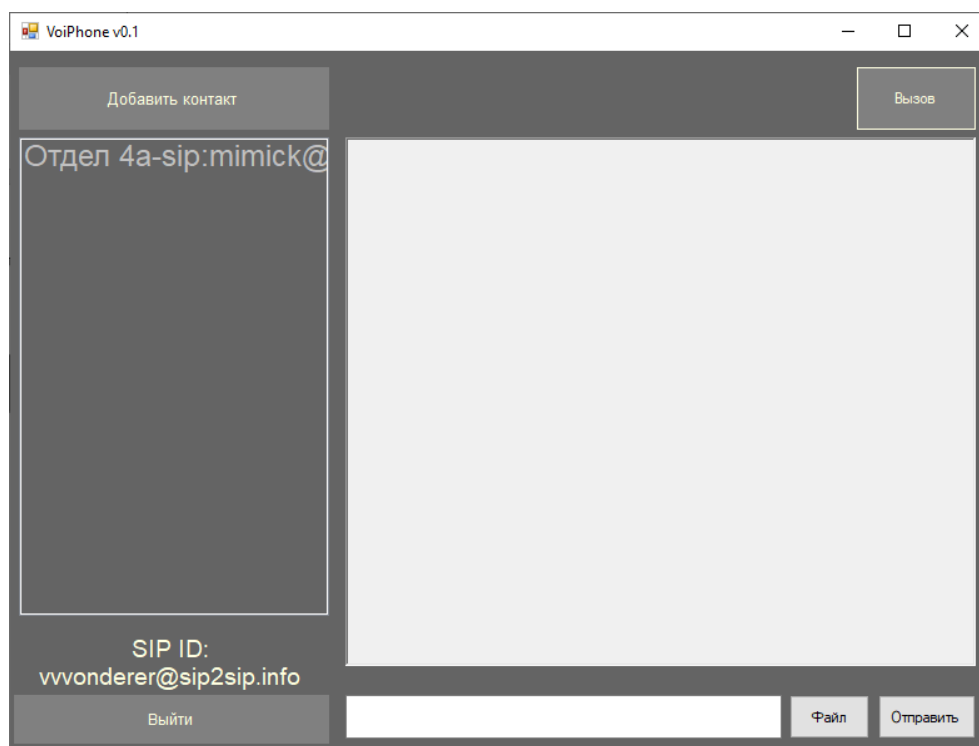


Рисунок 18 – Главная страница приложения после добавления контакта

Если ввести неверные данные, то программа выведет предупреждение, это показано на рисунке 19.

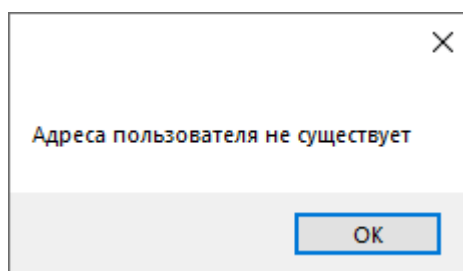


Рисунок 19 – Вывод предупреждения при неверном вводе данных

4.3 Вызов клиента SIP и завершение вызова

Чтобы протестировать вызов клиента SIP, необходимо выбрать контакт из контактной книжки нажав два раза по контакту и нажать кнопку вызов, показано на рисунке 20. В результате у вызываемого пользователя воспроизведется звуковой сигнал и всплывет диалоговое окно, рисунок 21.

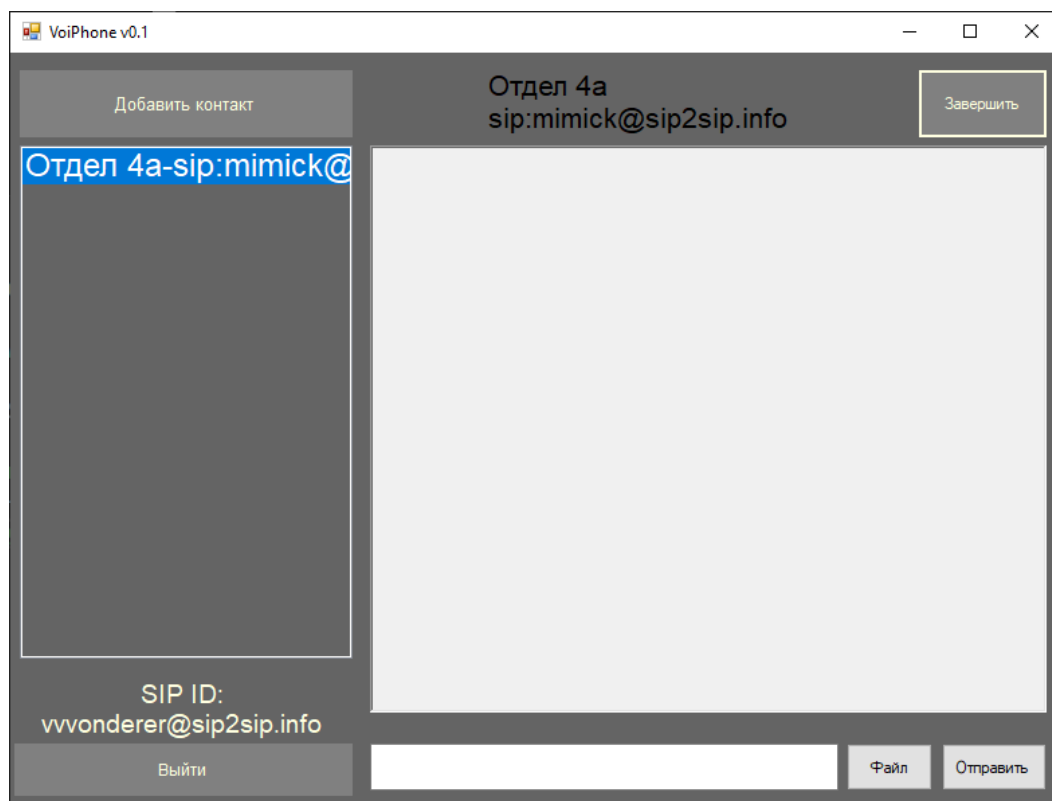


Рисунок 20 – Вызов клиента SIP

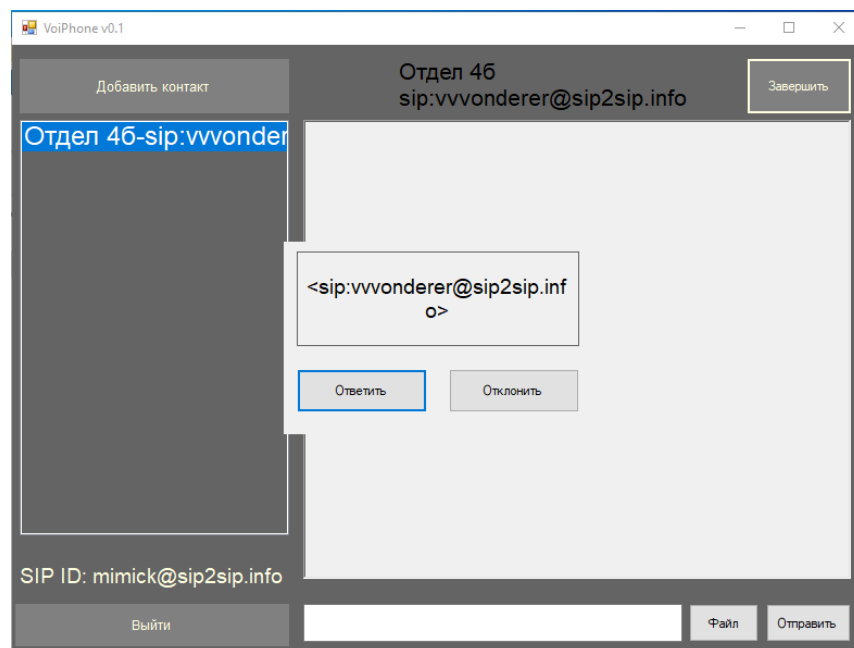


Рисунок 21 – Входящий вызов

Если нажать кнопку ответить то, в случае установки зашифрованной голосовой связи между двумя пользователями, в чате разговора появится сообщение об установке соединения, результат представлен на рисунке 22.

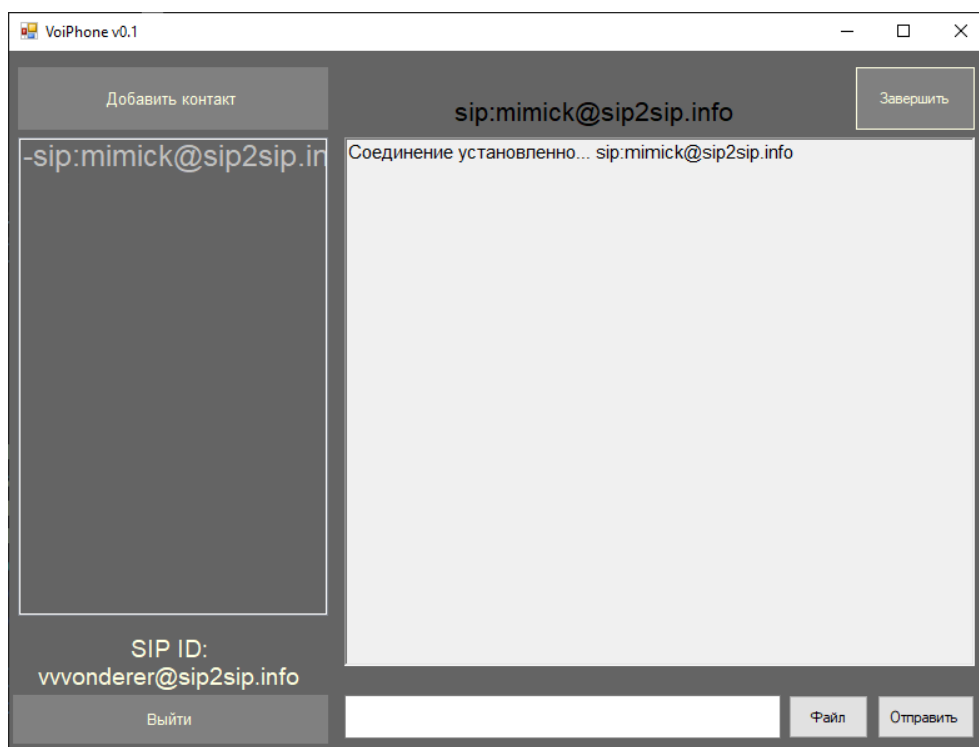


Рисунок 22 – Соединение установлено

Изм.	Лист	№ докум.	Подпись	Дата

ПГУ 1.090301.39.001 ПЗ

Лист

43

Чтобы закончить вызов нужно нажать на кнопку «Завершить» после чего воспроизведется звуковой сигнал, который сигнализирует о завершение вызова и в чате разговора появится сообщение о разрыве соединения, рисунок 23.

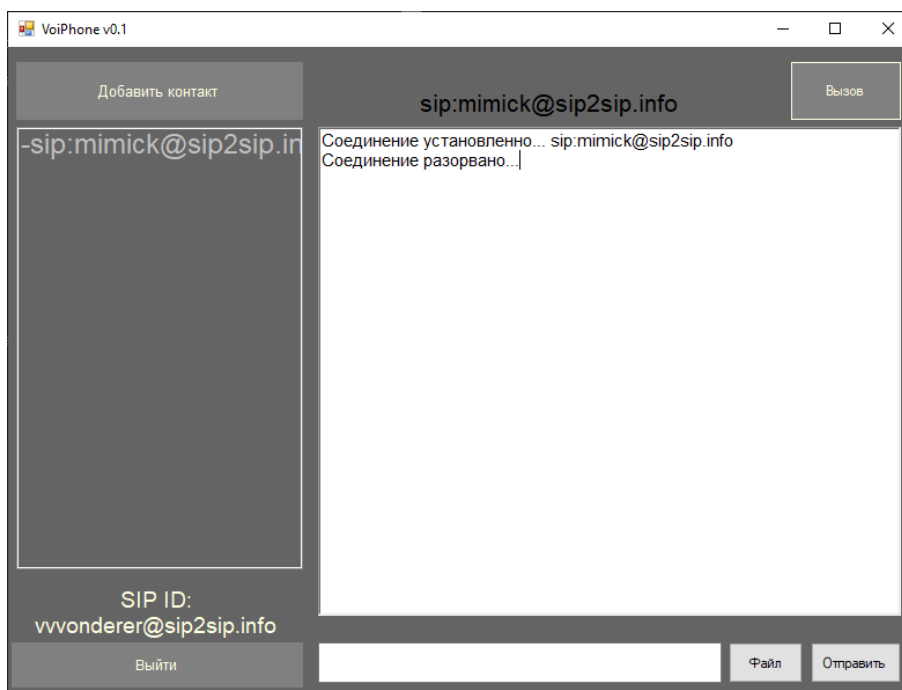


Рисунок 23 – Соединение разорвано

4.4 Отправка сообщений клиенту

Чтобы протестировать отправку сообщений клиенту нужно либо выбрать клиента из контактной книжки, либо отправить пользователю, с которым установлено соединение. Для тестирования будет отправлено сообщение пользователю, с которым установлено соединение, показано на рисунке 24 и рисунке 25.

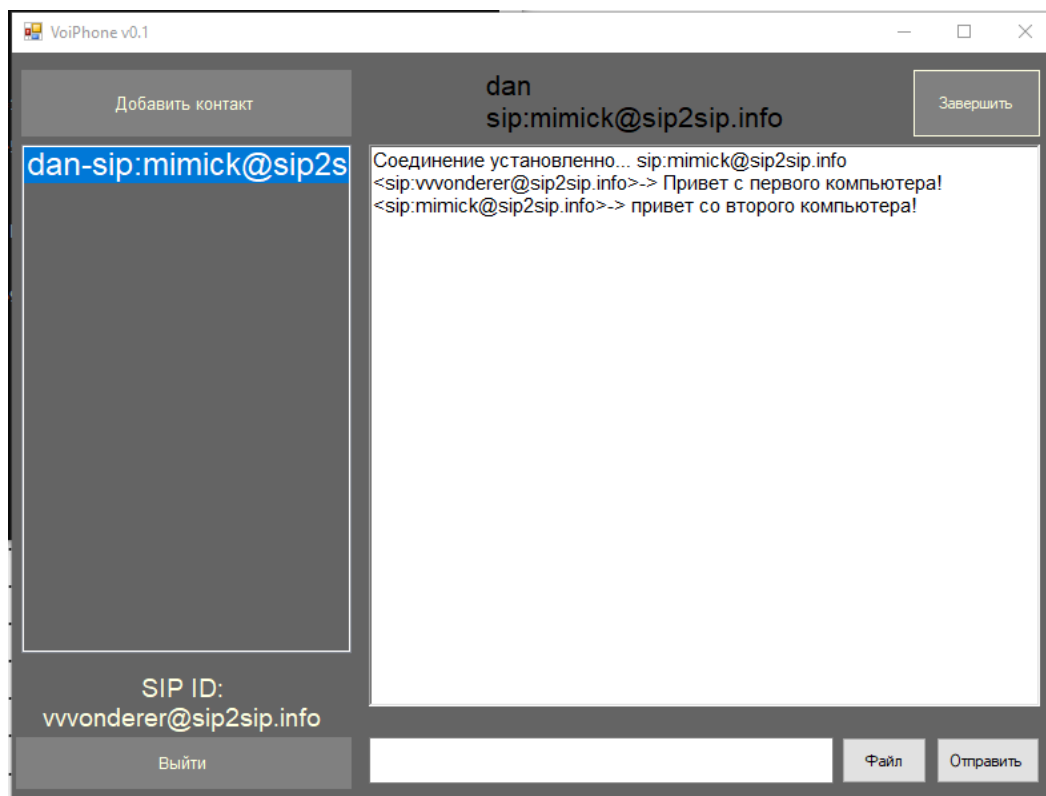


Рисунок 24 – Отправка сообщения пользователю, с которым установлено соединение

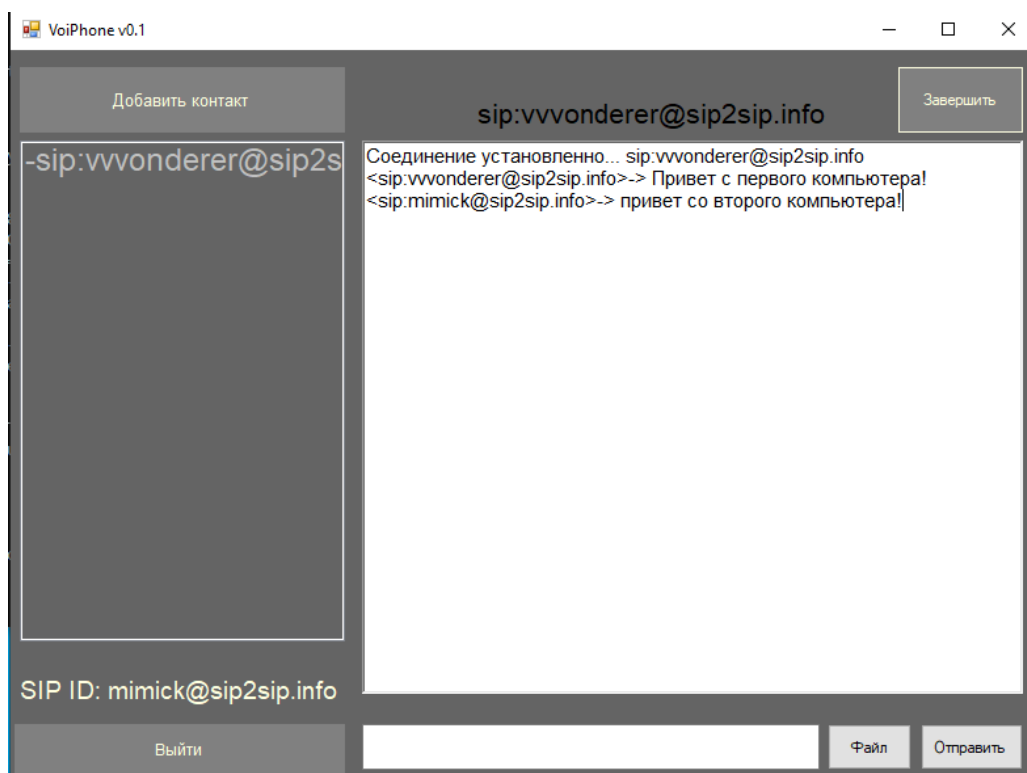


Рисунок 25 – Отправка сообщения пользователю, с которым установлено соединение

Так же для тестирования будет отправлено сообщение пользователю, с которым не установлено соединение, показано на рисунках 26 и 27.

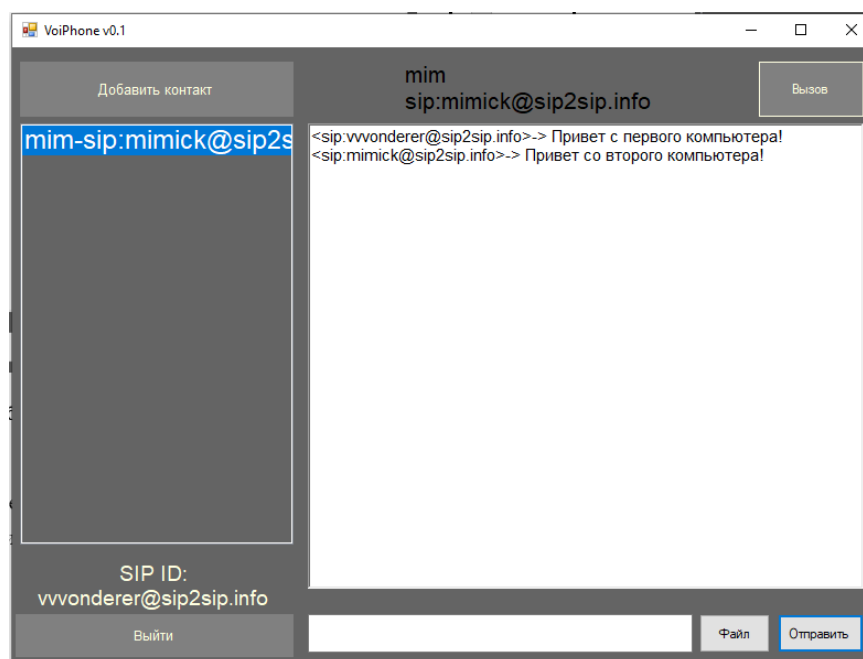


Рисунок 26 – Отправка сообщения пользователю, с которым не установлено соединение

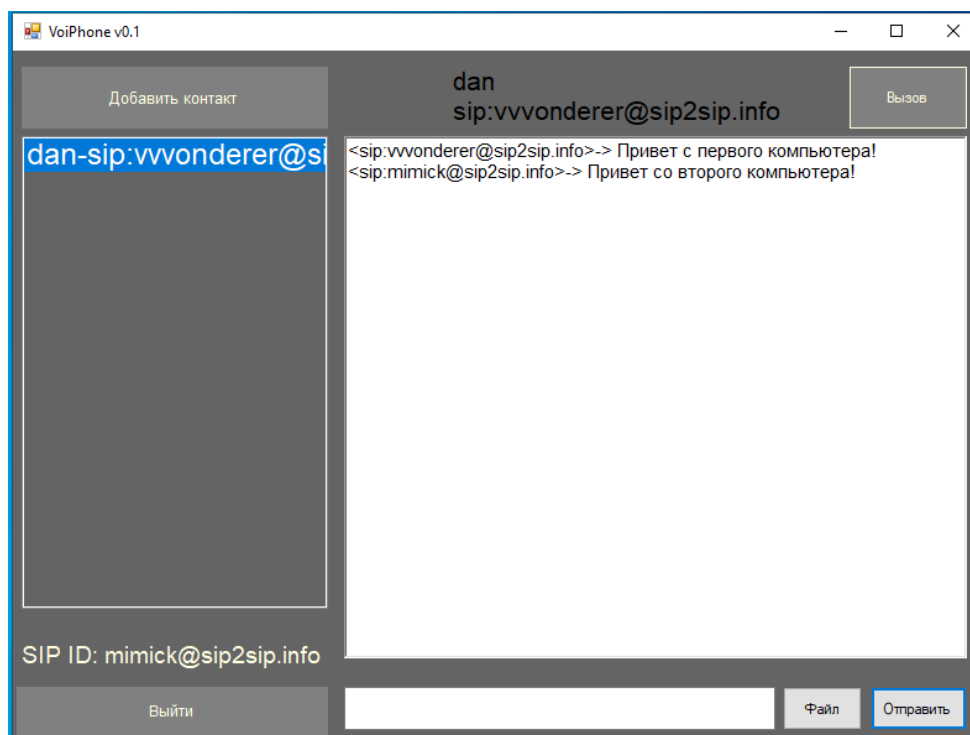


Рисунок 27 – Отправка сообщения пользователю, с которым не установлено соединение

4.5 Отправка файлов клиенту

Для проверки отправки файлов клиенту необходимо выбрать клиента из адресной книги. Например, в данном случае будет отправлен файл пользователю из контактной книжки, с которым соединение отсутствует, что продемонстрировано на рисунках 28 и 29.

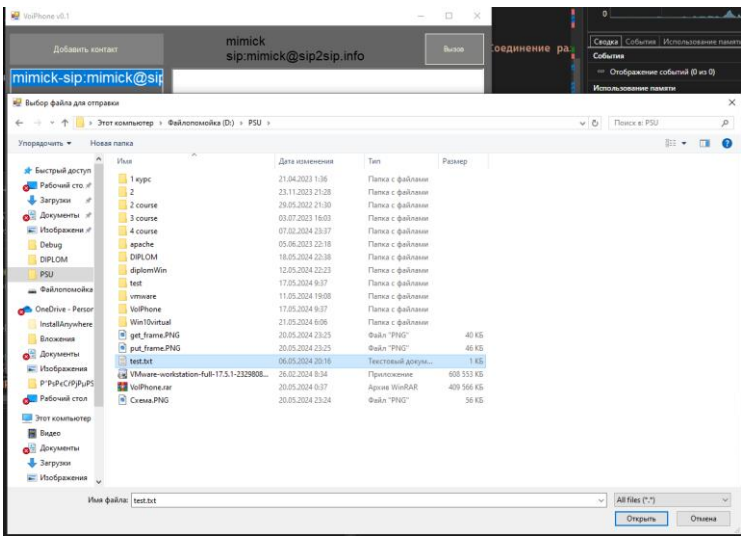


Рисунок 28 – Отправка сообщения пользователю, с которым соединение отсутствует

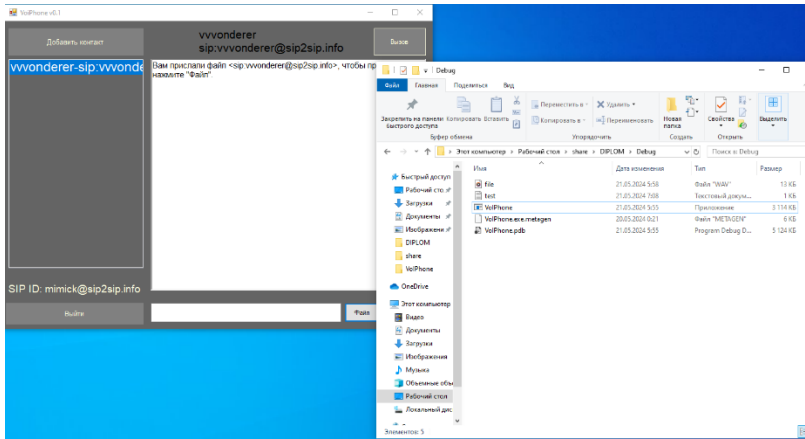


Рисунок 29 – Обработка отправленного файла на клиенте, которому был отправлен файл

Процесс тестирования прошел успешно, все функции приложения работают правильно, ожидаемый результат совпал с действительным, тестирование данных функций прошло успешно.

5 Безопасность жизнедеятельности

При взаимодействии с компьютером, необходимо соблюдать меры безопасности, чтобы не было неприятных последствий для здоровья после использования. Если пользователь ПК будет пренебрегать техникой безопасности, то проблемы со здоровьем будут неизбежны.

5.1 Вред здоровью при работе с ПК

Проблемы со здоровьем в процессе взаимодействия с компьютером имеют свойство накапливаться, то есть здоровье ухудшается не сразу, а постепенно. Наиболее опасными факторами являются:

- нагрузка на суставы. При неправильной посадке за компьютером, на суставы будет идти большая нагрузка, а именно на кисти рук, из-за чего может произойти онемение;
- нагрузка на спину. Опять же, если принять неправильно осанку при работе за компьютером, может появиться искривление позвоночника и даже грыжа межпозвоночного диска;
- нагрузка на органы зрения. При долгой работе с мониторами, будет ухудшаться здоровья глаз, впоследствии могут появиться синдромы «красного глаза» и «сухого глаза»;
- нагрузка на психику и нервную систему. При долгом взаимодействии с ПК, у пользователя могут возникать проблемы с памятью, а также может появиться депрессия и даже бессонница.

5.2 Снижение вреда для здоровья при работе с ПК

При соблюдении простых правил можно избежать серьезных проблем со здоровьем, для этого нужно:

- соблюдать необходимое расстояние до монитора. Правильным расстоянием до дисплея является расстояние, при котором пользователь может коснуться указательным пальцем до верхнего края монитора;

- соблюдать оптимальный угол экрана. Оптимальным углом является наклон монитора в 30 градусов;
- клавиатуру ставить в нужное положение. Клавиатура должна находиться на расстоянии 20-30 сантиметров от края стола;
- делать перерыв. Примерно каждый час необходимо делать разминку для органов зрения, а именно оторваться от монитора и делать гимнастику для глаз: рисовать различные фигуры глазами, фокусировать внимание на дальние объекты и переводить фокус с одних предметов на другие. Также примерно каждые полтора-два часа необходимо выполнять зарядку;
- выбрать правильный стул или кресло. При выборе данной мебели, необходимо приобрести такую, которая будет обеспечивать прямую осанку, при которой спина будет опираться в спинку.

5.3 Техника безопасности при работе с компьютером

При работе с компьютером необходимо соблюдать следующие правила, которые связаны с техникой безопасности:

- не оставлять компьютер без присмотра на длительное время;
- не использовать компьютер, который работает от питания без заземления;
- не работать за ПК, у которого есть повреждение в системном блоке или в его периферийных устройствах;
- не включать компьютер, если в помещении слишком высокая влажность;
- не класть на корпус компьютера или монитора посторонние предметы.

Заключение

В ходе выполнения данной выпускной квалификационной работы были получены навыки написания приложения для совершения вызовов, отправки сообщений и отправки файлов под операционную систему Windows, изучен стек технологий и библиотек позволяющих работать с IP-телефонией. Результатом работы является разработанное приложение «SIP-телефон», которое называется VoIPhone, а также отчёт, в котором представлены результаты тестирования разработанного программного обеспечения, описание разработки и подробный разбор его основных функциональных особенностей.

Разработанное приложение позволяет пользователю совершать и принимать вызовы, отправлять и принимать текстовые сообщения, а также отправлять и принимать файлы. После проведения тестирования было выяснено, что разработанное приложение соответствует всем поставленным в начале работы требованиям и работает корректно, не вызывая критических ошибок в системе, что является частой проблемой драйверов операционной системы Windows.

В дальнейшем приложение можно усовершенствовать путем добавления конференцсвязи, видео звонков, а также возможности синхронизации аккаунтов. Разработанный интерфейс пользователя можно улучшить путем усовершенствования дизайна для повышения комфорта использования.

Список использованных источников

1. Справочник по языку C++. [Электронный ресурс]/Режим доступа: <https://learn.microsoft.com/ru-ru/cpp/cpp/?view=msvc-170>, свободный. (дата обращения 23.04.2024).
2. Алина Крюкова. Труконф. Почему SIP лучше, чем H.323 [Электронный ресурс]/Режим доступа: <https://trueconf.ru/blog/reviews/pochemu-sip-luchshe-chem-h-323.html?ysclid=lxfxrwml2i200733351>, свободный. (дата обращения 23.04.2024).
3. Документация по PJSIP. [Электронный ресурс]/Режим доступа: <https://docs.pjsip.org/en/latest/overview/intro.html>, свободный. (дата обращения 20.04.2024).
4. Паттерны для новичков: MVC vs MVP vs MVVM. [Электронный ресурс]/Режим доступа: <https://habr.com/ru/articles/215605/>, свободный. (дата обращения 14.05.2024).

Приложение А

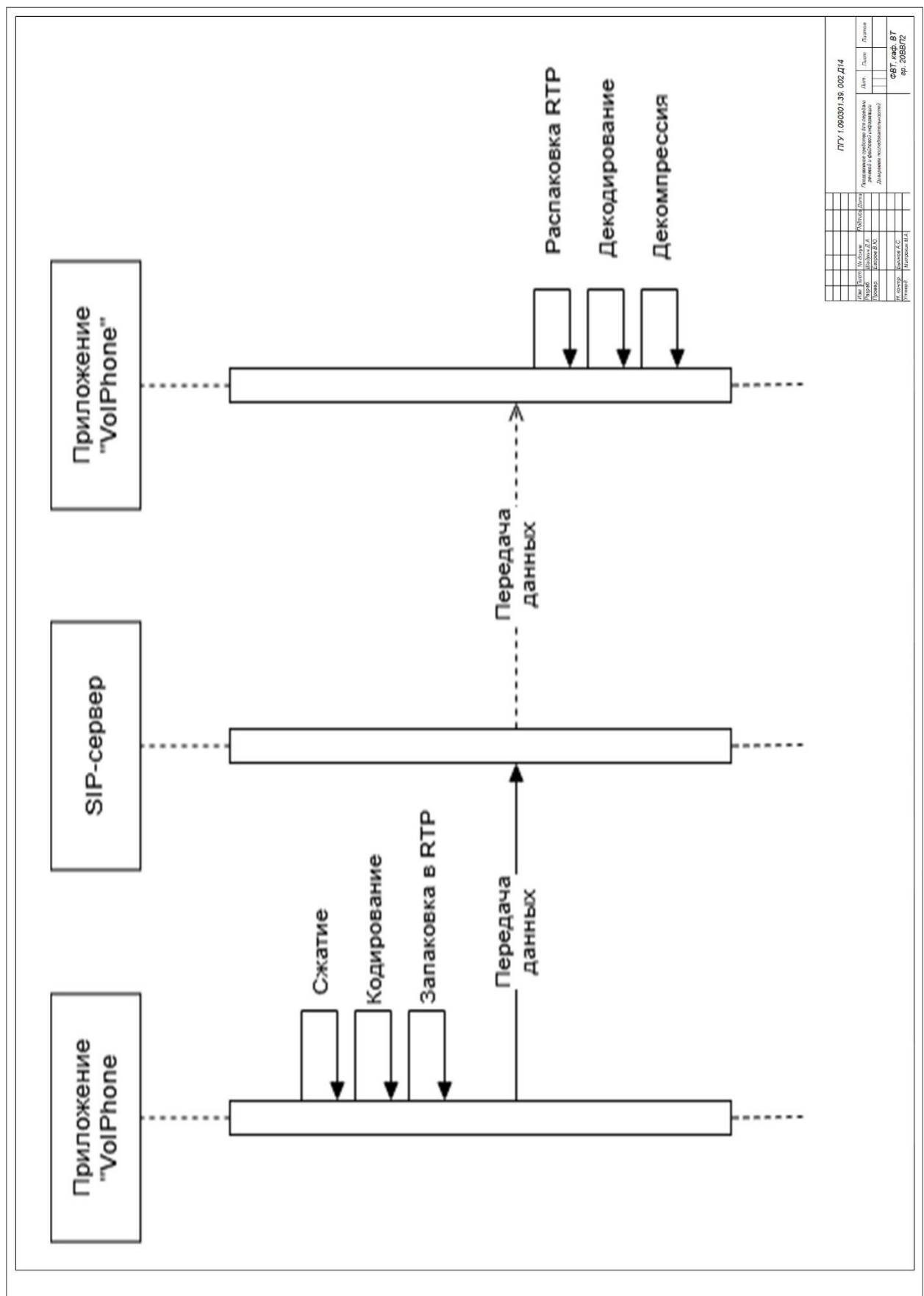
(Обязательное)

Программное средство для передачи речевой и файловой информации

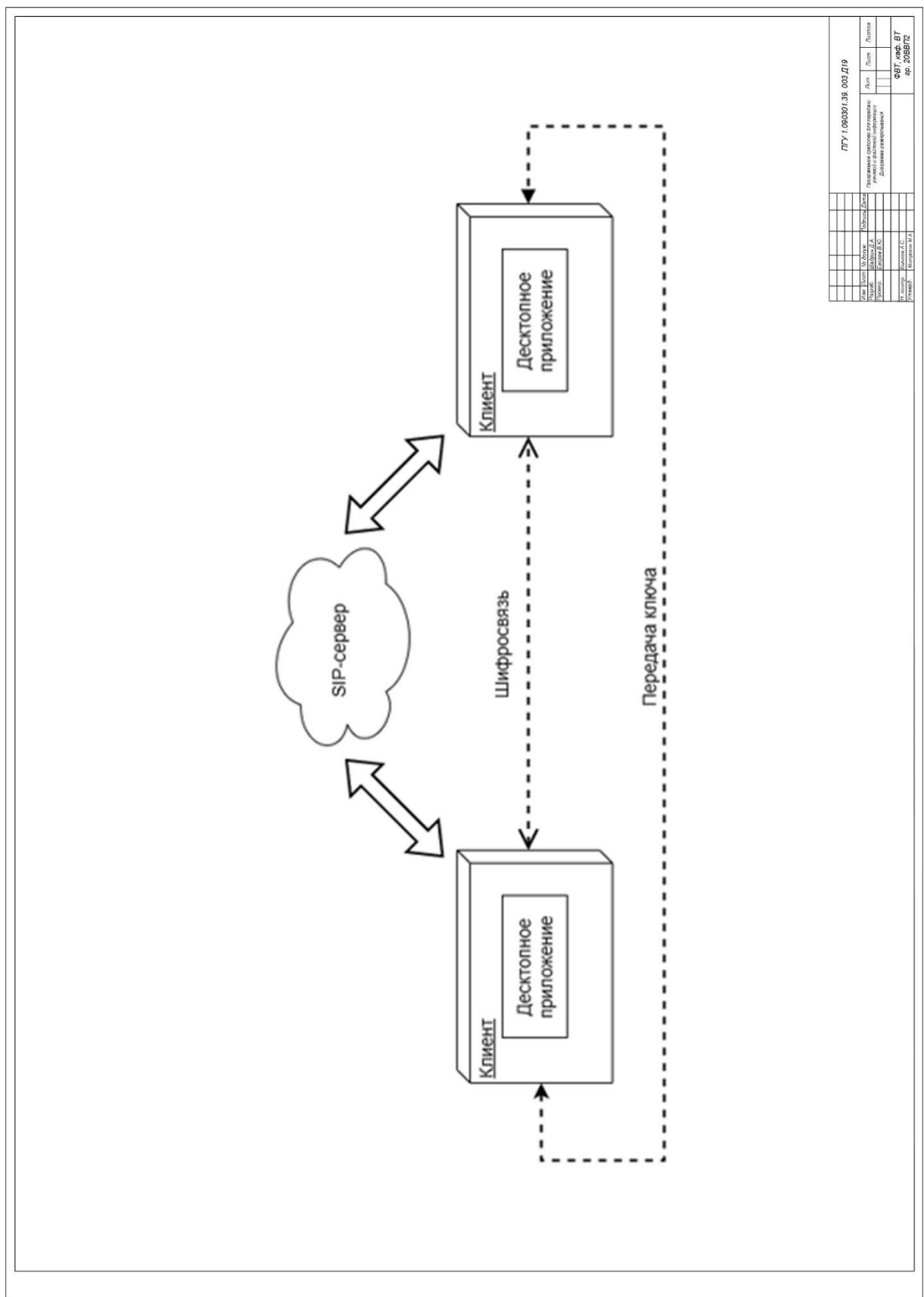
UML-диаграммы

					ПГУ 1.090301.39.001 ПЗ	Лист
						52
Изм.	Лист	№ докум.	Подпись	Дата		

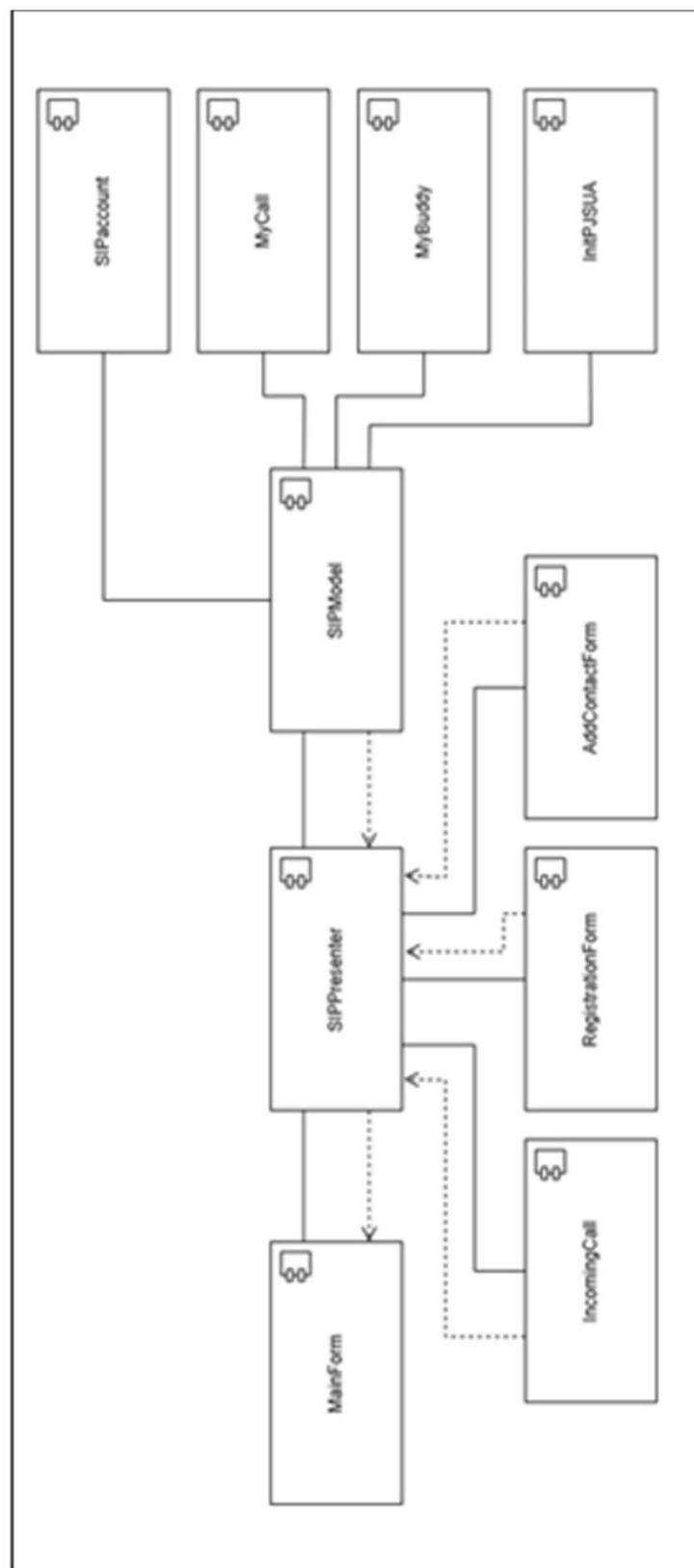
Приложение А.1-Диаграмма последовательностей



Приложение А.2-Диаграмма развертывания



Приложение А.3-Диаграмма компонентов

[illegible]

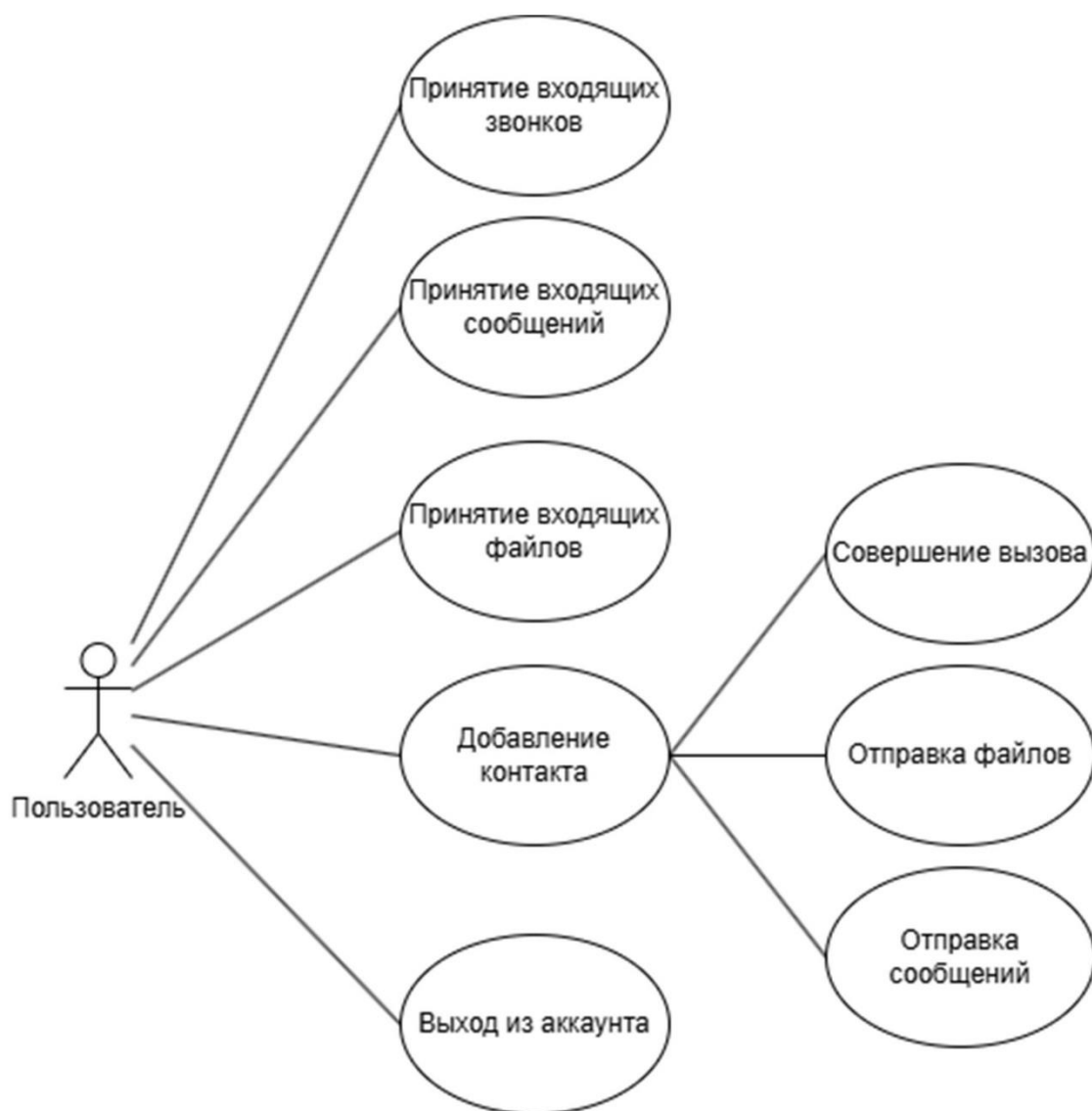
<i>Изм.</i>	<i>Лист</i>	<i>№ докум.</i>	<i>Подпись</i>	<i>Дата</i>

ПГУ 1,090301,39.001 ПЗ

Лист

55

Приложение А.4-Диаграмма вариантов использования



					ПГУ 1.090301.39.005 Д13		
Имя	Учлен	№ докум.	Подпись	Дата	Полномочные средства для проверки личности и фактически информации Добавлена графическая	Лист	Листов
Разработчик	Михайлов Д.А.						
Проверен	Григорьев В.Ю.						
И. контрол.	Рыжов А.В.					ФБТ, каф. ВТ	
Утвержден	Михайлов М.А.					сп. 20ВВП2	

Изм.	Лист	№ докум.	Подпись	Дата

ПГУ 1.090301.39.001 ПЗ

Лист

56

Приложение Б

(Обязательное)

Программное средство для передачи речевой и файловой информации

Листинг программы

					ПГУ 1.090301.39.001 ПЗ	Лист
						57
Изм.	Лист	№ докум.	Подпись	Дата		

Файл MainForm.hpp

```
#pragma once
#ifndef __MAINFORM_H
#define __MAINFORM_H
#include "SIPPresenter.h"
namespace VoIPhone {
    /// <summary>
    /// Сводка для MainForm
    /// </summary>
    enum StatusAcc {
        ACC_IS_REGISTERED,
        ACC_NOT_REGISTERED,
        ACC_CALL_ACTIVE,
        ACC_CALL_NO_ACTIVE
    };
    public ref class MainForm : public System::Windows::Forms::Form
    {
    public:
        MainForm(void)
        {
            InitializeComponent();
            gcnew SIPPresenter(this);
        }

    protected:
        /// <summary>
        /// Освободить все используемые ресурсы.
        /// </summary>
        ~MainForm()
        {
            if (components)
            {
                delete components;
            }
            destroyPJ();
        }

    public:
        System::Windows::Forms::ContextMenuStrip^ contextMenuStrip1;
        System::Windows::Forms::SplitContainer^ mainSplitContainer;
        System::Windows::Forms::Button^ loginRegButton;
        System::Windows::Forms::Label^ loginAccountInfo;
        System::Windows::Forms::Button^ addContactButton;
        System::Windows::Forms::ListBox^ contactsListBox;
        System::ComponentModel::IContainer^ components;
        System::Windows::Forms::RichTextBox^ chatRichTextBox;
        System::Windows::Forms::Label^ contactConInfoLabel;
        System::Windows::Forms::Button^ sendTextButton;
        System::Windows::Forms::Button^ sendFileButton;
        System::Windows::Forms::TextBox^ sendMessageTexBox;
        System::Windows::Forms::Button^ makeCallButton;

        int _inActiveCall = ACC_CALL_NO_ACTIVE;
        int _accRegFlag = ACC_NOT_REGISTERED;
        //Вызов входа в аккаунт
        event System::Action^ LoginRegButtonClicked;
        //Вызов выхода из аккаунта
        event System::Action^ ExitAccButtonClicked;
```

					ПГУ 1.090301.39.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		58

```

        //Вызов закрытия всех компонентов
        event System::Action^ destroyPJ;
        //Вызов добавления в контакты
        event System::Action^ addContactButtonClicked;
        //Вызов обновления выбранного контакта
        event System::Action^ updateSelectContact;
        //Вызов функции звонка
        event System::Action^ makeCall;
        //Отправка сообщений
        event System::Action^ sendMsg;
        //Отправка файла
        event System::Action^ sendFile;

#pragma region Windows Form Designer generated code
    /// <summary>
    /// Требуемый метод для поддержки конструктора – не
изменяйте
    /// содержимое этого метода с помощью редактора кода.
    /// </summary>
    void InitializeComponent(void)
    {
        this->mainSplitContainer = (gcnew
System::Windows::Forms::SplitContainer());
        this->addContactButton = (gcnew
System::Windows::Forms::Button());
        this->contactsListBox = (gcnew
System::Windows::Forms::ListBox());
        this->loginRegButton = (gcnew
System::Windows::Forms::Button());
        this->loginAccountInfo = (gcnew
System::Windows::Forms::Label());
        this->sendTextButton = (gcnew
System::Windows::Forms::Button());
        this->sendFileButton = (gcnew
System::Windows::Forms::Button());
        this->sendMessageTextBox = (gcnew
System::Windows::Forms::TextBox());
        this->makeCallButton = (gcnew
System::Windows::Forms::Button());
        this->contactConInfoLabel = (gcnew
System::Windows::Forms::Label());
        this->chatRichTextBox = (gcnew
System::Windows::Forms::RichTextBox());

        (cli::safe_cast<System::ComponentModel::ISupportInitialize>)(this
->mainSplitContainer)->BeginInit();
        this->mainSplitContainer->Panel1->SuspendLayout();
        this->mainSplitContainer->Panel2->SuspendLayout();
        this->mainSplitContainer->SuspendLayout();
        this->SuspendLayout();
        //
        // mainSplitContainer
        //
        this->mainSplitContainer->Dock =
System::Windows::Forms::DockStyle::Fill;
        this->mainSplitContainer->IsSplitterFixed = true;
        this->mainSplitContainer->Location =
System::Drawing::Point(0, 0);

```

Изм.	Лист	№ докум.	Подпись	Дата

```

        this->mainSplitContainer->Name =
L"mainSplitContainer";
        //
        // mainSplitContainer.Panel1
        //
        this->mainSplitContainer->Panel1->Controls->Add(this-
>addContactButton);
        this->mainSplitContainer->Panel1->Controls->Add(this-
>contactsListBox);
        this->mainSplitContainer->Panel1->Controls->Add(this-
>loginRegButton);
        this->mainSplitContainer->Panel1->Controls->Add(this-
>loginAccountInfo);
        //
        // mainSplitContainer.Panel2
        //
        this->mainSplitContainer->Panel2->Controls->Add(this-
>sendTextButton);
        this->mainSplitContainer->Panel2->Controls->Add(this-
>sendFileButton);
        this->mainSplitContainer->Panel2->Controls->Add(this-
>sendMessageTextBox);
        this->mainSplitContainer->Panel2->Controls->Add(this-
>makeCallButton);
        this->mainSplitContainer->Panel2->Controls->Add(this-
>contactConInfoLabel);
        this->mainSplitContainer->Panel2->Controls->Add(this-
>chatRichTextBox);
        this->mainSplitContainer->Size =
System::Drawing::Size(784, 561);
        this->mainSplitContainer->SplitterDistance = 261;
        this->mainSplitContainer->TabIndex = 0;
        //
        // addContactButton
        //
        this->addContactButton->BackColor =
System::Drawing::Color::Gray;
        this->addContactButton->FlatAppearance->BorderSize =
0;
        this->addContactButton->FlatStyle =
System::Windows::Forms::FlatStyle::Flat;
        this->addContactButton->Font = (gcnew
System::Drawing::Font(L"Arial", 9.75F,
System::Drawing::FontStyle::Regular,
System::Drawing::GraphicsUnit::Point,
        static_cast<System::Byte>(204)));
        this->addContactButton->ForeColor =
System::Drawing::SystemColors::Info;
        this->addContactButton->Location =
System::Drawing::Point(7, 13);
        this->addContactButton->Name = L"addContactButton";
        this->addContactButton->Size =
System::Drawing::Size(248, 50);
        this->addContactButton->TabIndex = 3;
        this->addContactButton->Text = L"Добавить контакт";
        this->addContactButton->UseVisualStyleBackColor =
false;

```

```

        this->addContactButton->Click += gcnew
System::EventHandler(this, &MainForm::addContactButton_Click);
        //
        // contactsListBox
        //
        this->contactsListBox->BackColor =
System::Drawing::SystemColors::WindowFrame;
        this->contactsListBox->Font = (gcnew
System::Drawing::Font(L"Arial", 18,
System::Drawing::FontStyle::Regular,
System::Drawing::GraphicsUnit::Point,
        static_cast<System::Byte>(204)));
        this->contactsListBox->ForeColor =
System::Drawing::Color::Silver;
        this->contactsListBox->FormattingEnabled = true;
        this->contactsListBox->ItemHeight = 27;
        this->contactsListBox->Location =
System::Drawing::Point(7, 69);
        this->contactsListBox->Name = L"contactsListBox";
        this->contactsListBox->Size =
System::Drawing::Size(248, 382);
        this->contactsListBox->TabIndex = 2;
        this->contactsListBox->DoubleClick += gcnew
System::EventHandler(this, &MainForm::contactsListBox_DoubleClick);
        //
        // loginRegButton
        //
        this->loginRegButton->BackColor =
System::Drawing::Color::Gray;
        this->loginRegButton->FlatAppearance->BorderSize = 0;
        this->loginRegButton->FlatStyle =
System::Windows::Forms::FlatStyle::Flat;
        this->loginRegButton->Font = (gcnew
System::Drawing::Font(L"Arial", 9,
System::Drawing::FontStyle::Regular,
System::Drawing::GraphicsUnit::Point,
        static_cast<System::Byte>(204)));
        this->loginRegButton->ForeColor =
System::Drawing::SystemColors::Info;
        this->loginRegButton->Location =
System::Drawing::Point(3, 514);
        this->loginRegButton->Name = L"loginRegButton";
        this->loginRegButton->Size =
System::Drawing::Size(252, 39);
        this->loginRegButton->TabIndex = 1;
        this->loginRegButton->Text = L"Войти";
        this->loginRegButton->UseVisualStyleBackColor = false;
        this->loginRegButton->Click += gcnew
System::EventHandler(this, &MainForm::loginRegButton_Click);
        //
        // loginAccountInfo
        //
        this->loginAccountInfo->Font = (gcnew
System::Drawing::Font(L"Arial", 14.25F,
System::Drawing::FontStyle::Regular,
System::Drawing::GraphicsUnit::Point,
        static_cast<System::Byte>(0)));

```

```

        this->loginAccountInfo->ForeColor =
System::Drawing::SystemColors::Info;
        this->loginAccountInfo->Location =
System::Drawing::Point(3, 465);
        this->loginAccountInfo->Name = L"loginAccountInfo";
        this->loginAccountInfo->Size =
System::Drawing::Size(252, 46);
        this->loginAccountInfo->TabIndex = 0;
        this->loginAccountInfo->Text = L"Войдите в SIP ";
        this->loginAccountInfo->TextAlign =
System::Drawing::ContentAlignment::MiddleCenter;
        //
        // sendTextButton
        //
        this->sendTextButton->Location =
System::Drawing::Point(429, 514);
        this->sendTextButton->Name = L"sendTextButton";
        this->sendTextButton->Size = System::Drawing::Size(78,
35);
        this->sendTextButton->TabIndex = 5;
        this->sendTextButton->Text = L"Отправить";
        this->sendTextButton->UseVisualStyleBackColor = true;
        this->sendTextButton->Click += gnew
System::EventHandler(this, &MainForm::sendTextButton_Click);
        //
        // sendFileButton
        //
        this->sendFileButton->Location =
System::Drawing::Point(358, 514);
        this->sendFileButton->Name = L"sendFileButton";
        this->sendFileButton->Size = System::Drawing::Size(64,
35);
        this->sendFileButton->TabIndex = 4;
        this->sendFileButton->Text = L"Файл";
        this->sendFileButton->UseVisualStyleBackColor = true;
        this->sendFileButton->Click += gnew
System::EventHandler(this, &MainForm::sendFileButton_Click);
        //
        // sendMessageTextBox
        //
        this->sendMessageTextBox->Location =
System::Drawing::Point(3, 514);
        this->sendMessageTextBox->Multiline = true;
        this->sendMessageTextBox->Name = L"sendMessageTextBox";
        this->sendMessageTextBox->Size =
System::Drawing::Size(349, 35);
        this->sendMessageTextBox->TabIndex = 3;
        //
        // makeCallButton
        //
        this->makeCallButton->BackColor =
System::Drawing::Color::Gray;
        this->makeCallButton->FlatStyle =
System::Windows::Forms::FlatStyle::Flat;
        this->makeCallButton->ForeColor =
System::Drawing::SystemColors::Info;
        this->makeCallButton->Location =
System::Drawing::Point(412, 13);

```

```

        this->makeCallButton->Name = L"makeCallButton";
        this->makeCallButton->Size = System::Drawing::Size(95,
50);

        this->makeCallButton->TabIndex = 2;
        this->makeCallButton->Text = L"Вызов";
        this->makeCallButton->UseVisualStyleBackColor = false;
        this->makeCallButton->Click += gcnew
System::EventHandler(this, &MainForm::makeCallButton_Click);
        //
        // contactConInfoLabel
        //
        this->contactConInfoLabel->Font = (gcnew
System::Drawing::Font(L"Arial", 15.75F,
System::Drawing::FontStyle::Regular,
System::Drawing::GraphicsUnit::Point,
        static_cast<System::Byte>(204)));
        this->contactConInfoLabel->Location =
System::Drawing::Point(87, 13);
        this->contactConInfoLabel->Name =
L"contactConInfoLabel";
        this->contactConInfoLabel->Size =
System::Drawing::Size(306, 50);
        this->contactConInfoLabel->TabIndex = 1;
        //
        // chatRichTextBox
        //
        this->chatRichTextBox->Font = (gcnew
System::Drawing::Font(L"Arial", 11.25F,
System::Drawing::FontStyle::Regular,
System::Drawing::GraphicsUnit::Point,
        static_cast<System::Byte>(204)));
        this->chatRichTextBox->Location =
System::Drawing::Point(3, 69);
        this->chatRichTextBox->Name = L"chatRichTextBox";
        this->chatRichTextBox->Size =
System::Drawing::Size(504, 422);
        this->chatRichTextBox->TabIndex = 0;
        this->chatRichTextBox->Text = L"";
        //
        // MainForm
        //
        this->AutoScaleDimensions = System::Drawing::SizeF(6,
13);

        this->AutoScaleMode =
System::Windows::Forms::AutoScaleMode::Font;
        this->BackColor =
System::Drawing::SystemColors::WindowFrame;
        this->ClientSize = System::Drawing::Size(784, 561);
        this->Controls->Add(this->mainSplitContainer);
        this->Name = L"MainForm";
        this->StartPosition =
System::Windows::Forms::FormStartPosition::CenterScreen;
        this->Text = L"VoIPPhone v0.1";
        this->mainSplitContainer->Panel1->ResumeLayout(false);
        this->mainSplitContainer->Panel2->ResumeLayout(false);
        this->mainSplitContainer->Panel2->PerformLayout();

```

```

        (cli::safe_cast<System::ComponentModel::ISupportInitialize>(this
->mainSplitContainer))->EndInit();
        this->mainSplitContainer->ResumeLayout(false);
        this->ResumeLayout(false);

    }
#pragma endregion
#pragma region FormsButtons
    private:
        System::Void loginRegButton_Click(System::Object^ sender,
System::EventArgs^ e) {
            LoginRegButtonClicked();
        }
        //Кнопка добавления контакта
        System::Void addContactButton_Click(System::Object^ sender,
System::EventArgs^ e) {
            addContactButtonClicked();
        }
        //Кнопка обновления информации о контактке
        System::Void contactsListBox_DoubleClick(System::Object^
sender, System::EventArgs^ e) {
            updateSelectContact();
        }
        //Кнопка вызова контакта
        System::Void makeCallButton_Click(System::Object^ sender,
System::EventArgs^ e) {
            makeCall();
        }
        //Кнопка отправки сообщения
        System::Void sendTextButton_Click(System::Object^ sender,
System::EventArgs^ e) {
            sendMsg();
        }
        System::Void sendFileButton_Click(System::Object^ sender,
System::EventArgs^ e) {
            sendFile();
        }
    };
#pragma endregion
}
#endif // !__MAINFORM_H

```

Файл MainForm.cpp

```

#include "MainForm.h"
#include <locale>
using namespace System;
using namespace System::Windows::Forms;

[STAThreadAttribute]
int main(array<String^>^ args) {
    setlocale(LC_ALL, "Rus");
    Application::SetCompatibleTextRenderingDefault(false);
    Application::EnableVisualStyles();
    VoIPhone::MainForm mainForm;
    Application::Run(%mainForm); //Запуск главного окна
}

```

Файл SIPModel.hpp

```
#pragma once
#ifndef __SIPMODEL_H
#define __SIPMODEL_H
#include "MyCall.h"
#include "SIPAccount.h"
#include "InitPJSUA.h"
#include "MyBuddy.h"
#include "regform/RegistrationForm.h"
#include "FriendsManager.h"
#include "AddContactForm/AddContactForm.h"
#include <thread>
namespace VoIPhone {
    ref class SIPPresenter;
    struct uData {
        std::string sipServ;
        std::string sipProxy;
        std::string sipDomain;
        std::string sipUsername;
        std::string sipPassword;
    };

    struct callData {
        pj::Call* call;
        pj::CallOpParam* prm;
    };

    class SIPModel {
    public:
        SIPModel() {}
        //Вызывается при входящем звонке
        SIPModel(SIPAccount* acc) {
            this->acc = acc;
        }
        ~SIPModel() {

        }

        //Инициализирует библиотеку PJSUA2
        bool initPJ(); //done
        //Деинициализирует библиотеку PJSUA2
        bool destroyPJ(); //done
        //Выполняет регистрацию аккаунта
        bool loginAccount(System::Collections::ArrayList^ destination,
SIPPresenter^ pres); //done
        //Выполняет создание Buddy
        bool buddyCreate(); //done
        //Выход из аккаунта
        bool exitAccount(); //done
        //Совершает вызов(System::String^)
        void makeCallSs(System::String^ destination); //done
        //Совершает вызов(std::string) больше не нужна
        callData makeCallNs(std::string destination);
        void CallThread(std::string callAddress, pj::Call* call,
pj::CallOpParam prm); //done
        //Сбрасывает все вызовы
        void hangupCalls(); //done
        //Отправляет сообщение
```

```

        void sendMessage(System::String^ msg, System::String^ fromUri)
const; //done
        //Добавление в контактную книжку
        System::String^ addContact(); //done
private:

        MyBuddy* buddy = nullptr;
        InitPJSUA core;
        // FriendsManager friends;
        class SIPaccount* acc = nullptr;
        bool _callActive = 0;
    };
}
#endif // !__SIPMODEL_H

```

Файл SIPModel.cpp

```

#include "SIPModel.h"
#include "pjsip/sip_msg.h"
#include "SIPPresenter.h"
bool VoIPhone::SIPModel::initPJ()
{
    pj::EpConfig ep_cfg;
    pj::TransportConfig tcfg;
    tcfg.port = 5060;
    try {
        core.libCreate();
        core.libInit(ep_cfg);
        core.transportCreate(PJSIP_TRANSPORT_UDP, tcfg);
        core.libStart();
        core.libRegisterThread("modelThread");
    }
    catch (pj::Error& err) {
        System::Console::WriteLine("Инициализация библиотеки
завершилась с ошибкой \r\n Error #" + gcnew
System::String(err.info().data()));
        return false;
    }
    return true;
}
bool VoIPhone::SIPModel::destroyPJ()
{
    try {
        core.hangupAllCalls();
        if (acc != nullptr) {
            acc->shutdown();
            delete acc;
        }
        if (this->exitAccount()) {
            core.libStopWorkerThreads();
            core.libDestroy();
        }
    }
    catch (pj::Error& err) {
        System::Console::WriteLine("Деинициализация библиотеки
завершилась с ошибкой \r\n Error #" + gcnew
System::String(err.info().data()));
    }
}

```

Изм.	Лист	№ докум.	Подпись	Дата

ПГУ 1.090301.39.001 ПЗ

Лист

66

```

        return false;
    }
    return true;
}

bool VoIPhone::SIPModel::loginAccount(System::Collections::ArrayList^
destination, SIPPresenter^ pres)
{
    //acc = new SIPaccount();
    acc = new SIPaccount(pres);
    System::Collections::Generic::List<System::String^>^ userData =
gcnew System::Collections::Generic::List<System::String^>();
    for each (auto item in destination)
    {
        userData->Add(dynamic_cast<System::String^>(item));
    }

    uData userDataNString;
    userDataNString.sipServ = (const
char*)(System::Runtime::InteropServices::Marshal::StringToHGlobalAnsi(
userData[0])).ToPointer();
    userDataNString.sipProxy = (const
char*)(System::Runtime::InteropServices::Marshal::StringToHGlobalAnsi(
userData[1])).ToPointer();
    userDataNString.sipDomain = (const
char*)(System::Runtime::InteropServices::Marshal::StringToHGlobalAnsi(
userData[2])).ToPointer();
    userDataNString.sipUsername = (const
char*)(System::Runtime::InteropServices::Marshal::StringToHGlobalAnsi(
userData[3])).ToPointer();
    userDataNString.sipPassword = (const
char*)(System::Runtime::InteropServices::Marshal::StringToHGlobalAnsi(
userData[4])).ToPointer();

    pj::AccountConfig acc_cfg;
    acc_cfg.idUri = "sip:" +
userDataNString.sipUsername + "@" + userDataNString.sipServ;
    acc_cfg.regConfig.registrarUri = "sip:" +
userDataNString.sipServ;
    acc_cfg.sipConfig.proxies.push_back("sip:" +
userDataNString.sipProxy);
    acc_cfg.sipConfig.authCreds.push_back(pj::AuthCredInfo("Digets",
"*, userDataNString.sipUsername, PJSIP_CRED_DATA_PLAIN_PASSWD,
userDataNString.sipPassword));
    try {
        acc->create(acc_cfg, true);
        this->buddyCreate();
    }
    catch (pj::Error& err) {
        System::Console::WriteLine("Авторизация аккаунта завершилась
с ошибкой(Create) \r\n Error #" + gcnew
System::String(err.info().data()));
        return false;
    }
    if (!acc->isValid()) {
        System::Console::WriteLine("Error in valid account");
        return false;
    }
    else {

```

```

        try {
            pj::PresenceStatus ps;
            ps.status = PJSUA_BUDDY_STATUS_ONLINE;
            // Optional, set the activity and some note
            ps.activity = PJRPID_ACTIVITY_BUSY;
            ps.note = "On the phone";
            acc->setOnlineStatus(ps);
        }
        catch (pj::Error& err) {
            System::Console::WriteLine("Инициализация библиотеки
завершилась с ошибкой(PresenceStatus) \r\n Error #" + gcnew
System::String(err.info().data()));
            return false;
        }
    }
    return true;
}

bool VoIPhone::SIPModel::buddyCreate()
{
    buddy = new MyBuddy;
    pj::BuddyConfig bdd_cfg;
    pj::AccountInfo acc_inf;
    acc_inf = acc->getInfo();
    bdd_cfg.uri = acc_inf.uri;
    try {
        buddy->create(*acc, bdd_cfg);
        buddy->subscribePresence(true);
    }
    catch (pj::Error& err) {
        System::Windows::Forms::MessageBox::Show("Инициализация
библиотеки завершилась с ошибкой(buddyCreate) \r\n Error #" + gcnew
System::String(err.info().data()));
        return false;
    }
    return true;
}

bool VoIPhone::SIPModel::exitAccount()
{
    try {
        if (acc != NULL) {
            acc->shutdown();
        }
    }
    catch (pj::Error& err) {
        System::Windows::Forms::MessageBox::Show("Выход из аккаунта
завершился с ошибкой \r\n Error #" + gcnew
System::String(err.info().data()));
        return false;
    }
    return true;
}

void VoIPhone::SIPModel::makeCalls(System::String^ destination)
{
    int startIndex = destination->IndexOf("\r\n");
    String^ resultString = destination->Substring(startIndex + 2);
    const char* chAddress = (const
char*)(System::Runtime::InteropServices::Marshal::StringToHGlobalAnsi(
resultString)).ToPointer();

```

```

        std::string userName = chAddress;
        System::Runtime::InteropServices::Marshal::FreeHGlobal(IntPtr((void*)chAddress));

        pj::Call* call = new MyCall(*acc);
        acc->calls.push_back(call);
        pj::CallOpParam prm(true);
        prm.opt.audioCount = 1;
        prm.opt.videoCount = 0;

        std::thread callTh([this, userName, call, prm] {
            CallThread(userName, call, prm);
        });
        callTh.detach();
    }
    VoIPhone::callData VoIPhone::SIPModel::makeCallNs(std::string
    destination)
    {
        VoIPhone::callData callData = {};
        pj::Call* call = new MyCall(*acc);
        acc->calls.push_back(call);
        pj::CallOpParam prm(true);
        prm.opt.audioCount = 1;
        prm.opt.videoCount = 0;
        std::string userName = destination;
        call->makeCall(destination, prm);
        callData.call = call;
        callData.prm = (pj::CallOpParam*)&prm;
        return callData;
    }
    void VoIPhone::SIPModel::CallThread(std::string callAddress, pj::Call*
    call, pj::CallOpParam prm)
    {
        pj_thread_desc desc;
        pj_thread_t* thread = 0;
        pj_thread_register("callIs", desc, &thread);

        try {
            call->makeCall(callAddress, prm);
        }
        catch (pj::Error& err) {
            std::cout << "Error with call " + err.info() << std::endl;
            acc->calls.pop_back();
            return;
        }
        _callActive = true;
    }
    void VoIPhone::SIPModel::hangupCalls()
    {
        try {
            for (size_t i = 0; i < acc->calls.size(); ++i)
                acc->calls.pop_back();
        }
        catch(...){}
        core.hangupAllCalls();
        _callActive = false;
    }

```

```

void VoIPhone::SIPModel::sendMessage(System::String^ msg,
System::String^ fromUri) const
{
    int          startIndex      = fromUri->IndexOf("\r\n");
    String^      resultString    = fromUri->Substring(startIndex + 2);
    const char*  chAddress       = (const
char*)(System::Runtime::InteropServices::Marshal::StringToHGlobalAnsi(
resultString)).ToPointer();
    //const char* chAddress       = (const
char*)(System::Runtime::InteropServices::Marshal::StringToHGlobalAnsi(
fromUri)).ToPointer();
    const char*  chMsg           = (const
char*)(System::Runtime::InteropServices::Marshal::StringToHGlobalAnsi(
msg)).ToPointer();
    std::string  userName        = chAddress;
    pj::BuddyConfig cfg;
    cfg.uri = userName;
    pj::Buddy*  buddy = new pj::Buddy();
    buddy->create(*acc, cfg);
    pj::SendInstantMessageParam simp;
    simp.content = chMsg;
    System::Runtime::InteropServices::Marshal::FreeHGlobal(IntPtr((vo
id*)chAddress));
    System::Runtime::InteropServices::Marshal::FreeHGlobal(IntPtr((vo
id*)chMsg));
    try {
        buddy->sendInstantMessage(simp);
    }
    catch (pj::Error& err) {
        std::cout << err.info() << std::endl;
    }
    delete buddy;
}

System::String^ VoIPhone::SIPModel::addContact()
{
    AddContactForm^ addContact = gcnew AddContactForm();
    addContact->ShowDialog();
    System::Collections::Generic::List<System::String^>^ contactInfo
= addContact->getInfo();
    if (contactInfo != nullptr) {
        //_friendsManager->addFriend(contactInfo[0],
contactInfo[1]);
        System::String^ req = contactInfo[0] + "-" + contactInfo[1];
        //System::Windows::Forms::MessageBox::Show("Адреса
пользователя не существует");
        return req;
    }
    return nullptr;
}

```

Файл SIPPresenter.cpp

```

#pragma once
#ifdef __SIPPRESENTER_H
#define __SIPPRESENTER_H
#include "SIPModel.h"
namespace VoIPhone {
    class SIPModel;
}

```

					ПГУ 1.090301.39.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		70

```

ref class MainForm;
ref class SIPPresenter{
public:
    SIPPresenter(MainForm^ v);
    void OnLoginRegButtonClicked();
    void OnExitButtonClicked();
    void OnAddContactButtonClicked();
    void destroyLib();
    void OnUpdateSelectedContact();
    void OnMakeCallButtonClicked();
    void OnSendMessageButtonClicked();
    void OnSendFileButtonClicked();
    void updateChatMsgTextBox(System::String^ url, System::String^
msg);
    void updateChat(array<Object>^ args);
    void UpdateChatHelper();
    array<Object>^ GetArgs();
    void answerCall(System::String^ uri);
    void updateCall();
private:
    array<Object>^ incMsg;
    array<Object>^ incCall;
    SIPModel* model;
    MainForm^ view;
    bool _destroy = 0;
};
}
#endif // !__SIPPRESENTER_H

```

Файл SIPPresenter.cpp

```

#include "SIPPresenter.h"
#include "MainForm.h"
#include "IncomigCallForm.h"
#include "SIPModel.h"
#include "SIPaccount.h"
VoIPhone::SIPPresenter::SIPPresenter(MainForm ^ v) {
    model = new SIPModel;
    view = v;
    if (model->initPJ()) {
        view->sendFile += gcnew
System::Action(this, &SIPPresenter::OnSendFileButtonClicked);
        view->sendMsg += gcnew
System::Action(this, &SIPPresenter::OnSendMessageButtonClicked);
        view->makeCall += gcnew
System::Action(this, &SIPPresenter::OnMakeCallButtonClicked);
        view->updateSelectContact += gcnew
System::Action(this, &SIPPresenter::OnUpdateSelectedContact);
        view->addContactButtonClicked += gcnew
System::Action(this, &SIPPresenter::OnAddContactButtonClicked);
        view->destroyPJ += gcnew
System::Action(this, &SIPPresenter::destroyLib);
        view->LoginRegButtonClicked += gcnew
System::Action(this, &SIPPresenter::OnLoginRegButtonClicked);
        view->ExitAccButtonClicked += gcnew
System::Action(this, &SIPPresenter::OnExitButtonClicked);
    }
    else

```

Изм.	Лист	№ докум.	Подпись	Дата

```

        System::Windows::Forms::MessageBox::Show("Ошибка инициализации
PJSIP");
    }
    void VoIPhone::SIPPresenter::OnLoginRegButtonClicked() {
        if (view->_accRegFlag == ACC_NOT_REGISTERED) {
            RegistrationForm^ regForm = gcnew RegistrationForm();
            regForm->ShowDialog();
            ArrayList^ accountInfo = regForm->getData();
            if (model->loginAccount(accountInfo, this)) {
                view->loginRegButton->Text = L"Выйти";
                view->loginAccountInfo->Text = L"SIP ID: " +
accountInfo[3]->ToString() + L"@" + accountInfo[2]->ToString();
                view->_accRegFlag = ACC_IS_REGISTERED;
            }
            else
                System::Windows::Forms::MessageBox::Show("Введен
неправильный логин или пароль!");
        }
        else if (view->_accRegFlag == ACC_IS_REGISTERED) {
            if (MessageBox::Show(
                L"Вы действительно хотите выйти?",
                L"Выход",
                MessageBoxButtons::YesNo,
                MessageBoxIcon::Exclamation
            ) == System::Windows::Forms::DialogResult::Yes) {
                OnExitButtonClicked();
            }
        }
    }
    void VoIPhone::SIPPresenter::OnExitButtonClicked() {
        if (model->exitAccount()) {
            view->loginRegButton->Text = L"Войти";
            view->loginAccountInfo->Text = L"Войдите в SIP";
            view->_accRegFlag = ACC_NOT_REGISTERED;
            view->contactsListBox->Items->Clear();
            view->contactConInfoLabel->Text = "";
        }
    }
    void VoIPhone::SIPPresenter::OnAddContactButtonClicked()
    {
        if (view->_accRegFlag == ACC_IS_REGISTERED) {
            view->contactsListBox->Items->Add(model->addContact());
        }
        else if (view->_accRegFlag == ACC_NOT_REGISTERED) {
            System::Windows::Forms::MessageBox::Show("Для начала войдите в
аккаунт");
        }
    }
    void VoIPhone::SIPPresenter::destroyLib()
    {
        if (!this->_destroy)
            model->destroyPJ();
        this->_destroy = 1;
    }
    void VoIPhone::SIPPresenter::OnUpdateSelectedContact()
    {
        if (view->contactsListBox->SelectedItem != nullptr) {
            view->contactConInfoLabel->Text = "";
        }
    }

```

Изм.	Лист	№ докум.	Подпись	Дата

```

        auto tx = view->contactsListBox->SelectedItem->ToString();
        array<System::String^>^ parts = tx->Split('-');
        for each (String ^ %part in parts) {
            part = part->Substring(0, 1) + part->Substring(1);
        }
        System::String^ correctedString = System::String::Join("\r\n",
parts);
        view->contactConInfoLabel->Text = correctedString;
        view->chatRichTextBox->Clear();
    }
}
void VoIPhone::SIPPresenter::OnMakeCallButtonClicked()
{
    if (view->_accRegFlag == ACC_IS_REGISTERED) {
        if (view->_inActiveCall == ACC_CALL_NO_ACTIVE) {
            if (view->contactConInfoLabel->Text != "") {
                model->makeCallSs(view->contactConInfoLabel->Text-
>ToString());
                view->_inActiveCall = ACC_CALL_ACTIVE;
                view->makeCallButton->Text = "Завершить";
            }
            else
                System::Windows::Forms::MessageBox::Show("Вы не
выбрали кому звонить");
        }
        else if (view->_inActiveCall == ACC_CALL_ACTIVE) {
            if (view->chatRichTextBox->Text != "") {
                view->chatRichTextBox->SelectedText = view-
>chatRichTextBox->Text + System::Environment::NewLine + "Соединение
разорвано...";
            }
            model->hangupCalls();
            view->_inActiveCall = ACC_CALL_NO_ACTIVE;
            view->makeCallButton->Text = "Вызов";
        }
    }
    else if (view->_accRegFlag == ACC_NOT_REGISTERED)
        System::Windows::Forms::MessageBox::Show("Для начала войдите в
аккаунт");
}
void VoIPhone::SIPPresenter::OnSendMessageButtonClicked()
{
    if (view->_accRegFlag == ACC_IS_REGISTERED) {
        model->sendMessage(view->sendMessageTextBox->Text->ToString(),
view->contactConInfoLabel->Text->ToString());
        view->sendMessageTextBox->Clear();
    }
    else if (view->_accRegFlag == ACC_NOT_REGISTERED) {
        MessageBox::Show("Для начала войдите в аккаунт");
    }
}
void VoIPhone::SIPPresenter::OnSendFileButtonClicked()
{
    OpenFileDialog^ openDlg = gcnew OpenFileDialog();
    openDlg->Title = "Выбор файла для отправки";
    openDlg->Filter = "Image
Files (*.BMP;*.JPG;*.GIF)|*.BMP;*.JPG;*.GIF|All files (*.*)|*.*";
    if (DialogResult::OK == openDlg->ShowDialog())

```

```

        {
            String^ fileName = openDlg->FileName;
            //model->makeCallNs (view->contactConInfoLabel->Text-
>ToString(), fileName);
        }
    }
    void VoIPhone::SIPPresenter::updateChatMsgTextBox(System::String^ url,
System::String^ msg)
    {
        incMsg = gcnew array<Object^> { url, msg };
        view->chatRichTextBox->Invoke(gcnew MethodInvoker(this,
&SIPPresenter::UpdateChatHelper), incMsg);
    }
    void VoIPhone::SIPPresenter::UpdateChatHelper()
    {
        array<Object^>^ args = dynamic_cast<array<Object^>^>(view-
>chatRichTextBox->Invoke(gcnew Func<array<Object^>^>(this,
&SIPPresenter::GetArgs))));
        updateChat(args);
    }
    array<System::Object^>^ VoIPhone::SIPPresenter::GetArgs()
    {
        return incMsg;
    }
    void VoIPhone::SIPPresenter::updateChat(array<Object^>^ args)
    {
        System::String^ url = safe_cast<System::String^>(args[0]);
        System::String^ msg = safe_cast<System::String^>(args[1]);

        if (view->chatRichTextBox->Text == "") {
            view->chatRichTextBox->SelectedText = url + "-> " + msg;
        }
        else {
            view->chatRichTextBox->SelectedText =
System::Environment::NewLine;
            view->chatRichTextBox->SelectedText = url + "-> " + msg;
        }
    }
    void VoIPhone::SIPPresenter::answerCall(System::String^ url)
    {
        incCall = gcnew array<Object^> {url};
        view->Invoke(gcnew MethodInvoker(this,
&SIPPresenter::updateCall));
    }
    void VoIPhone::SIPPresenter::updateCall()
    {
        System::String^ url = safe_cast<System::String^>(incCall[0]);
        int index = url->IndexOf('<');
        System::String^ resultUrl;
        System::String^ name;
        System::String^ address;
        if (index != -1)
        {
            if (index != 0)
            {
                name = url->Substring(1, index - 3)->Trim();
                address = url->Substring(index + 1, url->Length - index - 2)-
>Trim();
                resultUrl = name + "\r\n" + address;
            }
        }
    }

```

					ИГУ 1.090301.39.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		74

```

    }
    if (view->_inActiveCall == ACC_CALL_NO_ACTIVE) {
        view->_inActiveCall = ACC_CALL_ACTIVE;
        view->contactConInfoLabel->Text = resultUrl;
        view->contactsListBox->Items->Add(name + "-" + address);
        view->chatRichTextBox->Clear();
        view->chatRichTextBox->Text = "Соединение установлено... " +
address;
        view->makeCallButton->Text = "Завершить";
    }
    else if (view->_inActiveCall == ACC_CALL_ACTIVE) {
        model->hangupCalls();
        view->_inActiveCall = ACC_CALL_ACTIVE;
        view->contactConInfoLabel->Text = resultUrl;
        view->contactsListBox->Items->Add(name + "-" + address);
        view->chatRichTextBox->Clear();
        view->makeCallButton->Text = "Завершить";
    }
}

```

Файл MyCall.hpp

```

#pragma once
#ifdef __MYCALL_H
#define __MYCALL_H

#include "SIPaccount.h"

class MyAudioPlayer : public pj::AudioMediaPlayer
{
public:
    MyAudioPlayer() : AudioMediaPlayer() {}
    ~MyAudioPlayer() {}
};

class MyCall : public pj::Call
{
public:
    MyCall(pj::Account& acc, int call_id = PJSUA_INVALID_ID) :
pj::Call(acc, call_id) {
        myAcc = (SIPaccount*)&acc;

        wav_player = NULL;
    }
    ~MyCall() {
        if (wav_player) {
            delete wav_player;
        }
    }
    virtual void onCallState(pj::OnCallStateParam& prm); //когда
состояние вызова изменилось
    virtual void onCallMediaState(pj::OnCallMediaStateParam&
prm); //когда состояние микр или динамик изменилось
private:
    class SIPaccount* myAcc;
    MyAudioPlayer* wav_player;
};
#endif // !__MYCALL_H

```

Файл MyCall.cpp

```
#include "MyCall.h"
#include "Magma.h"
#include <fstream>
void MyCall::onCallState(pj::OnCallStateParam& prm)
{
    PJ_UNUSED_ARG(prm);

    pj::CallInfo ci = getInfo();
    std::cout << "*** Call: " << ci.remoteUri << " [" << ci.stateText
        << "]" << std::endl;

    if (ci.state == PJSIP_INV_STATE_CONFIRMED)
    {
        // do stuff
    }
    else if (ci.state == PJSIP_INV_STATE_DISCONNECTED)
    {
        // do stuff related to call disconnection
        //
        myAcc->removeCall(this);
        delete this;
        return;
    }
}

void MyCall::onCallMediaState(pj::OnCallMediaStateParam& media_param)
{
    pj::CallInfo ci = getInfo();
    pj::AudioMedia* aud_med = NULL;
    Magma magma("keyboard", "mono");
    for (unsigned i = 0; i < ci.media.size(); ++i) {
        if (ci.media[i].type == PJMEDIA_TYPE_AUDIO) {
            aud_med = (pj::AudioMedia*)getMedia(i);
            break;
        }
    }
    pj::AudioMediaRecorder wav_writer;
    pj::AudioMedia& mic_media =
    pj::Endpoint::instance().audDevManager().getCaptureDevMedia();
    try {
        wav_writer.createRecorder("file.wav");
        //не работает
        //magma.encrypt(Magma::Method::CFB, std::ifstream("file.wav",
    std::ios::binary), std::ofstream("Encrfile.wav", std::ios::binary));
        mic_media.startTransmit(wav_writer);

    }
    catch (pj::Error& err) {
    }

    pj::AudioMediaPlayer player;
    pj::AudioMedia& speaker_media =
    pj::Endpoint::instance().audDevManager().getPlaybackDevMedia();
    try {
        player.createPlayer("Encrfile.wav");
        //не работает
```

					ПГУ 1.090301.39.001 ПЗ	Лист
						76
Изм.	Лист	№ докум.	Подпись	Дата		

```

        //magma.decrypt (Magma::Method::CFB,
std::ifstream("Encrfile.wav", std::ios::binary),
std::ofstream("decr.wav", std::ios::binary));
        player.startTransmit(speaker_media);
    }
    catch (pj::Error& err) {
    }

    if (aud_med) {
        mic_media.startTransmit(*aud_med);
        aud_med->startTransmit(speaker_media);
    }
}

```

Файл SIPaccount.hpp

```

#pragma once
#ifndef __SIPACCOUNT_H
#define __SIPACCOUNT_H

#include "pjsua2.hpp"
#include "MyCall.h"
#include <iostream>
#include <gcroot.h>
#include "SIPPresenter.h"
public delegate void UpdateRichTextBoxDelegate(std::string text);
class SIPAccount : public pj::Account
{
public:
    SIPAccount();
    SIPAccount(VoIPhone::SIPPresenter^ pres);
    ~SIPAccount();
    //Функция состояния(Register/ Unregister)
    virtual void onRegState(pj::OnRegStateParam& prm);
    //Функция удаления звонка
    void removeCall(pj::Call* call);
    //Функция получения звонка
    virtual void onIncomingCall(pj::OnIncomingCallParam& iprm);
    virtual void onInstantMessage(pj::OnInstantMessageParam& iprm);
    void incall(pj::CallInfo* ci, pj::Call* call, pj::CallOpParam*
prm);
public:
    std::vector<pj::Call*> calls;
    gcroot<UpdateRichTextBoxDelegate^> del;
    gcroot<VoIPhone::SIPPresenter^> presenter;
};
#endif // !__SIPACCOUNT_H

```

Файл SIPaccount.cpp

```

#include "SIPaccount.h"
#include "DialogForm.h"
#include "IncomigCallForm.h"
#include <thread>
SIPAccount::SIPAccount()
{
}

```

```

SIPaccount::SIPaccount(VoIPhone::SIPPresenter^ pres)
{
    this->presenter = pres;
}

SIPaccount::~~SIPaccount()
{
    std::cout << "*** Account is being deleted: No of calls =" <<
calls.size() << std::endl;
    for (std::vector<pj::Call*>::iterator it = calls.begin(); it !=
calls.end(); )
    {
        delete (*it);
        it = calls.erase(it);
    }
}

void SIPaccount::onRegState(pj::OnRegStateParam& prm) {
    pj::AccountInfo ai = getInfo();
    std::cout << (ai.regIsActive ? "*** Register:" : "***
Unregister:")
        << " code=" << prm.code << std::endl;
}

void SIPaccount::removeCall(pj::Call* call)
{
    for (std::vector<pj::Call*>::iterator it = calls.begin();
it != calls.end(); ++it)
    {
        if (*it == call) {
            calls.erase(it);
            break;
        }
    }
}

void SIPaccount::onIncomingCall(pj::OnIncomingCallParam& iprm)
{
    pj::Call* call = new MyCall(*this, iprm.callId);
    pj::CallInfo ci = call->getInfo();
    pj::CallOpParam prm;

    std::cout << "*** Incoming Call: " << ci.remoteUri << " [" <<
ci.stateText << "]" << std::endl;
    calls.push_back(call);
    prm.statusCode = PJSIP_SC_OK;
    VoIPhone::IncomigCallForm^ incCall = gcnew
VoIPhone::IncomigCallForm(gcnew System::String(ci.remoteUri.c_str()));
    incCall->ShowDialog();
    bool state = incCall->getState();
    if (state) {
        call->answer(prm);
        presenter->answerCall(gcnew
System::String(ci.remoteUri.c_str()));
    }
    else {
        calls.pop_back();
        call->hangup(prm);
    }
}

```

```

    }
    /* std::thread answCall([this, ci, call, prm]{
        incall((pj::CallInfo*)&ci, call, (pj::CallOpParam*)&prm);
    });
    answCall.detach();*/
}

void SIPaccount::incall(pj::CallInfo* ci, pj::Call* call,
pj::CallOpParam* prm)
{
    pj_thread_desc desc;
    pj_thread_t* thread = 0;
    pj_thread_register("incomCall", desc, &thread);
    VoIPhone::DialogForm chatForm(*this, ci, call, prm);
    chatForm.ShowDialog();
    if (!this->calls.empty())
        call->hangup(prm);
}

void SIPaccount::onInstantMessage(pj::OnInstantMessageParam& iprm)
{
    presenter->updateChatMsgTextBox(gcnew
System::String(iprm.fromUri.c_str()), gcnew
System::String(iprm.msgBody.c_str()));
}

```

Файл AddContactForm.hpp

```

#pragma once

namespace VoIPhone {
    using namespace System;
    using namespace System::ComponentModel;
    using namespace System::Collections;
    using namespace System::Windows::Forms;
    using namespace System::Data;
    using namespace System::Drawing;

    /// <summary>
    /// Сводка для AddContactForm
    /// </summary>
    public ref class AddContactForm : public
System::Windows::Forms::Form
    {
    public:
        AddContactForm(void)
        {
            InitializeComponent();
            //
            //TODO: добавьте код конструктора
            //
        }

    protected:
        /// <summary>
        /// Освободить все используемые ресурсы.
        /// </summary>
        ~AddContactForm()

```

Изм.	Лист	№ докум.	Подпись	Дата

ПГУ 1.090301.39.001 ПЗ

Лист

79

```

        {
            if (components)
            {
                delete components;
            }
        }
private: System::Windows::Forms::Label^ contactNameLabel;
protected:
private: System::Windows::Forms::Label^ sipAddress;
private: System::Windows::Forms::TextBox^ enterNameTextBox;
private: System::Windows::Forms::TextBox^ enterSipAddressTextBox;
private: System::Windows::Forms::Button^ addContactButton;

private:
    /// <summary>
    /// Обязательная переменная конструктора.
    /// </summary>
    System::ComponentModel::Container ^components;
private: System::Windows::Forms::Label^ label1;
private: System::Windows::Forms::Label^ label2;
private:
    System::Collections::Generic::List<String^>^ dataList;
#pragma region Windows Form Designer generated code
    /// <summary>
    /// Требуемый метод для поддержки конструктора – не
изменяйте
    /// содержимое этого метода с помощью редактора кода.
    /// </summary>
    void InitializeComponent(void)
    {
        this->contactNameLabel = (gcnew
System::Windows::Forms::Label());
        this->sipAddress = (gcnew
System::Windows::Forms::Label());
        this->enterNameTextBox = (gcnew
System::Windows::Forms::TextBox());
        this->enterSipAddressTextBox = (gcnew
System::Windows::Forms::TextBox());
        this->addContactButton = (gcnew
System::Windows::Forms::Button());
        this->label1 = (gcnew
System::Windows::Forms::Label());
        this->label2 = (gcnew
System::Windows::Forms::Label());
        this->SuspendLayout();
        //
        // contactNameLabel
        //
        this->contactNameLabel->Location =
System::Drawing::Point(45, 67);
        this->contactNameLabel->Name = L"contactNameLabel";
        this->contactNameLabel->Size =
System::Drawing::Size(65, 23);
        this->contactNameLabel->TabIndex = 0;
        this->contactNameLabel->Text = L"Имя";
        //
        // sipAddress

```

```

        //
        this->sipAddress->Location =
System::Drawing::Point(45, 112);
        this->sipAddress->Name = L"sipAddress";
        this->sipAddress->Size = System::Drawing::Size(65,
23);
        this->sipAddress->TabIndex = 1;
        this->sipAddress->Text = L"Sip адресс";
        //
        // enterNameTextBox
        //
        this->enterNameTextBox->Location =
System::Drawing::Point(126, 64);
        this->enterNameTextBox->Name = L"enterNameTextBox";
        this->enterNameTextBox->Size =
System::Drawing::Size(164, 20);
        this->enterNameTextBox->TabIndex = 2;
        //
        // enterSipAddressTextBox
        //
        this->enterSipAddressTextBox->Location =
System::Drawing::Point(126, 109);
        this->enterSipAddressTextBox->Name =
L"enterSipAddressTextBox";
        this->enterSipAddressTextBox->Size =
System::Drawing::Size(164, 20);
        this->enterSipAddressTextBox->TabIndex = 3;
        //
        // addContactButton
        //
        this->addContactButton->Location =
System::Drawing::Point(98, 161);
        this->addContactButton->Name = L"addContactButton";
        this->addContactButton->Size =
System::Drawing::Size(135, 31);
        this->addContactButton->TabIndex = 4;
        this->addContactButton->Text = L"Добавить";
        this->addContactButton->UseVisualStyleBackColor =
true;
        this->addContactButton->Click += gcnew
System::EventHandler(this, &AddContactForm::addContactButton_Click);
        //
        // label1
        //
        this->label1->AutoSize = true;
        this->label1->Font = (gcnew
System::Drawing::Font(L"Microsoft Sans Serif", 8.25F,
System::Drawing::FontStyle::Italic,
System::Drawing::GraphicsUnit::Point,
        static_cast<System::Byte>(204)));
        this->label1->ForeColor =
System::Drawing::SystemColors::ControlDark;
        this->label1->Location = System::Drawing::Point(123,
48);
        this->label1->Name = L"label1";
        this->label1->Size = System::Drawing::Size(119, 13);
        this->label1->TabIndex = 5;
        this->label1->Text = L"Необязательное поле";

```

```

        //
        // label2
        //
        this->label2->AutoSize = true;
        this->label2->Font = (gcnew
System::Drawing::Font(L"Microsoft Sans Serif", 8.25F,
System::Drawing::FontStyle::Italic,
System::Drawing::GraphicsUnit::Point,
        static_cast<System::Byte>(204)));
        this->label2->ForeColor =
System::Drawing::SystemColors::ControlDark;
        this->label2->Location = System::Drawing::Point(123,
93);

        this->label2->Name = L"label2";
        this->label2->Size = System::Drawing::Size(152, 13);
        this->label2->TabIndex = 6;
        this->label2->Text = L"Например: 0000@sip2sip.info";
        //
        // AddContactForm
        //
        this->AutoScaleDimensions = System::Drawing::SizeF(6,
13);

        this->AutoScaleMode =
System::Windows::Forms::AutoScaleMode::Font;
        this->ClientSize = System::Drawing::Size(347, 219);
        this->Controls->Add(this->label2);
        this->Controls->Add(this->label1);
        this->Controls->Add(this->addContactButton);
        this->Controls->Add(this->enterSipAddressTextBox);
        this->Controls->Add(this->enterNameTextBox);
        this->Controls->Add(this->sipAddress);
        this->Controls->Add(this->contactNameLabel);
        this->FormBorderStyle =
System::Windows::Forms::FormBorderStyle::FixedDialog;
        this->Name = L"AddContactForm";
        this->StartPosition =
System::Windows::Forms::FormStartPosition::CenterScreen;
        this->Text = L"Добавить контакт";
        this->ResumeLayout(false);
        this->PerformLayout();

    }
#pragma endregion

    public: System::Collections::Generic::List<String^>^ getInfo() {
        return dataList;
    }

    private: System::Void addContactButton_Click(System::Object^
sender, System::EventArgs^ e) {
        if (this->enterSipAddressTextBox->Text == "")
            MessageBox::Show("Вы не ввели Sip адресс");
        else {
            System::Collections::Generic::List<String^>^ List =
gcnew System::Collections::Generic::List<String^>;
            List->Add(this->enterNameTextBox->Text->ToString());
            List->Add("sip:" + this->enterSipAddressTextBox->Text-
>ToString());

```

```

        dataList = List;
        Close();
    }

}
};
}

```

Файл RegistrationForm.cpp

```

#pragma once

namespace VoIPhone {

    using namespace System;
    using namespace System::ComponentModel;
    using namespace System::Collections;
    using namespace System::Windows::Forms;
    using namespace System::Data;
    using namespace System::Drawing;

    /// <summary>
    /// Сводка для RegistrationForm
    /// </summary>
    public ref class RegistrationForm : public
System::Windows::Forms::Form
    {
    public:
        RegistrationForm(void)
        {
            InitializeComponent();
            //
            //TODO: добавьте код конструктора
            //
        }

    protected:
        /// <summary>
        /// Освободить все используемые ресурсы.
        /// </summary>
        ~RegistrationForm()
        {
            if (components)
            {
                delete components;
            }
        }

    private: System::Windows::Forms::Button^ button1;
    private: System::Windows::Forms::Label^ accountNameLabel;
    private: System::Windows::Forms::TextBox^
enterAccountNameTextBox;

    private: System::Windows::Forms::Label^ label1;
    private: System::Windows::Forms::TextBox^
enterAccountPassTextBox;
    private: System::Windows::Forms::Label^ sipServLabel;
    private: System::Windows::Forms::Label^ sipProxyLabel;

```

Изм.	Лист	№ докум.	Подпись	Дата

ПГУ 1.090301.39.001 ПЗ

Лист

83

```

private: System::Windows::Forms::Label^ domainLabel;
private: System::Windows::Forms::TextBox^ sipServTextBox;
private: System::Windows::Forms::TextBox^ sipProxyTextBox;
private: System::Windows::Forms::TextBox^ domainTextBox;

protected:

private:
    /// <summary>
    /// Обязательная переменная конструктора.
    /// </summary>
    System::ComponentModel::Container ^components;

#pragma region Windows Form Designer generated code
    /// <summary>
    /// Требуемый метод для поддержки конструктора – не
изменяйте
    /// содержимое этого метода с помощью редактора кода.
    /// </summary>
    void InitializeComponent(void)
    {
        this->button1 = (gcnew
System::Windows::Forms::Button());
        this->accountNameLabel = (gcnew
System::Windows::Forms::Label());
        this->enterAccountNameTextBox = (gcnew
System::Windows::Forms::TextBox());
        this->label1 = (gcnew
System::Windows::Forms::Label());
        this->enterAccountPassTextBox = (gcnew
System::Windows::Forms::TextBox());
        this->sipServLabel = (gcnew
System::Windows::Forms::Label());
        this->sipProxyLabel = (gcnew
System::Windows::Forms::Label());
        this->domainLabel = (gcnew
System::Windows::Forms::Label());
        this->sipServTextBox = (gcnew
System::Windows::Forms::TextBox());
        this->sipProxyTextBox = (gcnew
System::Windows::Forms::TextBox());
        this->domainTextBox = (gcnew
System::Windows::Forms::TextBox());
        this->SuspendLayout();
        //
        // button1
        //
        this->button1->Location = System::Drawing::Point(88,
269);

        this->button1->Name = L"button1";
        this->button1->Size = System::Drawing::Size(131, 44);
        this->button1->TabIndex = 0;
        this->button1->Text = L"Войти";
        this->button1->UseVisualStyleBackColor = true;

```

```

        this->button1->Click += gcnew
System::EventHandler(this, &RegistrationForm::button1_Click);
        //
        // accountNameLabel
        //
        this->accountNameLabel->AutoSize = true;
        this->accountNameLabel->Location =
System::Drawing::Point(23, 135);
        this->accountNameLabel->Name = L"accountNameLabel";
        this->accountNameLabel->Size =
System::Drawing::Size(78, 13);
        this->accountNameLabel->TabIndex = 1;
        this->accountNameLabel->Text = L"Имя аккаунта\r\n";
        this->accountNameLabel->TextAlign =
System::Drawing::ContentAlignment::MiddleLeft;
        //
        // enterAccountNameTextBox
        //
        this->enterAccountNameTextBox->Location =
System::Drawing::Point(123, 132);
        this->enterAccountNameTextBox->Name =
L"enterAccountNameTextBox";
        this->enterAccountNameTextBox->Size =
System::Drawing::Size(167, 20);
        this->enterAccountNameTextBox->TabIndex = 2;
        this->enterAccountNameTextBox->Text = L"vvvonderer";
        //
        // label1
        //
        this->label1->AutoSize = true;
        this->label1->Location = System::Drawing::Point(23,
173);
        this->label1->Name = L"label1";
        this->label1->Size = System::Drawing::Size(45, 13);
        this->label1->TabIndex = 5;
        this->label1->Text = L"Пароль";
        this->label1->TextAlign =
System::Drawing::ContentAlignment::MiddleLeft;
        //
        // enterAccountPassTextBox
        //
        this->enterAccountPassTextBox->Location =
System::Drawing::Point(123, 170);
        this->enterAccountPassTextBox->Name =
L"enterAccountPassTextBox";
        this->enterAccountPassTextBox->Size =
System::Drawing::Size(167, 20);
        this->enterAccountPassTextBox->TabIndex = 6;
        this->enterAccountPassTextBox->Text = L"223344af";
        //
        // sipServLabel
        //
        this->sipServLabel->AutoSize = true;
        this->sipServLabel->Location =
System::Drawing::Point(23, 33);
        this->sipServLabel->Name = L"sipServLabel";
        this->sipServLabel->Size = System::Drawing::Size(63,
13);

```

```

        this->sipServLabel->TabIndex = 7;
        this->sipServLabel->Text = L"SIP сервер";
        //
        // sipProxyLabel
        //
        this->sipProxyLabel->AutoSize = true;
        this->sipProxyLabel->Location =
System::Drawing::Point(23, 69);
        this->sipProxyLabel->Name = L"sipProxyLabel";
        this->sipProxyLabel->Size = System::Drawing::Size(63,
13);

        this->sipProxyLabel->TabIndex = 8;
        this->sipProxyLabel->Text = L"SIP прокси";
        //
        // domainLabel
        //
        this->domainLabel->AutoSize = true;
        this->domainLabel->Location =
System::Drawing::Point(23, 101);
        this->domainLabel->Name = L"domainLabel";
        this->domainLabel->Size = System::Drawing::Size(42,
13);

        this->domainLabel->TabIndex = 9;
        this->domainLabel->Text = L"Домен";
        //
        // sipServTextBox
        //
        this->sipServTextBox->Location =
System::Drawing::Point(123, 30);
        this->sipServTextBox->Name = L"sipServTextBox";
        this->sipServTextBox->Size =
System::Drawing::Size(167, 20);
        this->sipServTextBox->TabIndex = 10;
        this->sipServTextBox->Text = L"sip2sip.info";
        //
        // sipProxyTextBox
        //
        this->sipProxyTextBox->Location =
System::Drawing::Point(123, 66);
        this->sipProxyTextBox->Name = L"sipProxyTextBox";
        this->sipProxyTextBox->Size =
System::Drawing::Size(167, 20);
        this->sipProxyTextBox->TabIndex = 11;
        this->sipProxyTextBox->Text = L"proxy.siphor.net";
        //
        // domainTextBox
        //
        this->domainTextBox->Location =
System::Drawing::Point(123, 101);
        this->domainTextBox->Name = L"domainTextBox";
        this->domainTextBox->Size = System::Drawing::Size(167,
20);

        this->domainTextBox->TabIndex = 12;
        this->domainTextBox->Text = L"sip2sip.info";
        //
        // RegistrationForm
        //

```

```

        this->AutoScaleDimensions = System::Drawing::SizeF(6,
13);
        this->AutoScaleMode =
System::Windows::Forms::AutoScaleMode::Font;
        this->ClientSize = System::Drawing::Size(316, 341);
        this->Controls->Add(this->domainTextBox);
        this->Controls->Add(this->sipProxyTextBox);
        this->Controls->Add(this->sipServTextBox);
        this->Controls->Add(this->domainLabel);
        this->Controls->Add(this->sipProxyLabel);
        this->Controls->Add(this->sipServLabel);
        this->Controls->Add(this->enterAccountPassTextBox);
        this->Controls->Add(this->label1);
        this->Controls->Add(this->enterAccountNameTextBox);
        this->Controls->Add(this->accountNameLabel);
        this->Controls->Add(this->button1);
        this->Name = L"RegistrationForm";
        this->StartPosition =
System::Windows::Forms::FormStartPosition::CenterScreen;
        this->Text = L"Вход в аккаунт";
        this->ResumeLayout(false);
        this->PerformLayout();

    }
#pragma endregion
public: ArrayList^ getData() {
    ArrayList^ accInfo = gcnew ArrayList();
    accInfo->Add(this->sipServTextBox->Text->ToString());
    accInfo->Add(this->sipProxyTextBox->Text->ToString());
    accInfo->Add(this->domainTextBox->Text->ToString());
    accInfo->Add(this->enterAccountNameTextBox->Text->ToString());
    accInfo->Add(this->enterAccountPassTextBox->Text->ToString());
    return accInfo;
}

private: System::Void button1_Click(System::Object^ sender,
System::EventArgs^ e) {
    Close();
}
};
}

```

Приложение В

(Обязательное)

Программное средство для передачи речевой и файловой информации

Презентация

ПРОГРАММНОЕ СРЕДСТВО ДЛЯ ПЕРЕДАЧИ РЕЧЕВОЙ И ФАЙЛОВОЙ ИНФОРМАЦИИ

Выполнил студент
группы 20ВВП2:
Шадрин Д.А.

Научный
руководитель:
К.Т.Н., доцент,
Егоров В.Ю.

ЦЕЛЬ РАБОТЫ

Цель работы – разработка программного средства для обмена речевой и файловой информацией.

Объектом разработки является клиентское приложение, которое выполняет следующие функции:

- ▶ Авторизация аккаунта, созданного на общедоступном интернет ресурсе IP-телефонии;
- ▶ Отправка зашифрованных сообщений;
- ▶ Отправка зашифрованных файлов;
- ▶ Совершение вызова с шифрованием голоса.

2

Изм.	Лист	№ докум.	Подпись	Дата

ПГУ 1.090301.39.001 ПЗ

Лист

89

АКТУАЛЬНОСТЬ ТЕМЫ

Актуальность данной темы обусловлена растущей популярностью IP-телефонии в бизнес-сфере благодаря своей простоте, надежности, обилием функций и возможности общаться в любое время и из любой точки мира.

3

СТЕК ИСПОЛЬЗУЕМЫХ ТЕХНОЛОГИЙ



Visual Studio



C++



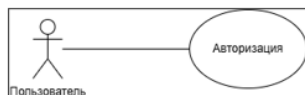
.NET



PJSIP

4

ВАРИАНТЫ ИСПОЛЬЗОВАНИЯ ПРИЛОЖЕНИЯ



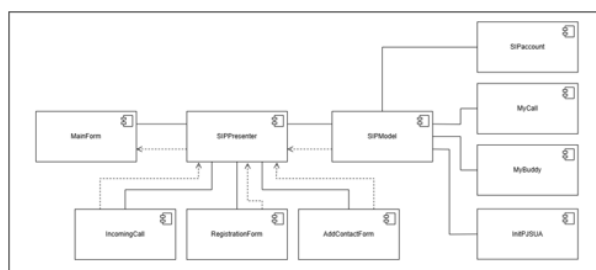
5

АРХИТЕКТУРА ПРИЛОЖЕНИЯ

ШАБЛОН ПРОЕКТИРОВАНИЯ MVP

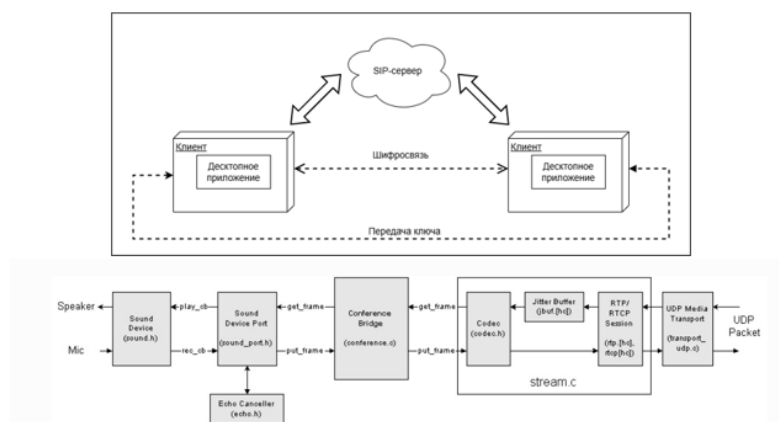
- **Model** - отвечает за работу с данными приложения. Она представляет из себя структуру данных, бизнес-логику и методы доступа к данным и не зависит от представления или презентера.
- **View** - отвечает за отображение данных для пользователя. Оно представляет собой интерфейс пользователя, который взаимодействует с моделью.
- **Presenter** - это связующее звено между моделью и представлением. Он обрабатывает пользовательские запросы, взаимодействует с моделью для получения или изменения данных.

ДИАГРАММА КОМПОНЕНТОВ



6

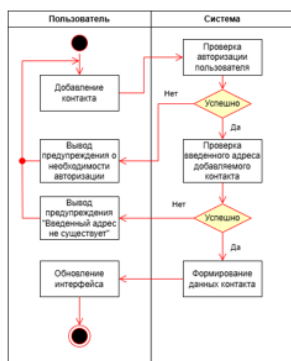
ПЕРЕДАЧА ЗАШИФРОВАННЫХ ДАННЫХ



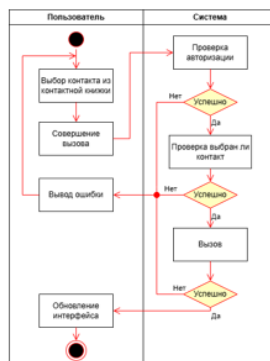
7

ДИАГРАММЫ ПОСЛЕДОВАТЕЛЬНОСТЕЙ

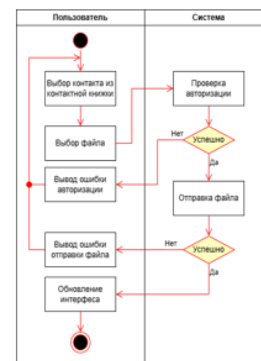
ДОБАВЛЕНИЕ КОНТАКТА



ВЫЗОВ КОНТАКТА



ОТПРАВКА ФАЙЛА

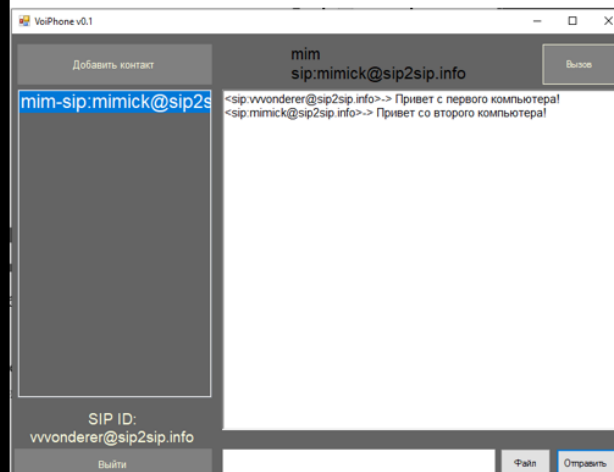


8

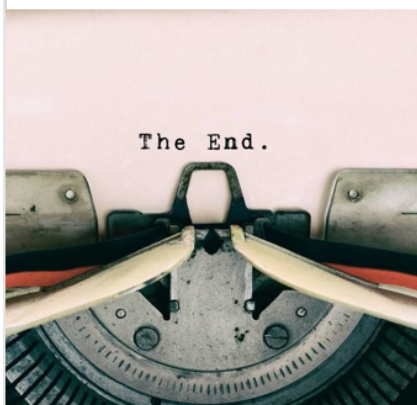
ЗАКЛЮЧЕНИЕ

В ходе выполнения данной выпускной квалификационной работы было разработано программное средство, которое позволяет обмениваться медиа данными.

Разработанное приложение позволяет пользователям совершать и принимать зашифрованные голосовые звонки, а также отправлять и принимать зашифрованные файлы и сообщения.



9



СПАСИБО ЗА ВНИМАНИЕ!

10

Изм.	Лист	№ докум.	Подпись	Дата

ПГУ 1.090301.39.001 ПЗ

Лист

93