

Пенза 2024

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

Факультет

вычислительной техники

Кафедра

«Вычислительная техника»

«Утверждаю»

Заведующий кафедрой

М.А. Митрохин

« 19 » 04 2024 г.

**ЗАДАНИЕ
НА БАКАЛАВРСКУЮ РАБОТУ**

1. Студент Баулин Александр Дмитриевич гр.20ВВП2 факультета вычислительной техники

2. Тема работы: «Мессенджер с шифрованием сообщений»

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

3. Руководитель бакалаврской работы к.т.н., доцент кафедры ВТ ПГУ Юрова О.В.

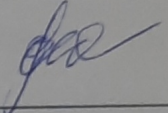
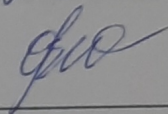
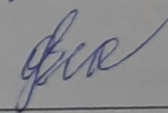
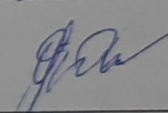
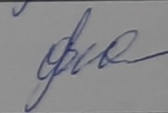
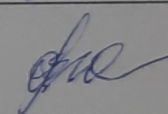
4. Задание на работу (назначение разработки, исходные данные и т.п.)

Разработать приложение «Мессенджер с шифрованием сообщений» для возможности обмена шифрованными сообщениями. Приложение должно позволять пользователям обмениваться сообщениями, редактировать настройки своего профиля. Также приложение должно позволять пользователям создавать новые чаты с пользователями. Приложение должно быть спроектировано с использованием клиент-серверной архитектуры. Клиентская и серверные части программы должны обладать понятным графическим интерфейсом. Сервер должен включать в себя хранение данных в базу данных, передачу данных между пользователями и удаление их из нее. Клиентская программа должна позволять пользователю просматривать чаты и сообщения в них и редактировать настройки своего профиля и самого приложения.

5. Перечень подлежащих разработке вопросов

Требуется произвести анализ предметной области: изучить существующие аналоги, изучить способы проектирования и разработки клиент-серверного приложения. Также необходимо изучить технологий для разработки приложения, такие как Windows Forms, MySQL, SQLite 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 «Мессенджер с шифрованием сообщений»	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 _____

(подпись, дата)

Юрова О.В.

Задание к исполнению принял студент _____

(подпись, дата)

Баулин А.Д.

Работу к защите допустить _____

Декан факультета

(подпись, дата)

Фионова Л.Р.

Реферат

Бакалаврская работа содержит 100 л., 35 рис., 7 табл., 12 источн. и 3 прил.
КЛИЕНТ-СЕРВЕРНАЯ АРХИТЕКТУРА, MYSQL, SQLITE, WINDOWS FORMS, JAVA, ANDROID STUDIO, VISUAL STUDIO, MVC, C#, .NET, JSON

Цель работы – разработать клиент-серверное приложение, позволяющее обмениваться шифрованными сообщениями.

Разработанное приложение должно позволять пользователям обмениваться шифрованными сообщениями между друг другом. Создавать новые чаты, просматривать историю сообщений и изменять настройки своего профиля и приложения.

Результатом проделанной работы является проект, который удовлетворяет поставленной задаче, а именно разработано, реализовано и протестировано приложение для обмена зашифрованными сообщениями. При разработке было сервера использовано большое количество технологий, которым относятся: язык программирования C#, платформа .NET, язык структурированных запросов SQL, Windows Forms для написания пользовательского интерфейса, среда разработки Visual Studio, для хранения данных выбрана СУБД MySQL.Для разработки клиентской части использовались: Язык программирования Java, среда разработки Android Studio, СУБД SQLite.

					ПГУ 1.090301.26.001 ПЗ				
Изм.	Лист	№ докум.	Подпись	Дата	«Мессенджер с шифрованием сообщений» Пояснительная записка	Лит.	Лист	Листов	
Разраб.		Баулин А.Д.		04.06.24					
Провер.		Юрова О.В.		04.06.24			2	100	
Н. Контр.		Бычков А.С.		04.06.24		ФВТ, каф. ВТ гр. 20ВВП2			
Утв.		Митрохин М.А.		04.06.24					

Содержание

Перечень сокращений и обозначений	5
Введение	6
1 Анализ предметной области	7
1.1 Анализ темы	7
1.2 Анализ аналогов приложения	9
1.3 Постановка задачи	14
2 Выбор технологий и средств разработки приложения	19
2.1 Язык программирования C#	19
2.2 .Net Framework	20
2.3 Visual Studio	21
2.4 MySQL	21
2.5 Json	22
2.6 Windows Forms	22
2.7 Java	23
2.8 Android Studio	23
2.9 SQLite	24
2.10 Gradle	24
2.11 Алгоритм шифрования «Кузнечик»	25
3 Архитектура приложения	27
3.1 Клиент-серверная архитектура	27
3.2 Архитектура приложения	28
4 Реализация приложения	31
4.1 Описание работы приложения	31
4.2 Структура БД	35
4.3 Программная реализация приложения	39
4.4 Основные алгоритмы взаимодействия клиента и сервера	47
5 Тестирование приложения	55
5.1 Регистрация пользователя администратором	55

5.2	Запуск сервера	56
5.3	Просмотр всех пользователей администратором	57
5.4	Просмотр журнала событий системы	57
5.5	Авторизация пользователя	58
5.6	Просмотр чатов пользователем	60
5.7	Просмотр чата пользователем	61
5.8	Отправка сообщения пользователем	62
5.9	Изменение настроек приложения пользователем	63
5.10	Просмотр своего профиля пользователем	65
5.11	Добавление чата пользователем	66
6	Безопасность жизнедеятельности	70
6.1	Вред здоровью при работе с ПК	70
6.2	Снижение вреда для здоровья при работе с ПК	70
6.3	Техника безопасности при работе с компьютером	71
	Заключение	72
	Список использованных источников	73
	Приложение А - UML - диаграммы	75
	Приложение А.1 - Диаграмма компонентов	76
	Приложение А.2 - Диаграмма прецедентов	77
	Приложение А.3 - Диаграмма деятельности	78
	Приложение А.4 - Диаграмма последовательностей	79
	Приложение Б - Листинг программы	80
	Приложение В - Презентация	95

Перечень сокращений и обозначений

В настоящей работе применяют следующие сокращения и обозначения

БД – база данных

ОС – операционная система

ПК – персональный компьютер

ПО – программное обеспечение

СУБД – система управления базой данных

CLR (Common Language Runtime) – исполняющая среда для байт-кода, в которой компилируется программа

FTP (File Transfer Protocol) – протокол передачи данных по сети

HTTP (Hypertext Transfer Protocol) – протокол прикладного уровня передачи данных

JSON (JavaScript Object Notation) – текстовый формат для обмена структурированными данными и для хранения их

MVC (Model-View-Controller) — шаблон (паттерн) программирования, разделяющий архитектуру приложения на три модуля

SQL (Structured Query Language) - язык структурированных запросов

TCP/IP (Transmission Control Protocol/Internet Protocol) – сетевая модель, описывающая процесс передачи цифровых данных

UML (Unified Modeling Language) – унифицированный язык моделирования

Введение

В современном мире, где общение всё больше перемещается в онлайн, конфиденциальность и безопасность данных становятся одними из главных приоритетов.

Раньше защищенные мессенджеры использовались в основном в государственных структурах, военных предприятиях, армии, в сфере бизнеса и финансовой сфере. Но теперь они востребованы практически везде, вплоть до обычных людей, которые хотят спокойно обмениваться информацией на любые темы, не боясь того, что какое-либо третье лицо увидит сообщение.

Существующие мессенджеры, несмотря на свою популярность, не всегда обеспечивают достаточный уровень защиты информации. Многие из них используют несовершенные алгоритмы шифрования, подверженные взломам и утечкам данных. Однако информационный мир не стоит на месте, поэтому те механизмы, которые раньше обеспечивали надежную защиту, сейчас считаются ненадежными. А также зарубежные мессенджеры не могут быть надежными для использования в нашей стране, так как есть вероятность утечки информации за границу. Решением данной проблемы является создание нового мессенджера с использованием алгоритма шифрования «Кузнечик».

Таким образом, важность и актуальность разработки мессенджера с использованием отечественного алгоритма шифрования, заключается в создании приложения способного обеспечить будущих пользователей надежной защитой их конфиденциальной информации и гарантировать безопасность общения.

					ПГУ 1.090301.26.001 ПЗ	Лист
						6
Изм.	Лист	№ докум.	Подпись	Дата		

1 Анализ предметной области

1.1 Анализ темы

Мессенджер – приложение для обмена сообщениями между людьми. Он необходим для того, чтобы два или более человека могли обмениваться информацией различного характера на любом расстоянии друг от друга. Защищенный мессенджер — это мессенджер, использующий для сокрытия информации от третьих лиц шифрование передаваемых данных. Такой вариант приложения для общения обычно используется для передачи конфиденциальной информации.

В пример можно привести корпоративные мессенджеры. Так как информация, которая обрабатывается на каком-либо предприятии может иметь коммерческий или секретный характер.

В современных защищенных мессенджерах имеется несколько подходов к шифрованию данных, а именно:

- симметричное;
- ассиметричное.

В симметричном виде шифрования для шифрации и дешифрации используется один ключ. Т.е. ключ является секретным, о нем могут знать только пользователи, участвующие в диалоге, иначе если ключ будет раскрыт, то получить информацию из зашифрованного сообщения не составит большого труда.

Из плюсов данного вида шифрования можно выделить: быструю скорость шифрования и дешифрования; простоту реализации; высокий уровень безопасности, при использовании достаточно длинных ключей.

Недостатком данного шифрования является передача ключа. Так как ключ приватный, то передавать его затруднительно, ведь при передаче его может перехватить третье лицо, после чего расшифровка сообщения будет только делом времени.

В асимметричном виде шифрования используется два ключа. Один из ключей является открытым, он доступен всем и используется для шифрования сообщений, отправляемых владельцу данного ключа. Для расшифрования этот ключ не используется. Второй ключ – закрытый, это приватный ключ пользователя, он используется для дешифрации.

Преимуществом данного вида шифрования является отсутствие проблемы передачи ключа, так как раскрытие ключа шифрования не дает возможности расшифровать сообщение.

Недостатками являются сложность реализации и понимания данного вида шифрования и значительные вычислительные ресурсы устройства.

Цель работы заключается в создании приложения для обмена сообщениями, ориентированного на использование в сферах деятельности, связанной с работой над секретной или коммерческой информации.

Сервер мессенджера будет разворачиваться на предприятии. Аккаунты пользователей будут создаваться администратором. Ввиду подобной специфики выбран симметричный вид шифрования, так как асимметричный имеет преимущество только в удобстве передачи ключа. Но при данном варианте использования ключ не по глобальной сети, что решает главную проблему симметричного шифрования.

Таким образом, задача разработки приложения для обмена сообщениями является сложной и ответственной задачей, требующей высокой квалификации и опыта в сфере разработки сетевых приложений. Однако, создание данного приложения является необходимым для обеспечения качественной, быстрой и защищенной связи, без которой, на сегодняшний день, работа любого предприятия становится практически невозможной.

1.2 Анализ аналогов приложения

1.2.1 Signal — Private Messenger

Signal относится к числу наиболее защищённых приложений для общения. Таким его делает использование нового алгоритма шифрования PQXDH, который применяется и другими мессенджерами, вроде WhatsApp. Так как автором Signal является некоммерческая организация, вряд ли стоит ждать от неё попыток монетизации пользовательских данных.

Шифрование в этом мессенджере основано на концепции совершенной прямой секретности. Это означает, что после каждого сообщения ключ шифрования меняется. Также применяется сочетание наиболее надёжных протоколов шифрования, в результате чего низкоуровневые криптографические алгоритмы делают приложение почти неприступным для взлома.

Signal также удобен в использовании и имеет достаточно удобный интерфейс. Поддерживается создание групп с числом участников до 1000, а в видеозвонках могут участвовать до 40 человек [1].

Вдобавок Signal позволяет посылать собеседникам фотографии и видеозаписи, которые исчезают через несколько секунд после просмотра, а также размывать лица на отправляемых снимках [2].

Из минусов можно отметить невозможность зарегистрироваться анонимно, поскольку при создании аккаунта нужно указывать номер телефона. Также пользователей в данном мессенджере не так много. В результате нужные вам люди могут не быть зарегистрированы здесь.

Интерфейс Signal представлен на рисунке 1.

					ПГУ 1.090301.26.001 ПЗ	Лист
						9
Изм.	Лист	№ докум.	Подпись	Дата		



Рисунок 1 - Окно диалога мессенджера Signal

1.2.2 WhatsApp

WhatsApp представляет собой самый популярный мессенджер в мире, количество пользователей в котором превышает 2 млрд. Здесь применяется сквозное шифрование.

Для безопасности аккаунт WhatsApp защищён двухфакторной проверкой. Также при отправке сообщений получателю они не сохраняются на серверах.

Популярностью WhatsApp обязан богатому выбору функциональных возможностей. Поддерживаются большие группы, исчезающие сообщения, в некоторых странах пересылка денег.

Из минусов можно отметить то, что WhatsApp связан с запрещенной на территории Российской Федерации организацией. Из этого следует, что между ними происходит обмен метаданными. Это якобы не затрагивает чаты и сообщения, но правоохранные органы и прочие организации могут использовать эти данные для идентификации отправителя и нахождения его местоположения.

Ещё WhatsApp пострадал от хакерской атаки группы Pegasus, благодаря чему правительства и организации разных стран получили возможность следить за пользователями мессенджера [1].

Интерфейс WhatsApp представлен на рисунке 2.

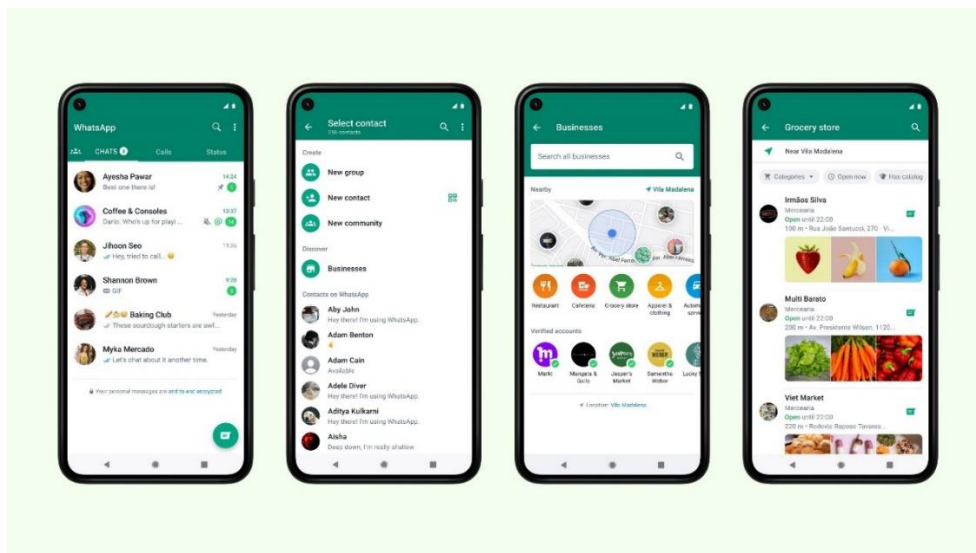


Рисунок 2 - Интерфейс мессенджера WhatsApp

1.2.3 Session

Session представляет собой бесплатное приложение для безопасного обмена текстовыми сообщениями. Оно анонимное и обладает шифрованием, не собирая при этом персональную информацию. Вместо указания номера телефона или адреса электронной почты в программе генерируются идентификаторы сеанса для регистрации и идентификации пользователей.

Используется передовой децентрализованный подход к маршрутизации и хранению сообщений. Сообщения зашифрованы и передаются через три узла, которыми управляет сообщество. В результате каждому отдельному узлу трудно собрать значительную информацию о сообщении.

Приложение Session достаточно простое в работе, в отличие от других анонимных программ подобного рода. В групповых чатах могут участвовать 100 человек, есть поддержка голосовых заметок и вложений.

Идеальных приложений не бывает, что относится и к Session. Функциональность этого мессенджера достаточно обширная, некоторые настройки отсутствуют. Например, нельзя поменять обои в защищённом чате.

Также на Android и iOS могут возникнуть трудности при попытке присоединиться к групповым чатам [1].

Интерфейс Session представлен на рисунке 3.

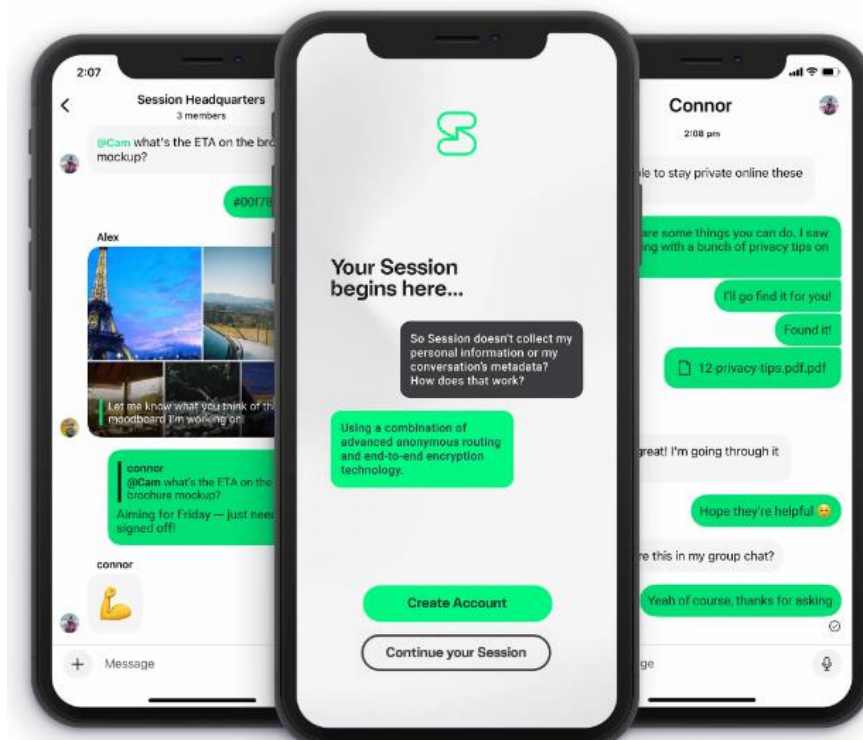


Рисунок 3 - Интерфейс мессенджера Session

1.2.4 Threema

Threema представляет собой приватный мессенджер, где ставка делает на безопасность пользовательских данных и где требуется купить платную подписку. Здесь применяется серьёзный протокол шифрования под названием Network and Cryptography Library (NaCL). Все сообщения зашифрованы, как и голосовые заметки и даже звонки. Наличие открытого исходного кода добавляет доверия к этому мессенджеру.

При создании учётной записи никакой личной информации указывать не нужно. Приложение создаёт уникальный ключ идентификации для связи с другими пользователями. Пользовательские сообщения регулярно удаляются с серверов. Любые чаты можно защитить пин-кодом.

Списки контактов хранятся локально на пользовательских устройствах, поэтому само приложение не имеет никаких данных о собственных пользователях. Группы в Threema децентрализованные, поэтому приложение не сохраняет информацию о том, в каких группах состоит тот или иной пользователь. В результате Threema представляет собой одну из лучших альтернатив WhatsApp.

Недостатками мессенджера является то, что несмотря на качество мессенджера с шифрованием, его минусом является необходимость платить.

В результате данный мессенджер является далеко не самым популярным, поэтому ваших собеседников в нём может не быть. Однако для тех, кому действительно нужен высокий уровень защиты данных, цена может оказаться не самой высокой [1].

Интерфейс Threema представлен на рисунке 4.

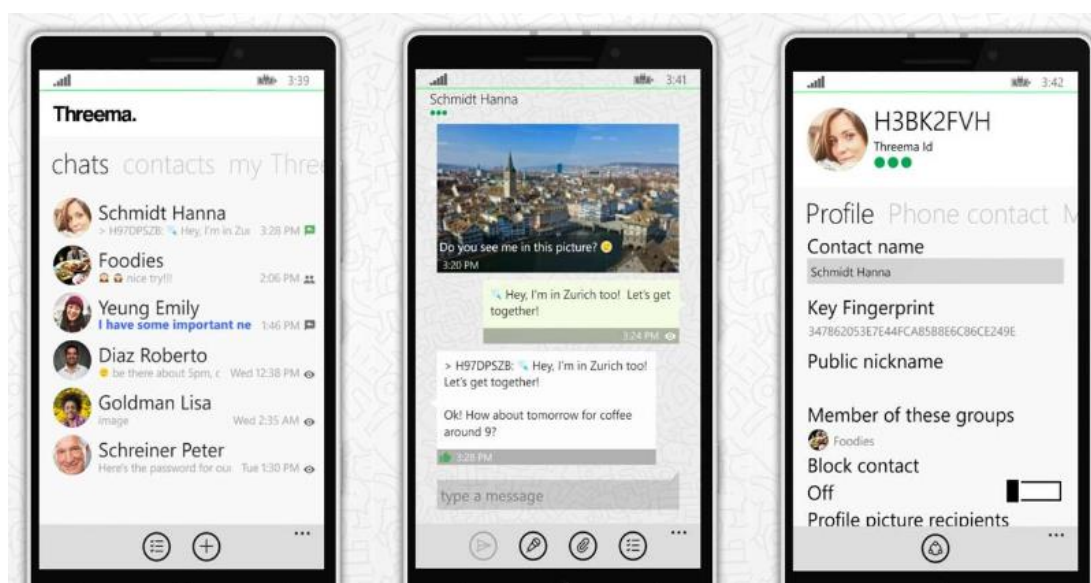


Рисунок 4 - Интерфейс мессенджера Threema

1.2.5 Вывод

По результатам анализа аналогов можно сделать такой вывод, что все мессенджеры имеют достаточно широкий функционал. Каждый мессенджер немного, но отличается своей направленностью, ввиду чего имеет отличающиеся от своих конкурентов недостатки

Результат сравнения существующих аналогов приведен в таблице 1.

Таблица 1 - Результат сравнения мессенджеров

Сравниваемые характеристики	«Signal»	«WhatsApp»	«Threema»	«Session»
Шифрование сообщений	+	+	+	+
Удобство пользовательского интерфейса	+	+	-	+
Является бесплатным	+	+	-	+
Открытый исходный код	+	-	+	-
Анонимность	-	-	+	+
Персонализация интерфейса	-	+	-	-

По итогам проведения сравнительного анализа данных мессенджеров, было принято решение разработать свой собственный мессенджер с использованием алгоритма определенного в ГОСТ Р 34.12-2015 как «Кузнечик». Приложение будет соответствовать основным требованиям, предоставляемым пользователем к современному мессенджеру с шифрованием сообщений. А именно: написание, чтение, отправка и получение; редактирование профиля; персонализация пользовательского интерфейса; шифрование сообщений.

1.3 Постановка задачи

Требуется разработать клиент-серверное приложение, позволяющее осуществлять обмен защищенными сообщениями между клиентами.

Современное мобильное приложение должно иметь пользовательский интерфейс, выполненный в современном стиле. Дизайн должен быть интуитивно понятным и легким в освоении и использовании.

Интегрированная среда разработки должна поддерживать средства отображения и редактирования графической части, написанной на выбранном языке программирования.

Не маловажным является выбор языка программирования, на котором будет написано приложение. Необходимо чтобы он позволял создавать сущности высокого уровня, обработку событий и доступ к функциям устройства. Эти возможности позволяют создать гибкое приложение со слабо связанной графической частью и бизнес-моделью. Это обеспечит высокий уровень расширяемости и масштабирования приложения.

Приложение должно реализовывать следующий функционал:

Со стороны клиента:

- форма авторизации;
- написание текста;
- возможность просмотра истории сообщений;
- отправка сообщения;
- чтение сообщений;
- последовательное отображение истории сообщений;
- сохранение диалога в локальную базу данных;
- редактирование профиля;
- шифрование передаваемых сообщений.

Со стороны сервера:

- форма регистрации нового пользователя;
- вывод системного журнала событий;
- возможность удаления существующего пользователя;
- многопоточная обработка запросов;

- возможность запуска сервера;
- возможность остановки сервера;
- хранение сообщений, не дошедших до получателя.

Минимальные системные требования для сервера:

- операционная система: Windows (7, 8, 10, 11);
- минимальная скорость интернета: 1 Мбит/сек;
- минимальный объем оперативной памяти: 2 ГБ (Windows 11 – 4 ГБ);
- минимальный объем свободного дискового пространства: 150 МБ.

Минимальные системные требования для клиента:

- операционная система: Android (9 и выше);
- минимальная скорость интернета: 1 Мбит/сек;
- минимальный объем оперативной памяти: 2 ГБ;
- минимальный объем свободного дискового пространства: 150 МБ.

Стоит учесть, что разрабатываемое приложение осуществляет передачу зашифрованных данных по сети. Ввиду чего необходимо учесть отсутствие у пользователя и администратора доступа к сети, и предоставить возможность использовать клиентскую и серверную часть приложения в режиме оффлайн.

Ввиду этого стоит разделить предоставляемый функционал на две категории, а именно:

- с подключением к сети;
- без подключения к сети.

На рисунке 5 представлена диаграмма использования приложения администратором и пользователем без подключения к сети.

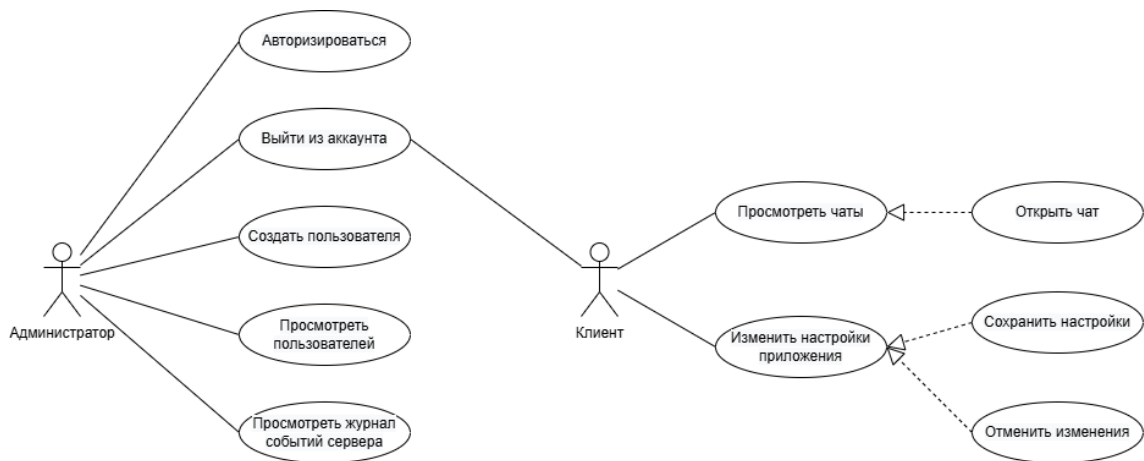


Рисунок 5 – UML диаграмма вариантов использования приложения без подключения к сети

Пользователь, не имеющий подключения к сети, имеет возможность пользоваться ограниченным набором функций, а именно:

- просмотр всех чатов;
- просмотр настроек профиля;
- просмотр истории сообщений конкретного чата;
- редактирование настроек приложения.

Также и администратор не может пользоваться всем функционалом серверного положения, ему доступны только данные функции:

- авторизация;
- выход из аккаунта;
- создание нового пользователя;
- просмотр всех пользователей;
- просмотр журнала событий сервера.

Пользователю и администратору, имеющим подключение к глобальной сети, доступен весь функционал клиентской части приложения. На рисунке 6 представлена диаграмма вариантов использования приложения с доступом к сети.

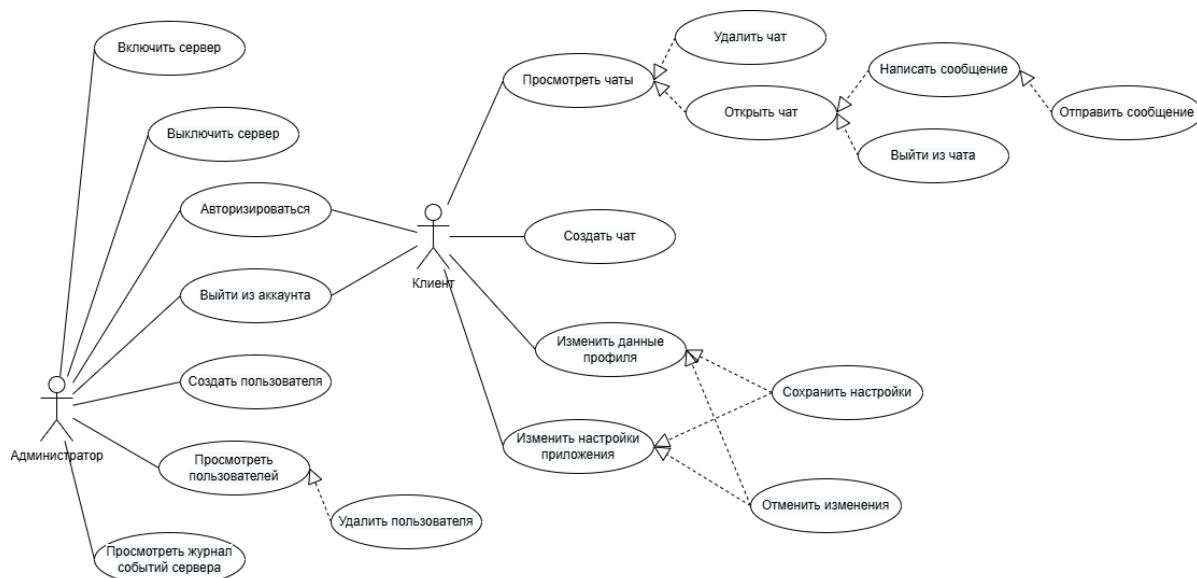


Рисунок 6 - UML диаграмма вариантов использования приложения с подключением к сети

Функциональные возможности, доступные пользователю с подключением к глобальной сети:

- авторизация;
- обмен сообщениями с другими пользователями;
- просмотр всех чатов;
- возможность просмотра истории сообщений;
- чтение сообщений;
- редактирование настроек профиля;
- редактирование настроек приложения.

Администратору, имеющему подключение к сети также доступен весь функционал серверной части приложения, а именно:

- создание нового пользователя;
- просмотр системного журнала событий;
- возможность удаления существующего пользователя;
- возможность запуска сервера;
- возможность остановки сервера.

2 Выбор технологий и средств разработки приложения

В данном разделе будут рассмотрены технологии и средства разработки, которые были использованы для реализации программы защищенного мессенджера «SecureMessenger».

Ниже приведены технологии и средства разработки, которые были выбраны для серверной и клиентской частей:

Серверная часть:

- C#;
- .NET;
- Visual Studio;
- MySQL;
- Json;
- Windows Forms;
- Java;
- Android Studio;
- SQLite;
- Gradle;
- алгоритм шифрования «Кузнечик».

2.1 Язык программирования C#

Язык программирования C# – это объектно-ориентированный и типобезопасный язык программирования, разработанный компанией Microsoft. Этот язык является универсальным и используется во многих направлениях таких как разработка десктопных приложений, написания игр, веб-сервисов, нейросетей и мобильных приложений.

C# - относительно молодой язык, но у него уже есть множество удобных библиотек для работы. Код, написанный на данном языке, является мультиплатформенным, виртуальная машина .NET Framework запустит его на

любом устройстве, где она установлена будь то компьютер, смартфон или умные часы [3].

Данный язык программирования был написан на языке C, и целью C# было заменить язык программирования Java, чтобы упростить и улучшить разработку различных приложений под Windows.

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

Для разработки мессенджера был выбран именно язык программирования C#, потому что компания Microsoft создала целую инфраструктуру для языка C# в виде различных библиотек и инструментов, с помощью которых можно быстро создать эффективное, надежное и удовлетворяющая всем требованиям приложение под операционную систему Windows.

Если сравнивать с другими высокоуровневыми языками, то выбор пал именно на C#, так как он лучше всего подходит для разработки десктопного приложения под операционную систему Windows. А если сравнивать с C подобными языками, такими как C и C++, то C# был выбран из-за упрощения и ускорения разработки за счет высокоуровневых абстракций, готовых решение в виде библиотек и отсутствия небезопасной работы с памятью.

2.2 .Net Framework

.Net – это бесплатный набор технологий, разработанных компанией Microsoft для удобного создания и выполнения приложений на различных устройствах и операционных системах.

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

В состав .Net входит CLR – общезыковая среда выполнения, она упрощает процесс разработки приложения, за счет использования специального языка IL,

который обеспечивает безопасную работу с памятью, сборку мусора и обработку исключений [4].

2.3 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 приложение 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 [5].

2.4 MySQL

MySQL — это популярная система управления базами данных. Первая версия этой СУБД была выпущена в 1994 году компанией MySQL AB. В 2008 году эту компанию приобрела Sun Microsystems, которую, в свою очередь, приобрела Oracle в 2010 году [6].

Основные особенности MySQL:

- высокая скорость работы;
- надежная и простая система безопасности;
- работа с большими объемами данных;
- плагины, позволяющие упростить и настроить работу под себя;
- бесплатная лицензия.

MySQL был использован в проекте из-за того, что он быстрый, надежный, бесплатный и имеет удобные приложения и библиотеки для работы с ним.

2.5 Json

Json (JavaScript Object Notation) – это формат для передачи и получения данных. Json имеет понятный и читаемый способ для представления структурированных данных, таких как массивы, объекты, структуры в виде текстовой строки. Этот формат данных широко используется для обмена данными между клиентом и сервером.

Формат данных Json имеет простую структуру, что в свою очередь позволяет написать Json-объект даже вручную. Также этот формат поддерживает большое количество различных типов данных, такие как строки, числа, массивы, объекты, логические значения, что позволяет представлять данные в разнообразных форматах.

Данный формат представления данных используется в приложении, так как уже реализован основной функционал для работы с ним, который легко и удобно использовать для передачи данных по сети. А также из-за того, что он имеет самый маленький объем json-файла при сериализации объекта класса среди других форматов для представления данных. Поэтому этот файл очень быстро передается по сети, что позволяет ускорить работу программы.

2.6 Windows Forms

Windows Forms — это платформа пользовательского интерфейса для создания классических приложений Windows. Она обеспечивает один из самых эффективных способов создания классических приложений с помощью визуального конструктора в Visual Studio. Такие функции, как размещение визуальных элементов управления путем перетаскивания, упрощают создание классических приложений.

В Windows Forms можно разрабатывать графически сложные приложения, которые просто развертывать, обновлять, и с которыми удобно работать как в автономном режиме, так и в сети. Приложения Windows Forms могут получать доступ к локальному оборудованию и файловой системе компьютера, на котором работает приложение [7].

2.7 Java

Java — это популярный язык программирования приложений, который применяют для многих задач, в том числе для мобильной разработки. Особенность Java в том, что написанный на нём код запускается в специальной среде — виртуальной машине Java, JVM. Эта среда либо уже установлена на устройстве, либо встраивается в приложение. Она позволяет запускать один и тот же код в разных операционных системах.

Плюсами данного языка программирования являются:

Большое количество библиотек и фреймворков. Java была выпущена в 1995 году, за время существования языка было написано множество вспомогательного кода в виде библиотек для облегчения написания кода.

Кроссплатформенность. В Java есть виртуальная Java-машина (JVM) — посредник между кодом и железом.

Объектно-ориентированный подход. Все эти сущности описывают в коде и учат взаимодействовать друг с другом. В итоге программа в стиле ООП состоит из отдельных блоков, которые хорошо расширяются и масштабируются. Поэтому язык Java подходит для разработки программ, которые планируют долго использовать и постоянно развивать [8].

2.8 Android Studio

Android Studio — это интегрированная среда разработки, специально созданная для разработки приложений под операционную систему Android. Это основной инструмент, который используется разработчиками для создания

высококачественных и инновационных мобильных приложений для платформы Android [9].

Android Studio предлагает широкий набор инструментов и функций, упрощающих процесс разработки приложений. В его состав входит редактор кода с функциями автозаполнения, отладчик, инструменты для разработки пользовательского интерфейса, система сборки приложений и многие другие полезные функции. Он также обеспечивает интеграцию с Android SDK, позволяя разработчикам создавать, тестировать и отлаживать приложения непосредственно в среде разработки.

2.9 SQLite

SQLite представляет библиотеку, которая написана на языке C (ANSI-C) и которая реализует движок реляционных баз данных. На сегодняшний день SQLite, возможно, самая используемая система баз данных.

В отличие от других систем баз данных, как MS SQL Server, MySQL, Postgres и т.д., для SQLite не требуется сервер базы данных. SQLite представляет встраиваемый движок базы данных, который обращается напрямую к файлу базы данных на диске. Соответственно для работы с базами данных нам не надо явным образом устанавливать или как-то конфигурировать SQLite.

SQLite имеет полноценную поддержку большинства возможностей, которыми обладают другие реляционные СУБД - таблицы, индексы, триггеры, представления [10].

2.10 Gradle

Gradle — это система автоматизации сборки, которая используется для управления жизненным циклом разработки проекта. Она предоставляет удобный и гибкий способ описания процесса сборки, зависимостей и других задач, связанных с разработкой программного обеспечения.

Система предоставляет широкий набор возможностей для автоматизации сборки и управления проектами. Вот некоторые из основных возможностей:

- управление зависимостями. Система обеспечивает простой способ управления зависимостями проекта. Зависимости могут быть определены в файле сборки, и Gradle автоматически загрузит их из удаленных репозиториях;

- многомодульные проекты. Система поддерживает организацию проектов в виде многомодульных структур. Каждый модуль может иметь свои собственные зависимости и настройки сборки;

- плагины. Gradle расширяется с помощью плагинов, которые добавляют дополнительную функциональность. Существует множество плагинов для интеграции с различными инструментами, фреймворками и языками программирования;

- специализированные задачи. Система предоставляет множество встроенных задач для выполнения различных операций, таких как компиляция, тестирование, упаковка приложений и многое другое. Разработчики также могут определять свои собственные задачи;

- инкрементальная сборка. Gradle поддерживает инкрементальную сборку, что означает, что он пересобирает только те части проекта, которые изменились, ускоряя процесс сборки;

- расширяемость. Сборщик предоставляет мощные средства для расширения его функциональности через плагины, пользовательские задачи и расширения [11].

2.11 Алгоритм шифрования «Кузнечик»

Рассмотрим наиболее важные и значимые преимущества данного алгоритма:

- высокая криптостойкость. Кузнечик является одним из наиболее стойких к взлому симметричных алгоритмов шифрования. Он был разработан российскими криптографами и прошел всестороннее тестирование, не показав ни одной уязвимости. Уровень его стойкости соответствует международным стандартам и превосходит многие другие алгоритмы, такие как AES и DES;

– высокая скорость работы. Кузнечик обладает высокой скоростью шифрования и расшифрования, что делает его пригодным для использования в различных приложениях, где требуется высокая производительность. Он оптимизирован для работы на современных процессорах и может использоваться даже на мобильных устройствах. Так как алгоритм является блочным, то может быть ускорен при распараллеливании вычисления по блокам;

– требует малого количество ресурсов. Код алгоритма имеет малый размер, что делает его реализацию достаточно простой. Это значительно снижает требования к ресурсам и позволяет использовать его в устройствах с ограниченной памятью;

– поддержка ГОСТ. Кузнечик одобрен Федеральной службой безопасности Российской Федерации (ФСБ РФ) и включен в список рекомендованных алгоритмов шифрования. Это означает, что он может быть использован для защиты конфиденциальной информации, в том числе государственной тайны.

В целом, Кузнечик является надежным, производительным и современным алгоритмом шифрования, который обладает множеством преимуществ по сравнению с другими алгоритмами.

Он идеально подходит для использования в приложениях, где требуется высокий уровень безопасности и конфиденциальности [12].

3 Архитектура приложения

3.1 Клиент-серверная архитектура

Клиент-серверная архитектура – это модель взаимодействия, между двумя основными лицами, взаимодействующими друг с другом, клиентом и сервером.

Взаимодействие между клиентом и сервером происходит за счет сетевых протоколов, таких как HTTP, TCP/IP, FTP. Данные протоколы определяют правила взаимодействия между клиентом и сервером. В основе этих правил лежит создание запросов от клиента и отправкой их на сервер. В свою очередь последний получает запрос, обрабатывает его, подготавливает данные и передает их в виде ответа клиенту.

Из плюсов данной архитектуры можно выделить следующие основные преимущества:

- масштабируемость. Нагрузка распределяется между серверами, количество которых можно легко увеличивать. Благодаря чему улучшается производительность всей системы и обрабатывается больше запросов от клиентов;

- централизованное управление. В данной архитектуре сервер выступает в роли центрального узла, осуществляющего полный контроль за всей системой. Благодаря этому можно легко обновлять и модифицировать инфраструктуру;

- безопасность. Благодаря централизованному управлению сервер обеспечивает полный контроль доступа и защиту всей информации.

Из основных недостатков клиент-серверной архитектуры можно выделить:

- зависимость от сети. Клиенты должны постоянно иметь соединение с сервером, при отсутствии сети приложение не будет правильно функционировать;

- полная зависимость клиента от сервера. Если сервер недоступен или работает со сбоями, то все клиенты окажутся нефункциональными и не смогут обновлять данные или обмениваться ими;
- перегрузка сервера. При большом количестве запросов сервер может не справиться и выключиться.

3.2 Архитектура приложения

В разделе «Архитектура приложения» будет рассмотрен паттерн программирования, а также взаимодействие компонентов в проекте. В качестве паттерна программирования был выбран шаблон MVC.

На данный момент одним из самых популярных шаблонов проектирования приложения является Model-View-Controller, в котором приложение делится на три основные части, а именно: модель, представление и контроллер. Каждая из этих частей выполняет свою определенную функцию, из-за чего упрощается разработка приложения и его поддержка. Архитектура серверной части данного проекта представлена на рисунке 7.

Модель 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 контроллера, а также обеспечивает изоляцию данных, что позволяет легко изменять данные без влияния на другие части приложения.

Представление отвечает за отображение данных для пользователя. Оно представляет собой интерфейс пользователя, который взаимодействует с моделью. Представление отображает данные из модели пользователя и передает пользовательские действия контроллеру.

Контроллер является связующим звеном между моделью и представлением. Он обрабатывает пользовательские запросы, взаимодействует с моделью для получения или изменения данных и передает данные в представление для отображения пользователю.

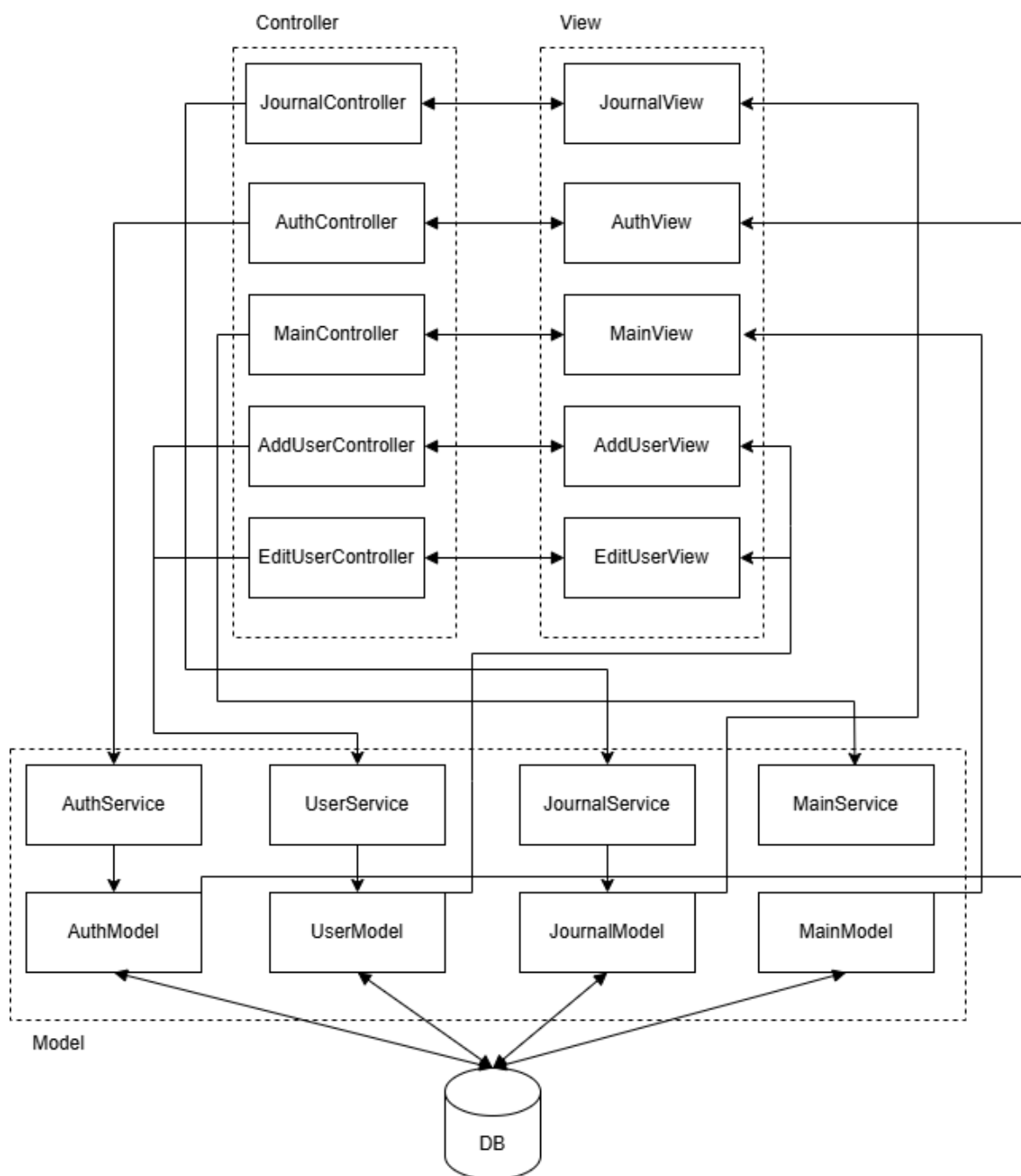


Рисунок 7 – Архитектура серверной части приложения

Основой реализации серверной части приложения будет: серверная часть приложения должна быть реализована на C# с использованием Windows Forms для создания пользовательского интерфейса; для взаимодействия с клиентами должен использоваться класс HttpListener; обмен данными между клиентом и сервером происходит с помощью Json файлов зашифрованных алгоритмом шифрования «Кузнечик»; взаимодействие с базой данных осуществляется через библиотеку

MySQL для C#, который предоставляет удобный доступ к базе данных MySQL. Для управления базой данных также может использоваться MySQL Command Line Client.

Также на рисунке 8 представлена UML диаграмма архитектуры клиентской части приложения.

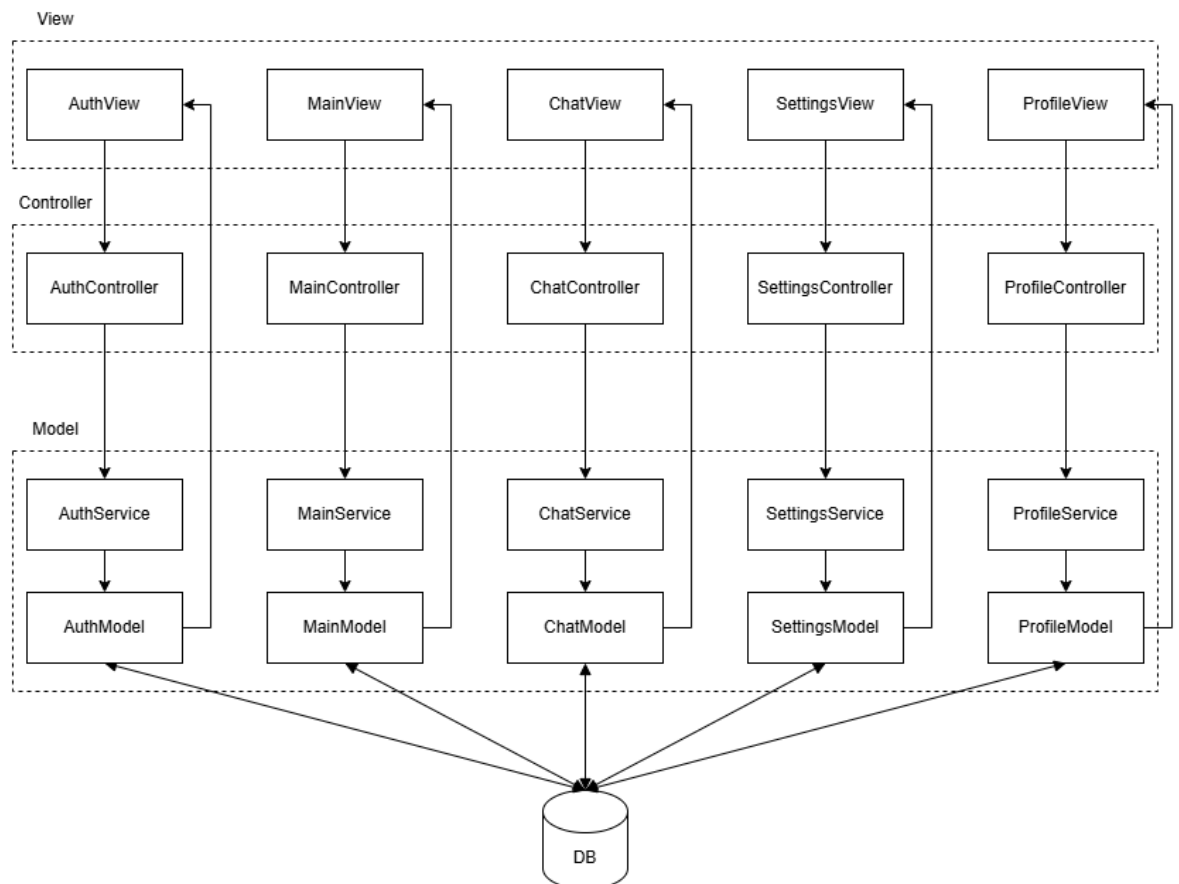


Рисунок 8 – Архитектура клиентской части приложения

Клиентская часть приложения разработана на Java; для взаимодействия с сервером должен использоваться okHttp; обмен данными между клиентом и сервером происходит с помощью Json файлов; для хранения информации используется база данных SQLite.

4 Реализация приложения

В этом разделе описан процесс разработки клиент-серверного приложения «SecureMessenger». Будут рассмотрены основные компоненты приложения, функционал и сущности, которые хранятся в базе данных. Цель этого раздела заключается в предоставлении всей необходимой информации о создании и внедрении приложения «SecureMessenger».

4.1 Описание работы приложения

Алгоритм работы приложения заключается в том, что пользователь взаимодействует с клиентской частью, написанной на Java. В зависимости от того, какие действия выполняет пользователь, будет формироваться и отправляться запрос на сервер. Сервер в свою очередь будет обрабатывать запрос. Если в процессе обработки запроса происходит ошибка, сервер сообщает об этом клиенту. В случае успешной обработки запроса сервер взаимодействует с базой данных для выполнения операций, таких как создание, редактирование, получение или удаление данных. После успешного выполнения операции с базой данных сервер по необходимости отправляет данные другому участнику чата или возвращает клиенту необходимые данные или подтверждение выполнения операции. Если при обращении к базе данных возникает ошибка, сервер обрабатывает эту ошибку и возвращает информацию о ней клиенту.

Последовательность выполняемых действий можно разделить на несколько групп, таких как:

- действия, выполняемые пользователем без взаимодействия с сервером. Клиентская часть для определенных действий не осуществляет взаимодействия с сервером. Взаимодействие идет только с локальными файлами или базой данных;
- действия, выполняемые пользователем для взаимодействия с сервером. Клиентская часть может осуществлять взаимодействия с сервером, например, обмен сообщениями с другими пользователями;

– действия, выполняемые администратором. Администратор может выполнять действия, осуществляющие только локальные для сервера действия, он не осуществляет взаимодействия с пользователями. Все его изменения влияют на серверную БД.

На рисунке 9 представлена UML диаграмма последовательности действий администратора.

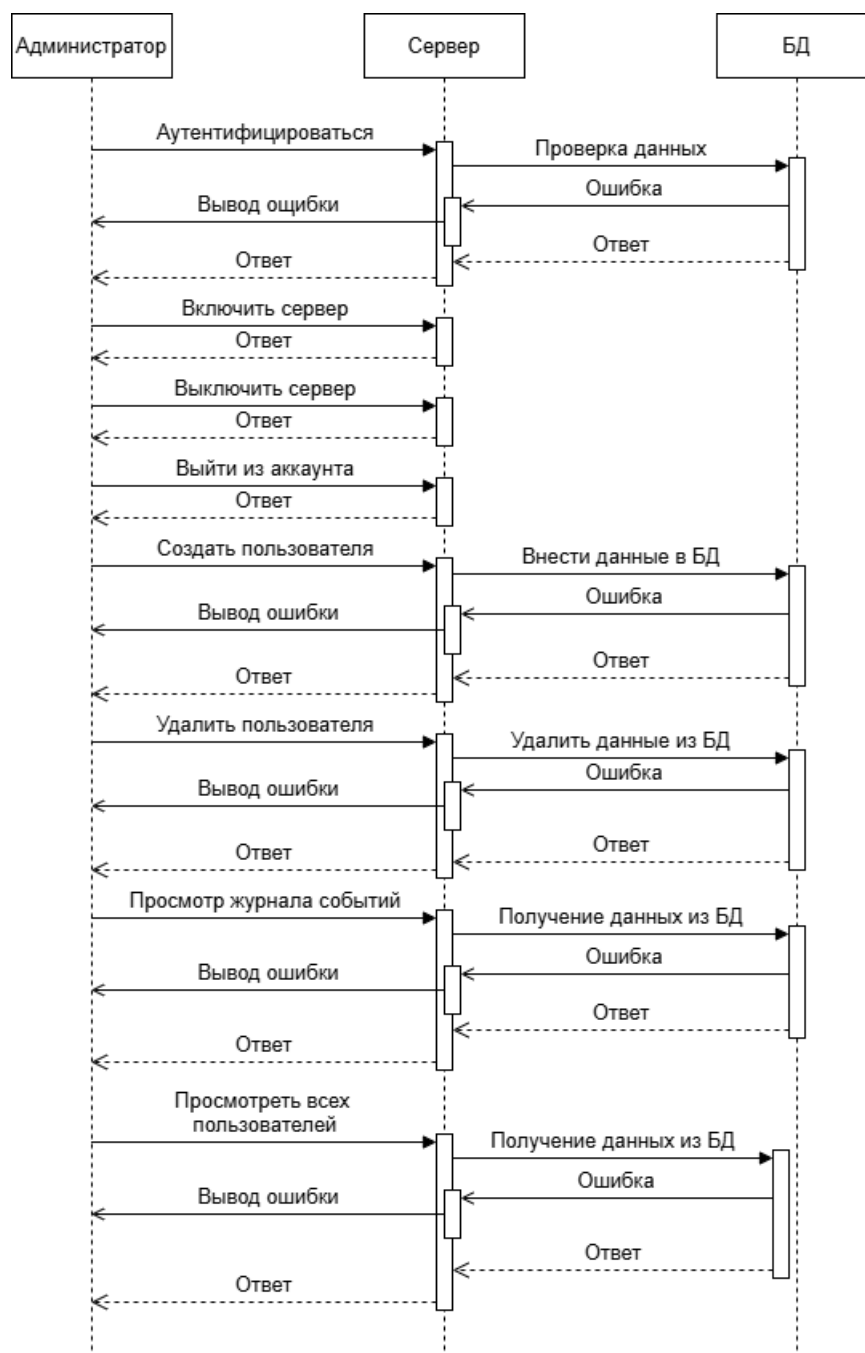


Рисунок 9 – UML диаграмма последовательности использования сервера

Последовательность действий, выполняемых пользователем представлена на рисунках 10 и 11.

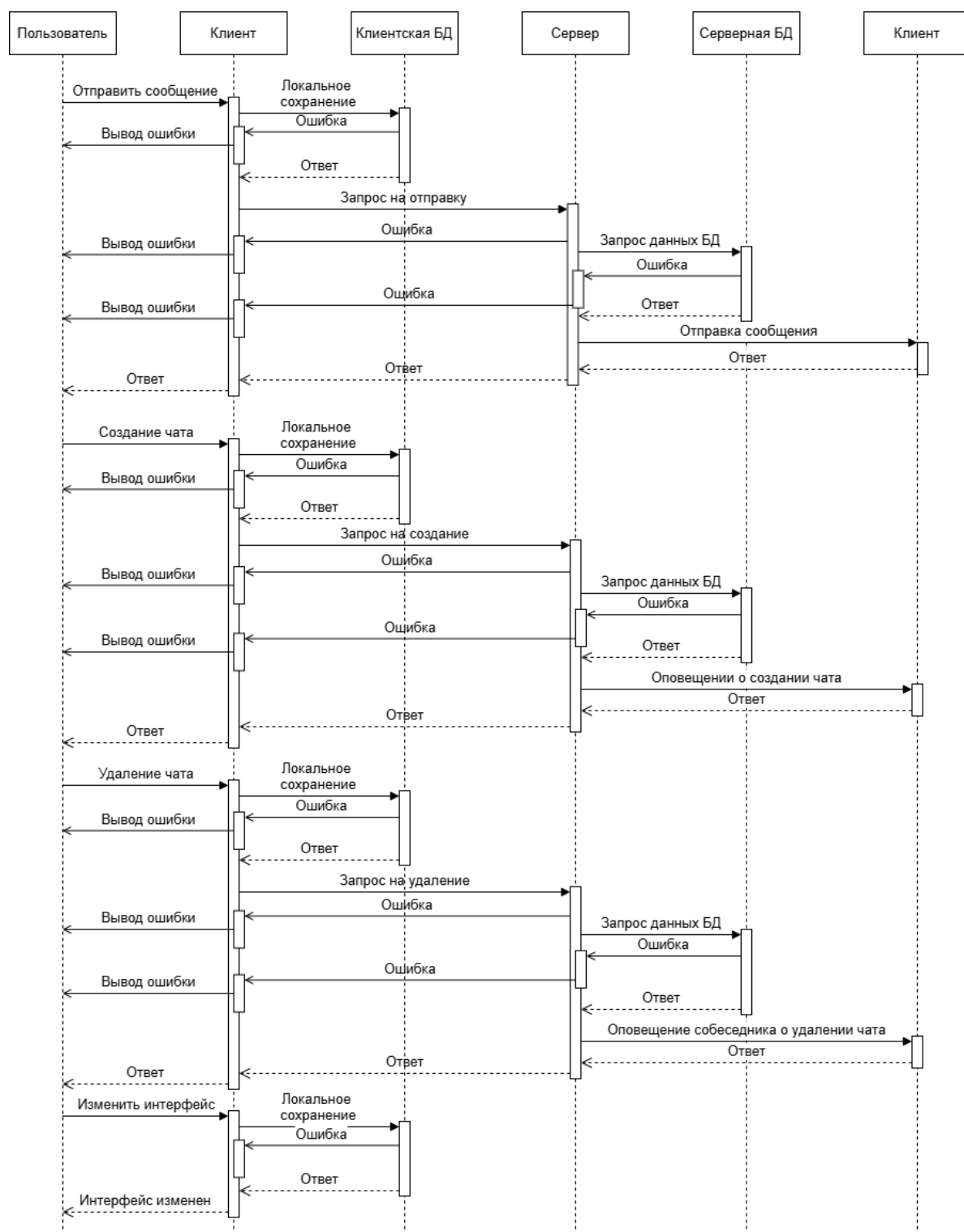


Рисунок 10 – UML диаграмма последовательности использования клиента

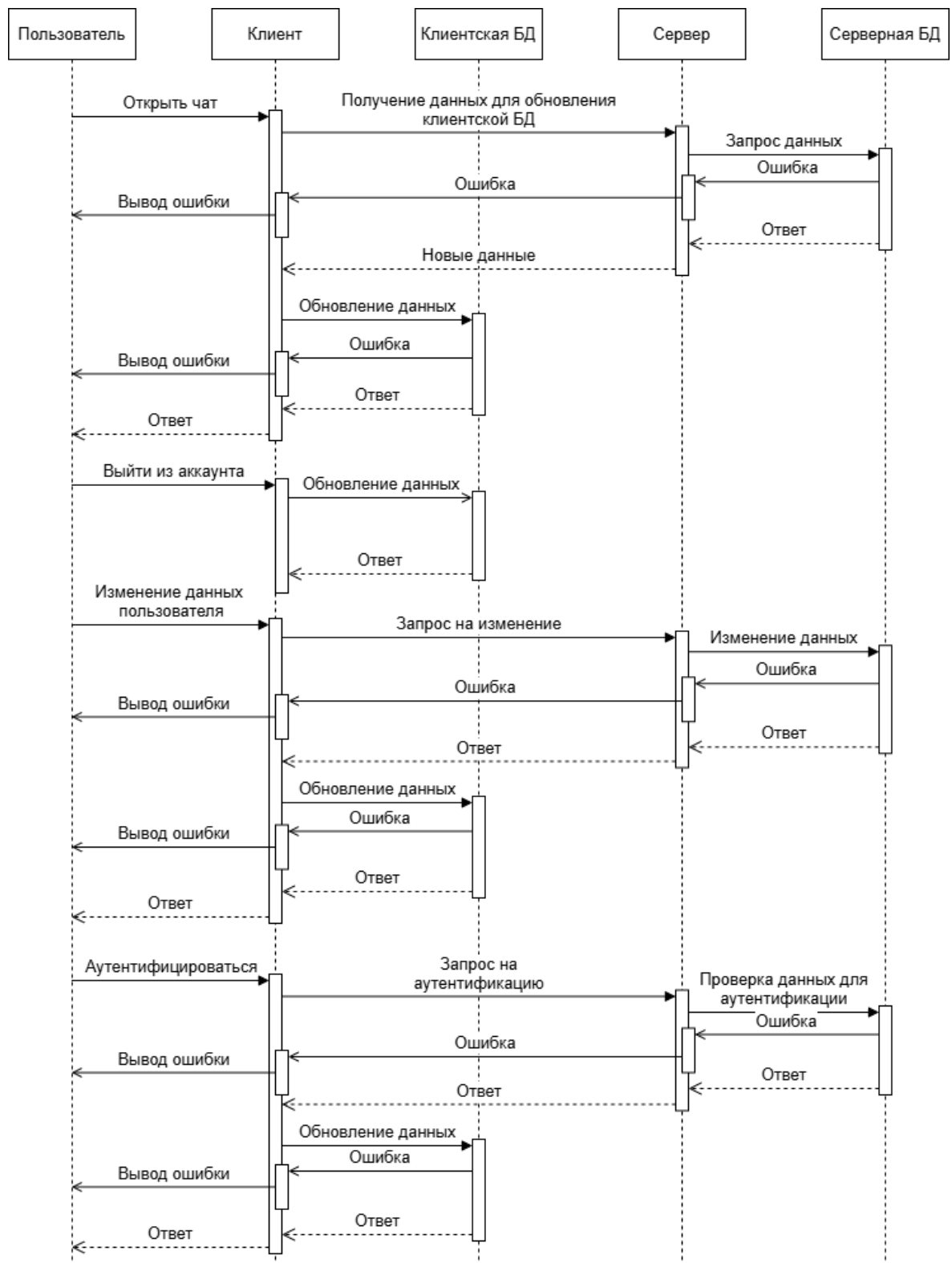


Рисунок 11 – Продолжение UML диаграммы последовательности использования клиента

Топология системы и распределение ее компонентов по физическим узлам представлены на рисунке 12.

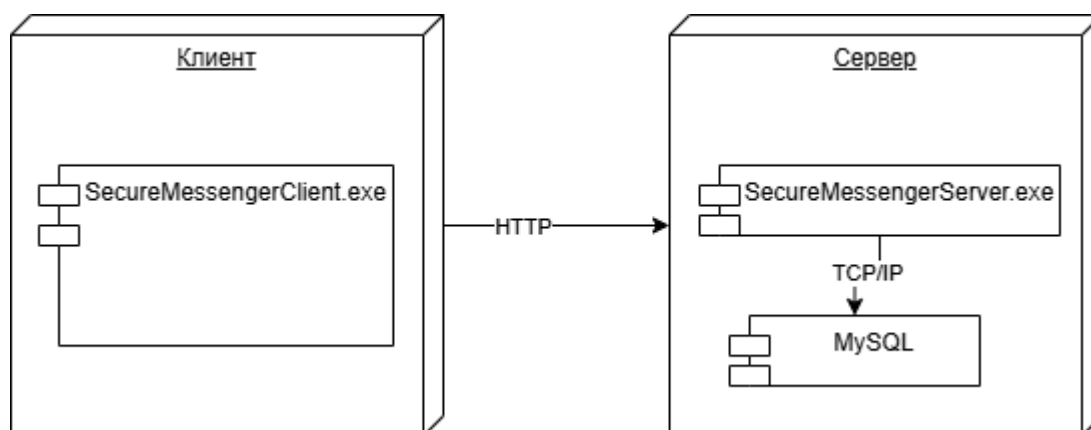


Рисунок 12 –UML диаграмма развертывания системы

Данная диаграмма развертывания позволяет понять, как физически должна быть развернута система на аппаратном обеспечении. Клиент и сервер, представленные в виде куба, являются физическими сущностями. Связанные между собой протоколом HTTP. Клиент — это мобильное устройство, на котором будет развернута клиентская часть приложения. Сервер – это персональный компьютер, на котором будет развернута серверная часть приложения, связанная с СУБД посредством протокола TCP/IP.

4.2 Структура БД

Для осуществления хранения данных приложения необходимо создать БД, которая будет хранить в себе, всю необходимую для обмена сообщениями информацию: таблицы, сущности, поля и атрибуты. Хорошо спроектированная база данных уменьшает объем занимаемой приложением памяти на жестком диске, за счет уменьшения количества сущностей, и уменьшает время разработки приложения.

Проведя анализ, было решено, что в серверной части разрабатываемого приложения будут использоваться следующие таблицы:

– таблица users, представленная в таблице 2. В ней будет содержаться основная информация о пользователях, которая будет использоваться для авторизации и идентификации пользователя в приложении;

– таблица uncompleted_transactions, представленная в таблице 3. Будет хранить основную информацию о незавершенных операциях между клиентами;

– таблица cached_messages, представленная в таблице 4. Данная таблица содержит в себе тексты сообщений незавершенных операций;

– таблица journal, представленная в таблице 5. Эта таблица будет хранить информацию журнала приложения.

Таблица 2 – Атрибуты сущности «users» (Пользователи)

Поле	Тип данных	Описание	Ключ
id_user	INT	Идентификатор пользователя	PK
login	VARCHAR	Логин пользователя	
password	VARCHAR	Пароль пользователя	
name	VARCHAR	Имя пользователя	
patronymic	VARCHAR	Отчество пользователя	
surname	VARCHAR	Фамилия пользователя	
ipaddress	VARCHAR	IP Адрес	
enckey	VARCHAR	Ключ шифрования	
status	CHAR	Статус пользователя	

Таблица 3 – Атрибуты сущности «uncompleted_transactions» (незавершённые операции)

Поле	Тип данных	Описание	Ключ
Id	INT	Идентификатор операции	PK
sender_id	INT	Идентификатор отправителя	FK
receiver_id	INT	Идентификатор получателя	FK

Поле	Тип данных	Описание	Ключ
creation_time	VARCHAR	Дата создания сущности	
status	CHAR	Статус операции	

Таблица 4 – Атрибуты сущности «cached_messages» (кэшированные сообщения)

Поле	Тип данных	Описание	Ключ
id	INT	Идентификатор	PK
transaction_id	INT	Идентификатор операции	FK
text_message	TEXT	Текст сообщения	

Таблица 5 – Атрибуты сущности «journal» (Статистика пользователя)

Поле	Тип данных	Описание	Ключ
id	INT	Идентификатор записи	PK
service	VARCHAR	Источник события	
text	TEXT	Описание	
creation_time	VARCHAR	Дата записи	

На рисунке 13 описана схема базы данных, серверной части разрабатываемого приложения, для наглядного рассмотрения взаимодействия таблиц.

На схеме отображены все поля и атрибуты всех таблиц приложения, а также все связи между таблицами.

Единственной несвязанной таблице является таблица «journal». Она не связана с другими таблицами так как единственное ее применение — это отслеживание событий сервера.

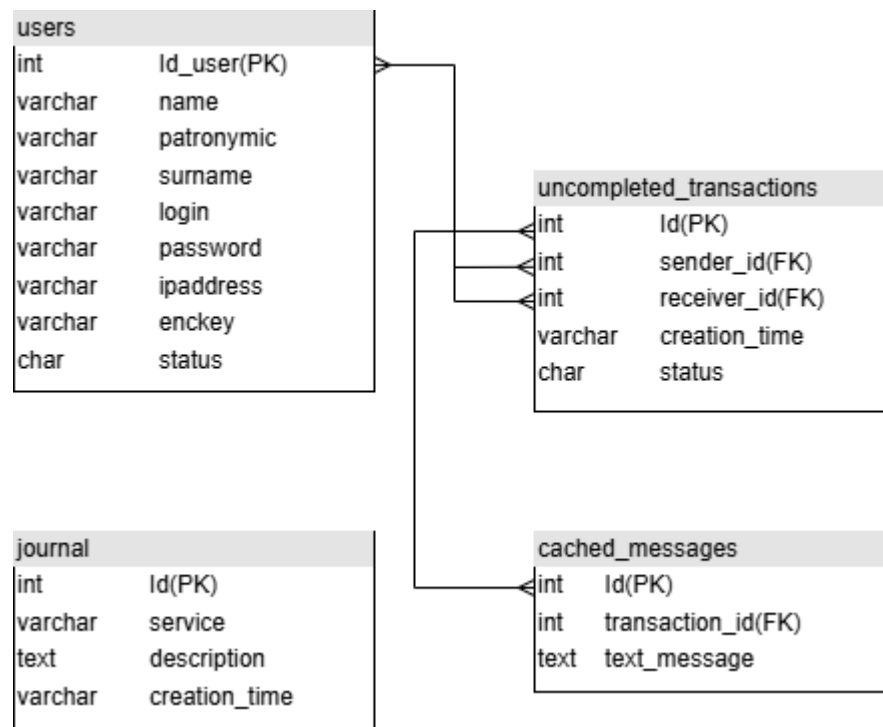


Рисунок 13 – Схема базы данных серверной части приложения

В клиентской части разрабатываемого приложения будут использоваться следующие таблицы:

- таблица messages, представленная в таблице 6. В ней будет содержаться информация о сообщениях, которая будет использоваться для авторизации и идентификации пользователя в приложении;
- таблица chats, представленная в таблице 7. Будет хранить основную информацию о чатах пользователя.

Таблица 6 – Атрибуты сущности «chats» (Чаты)

Поле	Тип данных	Описание	Ключ
id	INT	Идентификатор чата	PK
m_name	VARCHAR	Имя собеседника	
m_patronymic	VARCHAR	Отчество собеседника	
m_surname	VARCHAR	Фамилия собеседника	
login	VARCHAR	Логин собеседника	

Поле	Тип данных	Описание	Ключ
status	VARCHAR	Статус собеседника	
img_name	VARCHAR	Аватар собеседника	

Таблица 7– Атрибуты сущности «messages» (Сообщения)

Поле	Тип данных	Описание	Ключ
id	INT	Идентификатор сообщения	PK
chat_id	VARCHAR	Идентификатор чата	FK
text	VARCHAR	Текст сообщения	
datetime	DATETIME	Время создания	
status	INT	Статус сообщения	
type	BOOLEAN	Тип сообщения	

На рисунке 14 представлена схема базы данных клиентской части.

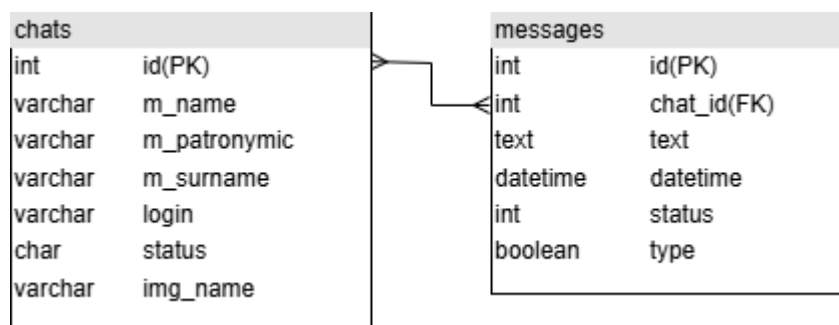


Рисунок 14 - Схема базы данных клиентской части приложения

Таким образом спроектирована логическая структура баз данных клиентской и серверной частей приложения, а также определены атрибуты для каждой сущности.

4.3 Программная реализация приложения

Данное клиент-серверное приложение содержит два программных модуля:

- SecureMessengerClient;
- SecureMessengerServer.

SecureMessengerClient – это программный модуль, который содержит реализацию компонентов представления, контроллера и модели, а именно пользовательский интерфейс, классы для взаимодействия с сервером и классы взаимодействия с локальной базой данных.

SecureMessengerServer – программный модуль, содержащий набор классов для полной реализации MVC паттерна. Функционал этих классов заключается в обработке запросов пользователей, взаимодействия с базой данных, пересылке данных между клиентами и реализации пользовательского интерфейса.

4.3.1 SecureMessengerClient

Данный программный модуль содержит в себе полную реализацию такого паттерна как MVC.

Далее будут рассмотрены основные классы компонента:

- MainActivity. Класс обработчик событий основного окна клиента приложения;
- DatabaseHandler. Класс, обеспечивающий взаимодействие с базой данных пользователя;
- HttpClient. С помощью данного класса клиент осуществляет взаимодействие с сервером.

Основными методами данных классов являются:

- SendRequest();
- AddChat();
- onCreate().

Каждый из этих методов будет подробнее описан далее.

Метод SendRequest() класса HttpClient осуществляет отправку запроса на сервер и получение ответа от сервера. Реализация метода приведена в листинге 1.

Листинг 1. SendRequest():

```
public SendRequest(String req){
```

```

MediaType JSON = MediaType.get("application/json; charset=utf-8");

OkHttpClient client = new OkHttpClient();
RequestBody body = RequestBody.create(req, JSON);
Request.Builder requestBuilder = new
Request.Builder().url(serverUrl).post(body);
Request request = requestBuilder.build();

try (Response response = client.newCall(request).execute()) {
    if (!response.isSuccessful()) {
        throw new IOException("Запрос к серверу не был успешен: " +
            response.code() + " " + response.message());
    }
    System.out.println(response.body().string());
} catch (IOException e) {
    System.out.println("Ошибка подключения: " + e);
}
}

```

Метод AddChat() класса DatabaseHandler осуществляет добавление нового чата в локальную базу данных. Реализация метода приведена в листинге 2.

Листинг 2. AddChat():

```

public void addChat(Chat chat) {
    SQLiteDatabase db = this.getWritableDatabase();
    ContentValues values = new ContentValues();
    values.put(DBContract.UserEntry.COLUMN_NAME_LOGIN,
chat.GetLogin());
    values.put(DBContract.UserEntry.COLUMN_NAME_NAME, chat.getName());

    db.insert(DBContract.UserEntry.TABLE_NAME, null, values);
    db.close();
}

```

Метод onCreate() класса MainActivity является одним из методов жизненного цикла основного окна клиента приложения и осуществляет привязку слушателей ко всем активным элементам данного окна. Реализация метода приведена в листинге 3.

Листинг 3. onCreate():

```

protected void onCreate(@Nullable Bundle savedInstanceState)
{
    super.onCreate(savedInstanceState);

    setContentView(R.layout.main_layout);

    ArrayList<Chat> chats = new ArrayList<Chat>();
    chatList = findViewById(R.id.chatsList);
    adapter = new ChatAdapter(this, R.layout.chat_item_layout, chats);
    chatList.setAdapter(adapter);

    addChatBtn = findViewById(R.id.add_chat_btn);
    confirmAddChatBtn = findViewById(R.id.conform_adding_chat_btn);
}

```

```

cancelAddChatBtn = findViewById(R.id.cancel_adding_chat_btn);
newUserLoginEdit = findViewById(R.id.new_chat_login_edit);

addChatBtn.setOnClickListener((View v)->
{
    Toast.makeText(MainActivity.this,
                    R.string.sample_text,
                    Toast.LENGTH_SHORT).show();

    confirmAddChatBtn.setClickable(true);
    cancelAddChatBtn.setClickable(true);
    newUserLoginEdit.setClickable(true);
    v.setClickable(false);

    confirmAddChatBtn.setVisibility(View.VISIBLE);
    cancelAddChatBtn.setVisibility(View.VISIBLE);
    newUserLoginEdit.setVisibility(View.VISIBLE);
    v.setVisibility(View.INVISIBLE);
});

confirmAddChatBtn.setOnClickListener((View v) ->
{
    String newUserLogin =
String.valueOf(newUserLoginEdit.getText());

    adapter.add(new Chat(0, newUserLogin, newUserLogin,
newUserLogin));
    adapter.notifyDataSetChanged();
});

cancelAddChatBtn.setOnClickListener((View v) ->
{
    Toast.makeText(MainActivity.this,
                    R.string.sample_text,
                    Toast.LENGTH_SHORT).show();

    confirmAddChatBtn.setClickable(false);
    addChatBtn.setClickable(true);
    newUserLoginEdit.setClickable(false);
    v.setClickable(false);

    confirmAddChatBtn.setVisibility(View.INVISIBLE);
    addChatBtn.setVisibility(View.VISIBLE);
    newUserLoginEdit.setVisibility(View.INVISIBLE);
    v.setVisibility(View.INVISIBLE);
});
}

```

4.3.2 SecureMessengerServer

Данный компонент содержит в себе реализацию пользовательского интерфейса, механизм взаимодействия с базой данных и обработки сетевых запросов.

Будут рассмотрены основные классы компонента, а именно:

					ПГУ 1.090301.26.001 ПЗ	Лист
						42
Изм.	Лист	№ докум.	Подпись	Дата		

- HttpServer. Данный класс реализует интерфейс сетевого взаимодействия сервера;
- DbConnector. С помощью данного осуществляются все взаимодействия с базой данных сервера;
- MainForm. Класс реализующий пользовательский интерфейс главного окна приложения.

Далее будут рассмотрены основные методы, описанных ранее классов сервера, такие как:

- dataTableGridView_CellClick();
- ServerControl_Click();
- UsersList_Click();
- Start();
- HandleRequests();
- Processor();
- Select();
- InsertInto().

Метод dataTableGridView_CellClick() является слушателем события нажатия на ячейку таблицы расположенной на главном экране. В зависимости от выведенной таблицы, метод предоставляет разные возможности. Для таблицы пользователей он открывает полную информацию о выбранном пользователе. Реализация метода приведена в листинге 4.

Листинг 4. dataTableGridView_CellClick():

```
private void dataTableGridView_CellClick(object sender,
DataGridViewCellEventArgs e)
{
    if (dataTableGridView.RowCount == 1)
        return;

    switch (selectedTable)
    {
        case SelectedTable.journal:
        {
            break;
        }
        case SelectedTable.users:
```

```

        {
            int userId =
(int)dataTableGridView.Rows[e.RowIndex].Cells[0].Value;
            var userInfoForm = new UserInfoForm(userId);
            userInfoForm.StartPosition =
FormStartPosition.CenterParent;
            userInfoForm.ShowDialog(this);

            break;
        }
        default:
            return;
    }
}

```

Метод `ServerControl_Click()` является слушателем события нажатия на кнопку главного окна приложения «Запуск сервера». Метод выполняет запуск прослушивания подключений от клиентов. Реализация метода приведена в листинге 5.

Листинг 5. `dataTableGridView_CellClick()`:

```

private void ServerControl_Click(object sender, EventArgs e)
{
    try
    {
        httpServer.Start();
        serverInfoShort.Text = $"Сервер запущен по адресу:
{httpServer.Url}\n";

        MessageBox.Show(this, "Сервер запущен", "Оповещение",
MessageBoxButtons.OK, MessageBoxIcon.Information);
    }
    catch (Exception ex){}
}

```

Метод `UsersList_Click()` является слушателем события нажатия на кнопку главного окна приложения «Все пользователи». Метод выполняет вывод всей таблицы пользователей сервера в таблицу главного экрана. Реализация метода приведена в листинге 6.

Листинг 6. `UsersList_Click()`:

```

private void UsersList_Click(object sender, EventArgs e)
{
    selectedTable = SelectedTable.users;
    userDbConnector.Select(dataTableGridView);
}

```

Метод `Start()` класса `HttpServer` выполняет запуск отдельного потока для прослушивания подключения и запуск пула потоков обработки полученных при прослушивании запросов. Реализация метода приведена в листинге 7.

Листинг 7. Start():

```
public void Start()
{
    listenerLoop = new Thread(HandleRequests);

    foreach (string prefix in Prefixes) httpListener.Prefixes.Add(prefix);

    listenerLoop.Start();

    for (int i = 0; i < requestProcessors.Length; i++)
    {
        requestProcessors[i] = StartProcessor(i, messages);
    }
}
```

Метод `HandleRequests()` класса `HttpServer` в отдельном потоке ожидает подключения клиентов, при подключении он добавляет в список новый объект запроса, который после забирает любой свободный поток обработки. Реализация метода приведена в листинге 8.

Листинг 8. HandleRequests():

```
private void HandleRequests()
{
    httpListener.Start();
    journalDbConnector.InsertInto(null, new JournalRecord("HttpServer",
"start listening"));

    try
    {
        while (httpListener.IsListening)
        {
            HttpListenerContext context = httpListener.GetContext();

            messages.Add(context);

            journalDbConnector.InsertInto(null, new
JournalRecord("HttpServer", "new connection" +
context.Request.RemoteEndPoint.ToString()));
        }
    }
    catch (Exception e)
    {
        Console.WriteLine(e.Message);
    }
}
```

Метод `Processor()` забирает из списка поступивших запросов последний запрос и осуществляет его обработку с последующей отправкой ответа клиенту. Реализация метода приведена в листинге 7.

Листинг 7. Processor():

```

private void Processor(int number, BlockingCollection<HttpListenerContext>
messages)
{
    try
    {
        for ( ; ; )
        {
            HttpListenerContext context = messages.Take();
            if (context == null)
                continue;

            HttpHandler httpHandler = new HttpHandler(context);
            httpHandler.Run();
        }
    }
    catch { }
}

```

Метод Select() выполняет Select запрос к определённой таблице базы данных для вывода этой таблицы или определенной ее строки. Реализация метода приведена в листинге 9.

Листинг 9. Select():

```

public bool Select(DataGridView dataGrid)
{
    using (MySqlConnection con = new
MySqlConnection(base.connectionString_))
    {
        using (MySqlDataAdapter a = new MySqlDataAdapter("SELECT * FROM
journal", con))
        {
            DataTable t = new DataTable();
            a.Fill(t);

            dataGrid.AutoGenerateColumns = true;
            dataGrid.DataSource = t;
        }
    }

    return true;
}

```

Метод InsertInto() осуществляет взаимодействие с определенной таблицей БД для добавления в нее элемента. Реализация метода приведена в листинге 10.

Листинг 10. InsertInto():

```

public bool InsertInto(DataGridView dataGrid, JournalRecord record)
{
    using (MySqlConnection con = new
MySqlConnection(base.connectionString_))
    {
        using (MySqlCommand cmd = new MySqlCommand())
        {

```

```

        con.Open();

        cmd.Connection = con;
        cmd.CommandText = "insert into journal (service, description,
creation_time)" +
                            " values(@service, @description,
@creation_time)";

        cmd.Parameters.AddWithValue("@service", record.Service);
        cmd.Parameters.AddWithValue("@description",
record.Description);
        cmd.Parameters.AddWithValue("@creation_time",
record.CreationDateTime);
        cmd.Prepare();

        cmd.ExecuteReader();
    }
}

return true;
}

```

4.4 Основные алгоритмы взаимодействия клиента и сервера

UML диаграммы деятельности разработаны для наглядного отображения последовательности действий между клиентской и серверной частями в процессе взаимодействия.

Для начала будет рассмотрен алгоритм авторизации пользователя в приложении. При открытии приложения пользователя попадает на страницу авторизации. Для авторизации необходимо заполнить соответствующие поля логина и пароля, далее необходимо нажать кнопку «Войти». После нажатия кнопки войти данные проходят проверку на валидность на устройстве. После чего данные отправляются на сервер для проверки. На сервере данные прогоняются по базе данных, для проверки действительности, если данные действительны, то пользователю отправляется ответ, зашифрованный его ключом, далее приложение расшифровывает полученные данные, и открывается главная страница приложения. После этого приложение загрузит данные из локальный БД на экран, а в это время в другом потоке будет происходить получение сообщений или изменений от других пользователей. После получения обновленных данных экран приложения перезагрузится.

Если данные введены некорректно или нет соединения с сервером, то выведется сообщение об ошибке и пользователь останется на экране авторизации.

Последовательность описанных действий представлена на рисунке 15.

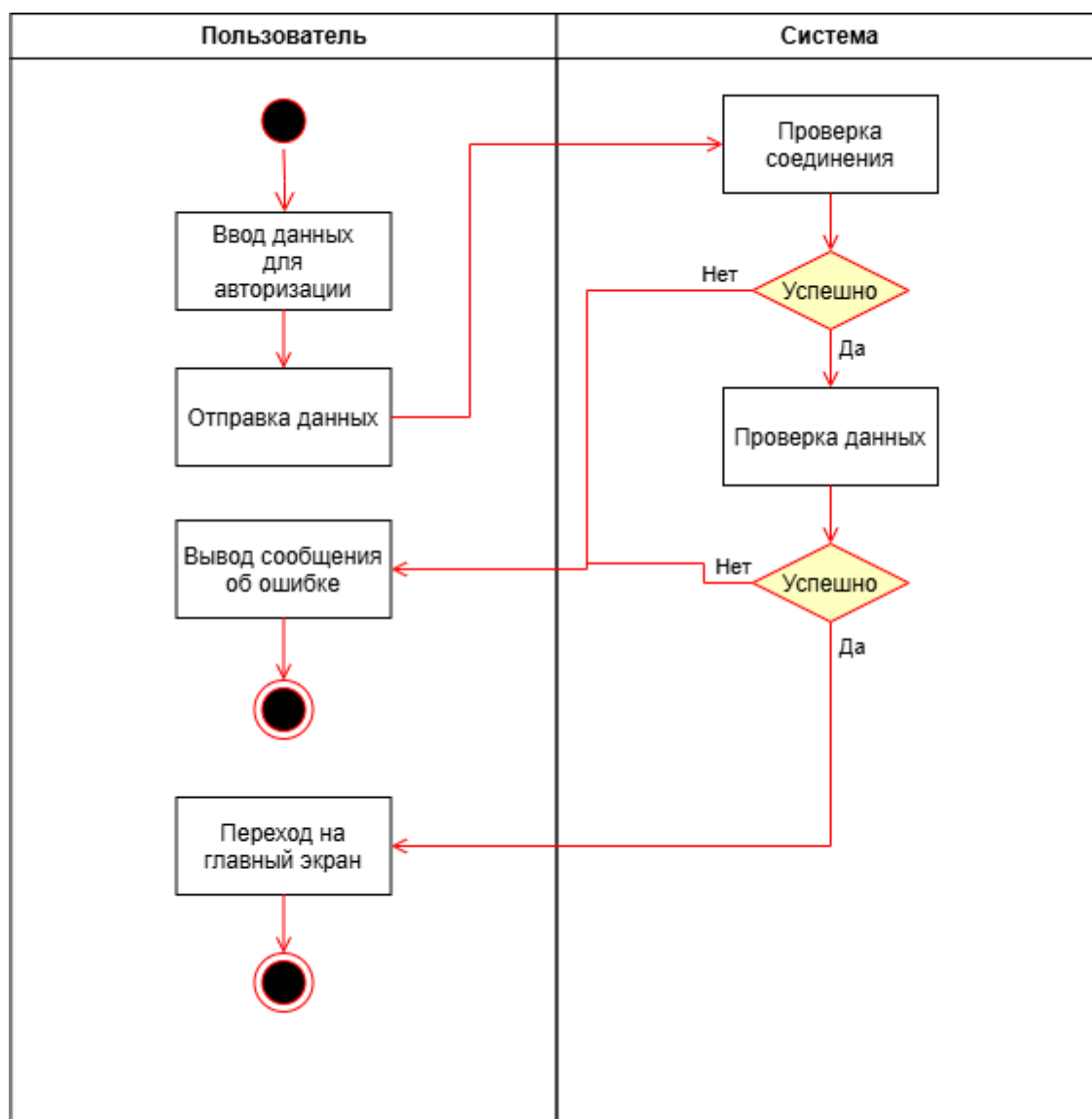


Рисунок 15 - UML диаграмма деятельность авторизации

Для регистрации нового пользователя, администратору нужно нажать на кнопку «Создать пользователя» в пользовательском интерфейсе серверной части приложения. Далее необходимо ввести данные пользователя, после введения данных и клике на кнопку «Создать» будет произведена проверка валидности и уникальности данных. Если логин пользователя будет не уникален выведется

сообщение об ошибке, тоже самое произойдет если пароль пользователя слишком короткий.

После этого для пользователя будет сгенерирован его уникальный ключ шифрования.

На рисунке 16 представлена UML диаграмма деятельности регистрации нового пользователя.

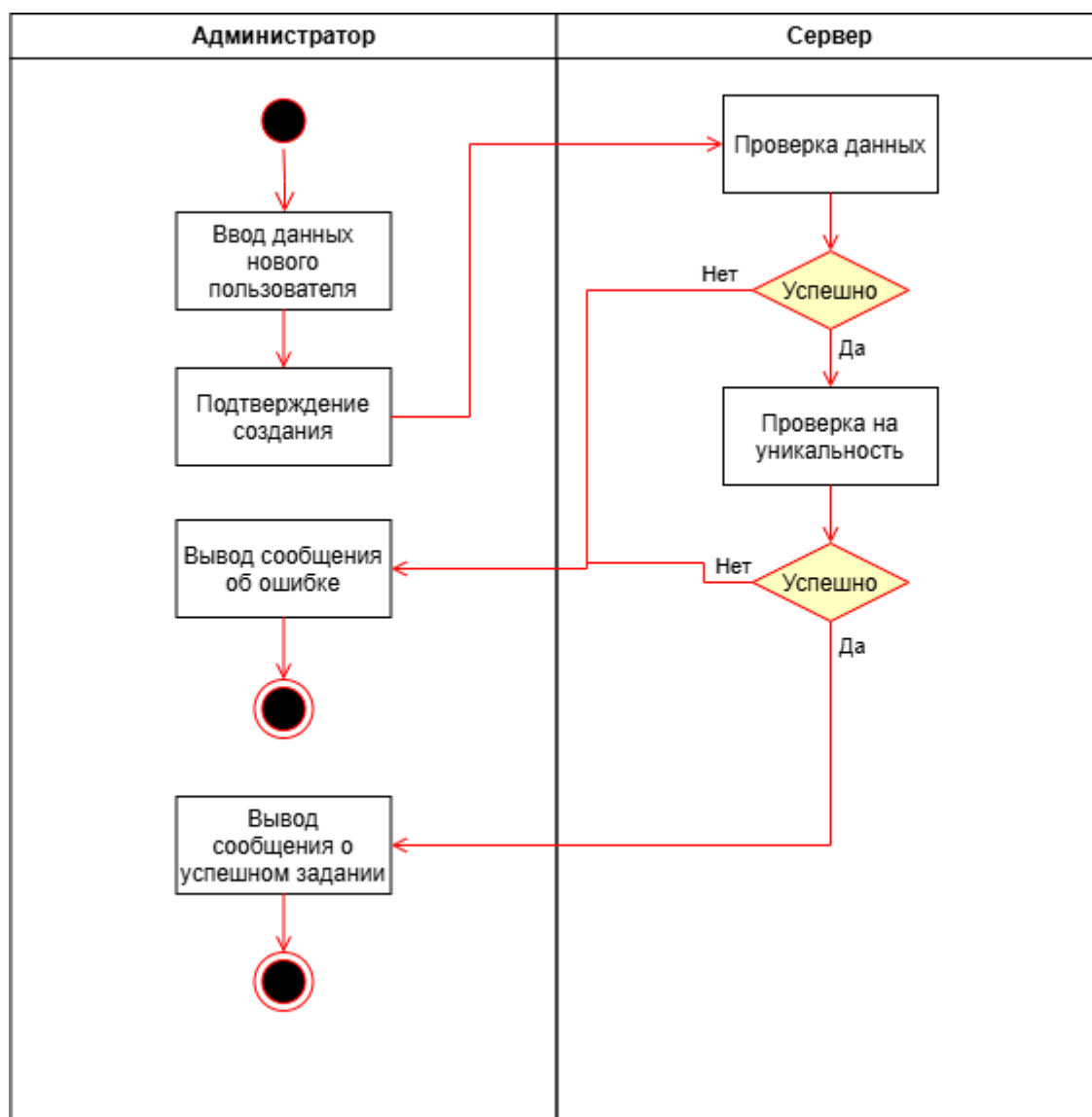


Рисунок 16 - UML диаграмма последовательности регистрации нового пользователя

Далее будет рассмотрен алгоритм изменения настроек пользователя. Для изменения настроек профиля необходимо перейти с главной страницы в окно профиля, это можно сделать кликом на три вертикальные точки расположенные в правом краю шапки приложения. Далее открывается меню приложения, из представленных вариантов, а именно «Профиль», «Настройки» и «Выйти» необходимо кликнуть на «Профиль», после чего произойдет переход на страницу настроек профиля пользователя. После изменения настроек на выбор есть два варианта дальнейших действий: отменить изменения или сохранить.

При клике на «Отмена», пользователя перекинет на главный экран приложения, а данные изменены не будут.

При клике на «Сохранить», сначала произойдет проверка измененных данных на валидность. Если не пройдена, то некорректно введенная строка данных будет подсвечена красным цветом, данные в свою очередь изменены не будут. Если проверка пройдена, то далее будет произведена проверка соединения, если соединение не установлено, то будет выведено сообщение об ошибке, а данные не изменятся, если соединение установлено, то произойдет шифровка данных ключом пользователя и отправка на сервер.

На сервере пройдет расшифровка переданных данных пользователем. Если полученные данные корректны, то они будут сохранены на сервере, а пользователю вернется сообщение о успешном сохранении данных.

UML диаграмма данного алгоритма представлена на рисунке 17.

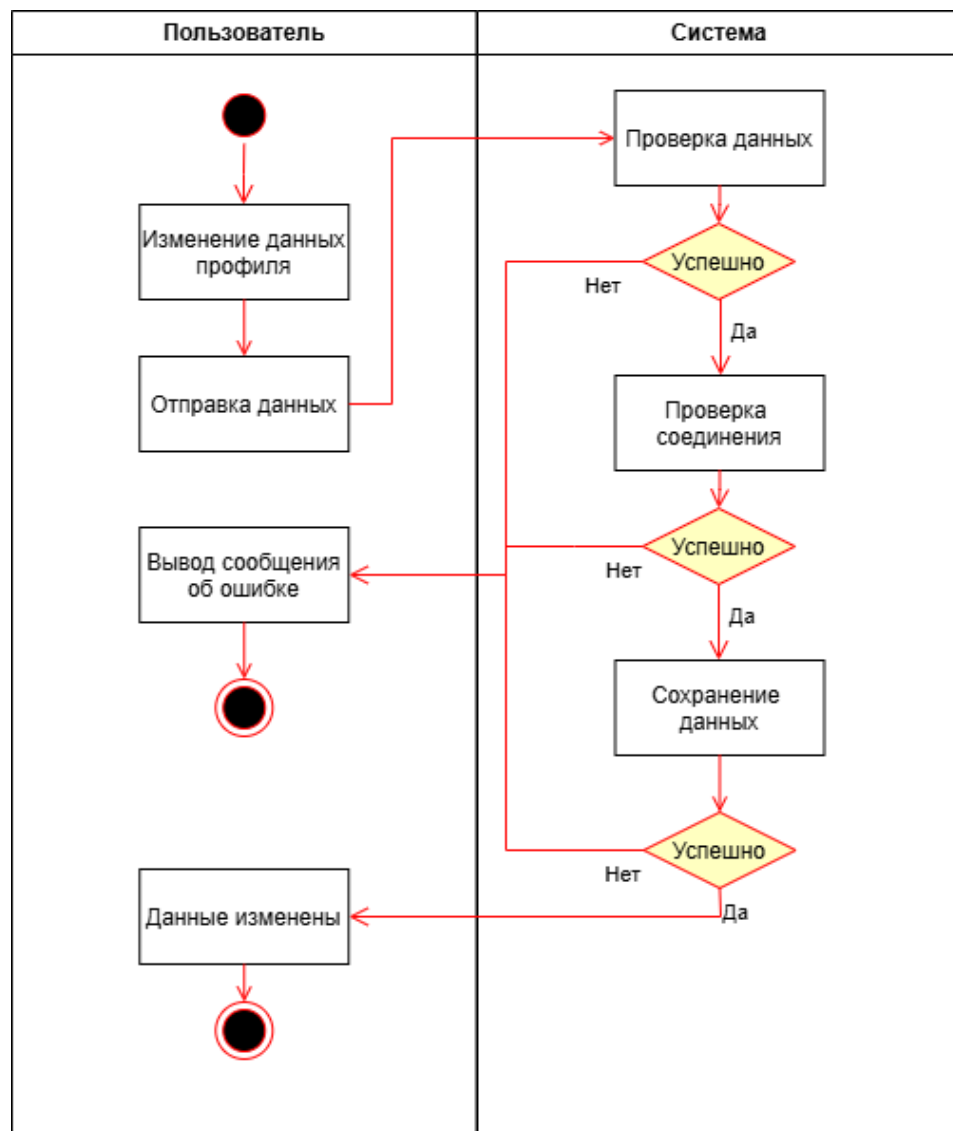


Рисунок 17 - UML диаграмма последовательности действий изменения настроек профиля

На рисунке 18 представлен алгоритм отправки сообщения. Для отправки сообщения необходимо перейти в чат с собеседником, сообщение которому нужно отправить. При переходе в окно чата, происходит обращение к локальной базе данных для вывода истории сообщений. Далее необходимо ввести текст сообщения, если при клике на кнопку «Отправить» в виде иконки сообщения со стрелочкой, направленной вправо, в поле ввода текста ничего нет, то сообщение отправлено не будет.

После клика на кнопку «Отправить» происходит проверка соединения, если соединения нет, то сообщение не будет отправлено. Если данные сообщения валидны и подключение к серверу присутствует, то данные для отправки начнут шифроваться ключом пользователя, после чего произойдет отправка данных на сервер.

На сервере данные расшифровываются ключом отправителя и шифруются ключом получателя, которому отправляется сообщение, после чего просматривается статус получателя, если его статус офлайн или сообщение не удалось доставить, то на сервере создается запись о неоконченной транзакции, а сообщение кэшируется. В таком случае отправителю сообщения, не придет ответа с сервера о успешной доставке сообщения. Кэшированные сообщения хранятся на сервере определенный период времени, после чего удаляются. Если сообщение успешно доставлено получателю, то сервер отвечает отправителю о успешной доставке и отправленное сообщение помечается как «Доставленное».

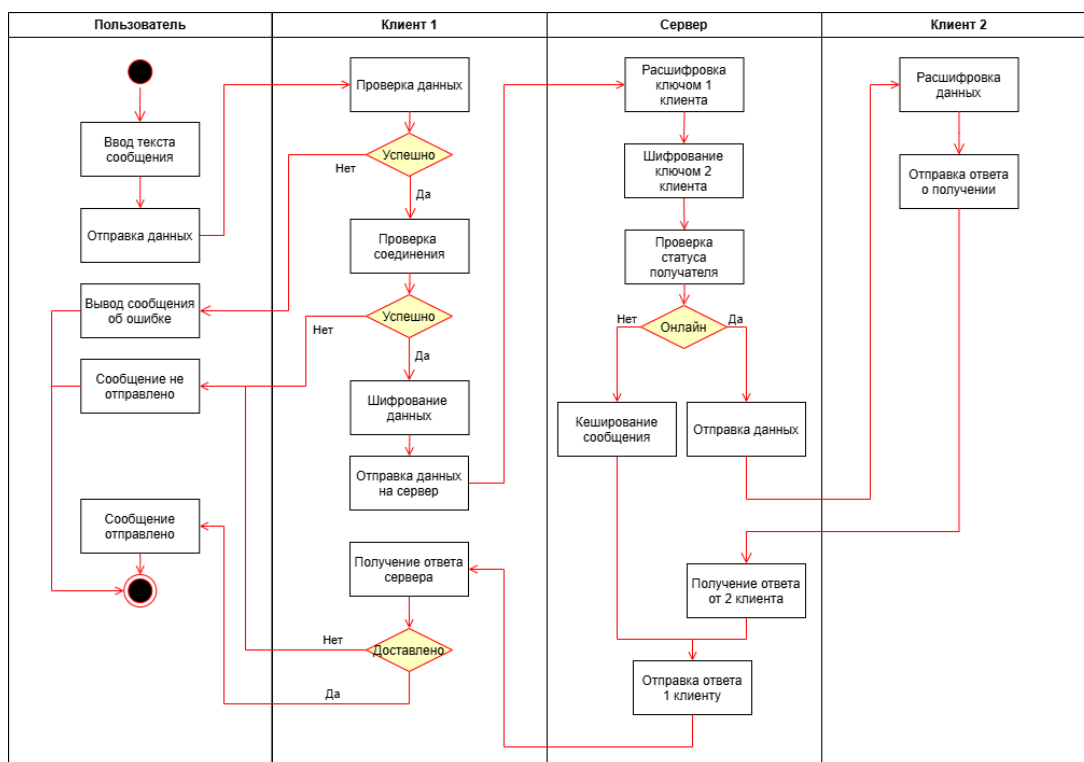


Рисунок 18 - UML диаграмма последовательности действий отправки сообщения

Для надежности работы приложения и устойчивости к потере соединения реализован механизм кеширования сообщений. Суть алгоритма состоит в том, что при какой-либо неудачной сетевой операции, сервер создает новую сущность в таблице незавершенных транзакций. Данная таблица содержит в себе такую информацию как id отправителя и id получателя, состояние транзакции и дата создания сущности. Также в случае, если незавершенная транзакция имеет статус «Не доставлено получателю», создается новая сущность в таблице кешированных сообщений, в таблице содержится информация о идентификаторе незавершённой транзакции, к которой принадлежит данное сообщение, также содержится и сам текст сообщения. Если транзакция имеет статус «Не доставлен ответ отправителю», новой сущности в таблице кэшированных сообщений не создается. Пример работы данного механизма:

Клиент 1 отправляет сообщение клиенту 2, сервер получает сообщение, далее просматривает статус клиента 2, если он не в сети, сервер создает новую незавершенную транзакцию и кеширует текст сообщения. Клиент 1, не дождавшись ответа разрывает соединение, а сообщение остается с меткой отправлено.

Когда клиент 2 снова заходит в приложение, он отправляет запрос на обновление своих данных и получает данные о незавершённой транзакции, участником которой он является. Кэшированное сообщение удаляется.

Далее сервер смотрит на статус клиента 1, если он не в сети, то статус транзакции меняется. Как только клиент 1 снова заходит в приложение, и получает с сервера данные, то транзакция удаляется, а отправленное сообщение меняет статус на доставленное. Данная последовательность представлена на рисунке 19.

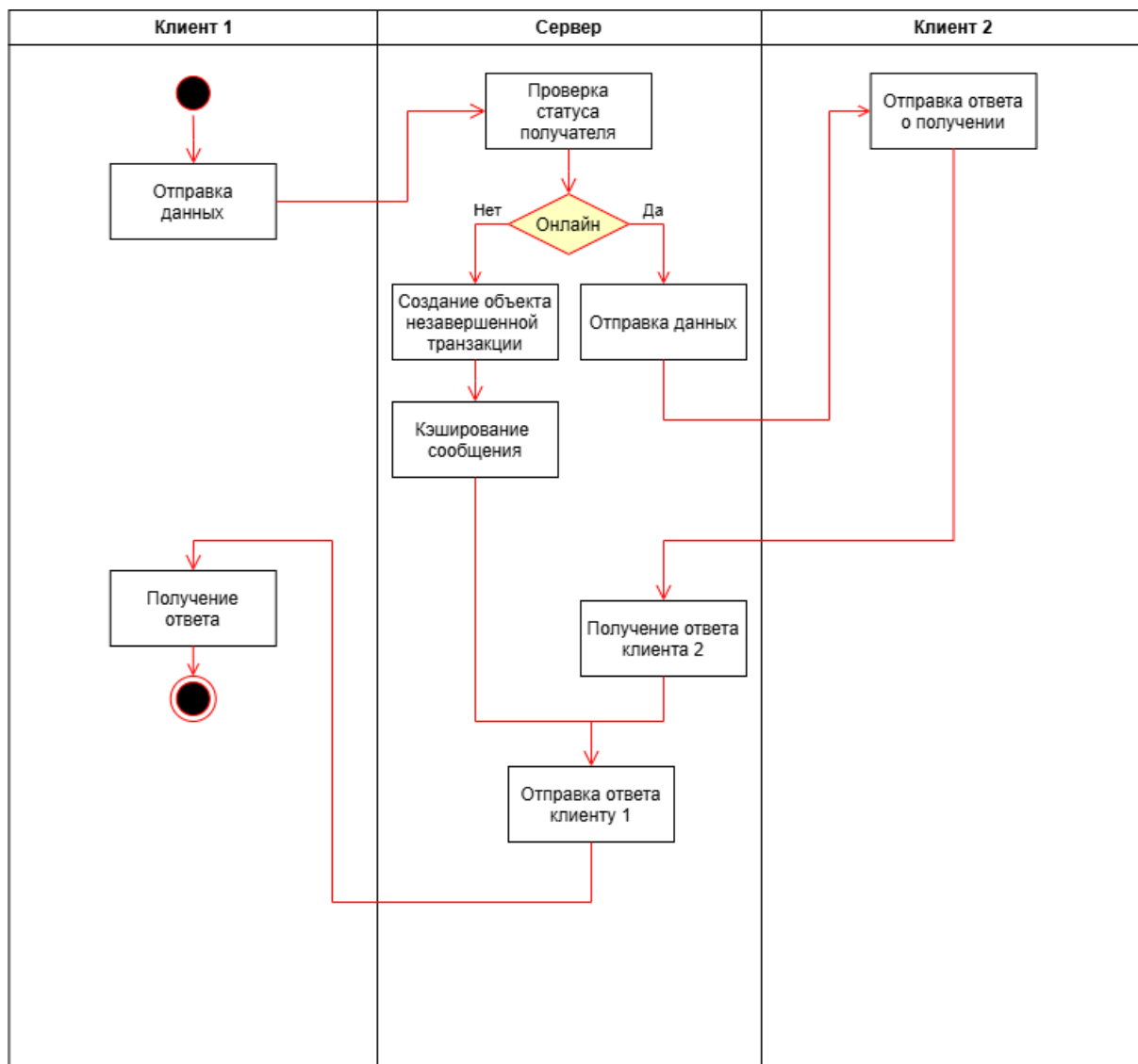


Рисунок 19 – UML диаграмма деятельности кэширования сообщения

Таким образом были подробно описаны и рассмотрены основные алгоритмы деятельности пользователя и администратора приложения, а также были рассмотрены некоторые алгоритмы программной реализации приложения.

5 Тестирование приложения

Для проверки работоспособности клиент-серверного приложения «Мессенджер с шифрованием сообщений» необходимо провести его тестирование.

Тестирование будет состоять из следующих действий:

- регистрация пользователя администратором;
- запуск сервера;
- просмотр всех пользователей администратором;
- просмотр журнала событий системы;
- авторизация пользователя;
- просмотр чатов пользователем;
- просмотр чата пользователем;
- отправка сообщения пользователем;
- изменение настроек приложения пользователем;
- просмотр своего профиля пользователем;
- добавление чата пользователем.

5.1 Регистрация пользователя администратором.

Чтобы протестировать регистрацию пользователя администратором, нужно перейти на страницу с формой регистрации и ввести следующие поля: логин, пароль, имя, отчество и фамилию. После того, как пользователь заполнит необходимые данные, в базу данных внесется новая запись о пользователе.

На рисунке 20 представлено окно регистрации нового пользователя.

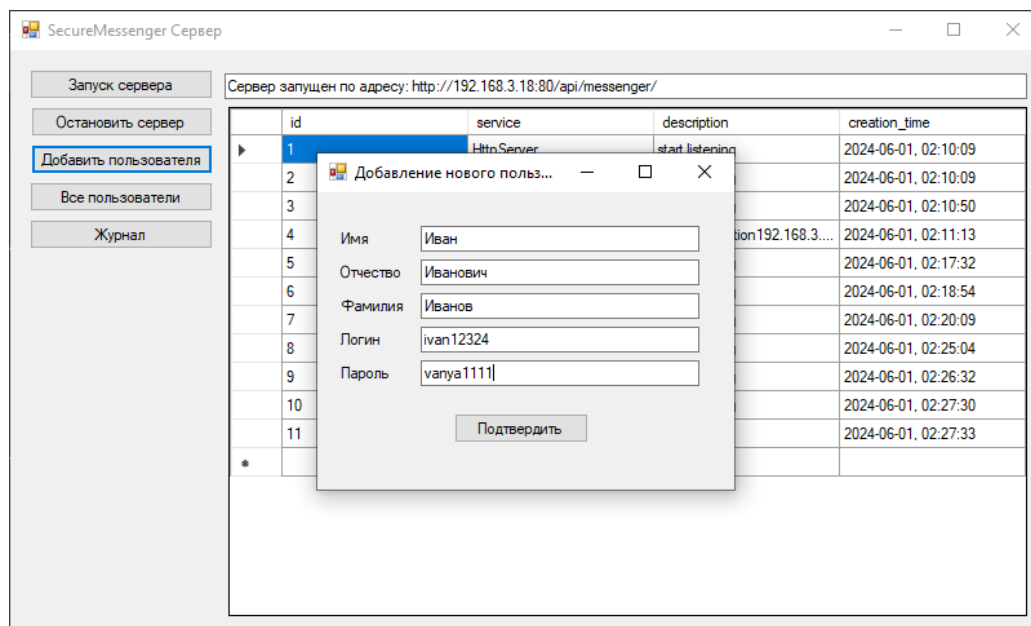


Рисунок 20 – Окно регистрации нового пользователя

5.2 Запуск сервера

Чтобы протестировать запуск сервера, нужно в главном окне приложения нажать на кнопку «Запустить сервер». После того, запуска сервера выведется окно о успешном запуске. На рисунке 21 представлено главное окно приложения с запущенным сервером.

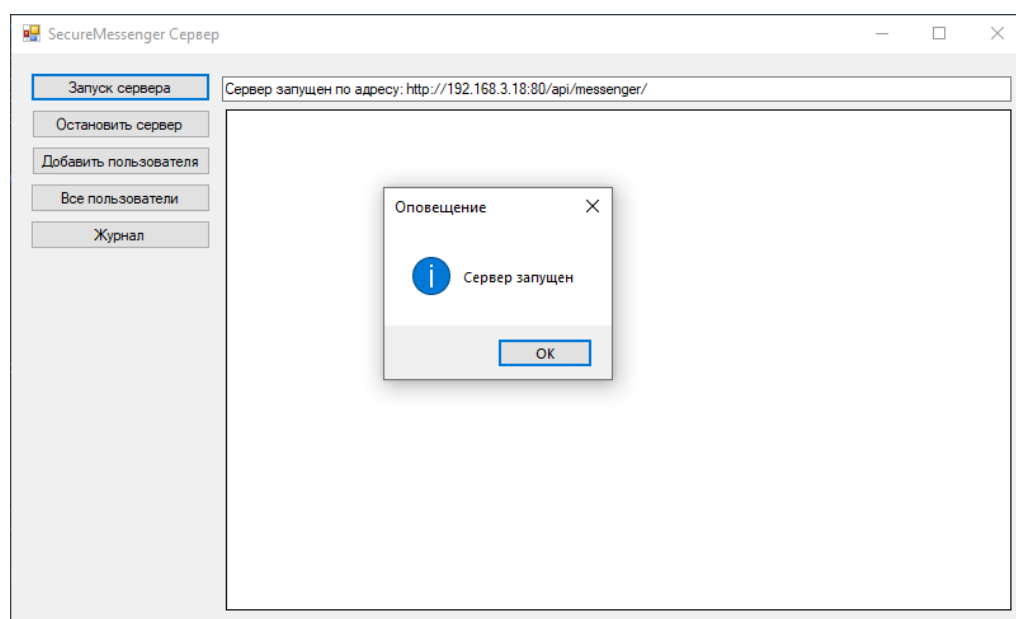
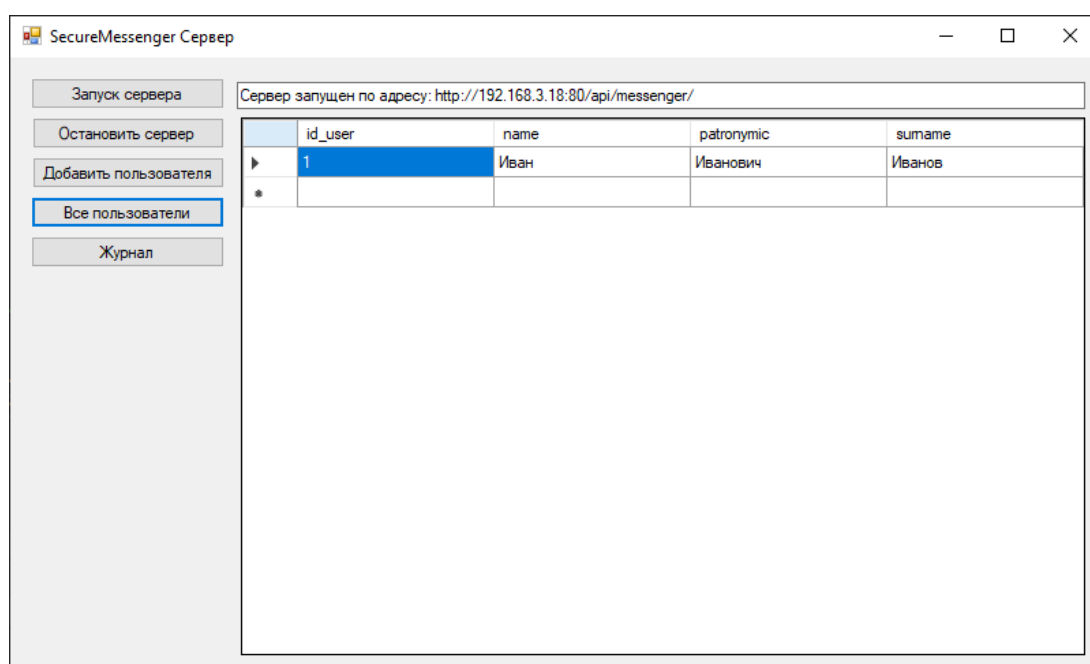


Рисунок 21 - Вывод оповещения о запуске сервера

5.3 Просмотр всех пользователей администратором.

Для тестирования такой функции сервера как просмотр всех пользователей необходимо на главном окне нажать на кнопку «Все пользователи». Список всех пользователей будет выведен в главном окне в виде таблицы. Данная таблица отображена на рисунке 22.



The screenshot shows a window titled "SecureMessenger Сервер". On the left is a sidebar with buttons: "Запуск сервера", "Остановить сервер", "Добавить пользователя", "Все пользователи" (highlighted), and "Журнал". The main area displays a table of users with the following data:

	id_user	name	patronymic	surname
▶	1	Иван	Иванович	Иванов
*				

Рисунок 22 - Таблица всех пользователей

5.4 Просмотр журнала событий системы

Для того чтобы протестировать функционал просмотра журнала событий сервера необходимо на главном окне приложения нажать на кнопку «Журнал». Как и при просмотре данных всех пользователей информация журнала будет выведена на таблицу главного окна сервера. На рисунке 23 представлен журнал сервера.

SecureMessenger Сервер				
Сервер запущен по адресу: http://192.168.3.18:80/api/messenger/				
Запуск сервера				
Остановить сервер				
Добавить пользователя				
Все пользователи				
Журнал				
	id	service	description	creation_time
▶	1	HttpServer	start listening	2024-06-01, 02:10:09
	2	HttpServer	start listening	2024-06-01, 02:10:09
	3	HttpServer	start listening	2024-06-01, 02:10:50
	4	HttpServer	new connection192.168.3...	2024-06-01, 02:11:13
	5	HttpServer	start listening	2024-06-01, 02:17:32
	6	HttpServer	start listening	2024-06-01, 02:18:54
	7	HttpServer	start listening	2024-06-01, 02:20:09
	8	HttpServer	start listening	2024-06-01, 02:25:04
	9	HttpServer	start listening	2024-06-01, 02:26:32
	10	HttpServer	start listening	2024-06-01, 02:27:30
	11	HttpServer	stopped	2024-06-01, 02:27:33
	12	HttpServer	start listening	2024-06-01, 02:37:01
	13	HttpServer	start listening	2024-06-01, 02:38:58
	14	AddUser	new user added	2024-06-01, 02:39:35
*				

Рисунок 23 - Журнал сервера

5.5 Авторизация пользователя

Чтобы протестировать авторизацию клиента необходимо запустить клиент приложения, после чего пользователь будет сразу перенаправлен на страницу авторизации приложения. Для авторизации необходимо ввести свой логин в поле «Логин» и пароль в поле «Пароль», далее нажать на кнопку «Войти». Далее происходит проверка введенных данных на некорректность, т.е. заполнены ли все поля и имеет ли пароль минимальную длину. После чего происходит проверка соединения с сервером, если проверка не успешно, то будет выведено всплывающее окно с соответствующим текстом. Если проверка успешно, то далее данные будут отправлены на сервер для проверки, в случае, когда пользователь с заданным логин и паролем существуют, пользователь будет перенаправлен на главную страницу приложения.

На рисунке 24 представлено окно авторизации клиента приложения с заполненными полями.

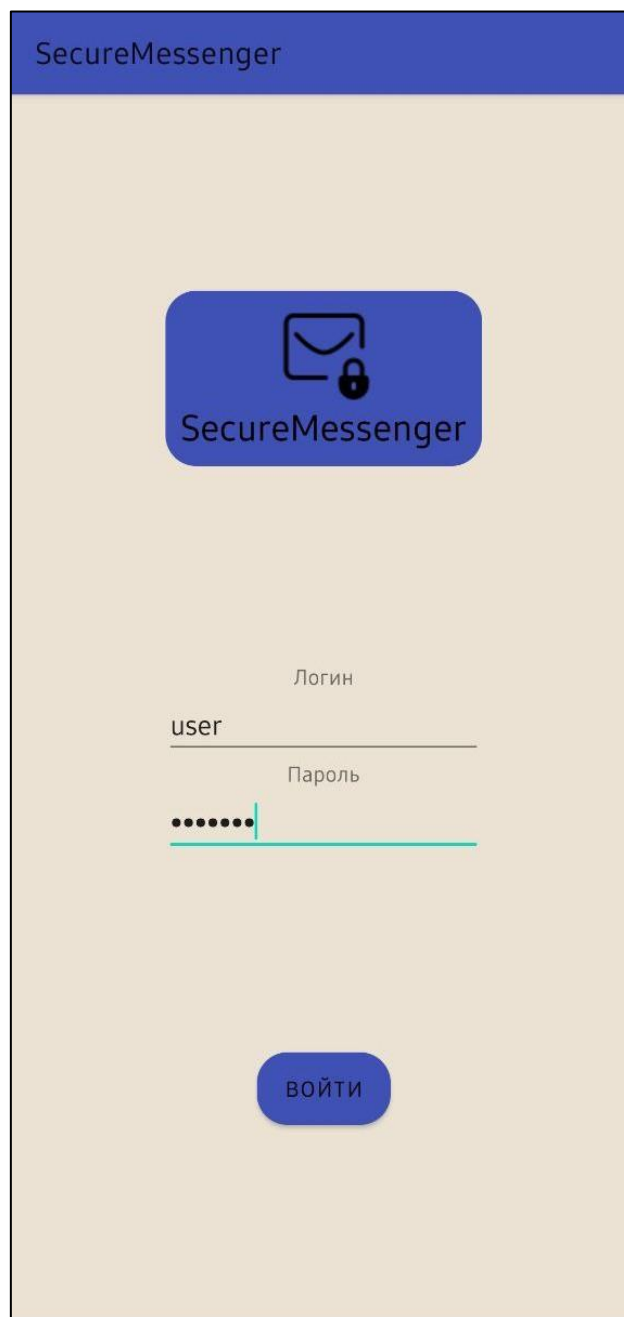


Рисунок 24 – Окно авторизации клиента приложения

После успешной авторизации пользователь попадет на главную страницу приложения. На которой отражены все уже созданные чаты с другими пользователями. А также кнопки меню и создания нового чата. Это отражено на рисунке 25.

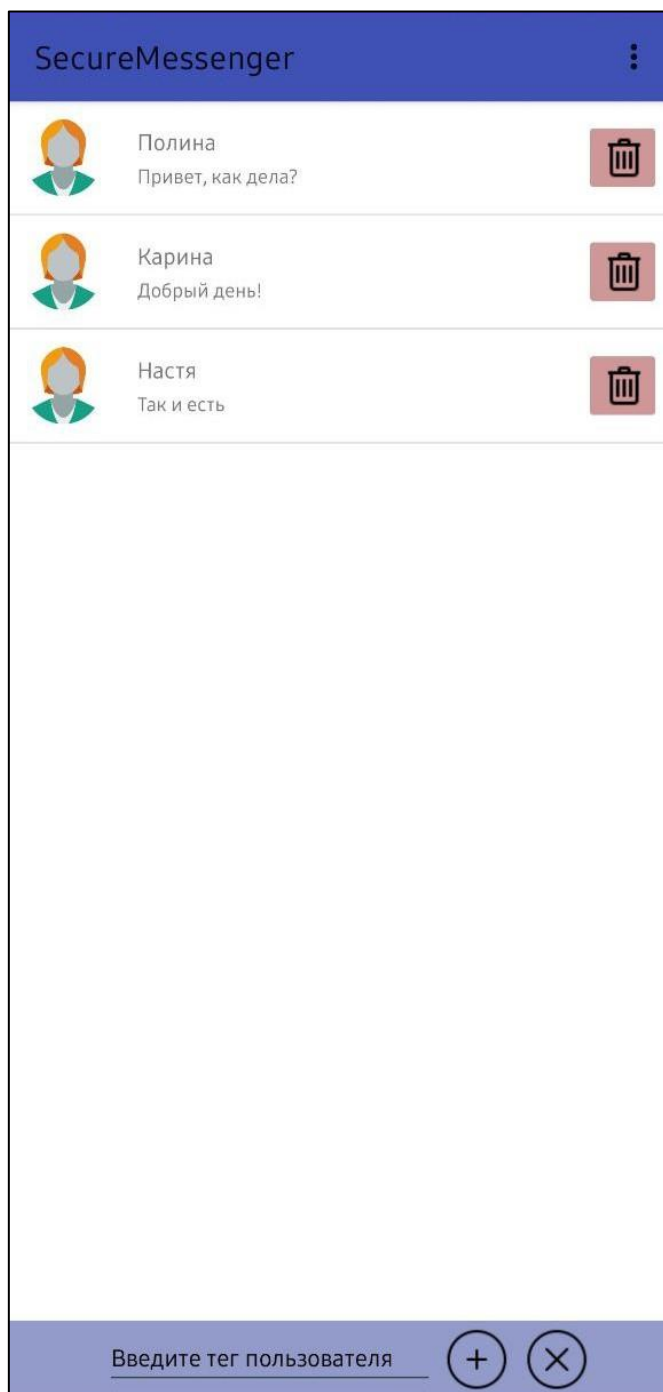


Рисунок 25 – Главная страница приложения

5.6 Просмотр чатов пользователем

Для просмотра всех чатов пользователем нужно авторизироваться в приложении после чего будет произведен переход на главный экран, где располагаются все чаты пользователя. На рисунке 26 представлено главное окно приложения со всеми чатами пользователя.

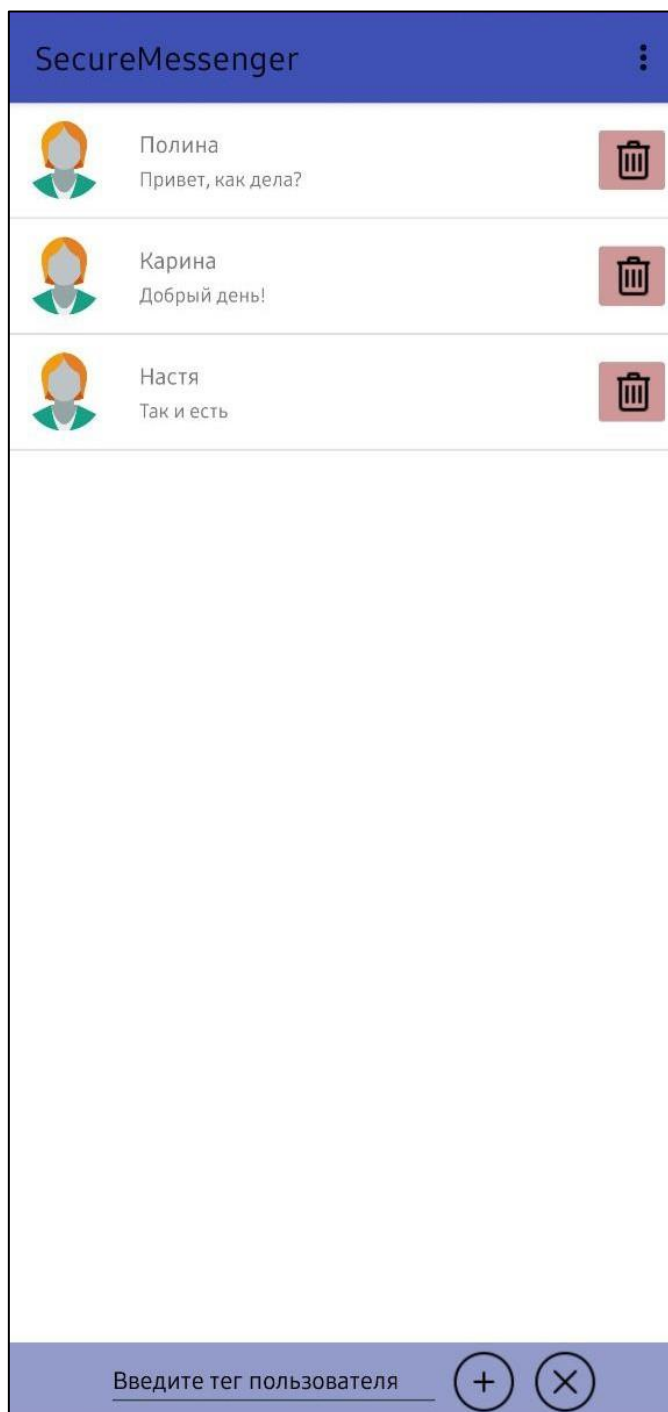


Рисунок 26 – Главное окно приложения с чатами пользователя

5.7 Просмотр чата пользователем

Для просмотра чата с другим пользователем нужно кликнуть на нужный чат после чего произойдет переход в окно чата, где будет располагаться история сообщений. На рисунке 27 представлено окно чата с историей сообщений.

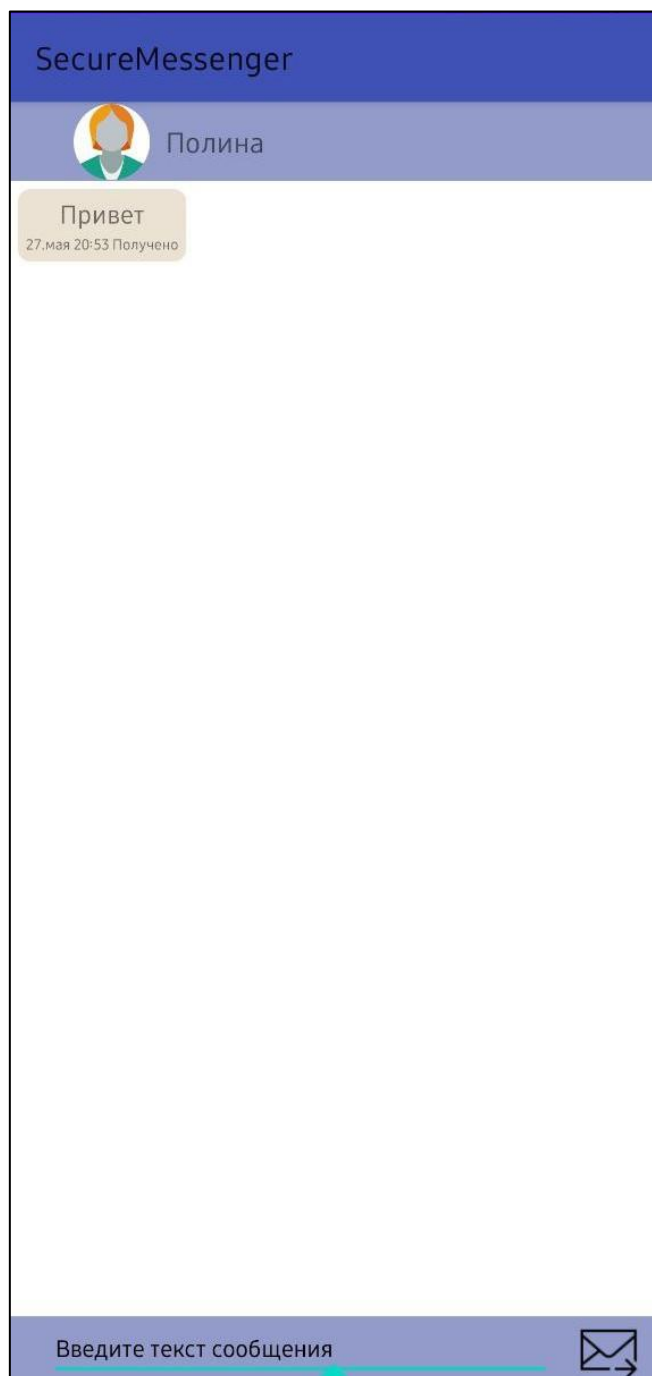


Рисунок 27 – Окно чата

5.8 Отправка сообщения пользователем

Для отправки сообщения необходимо перейти в чат с каким-либо пользователем после чего ввести текст и нажать на кнопку отправки в виде письма со стрелочкой вправо. После чего отправленное сообщение появится в истории чата.

					ПГУ 1.090301.26.001 ПЗ	Лист
						62
Изм.	Лист	№ докум.	Подпись	Дата		

На рисунке 28 представлена история сообщений с отправленным сообщением.

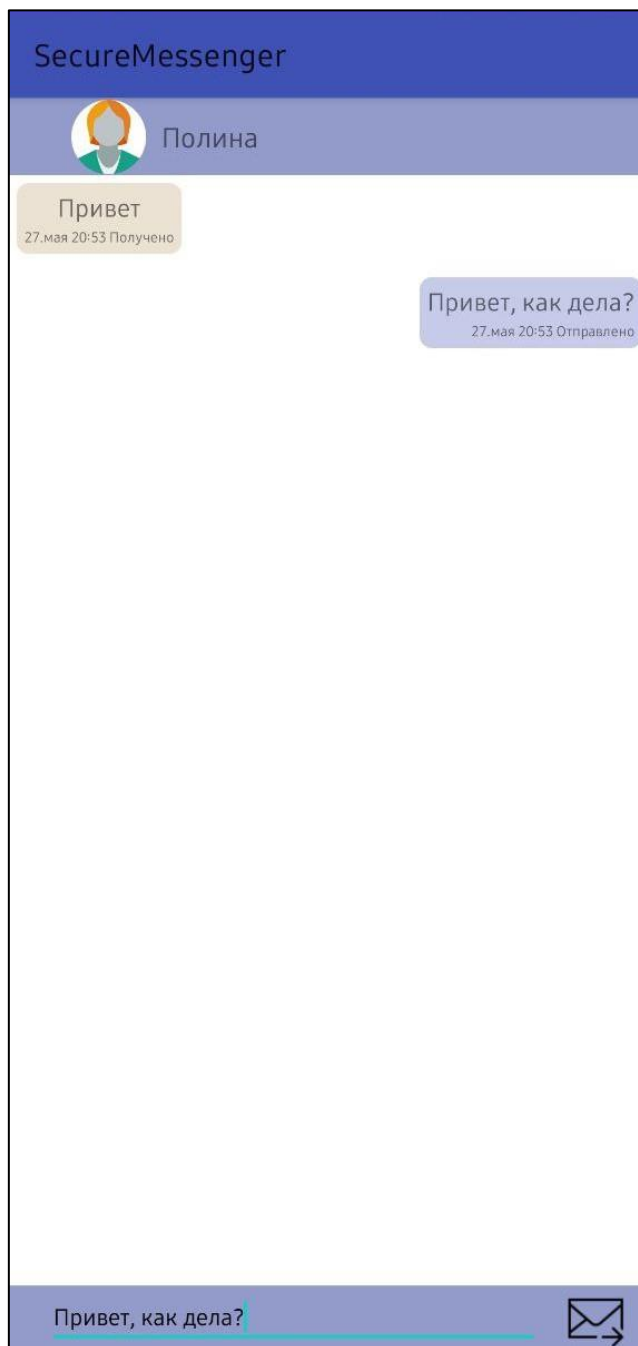


Рисунок 28 – Окно чата с отправленным сообщением

5.9 Изменение настроек приложения пользователем

Для изменения настроек приложения необходимо нажать на кнопку меню в виде трех вертикальных точек, расположенных справа в шапке приложения. Для

изменения настроек приложения необходимо изменить какую-либо опцию и нажать кнопку подтвердить, для отмены изменений нужно нажать на кнопку отмена.

На рисунке 29 представлено меню приложения.

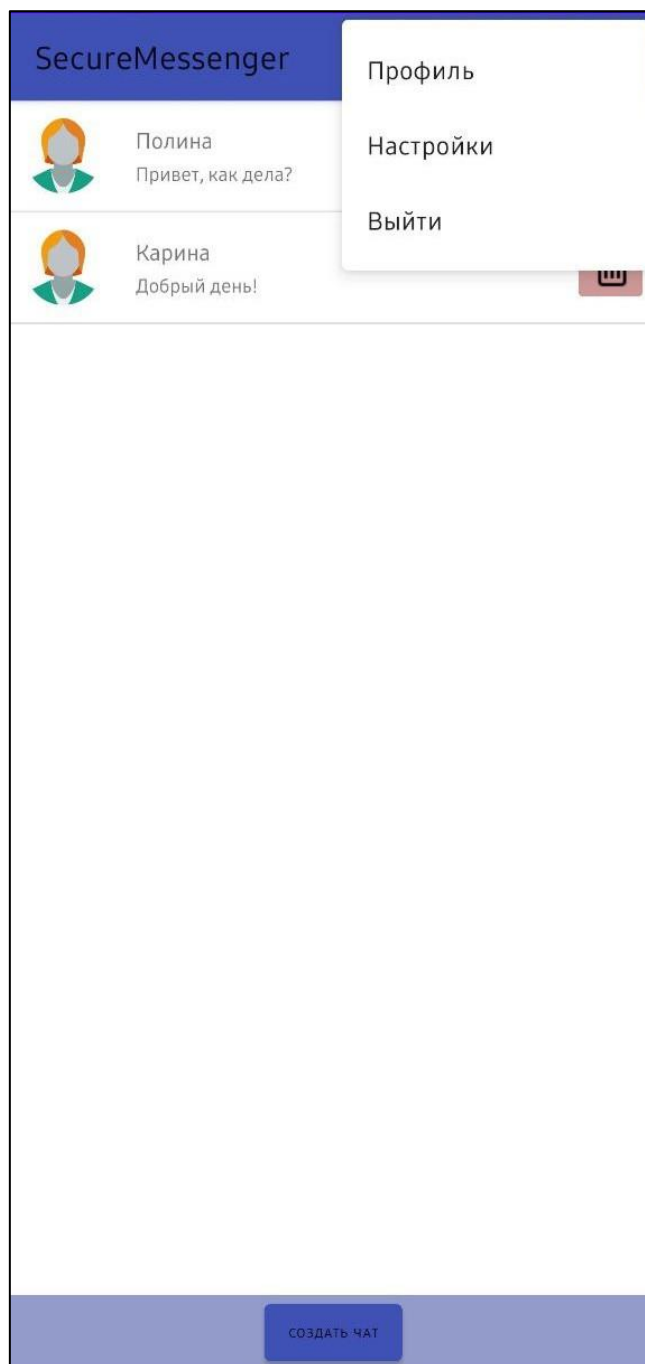


Рисунок 29 – Главное окно приложения с выведенным меню

Далее на рисунке 30 представлено открытое окно настроек приложения, с ранее описанными кнопками и списками опций.

Рисунок 30 – Окно настроек приложения

5.10 Просмотр своего профиля пользователем

Для просмотра всех чатов пользователем нужно авторизоваться в приложении после чего будет произведен переход на главный экран, где располагаются все чаты пользователя.

Для изменения настроек профиля необходимо нажать клавишу сохранить, для отмены изменений нужно нажать на клавишу отменить. На рисунке 31 представлено окно чатов.

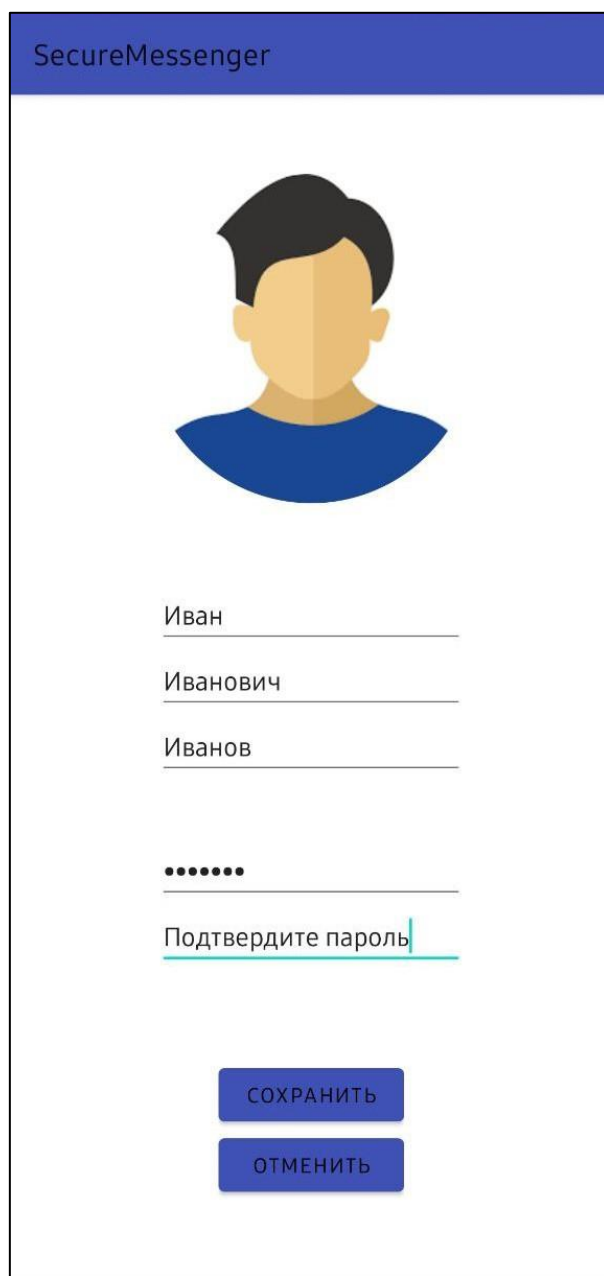


Рисунок 31 – Окно профиля

5.11 Добавление чата пользователем

Для добавления нового чата нужно нажать на кнопку «Создать чат» расположенную в нижней части главного окна.

На рисунке 32 представлена кнопка «Создать чат».

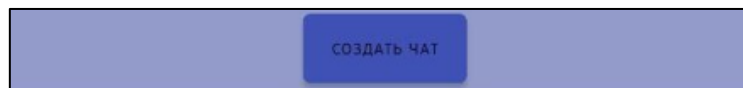


Рисунок 32 – Кнопка создания чата

После нажатия на клавишу появится поле ввода тега пользователя чата с которым нужно создать и две кнопки добавить и отмена. Кнопка отмены – кнопка с иконкой крестика возвращает нижнюю панель в начальное состояние. На рисунке 33 представлена измененная нижняя панель с полем ввода и кнопками.



Рисунок 33 – Измененная нижняя панель

При нажатии на кнопку добавления нового чата, будут произведены проверка введенных данных, введен ли тег пользователя, чат с которым необходимо, если проверка не пройдена, то будет выведено соответствующее сообщение о ошибке. Далее будет осуществлена проверка соединения с сервером, при проверке в случае отсутствия соединения также будет выведено сообщение об ошибке. Если проверка пройдена успешно, то данные будут отправлены на сервер. Сервер в свою очередь проверит переданные данные на корректность при расшифровке, а также проверит существует ли пользователь с переданным тегом. Если пользователь существует, то в ответном сообщении будет передана вся необходимая информация о нем для создания чата.

На рисунке 34 представлен пустой список чатов пользователя, до создания нового чата.



Рисунок 34 – Главное окно приложения с пустым списком чатов
пользователя

Далее на рисунке 35 представлен обновленный список, с новым чатов. Каждый новый чат создается как отдельный объект в прокручиваемом списке и содержит в себе аватар профиля собеседника, неполное имя, последнее сообщение из истории, а также кнопку удаления чата.

При нажатии на кнопку удаления очищаются все связанные с чатом записи в таблицах базы данных клиентской части приложения.

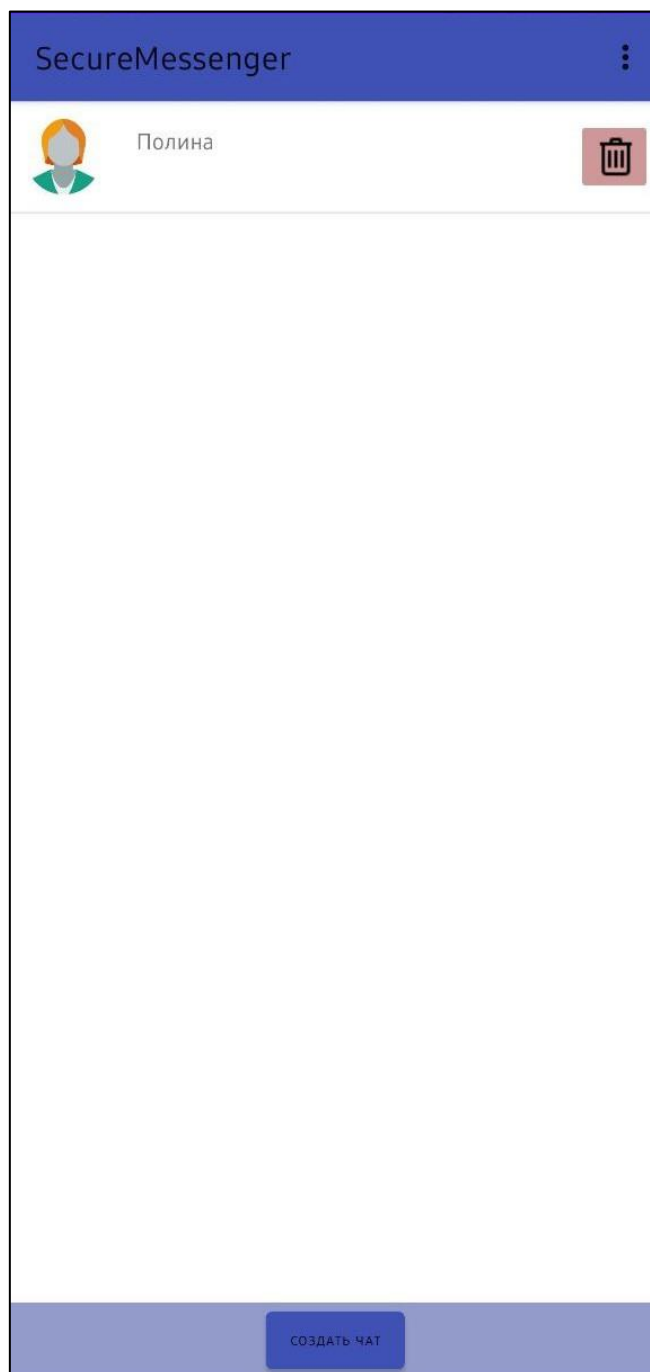


Рисунок 35 – Добавленный чат

Таким образом проведено тестирование клиентской и серверной части приложения «SecureMessenger».

6 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.

6.1 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 кистей рук, из-за чего может произойти даже онемение рук;

- нагрузка на спину. Опять же, если принять неправильно положение осанки при работе за компьютером, может появиться искривление позвоночника и даже грыжа межпозвоночного диска;

- нагрузка на органы зрения. При долгой работе с мониторами будет ухудшаться зрение, впоследствии могут появиться синдромы «красного глаза» и «сухого глаза»;

- нагрузка на психику и нервную. При долгом взаимодействии с ПК, у пользователя могут возникать проблемы с памятью, а также может появиться депрессия и даже бессонница.

6.2 Снижение вреда для здоровья при работе с ПК

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

					ПГУ 1.090301.26.001 ПЗ	Лист
						70
Изм.	Лист	№ докум.	Подпись	Дата		

- 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.

6.3 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 обмена зашифрованными сообщениями, под операционную систему Windows и Android, изучен стек технологий и библиотек .NET и Android. Результатом работы является разработанное клиент-серверное приложение «Мессенджер с шифрованием сообщением», которое называется SecureMessenger, а также отчет, в котором представлены результаты тестирования разработанного программного обеспечения, описание разработки и подробный разбор его основных функциональных особенностей.

Разработанное приложение позволяет пользователям обмениваться сообщениями. После проведения тестирования было выяснено, что разработанное приложение соответствует всем поставленным в начале работы требованиям и работает корректно, не вызывая критических ошибок в системе.

В дальнейшем приложения можно усовершенствовать путем добавления передачи изображений, создания групповых чатов и передачи голосовых сообщений. Разработанный интерфейс пользователя можно улучшать путем усовершенствования дизайна для повышения комфорта использования.

Список использованных источников

1. ТОП самых безопасных мессенджеров с шифрованием в 2024 году / [Электронный ресурс]. – URL: <https://trashexpert.ru/software/security/best-secure-and-encrypted-messaging-apps> (дата обращения 24.05.2024);
2. Что такое Signal и почему он безопаснее, чем WhatsApp и даже Telegram - Лайфхакер / [Электронный ресурс]. – URL: <https://lifehacker.ru/messendzher-signal/> (дата обращения 24.05.2024);
3. Обзор C#-Overview | Microsoft Learn / [Электронный ресурс]. – URL: <https://learn.microsoft.com/ru-ru/dotnet/csharp/> (дата обращения 30.04.2024);
4. Введение в .NET - .NET | Microsoft Learn / [Электронный ресурс]. – URL: <https://learn.microsoft.com/ru-ru/training/dotnet/> (дата обращения 30.04.2024);
5. Что такое интегрированная среда разработки Visual Studio? | Microsoft Learn / [Электронный ресурс]. – URL: <https://learn.microsoft.com/ru-ru/visualstudio/get-started/visual-studio-ide?view=vs-2022> (дата обращения 30.04.2024);
6. MySQL: обзор, плюсы и минусы – Timeweb Cloud / [Электронный ресурс]. – URL: <https://timeweb.cloud/blog/mysql-preimushchestva-i-nedostatki> (дата обращения 21.05.2024);
7. Что такое Windows Forms - Windows Forms .NET | Microsoft Learn / [Электронный ресурс]. – URL: <https://learn.microsoft.com/ru-ru/dotnet/desktop/winforms/overview/?view=netdesktop-8.0> (дата обращения 30.04.2024);
8. Языки программирования для создания приложений: топ языков для мобильной разработки / [Электронный ресурс]. – URL: <https://practicum.yandex.ru/blog/yazyki-programmirovaniya-dlya-mobilnoy-razrabotki/> (дата обращения 30.04.2024);
9. Android Studio: что это и как ей пользоваться — уроки для начинающих / Skillbox Media / [Электронный ресурс]. – URL: <https://skillbox.ru/media/code/chto-takoe-android-studio-i-kak-ey-polzovatsya/> (дата обращения 15.05.2024)
10. SQLite | Введение / [Электронный ресурс]. – URL: <https://metanit.com/sql/sqlite/> (дата обращения 30.04.2024);
11. Gradle: что это за система и как работает с Java / [Электронный ресурс]. – URL: <https://skillfactory.ru/gradle/> (дата обращения 30.04.2024);

					ПГУ 1.090301.26.001 ПЗ	Лист
						73
Изм.	Лист	№ докум.	Подпись	Дата		

12. GOST_R_3412-2015.pdf / [Электронный ресурс]. – URL:
https://tc26.ru/standard/gost/GOST_R_3412-2015.pdf (дата обращения 12.05.2024).

					<i>ПГУ 1.090301.26.001 ПЗ</i>	<i>Лист</i>
						74
<i>Изм.</i>	<i>Лист</i>	<i>№ докум.</i>	<i>Подпись</i>	<i>Дата</i>		

Приложение А

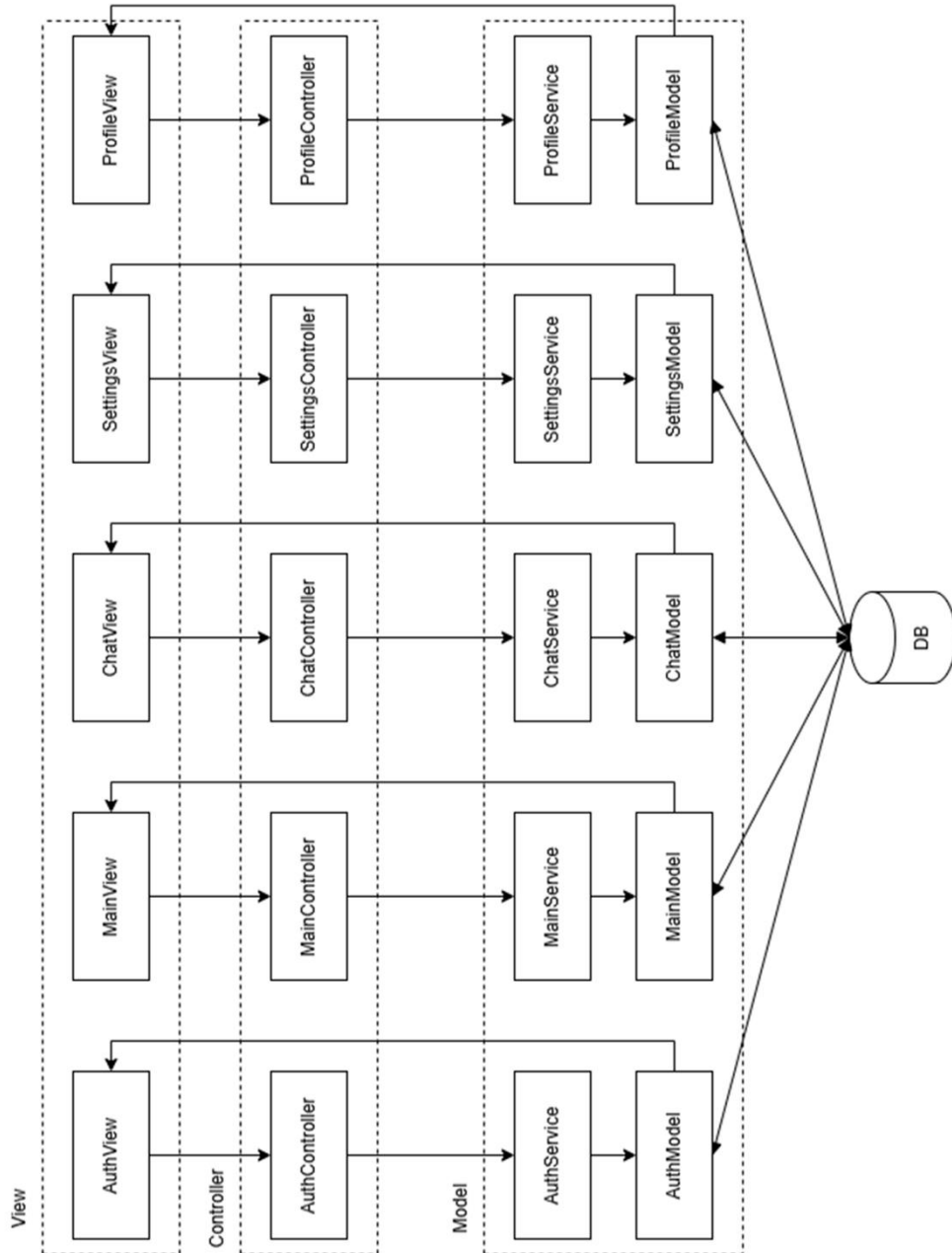
(Обязательное)

«Мессенджер с шифрованием сообщений»

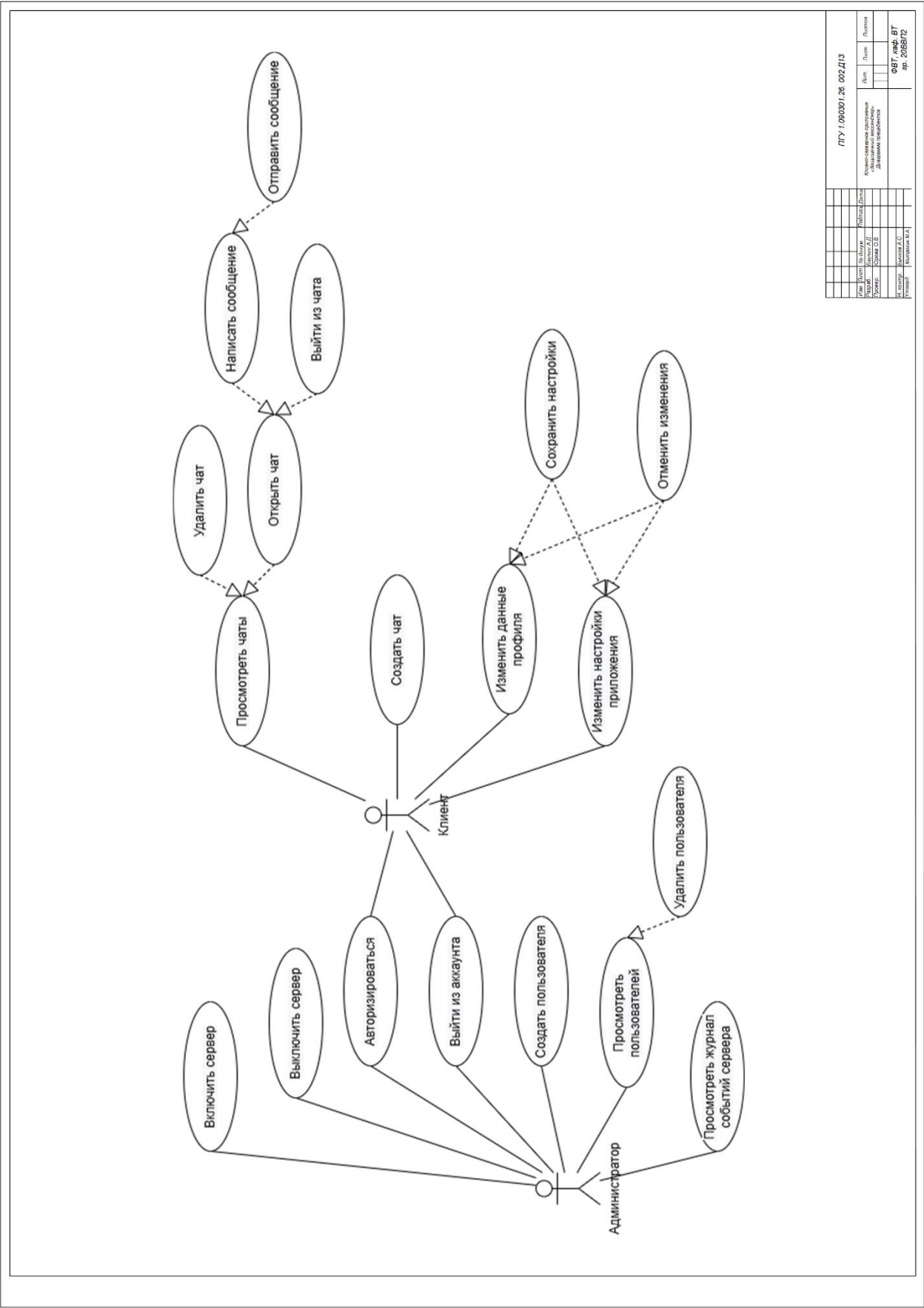
UML-диаграммы

					ПГУ 1.090301.26.001 ПЗ	Лист
						75
Изм.	Лист	№ докум.	Подпись	Дата		

Приложение А.1-Диаграмма компонентов

[illegible]

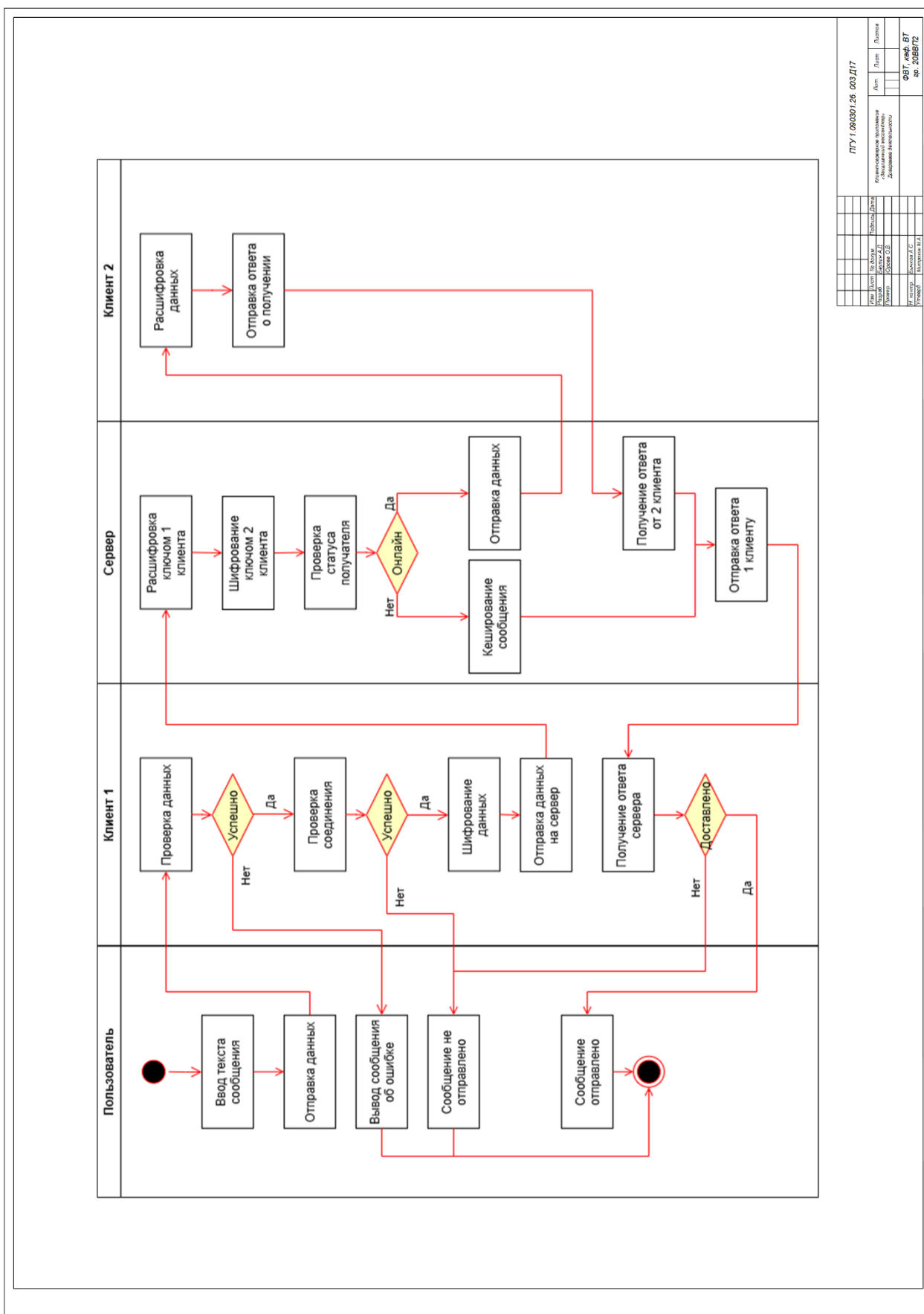
Приложение А.2-Диаграмма прецедентов



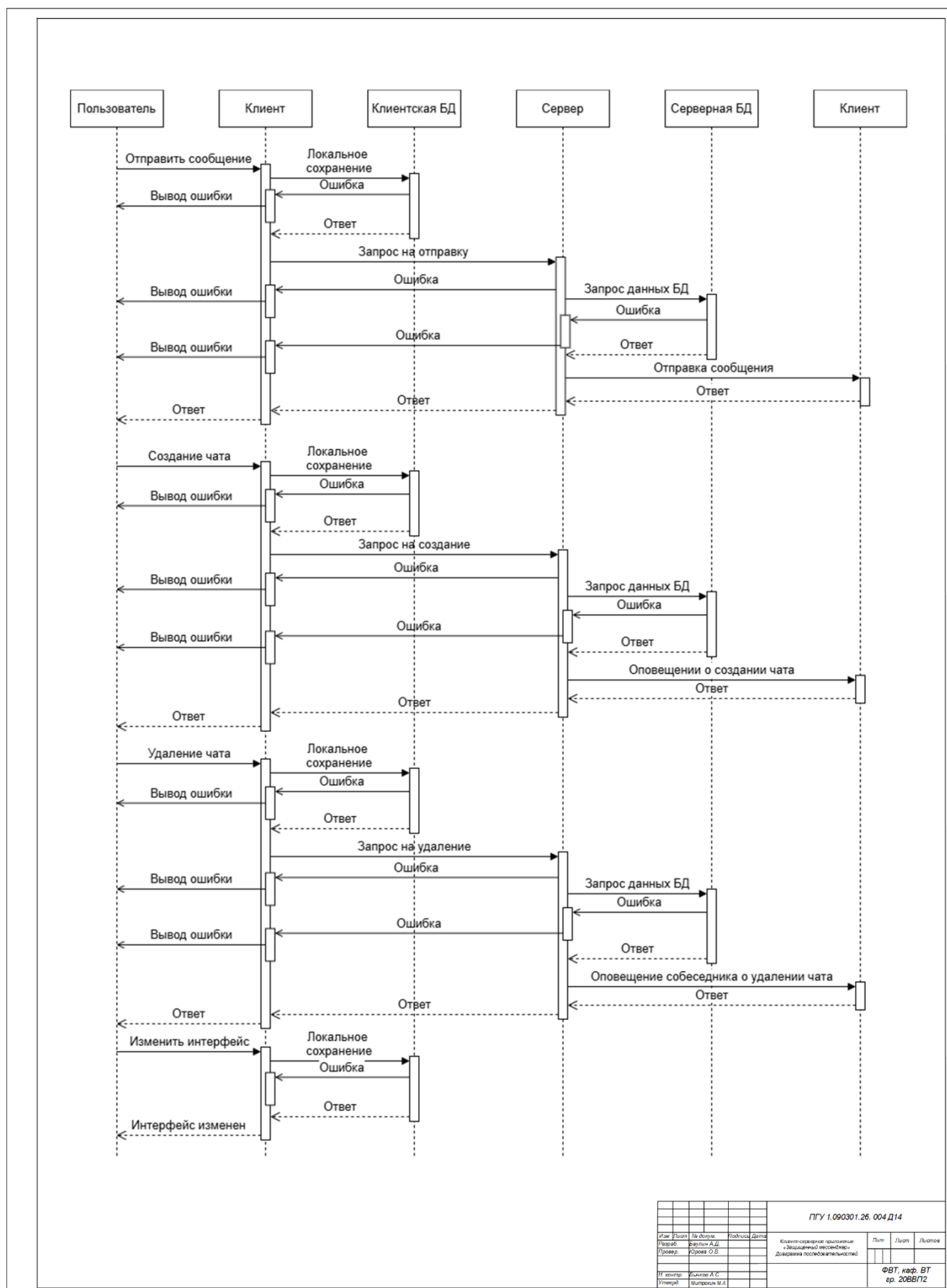
ПГУ 1.090301.26.002 Д13									
Имя	Иван	Фамилия	Иванов	Подпись	Д.И.И.	Адрес	Ленин	Район	
Пол	Мужской	Дата рождения	01.01.1990	Место рождения	Москва	Образование	Среднее	Специальность	
Т. домоф.	4-10-10	Место работы	М.П.	Подпись	М.П.	Дата	01.01.2013	Подпись	М.П.

Изм.	Лист	№ докум.	Подпись	Дата

Приложение А.3-Диаграмма деятельности



Приложение А.4-Диаграмма последовательностей



Приложение Б

(Обязательное)

«Мессенджер с шифрованием сообщений»

Листинг программы

					ПГУ 1.090301.26.001 ПЗ	Лист
						80
Изм.	Лист	№ докум.	Подпись	Дата		

Файл HttpServer.cs

```
using System;
using System.Linq;
using System.Net;
using System.Threading;
using MsgServer.Entities;
using MsgServer.Database;
using System.Collections.Concurrent;
using MsgServer.Utills;

namespace MsgServer.Net
{
    class HttpServer : IDisposable
    {
        private HttpListener httpListener;
        private Thread listenerLoop;
        private Thread[] requestProcessors;
        private BlockingCollection<HttpListenerContext> messages;
        private IPAddress hostIp;

        private JournalDbConnector journalDbConnector;

        public HttpServer(int threadCount)
        {
            requestProcessors = new Thread[threadCount];
            messages = new BlockingCollection<HttpListenerContext>();
            httpListener = new HttpListener();
            journalDbConnector = new JournalDbConnector();

            private String GetHostIp()
            {
                string host = Dns.GetHostName();
                IPAddress[] localIps = Dns.GetHostAddresses(host);

                hostIp = Array.Find<IPAddress>(localIps, ip => ip.AddressFamily
                == System.Net.Sockets.AddressFamily.InterNetwork);

                return hostIp.ToString();
            }

            public String Url { get { return httpListener.Prefixes.First(); } }

            public virtual string[] Prefixes
            {
                get { return new string[] {
                string.Format(@"http://{0}:{1}/{2}", GetHostIp(), Config.ServerPort,
                Config.ServerRoot) }; }
            }

            public void Start()
            {
                listenerLoop = new Thread(HandleRequests);
            }
        }
    }
}
```

```

        foreach (string prefix in Prefixes)
httpListener.Prefixes.Add(prefix);

        listenerLoop.Start();

        for (int i = 0; i < requestProcessors.Length; i++)
        {
            requestProcessors[i] = StartProcessor(i, messages);
        }

    public void Dispose() { Stop(); }

    public void Stop()
    {
        messages.CompleteAdding();

        foreach (Thread worker in requestProcessors) worker.Join();

        httpListener.Stop();
        listenerLoop.Join();

        journalDbConnector.InsertInto(null, new
JournalRecord("HttpServer", "stopped"));
    }

    private void HandleRequests()
    {
        httpListener.Start();
        journalDbConnector.InsertInto(null, new
JournalRecord("HttpServer", "start listening"));

        try
        {
            while (httpListener.IsListening)
            {
                HttpListenerContext context =
httpListener.GetContext();

                messages.Add(context);

                journalDbConnector.InsertInto(null, new
JournalRecord("HttpServer", "new connection" +
context.Request.RemoteEndPoint.ToString()));
            }
        }
        catch (Exception e)
        {
            Console.WriteLine(e.Message);
        }
    }

    private Thread StartProcessor(int number,
BlockingCollection<HttpListenerContext> messages)
    {
        Thread thread = new Thread(() => Processor(number, messages));
    }

```

```

        thread.Start();
        return thread;
    }

    private void Processor(int number,
BlockingCollection<HttpListenerContext> messages)
    {
        try
        {
            for ( ; ; )
            {
                HttpListenerContext context = messages.Take();
                if (context == null)
                    continue;

                HttpHandler httpHandler = new HttpHandler(context);
                httpHandler.Run();
            }
        }
        catch { }
    }
}

```

Файл MainForm.cs

```

using System;
using System.Windows.Forms;
using MsgServer.Net;
using MsgServer.Entities;
using MsgServer.Database;
using MsgServer.Utills;

namespace MsgServer.Forms
{
    public partial class MainForm : Form
    {
        enum SelectedTable
        {
            none = 0,
            users,
            journal
        }

        private HttpServer httpServer;
        private UserDbConnector userDbConnector;
        private JournalDbConnector journalDbConnector;
        private SelectedTable selectedTable = SelectedTable.none;

        public MainForm()
        {
            InitializeComponent();
            httpServer = new HttpServer(Config.ThreadsCount);
            userDbConnector = new UserDbConnector();
            journalDbConnector = new JournalDbConnector();
        }
    }
}

```

```

private void ServerControl_Click(object sender, EventArgs e)
{
    try
    {
        httpServer.Start();
        serverInfoShort.Text = $"Сервер запущен по адресу:
{httpServer.Url}\n";

        MessageBox.Show(this, "Сервер запущен", "Оповещение",
        MessageBoxButtons.OK, MessageBoxIcon.Information);
    }
    catch (Exception ex)
    {
    }
}

private void UsersList_Click(object sender, EventArgs e)
{
    selectedTable = SelectedTable.users;
    userDbConnector.Select(dataTableGridView);
}

private void ServerJournal_Click(object sender, EventArgs e)
{
    selectedTable = SelectedTable.journal;
    journalDbConnector.Select(dataTableGridView);
}

private void ServerStop_Click(object sender, EventArgs e)
{
    httpServer.Stop();

    MessageBox.Show("Сервер остановлен", "Оповещение",
    MessageBoxButtons.OK, MessageBoxIcon.Information);
}

private void AddUser_Click(object sender, EventArgs e)
{
    var addUserForm = new AddUserForm();
    addUserForm.StartPosition = FormStartPosition.CenterParent;
    addUserForm.ShowDialog(this);
}

private void dataTableGridView_CellClick(object sender,
DataGridViewCellEventArgs e)
{
    if (dataTableGridView.RowCount == 1)
        return;

    switch (selectedTable)
    {
        case SelectedTable.journal:
            {
                break;
            }
    }
}

```

```

        case SelectedTable.users:
        {
            int userId =
(int)dataTableGridView.Rows[e.RowIndex].Cells[0].Value;
            var userInfoForm = new UserInfoForm(userId);
            userInfoForm.StartPosition =
FormStartPosition.CenterParent;
            userInfoForm.ShowDialog(this);

            break;
        }
        default:
            return;
    }
}
}
}

```

Файл UserDbConnctor.cs

```

using MySql.Data.MySqlClient;
using System.Data;
using System.Windows.Forms;
using MsgServer.Entities;

namespace MsgServer.Database
{
    public class UserDbConnector : DbConnector
    {
        public UserDbConnector()
            : base("1111", "root", "msgserver", "localhost")
        {
        }

        public bool InsertInto(DataGridView dataGrid, User user)
        {
            using (MySqlConnection con = new
MySqlConnection(base.connectionString_))
            {
                using (MySqlCommand cmd = new MySqlCommand())
                {
                    con.Open();

                    cmd.Connection = con;
                    cmd.CommandText = "insert into users (name, patronymic,
surname, enckey, ipaddress, login, password, status)" +
                        " values(@name, @patronymic,
@surname, @enckey, @ipaddress, @login, @password, @status)";

                    cmd.Parameters.AddWithValue("@name", user.Name);
                    cmd.Parameters.AddWithValue("@patronymic",
user.Patronymic);
                    cmd.Parameters.AddWithValue("@surname", user.Surname);
                    cmd.Parameters.AddWithValue("@enckey", user.Enckey);
                    cmd.Parameters.AddWithValue("@ipaddress",
user.Ipaddress);
                    cmd.Parameters.AddWithValue("@login", user.Login);

```

					ПГУ 1.090301.26.001 ПЗ	Лист
						85
Изм.	Лист	№ докум.	Подпись	Дата		

```

        user.Password);
        cmd.Parameters.AddWithValue("@password",
        cmd.Parameters.AddWithValue("@status", user.Status);
        cmd.Prepare();

        cmd.ExecuteReader();
    }
}

return true;
}

public bool Select(DataGridView dataGrid)
{
    using (MySqlConnection con = new
        MySqlConnection(base.connectionString_))
    {
        using (MySqlDataAdapter a = new MySqlDataAdapter("SELECT
            id_user, name, patronymic, surname FROM users", con))
        {
            DataTable t = new DataTable();
            a.Fill(t);

            dataGrid.AutoGenerateColumns = true;
            dataGrid.DataSource = t;
        }
    }

    return true;
}

public bool Select(DataGridView dataGrid, int userId, out User
user)
{
    using (MySqlConnection con = new
        MySqlConnection(base.connectionString_))
    {
        using (MySqlCommand cmd = new MySqlCommand())
        {
            con.Open();

            cmd.Connection = con;
            cmd.CommandText = "SELECT * FROM users WHERE
            id_user=@id_user";

            cmd.Parameters.AddWithValue("@id_user", userId);
            cmd.Prepare();

            using (MySqlDataReader reader = cmd.ExecuteReader())
            {
                reader.Read();

                user = new User(reader.GetString("name"),
                reader.GetString("patronymic"),
                reader.GetString("surname"),
                reader.GetString("login"),

```

```

        reader.GetString("password"));

        user.Id = reader.GetInt32("id_user");
    }
}

return true;
}

public bool Select(DataGridView dataGrid, string login, out User
user)
{
    using (MySQLConnection con = new
MySQLConnection(base.connectionString_))
    {
        using (MySQLCommand cmd = new MySQLCommand())
        {
            con.Open();

            cmd.Connection = con;
            cmd.CommandText = "SELECT * FROM users WHERE
login=@login";

            cmd.Parameters.AddWithValue("@login", login);
            cmd.Prepare();

            using (MySQLDataReader reader = cmd.ExecuteReader())
            {
                reader.Read();

                user = new User(reader.GetString("name"),
reader.GetString("patronymic"),
reader.GetString("login"),
reader.GetString("surname"),
reader.GetString("password"));

                user.Id = reader.GetInt32("id_user");
            }
        }
    }

    return true;
}

public bool Delete(DataGridView dataGrid, User user)
{
    using (MySQLConnection con = new
MySQLConnection(base.connectionString_))
    {
        using (MySQLCommand cmd = new MySQLCommand())
        {
            con.Open();

            cmd.Connection = con;

```

```

        cmd.CommandText = "DELETE FROM users WHERE id_user =
@id_user";

        cmd.Parameters.AddWithValue("@id_user", user.Id);
        cmd.Prepare();

        cmd.ExecuteReader();
    }

    return true;
}

public bool Update(DataGridView dataGrid, User user)
{
    using (MySQLConnection con = new
MySQLConnection(base.connectionString_))
    {
        using (MySQLCommand cmd = new MySQLCommand())
        {
            con.Open();

            cmd.Connection = con;
            cmd.CommandText = "UPDATE users SET name=@name,
patronymic=@patronymic, " +
                                "surname=@surname, login=@login,
password=@password " +
                                "WHERE id_user= @id_user;";

            cmd.Parameters.AddWithValue("@id_user", user.Id);
            cmd.Parameters.AddWithValue("@name", user.Name);
            cmd.Parameters.AddWithValue("@patronymic",
user.Patronymic);

            cmd.Parameters.AddWithValue("@surname", user.Surname);
            cmd.Parameters.AddWithValue("@login", user.Login);
            cmd.Parameters.AddWithValue("@password",
user.Password);

            cmd.Prepare();

            cmd.ExecuteReader();
        }
    }

    return true;
}
}
}

```

Файл User.cs

```

using MsgServer.Utills;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

```

```

namespace MsgServer.Entities
{
    public class User
    {
        enum NetStatus
        {
            none=0,
            online,
            offline
        }

        private int id_;
        private String name_;
        private String patronymic_;
        private String surname_;
        private String ipaddress_;
        private String enckey_;
        private String login_;
        private String password_;
        private NetStatus netStatus_;
        public User(String name, String patronymic, String surname, int id,
            String ipaddress, String enckey, String login, String
password)
        {
            name_ = name;
            patronymic_ = patronymic;
            surname_ = surname;
            id_ = id;
            ipaddress_ = ipaddress;
            enckey_ = enckey;
            login_ = login;
            password_ = password;
        }

        public User(String name, String patronymic, String surname,
            String login, String password)
        {
            name_ = name;
            patronymic_ = patronymic;
            surname_ = surname;
            id_ = 0;
            ipaddress_ = "192.168.1.3";

            enckey_ = Crypto.GetRandomKey();

            login_ = login;
            password_ = password;
            netStatus_ = NetStatus.offline;
        }

        public int Id{ get { return id_; } set { id_ = value; } }
        public String Name{ get { return name_; } set { name_ = value; } }
        public String Patronymic { get { return patronymic_; } set {
patronymic_ = value; } }
        public String Surname { get { return surname_; } set { surname_ =
value; } }
    }
}

```

```

        public String Ippaddress { get { return ippaddress_; } set {
ippaddress_ = value; } }
        public String Enckey { get { return enckey_; } set { enckey_ =
value; } }
        public String Login { get { return login_; } set { login_ = value;
} }
        public String Password { get { return password_; } set { password_
= value; } }

        public String Status
        {
            get
            {
                switch(netStatus_)
                {
                    case NetStatus.online: return "online";
                    case NetStatus.offline: return "offline";
                    default: return "offline";
                }
            }

            set
            {
                switch (value)
                {
                    case "online":
                    {
                        netStatus_ = NetStatus.offline;
                        break;
                    }
                    case "offline":
                    {
                        netStatus_ = NetStatus.offline;
                        break;
                    }
                    default:
                    {
                        netStatus_ = NetStatus.offline;
                        break;
                    }
                }
            }
        }
    }
}

```

Файл MainActivity.java

```

package com.example.securemessenger.activities;

import android.content.Intent;
import android.os.Bundle;
import android.util.Log;
import android.view.Menu;
import android.view.MenuItem;
import android.view.View;
import android.widget.Button;

```

					ПГУ 1.090301.26.001 ПЗ	Лист
						90
Изм.	Лист	№ докум.	Подпись	Дата		

```

import android.widget.EditText;
import android.widget. ImageButton;
import android.widget.ListView;
import android.widget.Toast;

import androidx.annotation.Nullable;
import androidx.appcompat.app.AppCompatActivity;

import com.example.securemessenger.R;
import com.example.securemessenger.list_iltems.ChatAdapter;
import com.example.securemessenger.objects.Chat;

import java.util.ArrayList;

public class MainActivity extends AppCompatActivity
{
    String main_activity = "MainActivity";
    Button addChatBtn = null;
    ImageButton confirmAddChatBtn = null;
    ImageButton cancelAddChatBtn = null;
    EditText newUserLoginEdit = null;
    ListView chatList = null;
    ChatAdapter adapter = null;

    @Override
    protected void onCreate(@Nullable Bundle savedInstanceState)
    {
        super.onCreate(savedInstanceState);

        setContentView(R.layout.main_layout);

        ArrayList<Chat> chats = new ArrayList<Chat>();
        chatList = findViewById(R.id.chatsList);
        adapter = new ChatAdapter(this, R.layout.chat_item_layout, chats);
        chatList.setAdapter(adapter);

        addChatBtn = findViewById(R.id.add_chat_btn);
        confirmAddChatBtn = findViewById(R.id.conform_adding_chat_btn);
        cancelAddChatBtn = findViewById(R.id.cancel_adding_chat_btn);
        newUserLoginEdit = findViewById(R.id.new_chat_login_edit);

        addChatBtn.setOnClickListener((View v)->
        {
            Toast.makeText(MainActivity.this,
                           R.string.sample_text,
                           Toast.LENGTH_SHORT).show();

            confirmAddChatBtn.setClickable(true);
            cancelAddChatBtn.setClickable(true);
            newUserLoginEdit.setClickable(true);
            v.setClickable(false);

            confirmAddChatBtn.setVisibility(View.VISIBLE);
            cancelAddChatBtn.setVisibility(View.VISIBLE);
            newUserLoginEdit.setVisibility(View.VISIBLE);
            v.setVisibility(View.INVISIBLE);
        }
    }
}

```

```

    });

    confirmAddChatBtn.setOnClickListener((View v) ->
    {
        String newUserLogin =
String.valueOf(newUserLoginEdit.getText());

        adapter.add(new Chat(0, newUserLogin, newUserLogin,
newUserLogin));
        adapter.notifyDataSetChanged();
    });

    cancelAddChatBtn.setOnClickListener((View v) ->
    {
        Toast.makeText(MainActivity.this,
            R.string.sample_text,
            Toast.LENGTH_SHORT).show();

        confirmAddChatBtn.setClickable(false);
        addChatBtn.setClickable(true);
        newUserLoginEdit.setClickable(false);
        v.setClickable(false);

        confirmAddChatBtn.setVisibility(View.INVISIBLE);
        addChatBtn.setVisibility(View.VISIBLE);
        newUserLoginEdit.setVisibility(View.INVISIBLE);
        v.setVisibility(View.INVISIBLE);
    });

}

@Override
public boolean onCreateOptionsMenu(Menu menu) {

    getMenuInflater().inflate(R.menu.main_menu, menu);

    Log.i(main_activity, "menu is opened");

    return true;
}

@Override
public boolean onOptionsItemSelected(MenuItem item) {
    int id = item.getItemId();

    if (id ==R.id.action_profile ){
        Intent myActivity = new Intent(this, ProfileActivity.class);

this.startActivity(myActivity);
    }
    else if (id == R.id.action_settings ){
        Intent myActivity = new Intent(this, SettingsActivity.class);

        this.startActivity(myActivity);
    }
}

```

```

        else {
            Intent myActivity = new Intent(this,
AuthorizationActivity.class);

            this.startActivity(myActivity);
        }

        return super.onOptionsItemSelected(item);
    }
}

```

Файл MessageAdapter.java

```

package com.example.securemessenger.list_iltems;

import android.content.Context;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.ArrayAdapter;
import android.widget.ImageButton;
import android.widget.TextView;

import com.example.securemessenger.R;
import com.example.securemessenger.objects.Chat;
import com.example.securemessenger.objects.Message;

import java.util.ArrayList;

public class MessageAdapter extends ArrayAdapter<Message> {
    private final LayoutInflater inflater;
    private final int layout;
    ArrayList<Message> messages = null;

    public MessageAdapter(Context context, int resource, ArrayList<Message>
messages) {
        super(context, resource, messages);
        this.inflater = LayoutInflater.from(context);
        this.layout = resource;
        this.messages = messages;
    }

    public View getView(int position, View convertView, ViewGroup parent) {

        final MessageAdapter.ViewHolder viewHolder;

        if(convertView==null){
            convertView = inflater.inflate(this.layout, parent, false);
            viewHolder = new MessageAdapter.ViewHolder(convertView);
            convertView.setTag(viewHolder);
        }
        else{
            viewHolder = (MessageAdapter.ViewHolder) convertView.getTag();
        }

        Message message = messages.get(position);
    }
}

```

```

        switch(message.getMsgType())
        {
            case received: {
                convertView =
inflater.inflate(R.layout.received_item_layout, null);
                TextView msgText =
convertView.findViewById(R.id.received_message_text);
                TextView msgDate =
convertView.findViewById(R.id.received_date);
                TextView msgStatus =
convertView.findViewById(R.id.received_status);

                msgText.setText(message.getText());
                msgDate.setText(message.getDate());
                msgStatus.setText(message.getStatus());

                break;
            }
            case sended: {
                convertView = inflater.inflate(R.layout.sended_item_layout,
null);

                TextView msgText =
convertView.findViewById(R.id.sended_message_text);
                TextView msgDate =
convertView.findViewById(R.id.sended_date);
                TextView msgStatus =
convertView.findViewById(R.id.sended_status);

                msgText.setText(message.getText());
                msgDate.setText(message.getDate());
                msgStatus.setText(message.getStatus());
                break;
            }
            default:
                throw new IllegalStateException("Unexpected value: " +
message.getMsgType());
        }

        return convertView;
    }

    private static class ViewHolder {
        /*final ImageButton removeChatBtn;
        final TextView nameView, lastMsgView;*/
        ViewHolder(View view){
            /*removeChatBtn =
view.findViewById(R.id.remove_chat_img_button);
            nameView = view.findViewById(R.id.name_view);
            lastMsgView = view.findViewById(R.id.last_message_view);*/
        }
    }
}

```

Приложение В

(Обязательное)

«Мессенджер с шифрованием сообщений»

Презентация

					ПГУ 1.090301.26.001 ПЗ	Лист
						95
Изм.	Лист	№ докум.	Подпись	Дата		

«МЕССЕНДЖЕР С ШИФРОВАНИЕМ СООБЩЕНИЙ»

Автор работы:

студент группы 20ВВП2 Баулин Александр Дмитриевич

Научный руководитель:

к.т.н., доцент, Юрова Ольга Викторовна

Цель и объекты работы

Цель работы – разработка приложения для обмена зашифрованными сообщениями.

Объектами разработки являются программы клиента и сервера, которые выполняют следующие функции:

- Отображение пользовательских данных через интерфейс клиентской программы и отображение данных администратора через интерфейс серверной программы;
- Обработка пользовательских данных и передача их от клиента к клиенту через сервер;
- Обмен сообщениями, создание чатов и изменение настроек клиентской части приложения.

Актуальность темы

В современном мире в связи с распространением цифровых технологий все больше людей используют мессенджеры для общения как в личных, так и в рабочих целях. При этом пользователям важно, чтобы передаваемая информация была защищена от несанкционированного доступа и прочтения третьими лицами.

Исходя из этого можно сделать вывод, что актуальность темы заключается в создании приложения способного обеспечить будущих пользователей надежной защитой их конфиденциальной информации и гарантировать безопасность общения.

Постановка задачи

Требуется разработать клиент-серверное приложение, позволяющее осуществлять обмен зашифрованными сообщениями между клиентами.

Для достижения поставленной цели, необходимо выполнить следующие основные задачи:

1. Провести анализ предметной области;
2. Разработать архитектуру приложения;
3. Определить используемые в разработке технологии;
4. Реализовать приложение;
5. Протестировать работу приложения.

Варианты использования приложения

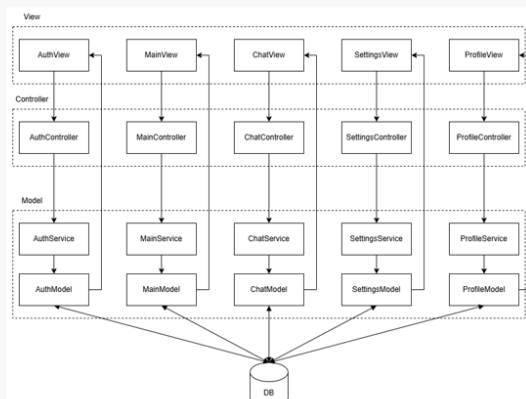
С подключением к сети



Без подключения к сети



Архитектура приложения



Шаблоном проектирования был выбран паттерн Model-View-Controller (MVC).

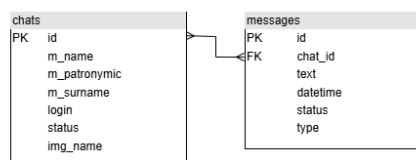
Модель - представляет собой данные и бизнес-логику приложения.

Представление - отвечает за отображение данных пользователю.

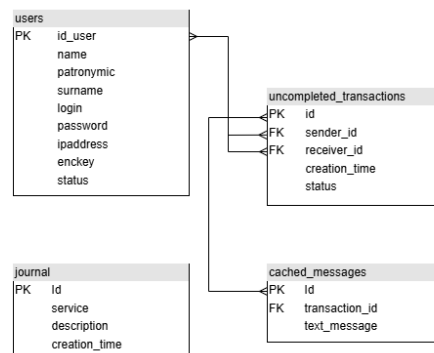
Контроллер - отвечает за обработку пользовательских запросов и управление взаимодействием между моделью и представлением.

Структуры баз данных

Клиент



Сервер



Стек используемых технологий

Клиент:



Сервер:



Заключение

В ходе выполнения данной выпускной квалификационной работы были разработаны программы сервера и клиента, которые позволяют обмениваться шифрованными сообщениями.

Клиентская часть разработанного приложения позволяет пользователям обмениваться шифрованными сообщениями, создавать чаты и менять настройки пользовательского интерфейса.

Серверная часть приложения позволяет создавать новых пользователей и просматривать журнал событий приложения.

Спасибо за внимание!

					ПГУ 1.090301.26.001 ПЗ	Лист
						100
Изм.	Лист	№ докум.	Подпись	Дата		