**МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ**

**РОССИЙСКОЙ ФЕДЕРАЦИИ**

федеральное государственное автономное образовательное учреждение

высшего образования

«Санкт-Петербургский политехнический университет Петра Великого»

(ФГАОУ ВО «СПБПУ»)

**Институт среднего профессионального образования**

|  |  |
| --- | --- |
|  | **Допустить к защите**  Заместитель директора по  учебно-методической работе  \_\_\_\_\_\_\_\_\_\_\_\_ \_\_ Е. Г. Конакина\_\_  (подпись) (И.О.Ф.)  «\_\_\_\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 202\_\_ г. |
|  |

**ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА**

Тема Разработка программного обеспечения для создания виртуального пространства с поддержкой чата

специальность 09.02.07 группа 42919/1

Студент (ка) Сизых Д. Н.

(подпись) (ФИО)

Руководитель Панченко М. С.

(подпись) (ФИО)

Санкт-Петербург

2022

**СОДЕРЖАНИЕ**

[Введение 4](#_Toc104667785)

[1 Общая часть 6](#_Toc104667786)

[1.1 Анализ предметной области 6](#_Toc104667787)

[1.2 Анализ рынка существующих решений 6](#_Toc104667788)

[1.3 Обоснование и выбор методики, технологии и инструментальных средства разработки 10](#_Toc104667789)

[1.4 Технология WebSocket 12](#_Toc104667790)

[2 Специальная часть 13](#_Toc104667791)

[2.1 Постановка задачи 13](#_Toc104667792)

[2.2 Диаграмма вариантов использования 16](#_Toc104667793)

[2.3 Общее описание структуры программного обеспечения 17](#_Toc104667794)

[2.4 Архитектура серверного приложения 18](#_Toc104667795)

[2.5 Архитектура клиентского приложения 22](#_Toc104667796)

[2.6 Реализация серверного приложения 28](#_Toc104667797)

[2.7 Реализация клиентского приложения 33](#_Toc104667798)

[2.8 Тестирование полученного программного обеспечения 51](#_Toc104667799)

[2.9 Оценка качества разработанного программного обеспечения 54](#_Toc104667800)

[2.10 Текст программы 55](#_Toc104667801)

[2.11 Руководство оператора 55](#_Toc104667802)

[3 Экономическая часть 56](#_Toc104667803)

[3.1 Область применения программного продукта и его преимущества перед аналогичным программным продуктом 56](#_Toc104667804)

[3.2 Трудоемкость разработки программного продукта, квалификация исполнителя и его оклад 56](#_Toc104667805)

[3.3 Расчет затрат на разработку 58](#_Toc104667806)

[3.4 Расчет цены и прибыли 63](#_Toc104667807)

[4 Техника безопасности и охрана труда 66](#_Toc104667808)

[4.1 Анализ условий труда программиста, работающего в компании ООО «Омега» 66](#_Toc104667809)

[4.2 Расчет искусственного освещения рабочего места программиста на предприятии ООО «Омега» 68](#_Toc104667810)

[4.3 Электробезопасность на предприятии ООО «Омега» 70](#_Toc104667811)

[4.4 Пожарная безопасность на предприятии ООО «Омега» 72](#_Toc104667812)

[Заключение 75](#_Toc104667813)

[Список использованных источников 77](#_Toc104667814)

[Приложение А (справочное) Описание таблиц баз данных 79](#_Toc104667815)

[Приложение Б (справочное) Интерфейс браузерного клиента 85](#_Toc104667816)

[Приложение В (справочное) Исходный код 91](#_Toc104667817)

[Приложение Г (справочное) Руководоство оператора 101](#_Toc104667818)

# **ВВЕДЕНИЕ**

Современное развитие технологий практически ни для кого не оставило секретом само понятие «виртуальный». Это понятие широко используется в современной культуре для обозначения самых разных форм взаимодействия не только между человеком и машиной, но и людьми друг с другом. «Виртуальное общение», различного рода «виртуальные встречи» практически не вызывают у современных людей чувства недоумения или полного непонимания. Можно сказать, что наряду с типом взаимодействия «человек-машина» появляется и новая схема «человек-машина-человек». Таким образом, виртуальный мир постепенно охватывает человека, вовлекает в свои границы и стирает подчас границы мира реального.

Для обозначения сферы общественных отношений, возникающих в связи с использованием различных электронных устройств, в современной науке используется ряд терминов: «киберпространство», «сетевое пространство», «среда Интернет», «виртуальное пространство» и другие. Виртуальное пространство - это область технических, технологических и социальных отношений, возникающих, изменяющихся и прекращающихся в процессе использования компьютерной или иной электронной технической сети по поводу информации, информационных ресурсов, информационных услуг и средств связи.

Виртуальное пространство, обладая такими характеристиками, как глобальность, отсутствие национальных границ, интерактивность, анонимность, с одной стороны, предоставляет личности беспрецедентные возможности для реализации прав и свобод, а с другой - широко используется для совершения различного рода неправомерных действий.

В связи с этим, в настоящее время широкое распространение получили всевозможные средства виртуального общения и альтернативные реальности, выраженные в виде различных виртуальных пространств.

Объектом выпускной квалификационной работы является процесс взаимодействий между людьми с использованием информационных технологий.

Предметом выпускной квалификационной работы является программное обеспечение для создания виртуального пространства внутри которого происходит общение людей.

Целью выпускной квалификационной работы является создание программного обеспечения на удобных для конечного пользователя платформах, которое без затруднений позволит ему исполнять свои личные потребности посредством данного приложения.

В соответствии с поставленной целью в работе определены следующие задачи:

1. Проанализировать предметную область и аналогичные проекты в сети Интернет;
2. Изучить требования конечного пользователя по функциональности будущего приложения;
3. Провести проектирование будущего программного обеспечения;
4. Реализовать программное обеспечение, представляющие собой серверную и клиентскую части в виде приложений на разных платформах;
5. Провести тестирование разработанного программного обеспечения.
6. **Общая часть**
   1. **Анализ предметной области**

В современных условиях глобальной информатизации и построения информационного общества широкое применение во всех сферах жизнедеятельности получили информационные и телекоммуникационные технологии. Их повсеместное и многоцелевое использование порождает возникновение виртуального пространства, ставшего в настоящее время основным способом коммуникации, методом ведения бизнес-операций и средством самовыражения.

В настоящее время на лицо наличие безусловной тенденции к интенсивному распространению информационных отношений, возникающих в виртуальном пространстве и проникновению их во все сферы жизнедеятельности.

Исходя из этого у группы людей появляется нужда в личном виртуальном пространстве, которое могло бы использоваться в своих целях по необходимости. В связи с этим было принято решение разработать программное обеспечение, генерирующее это так называемое виртуальное пространство, в котором люди могли бы создавать свои собственные профили (аватары) по своему усмотрению и взаимодействовать с другими подобными им профилями (аватарами).

* 1. **Анализ рынка существующих решений**

Перед разработкой собственного продукта необходимо для начала изучить рынок с уже существующими аналогами. В настоящее время существует сравнительно большое количество готовых программных решений, которые предоставляют людям использование своего виртуального пространства в зависимости от их желаний и требований.

К таковым решениям относятся, например, следующие продукты:

1. Социальная сеть «ВКонтакте»;
2. Мессенджер «WhatsApp»;
3. Многопользовательская компьютерная игра «VRChat».

Социальная сеть «ВКонтакте» - наиболее известная социальная сеть Рунета, изображено на рисунке 1. Здесь можно создать свою личную страницу, написать заметку, добавить аудио или видео, общаться с другими участниками «ВКонтакте», подписываться на обновления групп или пользователей, организовывать встречи или группы.

Достоинства:

1. Большое сообщество;
2. Постоянное развитие сервиса;
3. Разнообразие предоставляемых возможностей.

Недостатки:

1. Рост платного функционала;
2. Ограниченность данных вашего аккаунта настоящей информацией;
3. Наличие обязательной верификации пользователей.

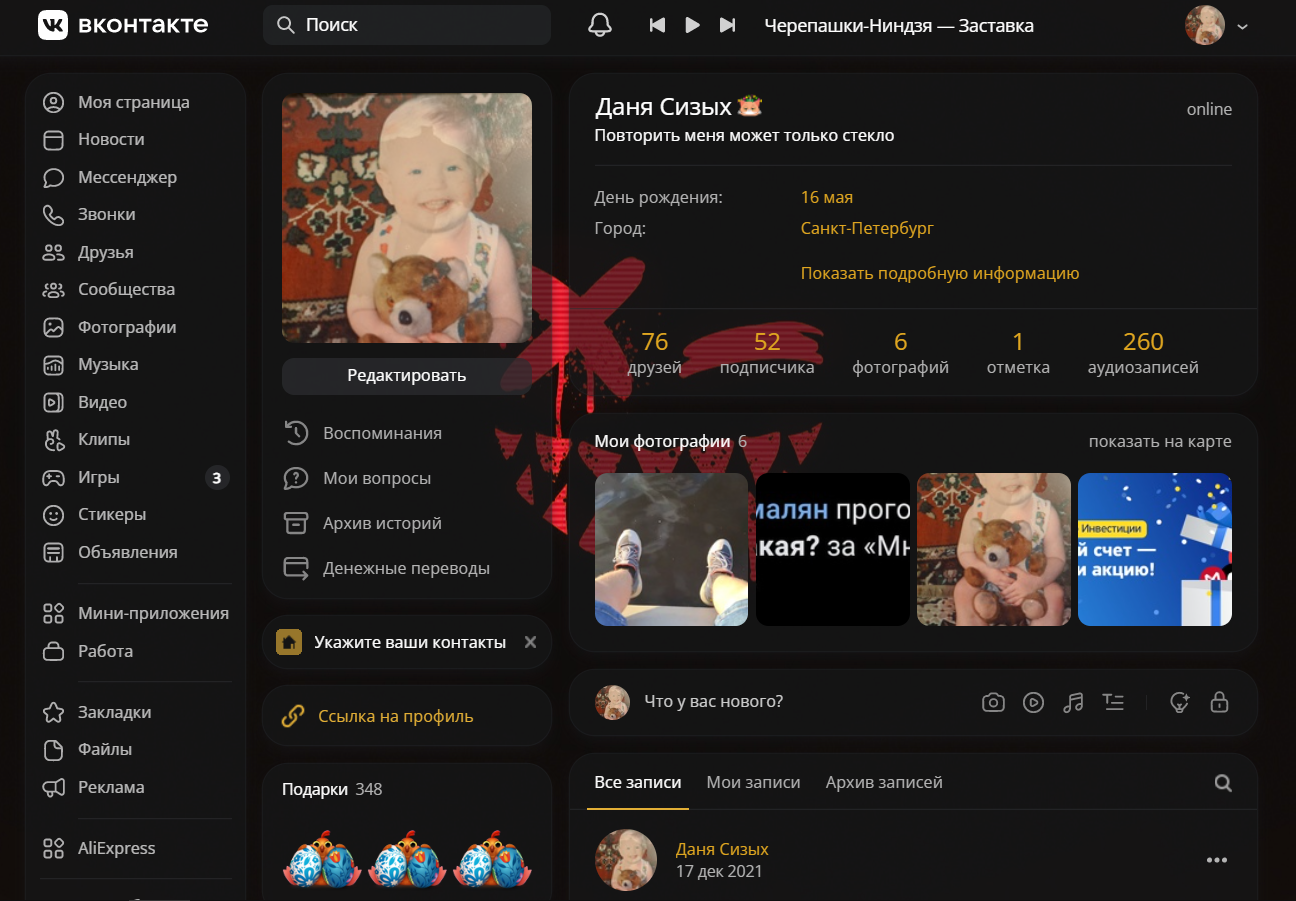


Рисунок 1 – Социальная сеть «ВКонтакте»

«WhatsApp» – американский бесплатный сервис обмена мгновенными сообщениями и голосовой связи по IP, принадлежащий компании Meta, изображено на рисунке 2. Она позволяет пользователям отправлять текстовые и голосовые сообщения, совершать голосовые и видеозвонки, обмениваться изображениями, документами, местоположением пользователя и другим контентом.

Достоинства:

1. Самый популярный мессенджер в мире;
2. Бесплатный сервис;
3. Наличие нескольких поддерживаемых платформ.

Недостатки:

1. Запрет одновременного использования на разных платформах;
2. Устаревший интерфейс и функционал;
3. Проблемы с безопасностью.

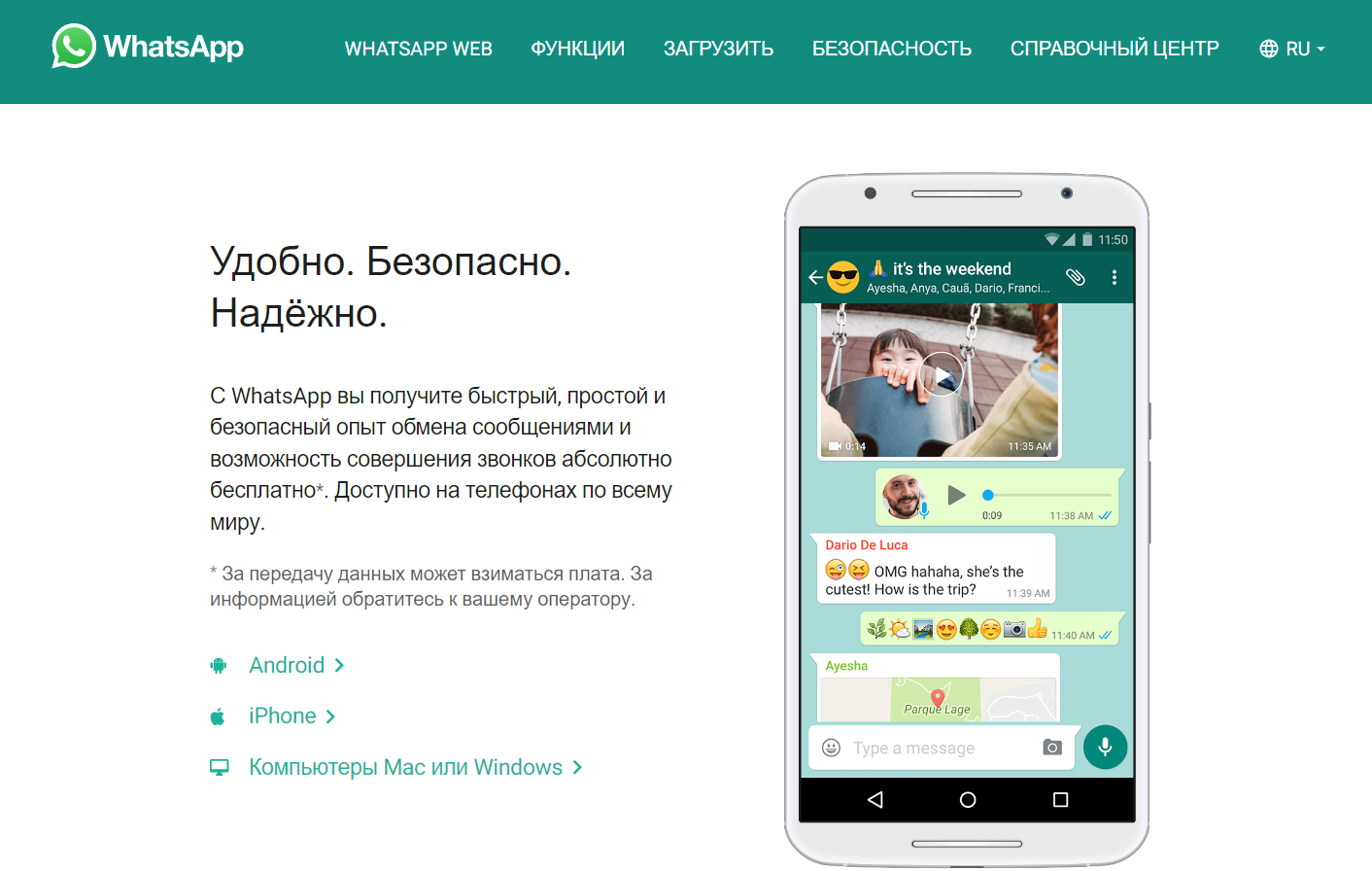


Рисунок 2 – Мессенджер «WhatsApp»

VRChat — это онлайн-сообщество, доступное для Windows

Mixed Reality, HTC Vive, Oculus Rift или просто для обычной клавиатуры, мыши и монитора, изображено на рисунке 3. Это программное обеспечение-симулятор, которое разработчики чаще называют «песочницей». Администраторы создают комнаты с нужной геометрией, а пользователи подключаются к ним. Это площадка для встреч, где вы можете создать свой собственный профиль, свой уникальный мир, играть в игры, смотреть фильмы и просто гулять, как в реальной жизни. Вместо безликих конференций Skype детище команды VRChat Inc. предлагает общаться в декорациях множества миров, начиная от лоснящегося от официоза зала и заканчивая космической станцией где-то на задворках вселенной.

Достоинства:

1. Трехмерное виртуальное пространство;
2. Бесплатность и открытость платформы;
3. Легкая адаптивность к пользовательскому контенту.

Недостатки:

1. Необходимость высокопроизводительного оборудования;
2. Отсутствие функционального развития.



Рисунок 3 – онлайн-сообщество VRChat

Все перечисленные решения имеют свои достоинства и недостатки. Но у всех присутствует верификация пользователей в той или иной степени, что существенно сужает возможности людей, ограничивая их в количестве создаваемых учетных записей в системе для проецирования различных представлений себя в виртуальном пространстве.

* 1. **Обоснование и выбор методики, технологии и инструментальных средств разработки**

Для решения задачи создания данного программного продукта необходимо изначально определиться с языковым и техническим стеком для разработки проекта. В подзадачи были определены необходимость создания серверной и клиентской части приложения [1, 2].

По требованиям клиентское приложение может существовать в разных вариациях, представляя собой мобильное приложение для операционной системы Android и браузерное решение для настольных платформ. Серверное же приложение представляет собой продукт, запускаемый на настольных компьютерах, имеющий бесперебойный источник питания.

Чтобы выбирать средства разработки необходимо иметь некую основу, которая будет основополагающим принципом при выборе продукта. Так, в основе инструментального стека должны лежать продукты известных компаний, вокруг которых существует живое сообщество разработчиков, поддерживающих свой продукт и наполняющих его новым и удобным функционалом. Но главным фактором выбора является личный опыт использования таких продуктов. Под такие рассуждения подходят компании Microsoft и JetBrains. Первая имеет в своем арсенале самую популярную IDE для разработки разно-платформенных приложений – Visual Studio, и не менее популярную систему управления базами данных – SQL Server. Вторая же является амбассадором IDE по разработке веб-приложений – WebStrom, а также мобильных приложений для операционной системы Android – Android Studio.

Все выше перечисленные продукты этих компаний отличаются большим функционалом и широкими возможностями для модификаций с помощью встроенной системы модулей.

Выбор же SQL Server обусловлен тем, что она является одной из наиболее популярных систем управления базами данных в мире. Помимо этого, она поддерживают технологию ADO.NET, которая является посредником между базой данных и приложением на C#. Данная СУБД подходит для самых различных проектов: от небольших приложений до больших высоконагруженных проектов. Она характеризуется такими особенностями как: производительность, надежность, безопасность и простота.

Из-за масштабов разработки и ориентированности на разные платформы нет возможности использования какого-либо конкретного языка программирования. Таким образом, существует необходимость определения стека языковых средств, которые будут участвовать в разработке. Из всех характеристик языка программирования есть несколько, являющихся основополагающими в плане выбора. К таковым причисляется наличие опыта работы с конкретным языком и вытекающим из этого скорость разработки. В ходе обучения было затронуто немалое количество языков для написания программ, из которых будут по параметрам выше выделены несколько, а именно: C#, C++, Kotlin, JavaScript и PHP. C# [3, 4] и C++ [5] будут использованы для написания серверной части программного обеспечения; Kotlin [6] возьмет на себя участь мобильной разработки, а JavaScript с PHP сформируют web-версию клиентского приложения [7, 8, 9].

* 1. **Технология WebSocket**

Существуют разные способы передачи данных от браузера или приложения к серверам и обратно.

WebSocket – протокол полнодуплексной связи (может передавать и принимать одновременно) поверх TCP-соединения, предназначенный для обмена сообщениями между клиентом и веб-сервером в режиме реального времени.

WebSocket разработан для создания приложений в web-браузерах и web-серверах, но он может быть использован для любого клиентского или серверного приложения. Протокол WebSocket – это независимый протокол, основанный на протоколе TCP.

Специальные надстройки протоколов передачи данных кодируют информацию на стороне отправителя и раскодируют на стороне получателя, оставляя ее зашифрованной для любых посредников. Так достигается безопасный транспортный уровень.

Для установления соединения WebSocket клиент и сервер используют протокол, похожий на HTTP. Клиент формирует особый HTTP-запрос, на который сервер отвечает определенным образом.

Преимущества WebSocket:

1. двустороннее соединение;
2. снижение используемого трафика;
3. безопасность;
4. минимальная задержка передачи данных.
5. **Специальная часть**
   1. **Постановка задачи**

Постановка задачи по структуре соответствует программному документу «Техническое задание» и включает в себя следующие подпункты:

1. Основание для разработки;
2. Назначение разработки;
3. Требования к программе.
   * 1. **Основание для разработки**

Основанием для разработки является полученное задание от лица сотрудников компании ООО «Омега».

Наименование темы разработки – «Разработка программного обеспечения для создания виртуального пространства с поддержкой чата».

Условное обозначение темы разработки (шифр темы) – «ReChat».

* + 1. **Назначение разработки**

Данное разрабатываемое программное обеспечение предназначено для решения задачи предоставления пользователям возможности использования виртуального пространства с поддержкой чата. Эксплуатация программы будет осуществляться в личных целях заинтересованного круга лиц, которым выступает заказчик разработки.

* + 1. **Требования к программе**
       1. **Требования к функциональным характеристикам**

Данное программное обеспечение должно обеспечить возможность выполнения всего набора ниже перечисленных функциональных характеристик.

Для серверного приложения таковыми являются:

1. создание сессий подключения;
2. параллельная работа всех подключений;
3. тайм-аут отключения бездействующих подключений;
4. хранение созданных сессий на протяжении всего времени соединения;
5. обработка получаемых данных;
6. ретрансляция операций к серверу баз данных и к клиенту;
7. логирование проводимых операций.

Для клиентского приложения функциональными характеристиками являются следующее:

* 1. Функции регистрации нового пользователя системы;
  2. Функции авторизации пользователя путем ввода логина и пароля;
  3. Функции автоматической авторизации пользователя путем верификации токена устройства;
  4. Функции установки, смены, удаления главного изображения профиля пользователя;
  5. Функции смены отображаемого имени пользователя;
  6. Функции установки, смены, удаления информации о профиле пользователя;
  7. Функции смены статуса видимости пользователя среди подключенных участников системы;
  8. Функции смены статуса приватности информации о профиле пользователя;
  9. Функции создания личных диалогов с другим пользователем;
  10. Функции отправки и приемки текстовых сообщений и изображений в режиме реального времени;
  11. Функции создания групповых бесед с другими пользователями;
  12. Функции установки, смены и удаления информации о групповой беседе;
  13. Функции удаления групповой беседы и/или выхода из нее;
  14. Функции ранжирования прав доступа участников в групповой беседе;
  15. Функции информирования новых сообщений в диалоговых каналах;
  16. Функции просмотра профиля другого пользователя;
  17. Функции поиска профиля другого пользователя;
  18. Функции добавления и удаления профиля другого пользователя в список друзей.
      + 1. **Требования к техническим средствам**

Для нормального функционирования данного программного обеспечения, а именно серверного приложения, в состав технических средств должен входить компьютер, включающий в себя:

1. процессор x86 с тактовой частотой, не менее 2,5 ГГц;
2. оперативную память объемом, не менее 4 Гб;
3. постоянный накопитель объемом, не менее 50 Гб;
4. видеокарту, сетевую карту, монитор, мышь, клавиатура;
5. подключение к сети Интернет.

Для функционирования клиентского приложения на настольной платформе необходим компьютер с теми же техническими параметрами, что и для функционирования сервера, а для мобильной платформы следует иметь мобильное устройство, включающее:

1. процессор x86 с тактовой частотой, не менее 1 ГГц;
2. оперативную память объемом, не менее 1 Гб;
3. размер встроенной памяти от 4 Гб;
4. подключение к сети Интернет.
   * + 1. **Требования к информационной и программной совместимости**

Для полноценного функционирования данного программного обеспечения для серверного приложения необходимо наличие операционной системы Windows 7 и выше, и Microsoft SQL Server.

Для полноценного функционирования данного программного обеспечения для клиентской части на настольной платформе необходимо наличие операционной системы, которая может предоставлять пользователю свой GUI и иметь браузер, или возможность его инсталлирования, для мобильной платформы необходимо иметь операционную систему Android 10 и выше.

* 1. **Диаграмма вариантов использования**

В ходе анализа был выделен основной актер системы – Пользователь, второстепенный актер – Гость. Пользователь может использовать все функции клиентского приложения, которые изложены ниже:

1. Управлять аккаунтом: изменить данные профиля, изменить приватность данных, изменить онлайн-статус, деавторизоваться;
2. Управлять списком друзей: добавить друга, удалить друга, найти пользователя;
3. Создать личный диалог с другим участником;
4. Создать групповой диалог, выбрав участника (-ов) и задав название;
5. Открыть личный диалог для отправки текстового сообщений или изображения;
6. Открыть групповой диалог для отправки текстового сообщения или изображения, для просмотра данных о диалоге, для управления рангами участников.

Пользователь обобщает действия второстепенного актера, который в свою очередь может:

1. Зарегистрироваться;
2. Авторизоваться.

Данные связи вариантов использования функций актерами системы демонстрируются на рисунке 4.



Рисунок 4 – Диаграмма вариантов использования

* 1. **Общее описание структуры программного обеспечения**

Программное обеспечение представляет собой стек из серверного и клиентского приложений. Структурный принцип, по которому создан данный описываемый проект представляет из себя трехуровневую архитектуру клиент-серверного приложения [10]. В данной архитектуре имеется три точки для работы приложения: клиент, сервер для обработки и сервер базы данных.

Сервер для обработки, представляющий из себя настольное приложение, исполняемое в среде Windows, обрабатывает все приходящие к нему запросы и перенаправляет их в нужный узел в режиме реального времени. Данный сервер имеет на вооружении технологию веб-сокетов.

Это позволяет ему создавать ему интерактивное соединение со своими клиентами. Хранение же информации происходит в базе данных. Таким образом, все обращения клиентов к базе данных происходят через выше упомянутый сервер для обработки, выступающий в роли промежуточного программного обеспечения. Вследствие такого устройства повышается гибкость работы и производительность.

В системе присутствуют изображения (с которыми могут взаимодействовать пользователи), для хранения их в базе данных используется веб-сервер. Он хранит сами изображения, а в базе данных хранится путь, по которому их можно получить.

Исходя из требований клиентское приложение может существовать в разных вариациях, представляя собой мобильное приложение для операционной системы Android и браузерное решение для настольных платформ. Благодаря разделению на разные части разработка может оставаться гибкой в выборе решения для разных задач.

В теории обе части являются свободно заменяемыми. При необходимости имеется возможность заменить серверную часть без нужды внесения изменений в клиентской части и наоборот. Такой подход называется loose coupling, то есть отказ от сильной взаимозависимости частей. С таким подходом один сервер может обслуживать несколько разных клиентских приложений.

* 1. **Архитектура серверного приложения**
     1. **Проектирование базы данных**

База данных создается для хранения в ней информации и получения этой информации при необходимости [11]. Для создания данного проекта база данных является неотъемлемой частью всей системы. В ней будут храниться все данные, вносимые пользователями, и данные необходимые для функционирования и верной работоспособности всей системы в целом.

Реляционная модель базы данных – это модель, которая описывает как организовать данные в таблицах и как определить связи между этими таблицами. Правила реляционной модели диктуют, как информация должна быть организована в таблицах и как таблицы связаны друг с другом. В конечном счете результат можно предоставить в виде диаграммы базы данных или, если точнее, диаграммы «сущность-связь».

На рисунке 5 демонстрируется ER-модель, отражающая выше упомянутую концепцию модели «сущность-связь».

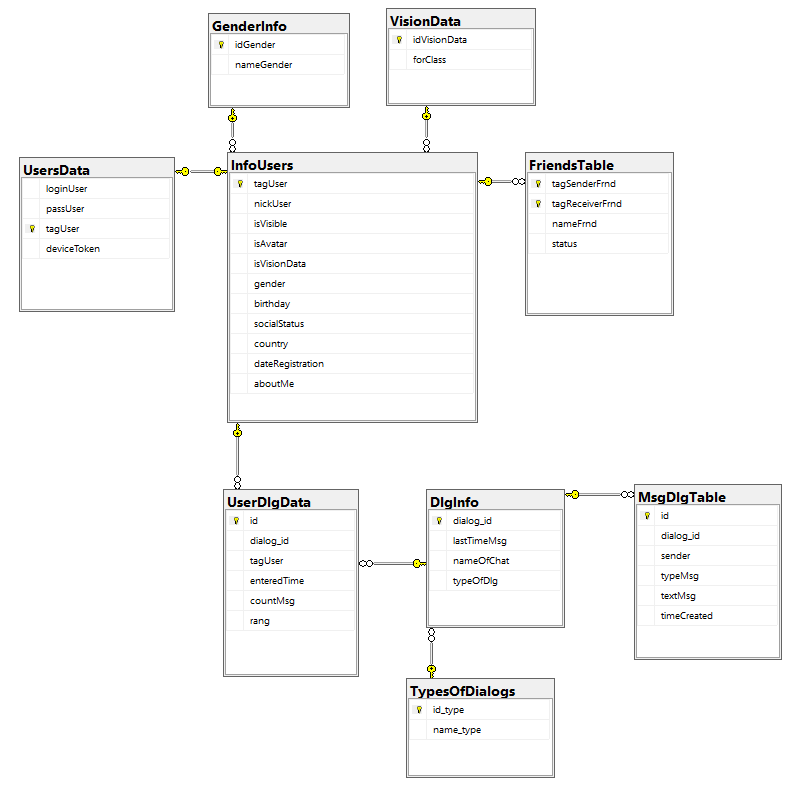


Рисунок 5 – ER-диаграмма базы данных

База данных содержит девять таблиц: UsersData, InfoUsers, GenderInfo, VisionData, FriendsTable, DlgInfo, UserDlgData, TypesOfDialogs, MsgDlgTable.

В таблице UsersData хранятся данные необходимые для авторизации пользователей.

В таблице InfoUsers хранится вся информация о пользователе, большинство из которой он задает сам и может изменять.

В таблице GenderInfo хранятся типы гендера, который пользователь может задать для своего профиля.

В таблице VisionData хранятся типы приватности, которую пользователь может задать для своих данных.

В таблице FriendsTable хранятся записи о дружеских отношениях между пользователями.

В таблице DlgInfo хранится информация о всех существующих диалогах в системе.

В таблице UserDlgData хранится информация об участниках в каждом конкретном диалоге.

В таблице TypesOfDialogs хранится информация о возможных типажах диалога.

В таблице MsgDlgTable хранятся записи о всех сообщениях пользователей.

В приложении А представлено подробное описание данных этих таблиц.

В таблице 1 представлено описание связей таблиц в базе данных.

Таблица 1 – Описание связей между сущностями в базе данных сервера

| **№** | **Название связи** | **Сущности, участвующие в связи** | **Назначение** |
| --- | --- | --- | --- |
| 1 | 1:1 | UsersData – InfoUsers | Один единственный пользователь имеет единственный комплект информации о себе. |
| 2 | 1:М | GenderInfo – InfoUsers | Один конкретный типаж пола может быть у множества пользователей. |
| 3 | 1:М | VisionData – InfoUsers | Один конкретный тип приватности может быть у множества пользователей. |
| 4 | 1:М | InfoUsers - FriendsTable | Один конкретный пользователь может иметь множество друзей (отсюда следует множество комбинаций тегов) |
| 5 | 1:М | InfoUsers – UserDlgData | Один конкретный пользователь может состоять в множестве диалогов. |
| 6 | 1:М | DlgInfo – UserDlgData | Каждый диалог может содержать множество участников. |
| 7 | 1:М | TypesOfDialogs – DlgInfo | Один тип диалога может встречаться в множестве чатов. |
| 8 | 1:М | DlgInfo – MsgDlgTable | Каждый диалог может содержать множество сообщений. |

* + 1. **Web-сервер для изображений**

Большинство веб-приложений так или иначе используют файлы: аудио, видео, изображения, текстовые и другие типы файлов.

Есть два способа сохранения файлов - в базе данных и в файловой системе, и хранение файлов в базе данных - не всегда является идеальным способом.

Microsoft рекомендует сохранять в базе данных файлы, размер которых не более 1 МБ. Все остальное - в файловой системе. Так или иначе у обоих способов есть свои плюсы. Например, при хранении в базе данных соблюдается целостность данных. В то же время работа с базой данных - это одно из узких мест веб-приложения. И чем больше запросы к базе будут оптимизированы, и чем их будет меньше, тем больше будет производительность. А доступ к файловой системе, как правило, будет происходить быстрее, чем доступ к базе данных.

Исходя из этого было принято решение о создание веб-сервера, который будет хранить в себе изображения пользователей (изображения профилей, изображения в диалогах). В базе данных же будут храниться лишь пути, по которым можно получить эти изображения. Данный сервер имеет несколько PHP-файлов, предназначенных для отработки функций загрузки, смены и удаления изображений.

* + 1. **Проектирование сервера**

WebSocket-сервер необходим для реализации всей концепции системы о взаимодействиях в режиме реального времени. К нему подключаются все пользователи, которые используют клиентские приложения. К нему же имеет подключение и сервер базы данных.

Данный сервер открывает, хранит и завершает сессии пользователей, а также принимает сообщения от них и обрабатывает их. После обработки сервер перенаправляет сообщение в зависимости от результатов обработки.

* 1. **Архитектура клиентского приложения**
     1. **Платформы клиентских приложений**

По требованиям заказчика клиентское приложение должно было быть исполнено на двух разных платформах.

Первый вариант предполагал мобильное использование [12], поэтому одной из платформ является операционная система Android.

Второй вариант заключался в использовании клиента с помощью настольных систем. Исходя из этого следующей платформой был выбран браузер, что позволит пользователям пользоваться веб-приложением с любого компьютера без нужды установки лишних программ.

Обе платформы имеют свою уникальную структуру, но каждая использует собственную базу данных одинаковой структуры для хранения данных и сокращения обращений к серверу.

* + 1. **Мобильная платформа**

Структура мобильного приложения представляет из себя разделение классов по своим функциональным особенностям. Происходит деление на классы активностей, на классы адаптеров, на вспомогательные классы, на классы данных, на классы типа «интерфейс», на классы, отвечающие за интерфейс приложения. Демонстрация деления представлена на рисунке 6.

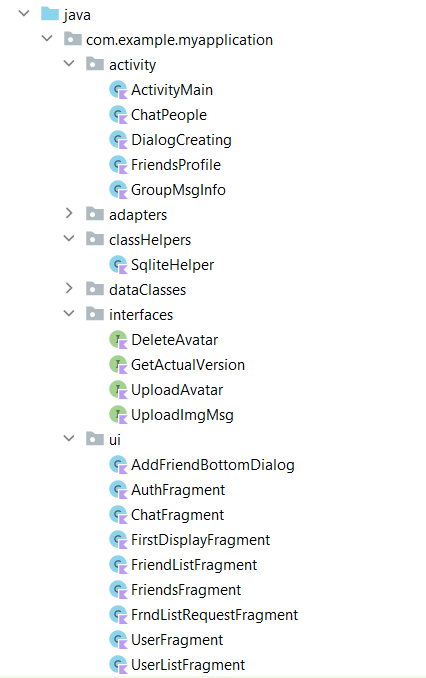


Рисунок 6 – Структура мобильного приложения

Архитектурно приложение является модульным за счет использования фрагментации интерфейса. Каждый фрагмент – это модульная часть активности, у которой есть свой жизненный цикл и свои обработчики различных событий.

Имеется один головной класс – ActivityMain, находящийся в каталоге - activity. Он используется при запуске приложения и обрабатывает основные функции. Он же является родительским контейнером для всех фрагментов приложения. В нем происходит логическая обработка и смена интерфейсов. Остальные классы в этой же подпапке представляют активности, вызываемые для различных функций системы:

1. ChatPeople – активность для диалога;
2. DialogCreating – активность для создания группового диалога;
3. FriendsProfile – активность для открытия профиля другого пользователя;
4. GroupMsgInfo – активность для открытия информации о групповом диалоге.

Каталог adapters включает в себя собственные адаптеры необходимые для заполнения списков ListView и других подобных ему данными для отображения.

Каталог classHelpers включает в себя вспомогательный класс, который нацелен только на работу со встроенной базой данных. Все действия данного с базой происходят посредством использования функций этого класса.

Каталог dataClasses наполнена классами данных (data), необходимые для формирования структур данных при работе с сервером.

Каталог interfaces включает в себя классы типа «интерфейс», отвечающие за формирование запросов к веб-серверу с изображениями и работу с ним.

Каталог ui вмещает в себя фрагменты (экраны) приложения:

1. AddFriendBottomDialog – экран добавления новых участников в групповую беседу;
2. AuthFragment – экран авторизации или регистрации пользователя;
3. ChatFragment – экран списка диалогов;
4. FirstDisplayFragment – загрузочный экран;
5. FriendsListFragment – экран списка друзей;
6. FriendsFragment – экран отдела функций с другими пользователями;
7. FrndListRequestFragment – экран списка запросов в друзья от других пользователей;
8. UserFragment – экран профиля пользователя;
9. UserListFragment – экран списка онлайн-пользователей в системе.

Все ресурсы, используемые приложением, хранятся в каталоге res, изображено на рисунке 7.

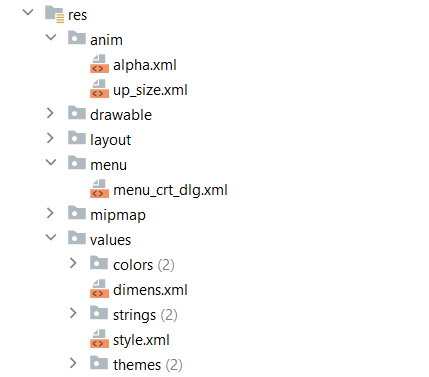


Рисунок 7 – Каталог res с ресурсами приложения

* + 1. **Web-платформа**

Структура веб-приложения представляет древовидную форму с блочными элементами. Каждому разделу и подразделу отводится отдельная ветка. Таким образом смысловая нагрузка делится между основными разделами и их подразделами.

Функционально и стилистически веб-версия повторяет мобильную платформу: она имеет те же возможности и логику обращения с интерфейсом.

Файловая структура данного приложения представлена на рисунке 8.

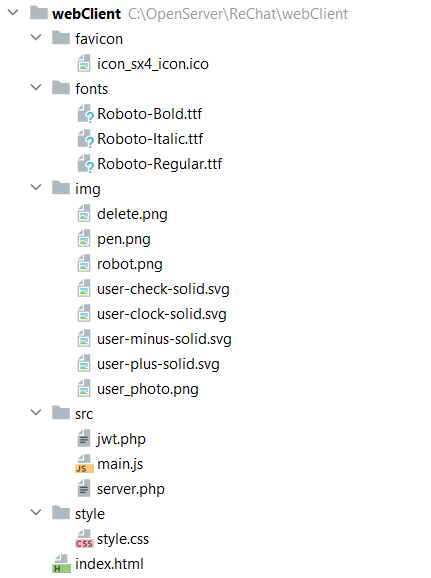


Рисунок 8 – Файловая структура веб-приложения

На изображении выше видно, что имеется основной каталог – webClient. Он в свою очередь включает в себя файл index.html, хранящий html-разметку, и подкаталоги:

1. favicon – предназначен для хранения иконки;
2. fonts – предназначен для хранения шрифтов;
3. img – предназначен для хранения изображений;
4. src – предназначен для хранения файлов с логикой работы;
5. style – предназначен для хранения используемых стилей разметки.
   * 1. **Проектирование базы данных**

Для клиентских приложений необходимо иметь базу данных для хранения пользовательских данных и уменьшения количества обращений к серверу. Для этой цели на обеих платформах клиентских приложений была разработана база данных одинаковой структуры, представленной на рисунке 9.

Эта база данных состоит из семи таблиц: DataGroupInfo, FriendsTable, ListUsersChat, MsgDlgTable, OnlineUsers, UserDlgTable, allUserInfoTable.

В таблице DataGroupInfo хранится информация о текущей открытом групповом диалоге.

В таблице FriendsTable хранится информация о дружеских отношениях пользователя с другими пользователями.

В таблице ListUsersChat хранится список диалогов, отображаемых пользователю.

В таблице MsgDlgTable хранится информация о сообщениях.

В таблице OnlineUsers хранится информация о пользователях, находящихся в статусе онлайн.

В таблице UserDlgTable хранится информация о диалогах.

В таблице allUserInfoTable хранится информация об открытом профиле пользователя.

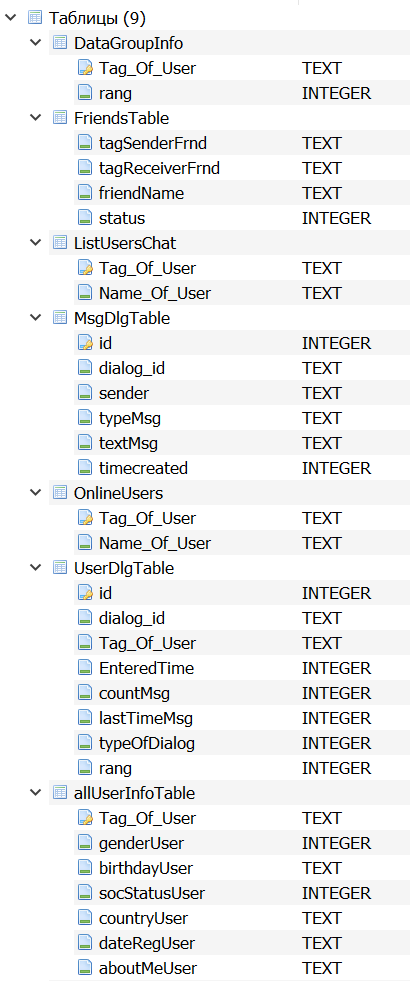


Рисунок 9 – Структура базы данных клиентских приложений

* 1. **Реализация серверного приложения**
     1. **База данных**

Создание базы данных, ее поддержка и обеспечение доступа пользователей к ней осуществляется централизованно с помощью специального программного инструментария – системы управления базы данных.

СУБД, которая была использована для создания базы данных – это, как ранее упоминалось, SQL Server, разработанная корпорацией Microsoft.

Основной используемый язык запросов — Transact-SQL, создан совместно Microsoft и Sybase. Transact-SQL является реализацией стандарта ANSI/ISO по структурированному языку запросов (SQL) с расширениями.

Проектирование базы данных для проекта началось с разработки моделей данных, которые содержат несколько высокоуровневых сущностей и связей, затем работа продолжилась в виде серии нисходящих уточнений низкоуровневых сущностей, связей и относящихся к ним атрибутов.

После полученной модели данных началось описание реляционных таблиц и ограничений для них на основе ранее собранной информации, определение конкретных структур хранения данных и методов доступа к ним. По получении удовлетворительных результатов проделанной работы начался этап реализации физической версии базы данных параллельно с разработкой приложений, в которых проектировались транзакции, представляющих некоторые события реального мира.

* + 1. **Взаимодействие с базой данных**

Клиентские приложения имеют соединение лишь с сервером, и они никак не могут взаимодействовать с базой данных напрямую. Им необходимо отправить нужный запрос на сервер, который будет обработан и уже после перенаправлен на сервер базы данных. Взаимодействие же с базой будет уже осуществлять данный сервер.

Для реализации этого самого взаимодействия было решено использовать технологию ADO.NET, включающую набор классов, предоставляющих возможность отправлять запросы к базе данных, устанавливать подключения, получать ответ от базы данных и производить ряд других операций.

Схематично архитектуру ADO.NET можно представить следующим образом, изображенном на рисунке 10.

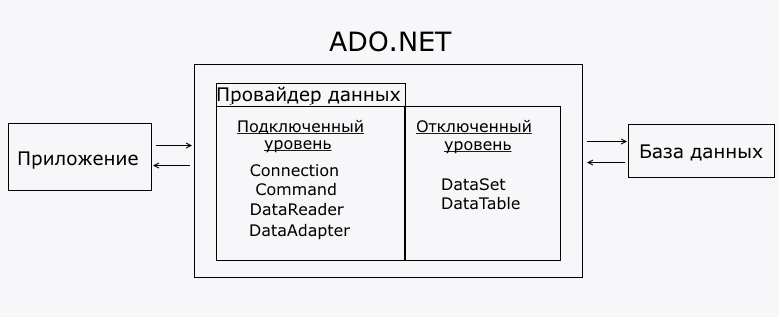


Рисунок 10 – Схематичная архитектура ADO.NET

Для начала работы необходимо определить функциональность провайдера для баз данных MS SQL Server путем подключения пространства имен «System.Data.SqlClient».

Чтобы подключаться к базе данных, необходимо создать и использовать объект SqlConnection. В конструктор объекту SqlConnection передается строка подключения, которая инициализирует объект, изображено на рисунке 11. Чтобы использовать этот объект и подключаться к базе данных, мы должны выполнить его метод Open(), а после завершения работы с базой данных нам надо вызвать метод Close() для закрытия подключения.

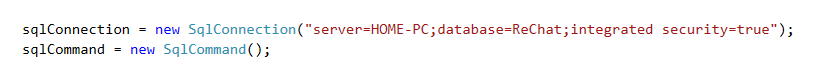


Рисунок 11 – Использование объекта SqlConnection

После установки подключения можно выполнять операции с базой данных. Провайдер для MS SQL предоставляет реализацию этого в виде класса SqlCommand. Этот класс инкапсулирует sql-выражение, которое должно быть выполнено. С помощью свойства CommandText устанавливается SQL-выражение, которое будет выполняться, изображено на рисунке 12.

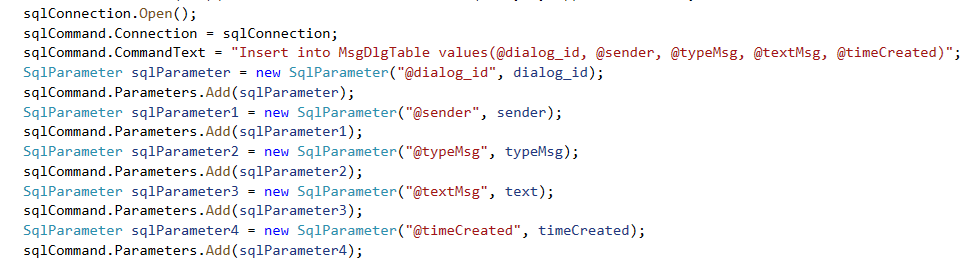


Рисунок 12 – Задание sql-выражения свойству CommandText

Для выполнения команды применяется один из методов SqlCommand: ExecuteNonQuery, ExecuteReader, ExecuteScalar, изображено на рисунке 13.



Рисунок 13 – Метод ExecuteNonQuery для выполнения команды

* + 1. **WebSocket-сервер**

Для создания WebSocket-сервера, обеспечивающего связь между клиентом и сервером в режиме реального времени, используется библиотека uWebSockets. Код серверного приложения написан на языке C++ с использованием ранее упомянутой библиотеки. Такое решение является оптимизированным с точки зрения скорости и объема памяти, и позволяет отправлять зашифрованные сообщения быстрее, чем большинство альтернативных серверов могут отправлять даже незашифрованные сообщения в открытом тексте. О надежности использования данного решения говорит тот факт, что uWebSockets управляет многими крупнейшими криптобиржами в мире, ежедневно обрабатывая объемы торговли на несколько миллиардов долларов.

Сервер состоит из главной функции – main, вызываемой при старте приложения, и множества сторонних функций, необходимых для обработки данных курсирующих через сервер.

В данной функции происходит обращение к пространству имен подключенной библиотеки и выполнение функции создания «приложения» этого сервера, изображено на рисунке 14.

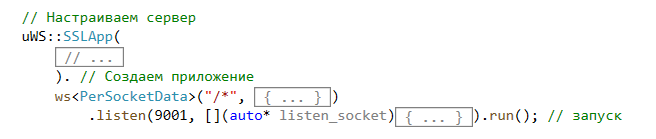


Рисунок 14 – Функция создания приложения

Эта функция принимает на себя ряд настроек, изображено на рисунке 15, среди которых располагаются:

1. Таймаут подключения пользователя, где указывается время, через которое пользователь будет отключен от сервера в случае бездействия;
2. Функция, вызываемая при открытии нового соединения (подключения нового клиента);
3. Функция, вызываемая при получении любого сообщения сервером от клиента;
4. Функция, вызываемая при закрытии соединения с клиентом (любое отключение клиента от сервера).

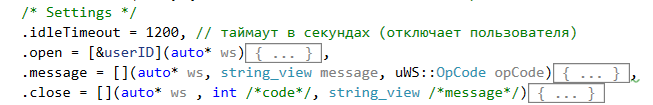


Рисунок 15 – Определение настроек сервера

После определения нужных настроек и их поведения, вызывается внутренняя функция «прослушивания». Данная функция устанавливает порт, который будет «прослушиваться» и с которым будет происходить взаимодействие. Затем наступает заключающий этап, где происходит вызов функции запуска «приложения».

При подключении нового клиента к серверу, вызывается функция open, которая добавляет новую сессию в виде структуры «имя-идентификатор» в список остальных сессий. Идентификатор (тег пользователя) – это уникальный восьмизначный набор из букв латинского алфавита и арабских цифр, генерируемый при подключении нового клиента к серверу. Впоследствии этот идентификатор может изменить у сессии, если клиент с данным подключением произведет авторизацию в системе в уже существующий ранее профиль. Имя – имя пользователя (также подвержено изменению). После этого при получении сообщений от клиента, вызывается функция message. Там полученные данные в виде строки проходят обработку и выполняются различные операции в зависимости от результатов обработки. В случае необходимости, строка форматируется в объект JSON. При завершении сеанса вызывается функция close: происходит удаление текущей сессии из списка.

* 1. **Реализация клиентского приложения**
     1. **База данных**

Чтобы хранить данные на клиентском устройстве необходимо создать базу данных. Для мобильной платформы характерно использование мобильной версии sqlite, для браузера же – web sql. Обе базы данных имеют схожую концепцию, набор возможностей и язык запросов. К сожалению, Android Studio и WebStrom не располагают возможностями для управления базами данных при помощи интерфейса, поэтому все взаимодействия выполняются посредством программного кода, изображено на рисунках 16 и 17. Таким образом, были созданы все таблицы и возможные действия с ними.

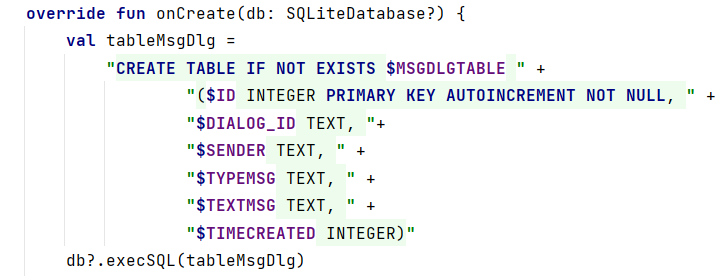


Рисунок 16 – Создание таблицы в базе данных

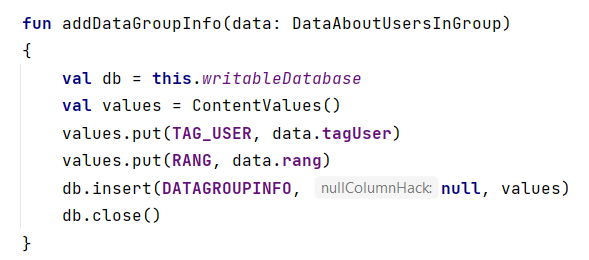


Рисунок 17 – Вставка данных в таблицу

* + 1. **Работа WebSocket**

Для того, чтобы иметь возможность обмениваться с сервером данными в режиме реального времени клиентскому приложению также необходимо иметь реализацию WebSocket’ов. Мобильное приложение использует бибилотеку Java-WebSocket, браузеры же имеют нативную поддержку данной технологии.

Обе платформы имеют одинаковую структуру осуществления использования WebSocket’ов.

Для того чтобы открыть соединение этого типа, создается объект WebScoket’а, которому в параметрах указывается адрес сервера.

Затем описываются события данного соединения в виде коллбек-функций, изображено на рисунке 18: OnOpen (соединение установлено), OnMessage (получены данные), onError (ошибка соединения), OnClose (закрытие соединения). После установки соединения выполняется код, находящийся в блоке OnOpen(), запускающий синхронизацию данных с сервером, если пользователь авторизован, или же запускающий окно авторизации, если нет. Данный блок является отправной точкой работы всей системы взаимодействия.



Рисунок 18 – Описание событий WebSocket-соединения

При получении данных происходит их декодирование, определяющее необходимые функции для выполнения операций. Происходит это в блоке OnMessage(), в котором вызывается функция для обработки пришедших данных – parseMessage(string). Частичный пример кода из этой функции представлен на рисунке 19. В процессе декодирования используется десериализация JSON-объектов: в мобильной версии для этого служат созданные заранее классы данных (Data) по выявленной структуре, в браузере же есть автоматическая структуризация данных, что освобождает время от написания подобных классов.



Рисунок 19 – Процесс декодирования полученных данных от сервера

В случае, если соединение с сервером прерывается по любой возможной причине, то вызывается блок OnClose(), в котором происходит смена отображаемого экрана на окно загрузки и предпринимаются попытки переподключения к серверу.

* + 1. **Функциональность приложения**

В приложение все действия выполняются после указания выполнения сервером этих самых действий. Весь принцип построен по следующей последовательности операций:

1. приложение выполняет какое-либо действие;
2. создается запрос к серверу на основе этого действия и отправляется;
3. приложение принимает ответные данные от сервера;
4. приложение выполняет действия на основе полученных данных.

Блокировка интерфейса в случае ожидания ответа от сервера полностью исключена, потому что все операции происходят в разных потоках относительно друг друга и относительно интерфейса. Также отсутствует вариант превышения ожидания времени ответа от сервера. Ответ будет получен, когда он будет обработан. Ограничения по времени нет.

Разработанное приложение включает в себя более десяти различных экранов с представлениями данных, поэтому при описании функций приложения используются снимки экранов интерфейса мобильного решения для более компактного вида. Снимки экранов интерфейса браузерной версии представлены в приложении Б на рисунках Б.1 – Б.12.

* + - 1. **Авторизация и регистрация**

Регистрация происходит следующим образом, что гость заполняет поля, изображено на рисунке 20, затем запрос отправляется на сервер, обрабатывается там и высылается ответ в зависимости от успешности. Пользователь оповещается о статусе успеха операции. Авторизация, изображено на рисунке 21, с помощью логина и пароля происходит по тому же принципу, но после верификации введенных данных сервером, экран пользователя меняется на загрузочное ожидание и происходит синхронизация данных с сервером, изображено на рисунке 22.

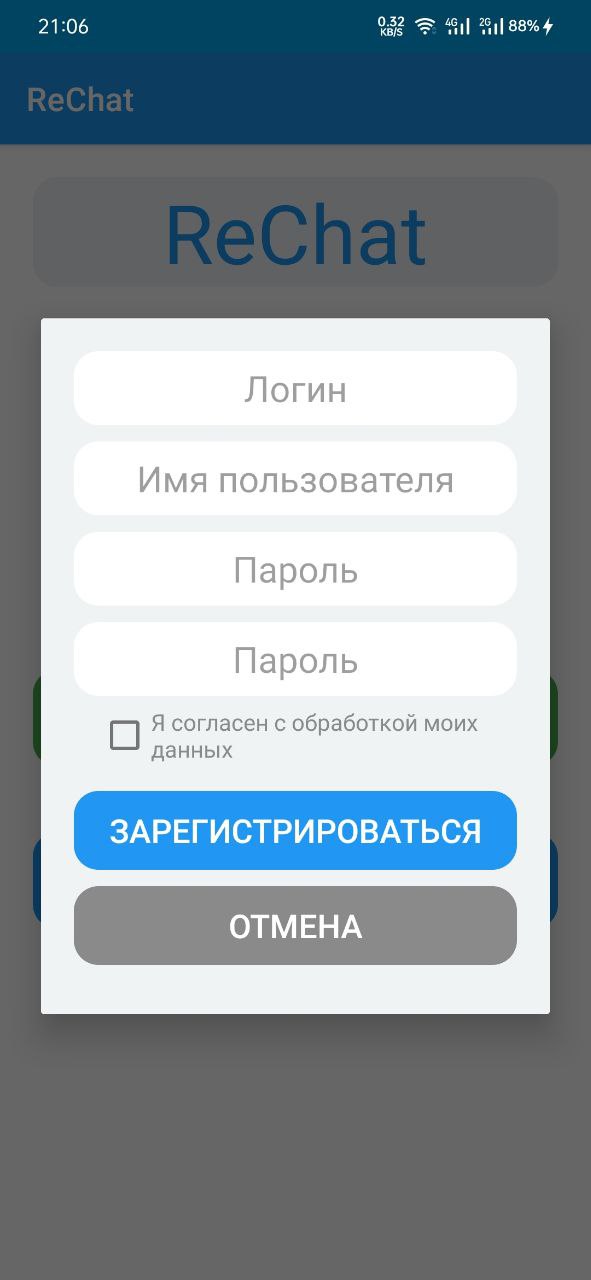


Рисунок 20 – Экран регистрации



Рисунок 21 – Экран авторизации

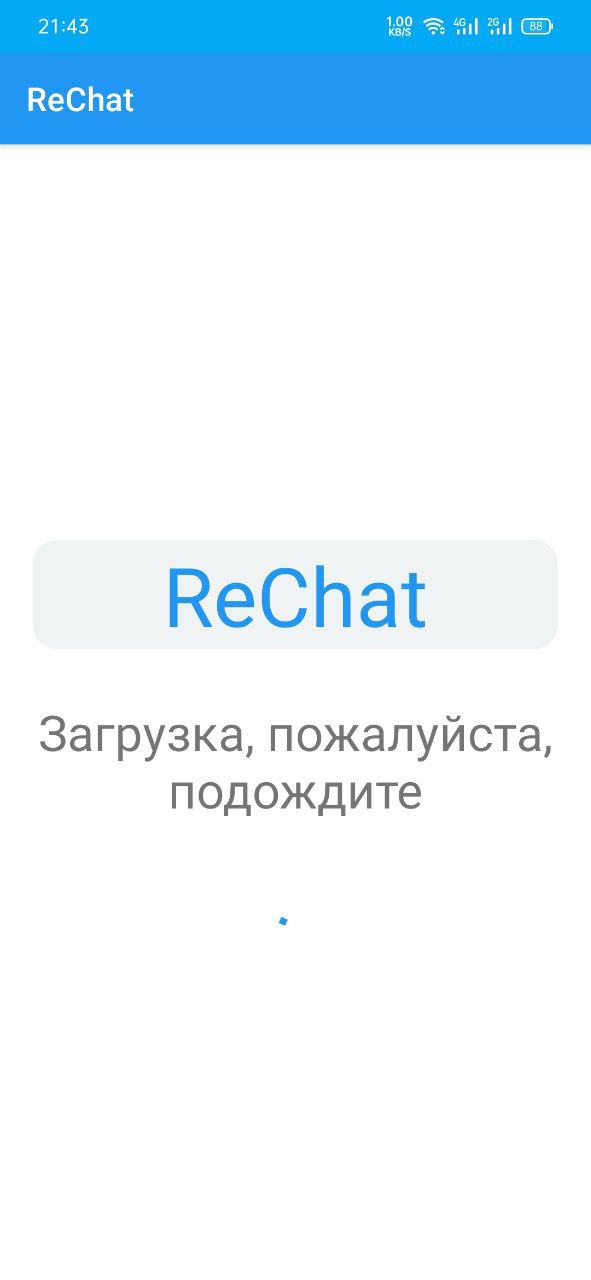


Рисунок 22 – Экран загрузки

Также разработана система сохранения авторизованного аккаунта на устройстве. Иначе говоря, автоматическая авторизация пользователя по токену устройства. При успешной авторизации по логину и паролю происходит генерация сервером уникального токена по системе JSON Web Token. JWT - веб-стандарт, который определяет способ передачи данных о пользователе в формате JSON в зашифрованном виде [13]. После генерации данного токена он сохраняется в базе данных и отправляется пользователю, сохраняясь на его устройстве. При следующем подключении к серверу происходит проверка наличия токена для авторизации на устройстве. Если проверка окончилась успехом, то токен отправляется в запросе на сервер, после чего происходит проверка на совпадение с токеном, хранящимся в базе данных. В другом случае происходит переход на экран авторизации по логину и паролю. Затем клиентское устройство получает ответ от сервера. При удаче начинается процесс синхронизации данных с сервером, в ином случае пользователю необходимо будет провести авторизацию при помощи логина и пароля.

Пароль в базе данных хранится в хешированном виде, поэтому все операции с паролем происходят тоже в его хешированном виде.

Клиентское приложение также имеет систему JWT-токенов, которую он использует в процессах синхронизации данных с сервером. После получения данных от сервера клиентское приложение сверяет токены в полученных данных и хранящийся в памяти, используемый в запросе. Данные будут приняты и обработаны лишь в случае их совпадения.

* + - 1. **Навигация в приложении**

В приложении присутствует три основных раздела: профиль, диалоги и люди. Эти разделы представляют собой фрагменты, которые вставляются в родительский контейнер. Таким образом, происходит смена разделов. Переключение между фрагментами происходит посредством выбора конкретного раздела в нижней панели навигации.

После выбора раздела отрабатывается код, отрисовывающий фрагмент, и производящий замену в родительском контейнере, изображено на рисунке 23.



Рисунок 23 – Замена фрагментов для смены экрана

* + - 1. **Редактирование данных профиля**

Профиль пользователя, изображено на рисунке 24, является одним из основных разделов приложения. Он содержит в себе следующие элементы: изображение профиля, имя пользователя, тег пользователя, онлайн-статус профиля, видимость данных профиля, пол, день рождения, социальный статус, страна, дата регистрации, данные о пользователе, выход из текущего профиля.

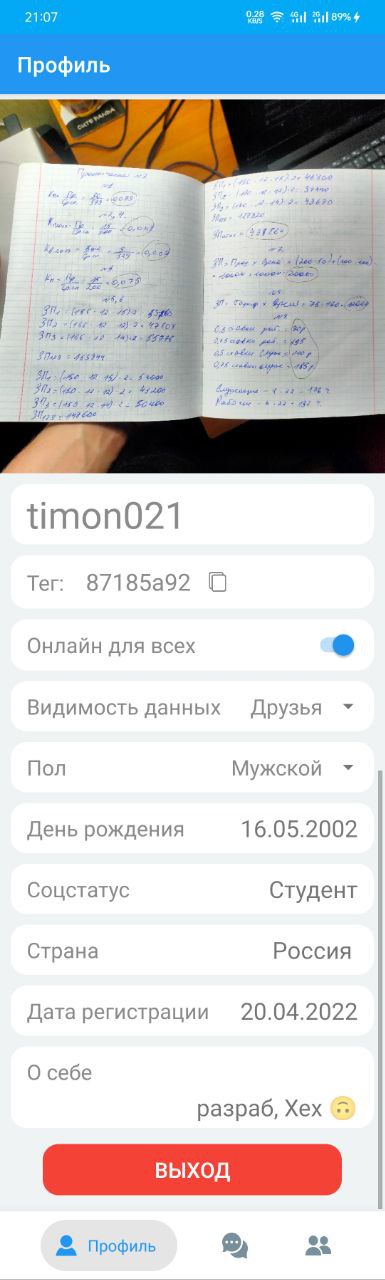


Рисунок 24 – Профиль пользователя

Все поля могут быть отредактированы пользователем по своему усмотрению, исключая данные о дате регистрации профиля в системе и тег пользователя, - это неизменяемые данные.

Тег пользователя – уникальный набор цифр и букв. Зная тег, можно выполнить поиск профиля.

Изображение профиля загружается из локального хранилища устройства. При установке, смене или удалении изображения производится запрос на соответствующую операцию на веб-сервер с хранилищем изображений. В случае успешности операции происходит регистрация изменений в базе данных.

Онлайн-статус, имея два положения, либо отображает текущий профиль в списке онлайн-пользователей для всех остальных, либо нет.

Видимость данных отвечает за приватность отображения данных пользователя. Настройка имеет следующие возможные параметры: все (любой пользователь), друзья, никто. В зависимости от выбранного параметра определенные данные профиля будут отображаться лишь конкретному кругу пользователей. К таковым данным относятся все поля за исключением изображения профиля, имени и тега – эти данные отображаются всегда.

Поле с датой рождения позволяет установить дату рождения пользователя при помощи диалога DatePicker. Результирующие данные диалога будут валидными для создания запроса к серверу лишь при выборе даты до дня выполнения данной операции. Все остальные изменяемые поля с данными имеют ограничения на максимальное количество используемых символов вопреки нагрузки базы данных большим объемом значений. Каждое из них при запросе на редактирование вызывает диалоговое окно с полем для ввода новых данных.

При изменении данных профиля пользователя происходит следующий процесс: заполняется запрос на обновление определенных данных, после он отправляется на сервер, данные сохраняются в базе данных, и клиенту приходит ответ. При успешной операции данные на будут изменены. Пока клиент не получит подтверждение успешного обновления данных от сервера, данные на клиенте не будут обновлены.

Выход из текущего профиля позволяет провести авторизацию в другом или зарегистрировать новый при желании. При деавторизации происходит полная очистка данных о прошлом пользователе.

* + - 1. **Чаты**

Раздел чатов является вторым основным разделом приложения и представляет из себя список чат-каналов. Здесь присутствуют как личные диалоги, так и групповые беседы. Среди всех есть особенный чат-канал, именуемый «GLOBAL CHAT». Этот чат является общим для всех пользователей системы. Никто не может модерировать или как-либо изменять его. На него пользователь подписывается автоматически во время процесса регистрации. В этом чате указываются лишь те авторы сообщений, которые находятся в списке друзей, остальные остаются неизвестными.

Все диалоги сортируются в режиме реального времени по последнему сообщению в них. Также присутствуют индикаторы непрочитанных (иначе говоря - новых) сообщений на чат-каналах, которые также обновляются в режиме реального времени.

Демонстрация раздела чатов представлена на рисунке 25.

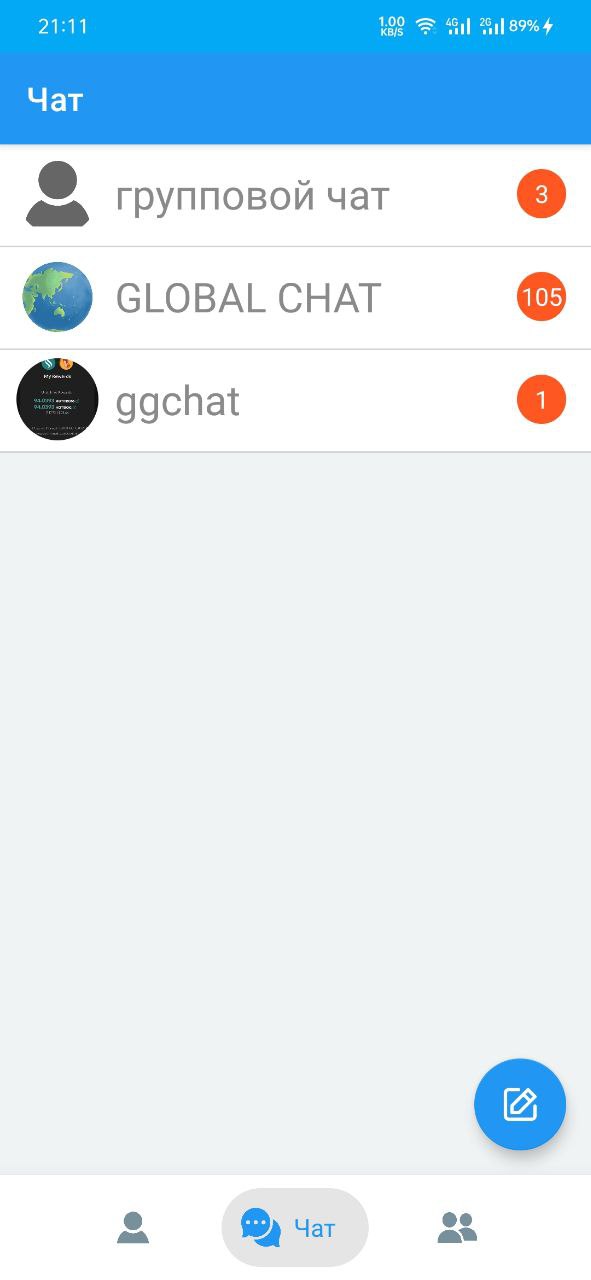


Рисунок 25 – Раздел чатов с индикаторами

Для создания группового чата необходимо нажать на плавающую кнопку в правом нижнем углу в разделе чатов, открывающую дополнительную активность, изображено на рисунке 26.

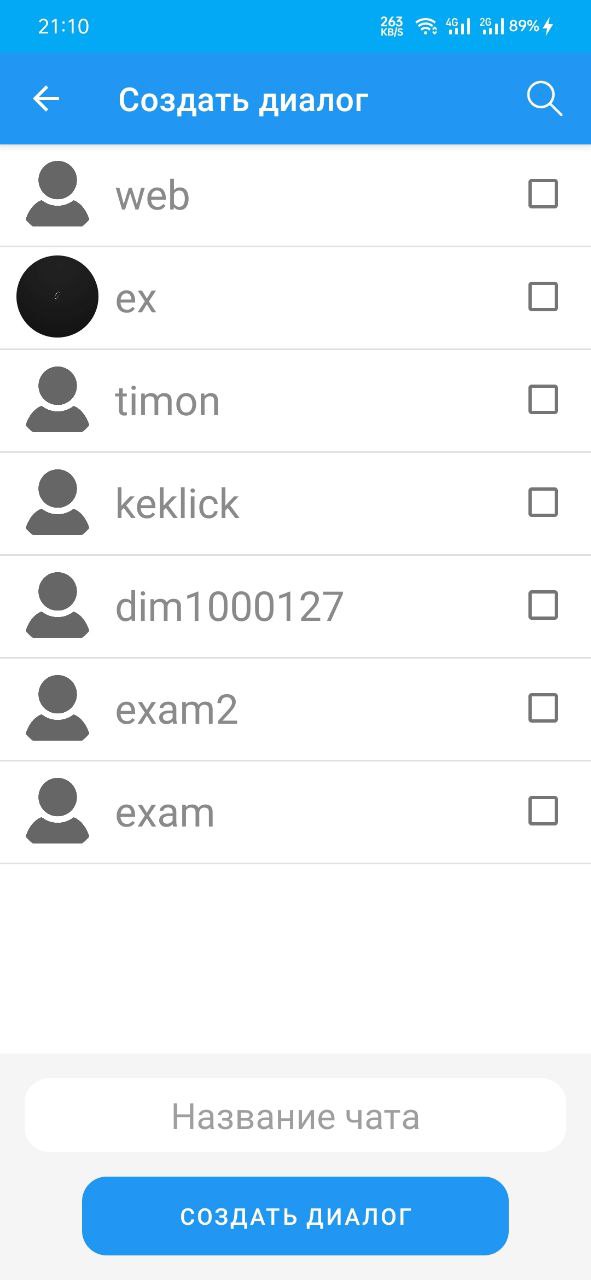


Рисунок 26 – Создание группового диалога

При создании нового группового диалога выводится весь список друзей пользователя (для ускорения поиска присутствует поиск по имени, расположенный в ActionBar). Запрос на создание будет отправлен на сервер лишь после задания названия и выбора хотя бы одного участника. Сервер оповестит каждый клиент, который в свою очередь после этого начнет загружать информацию о новом чате. После создания он отобразится первым пунктом в списке чатов у всех клиентов.

При выборе диалога запускается дополнительная активность, которая восстанавливает сообщения из локальной базы данных. Вся активность имеет срок жизни, пока не будет полностью закрыта возвращением на предыдущий экран.

В зависимости от типа открытого чата будут отличаться дальнейшие возможности, а именно: в личных диалогах при нажатии на информационную часть в ActionBar будет произведено открытие профиля пользователя, с которым был открыт диалог, изображено на рисунке 27; в групповых диалогах, изображено на рисунке 28, информационная часть недоступна для клика, но вместо этого присутствует возможность открыть настройки этого чата и информацию о нем, изображено на рисунке 29.

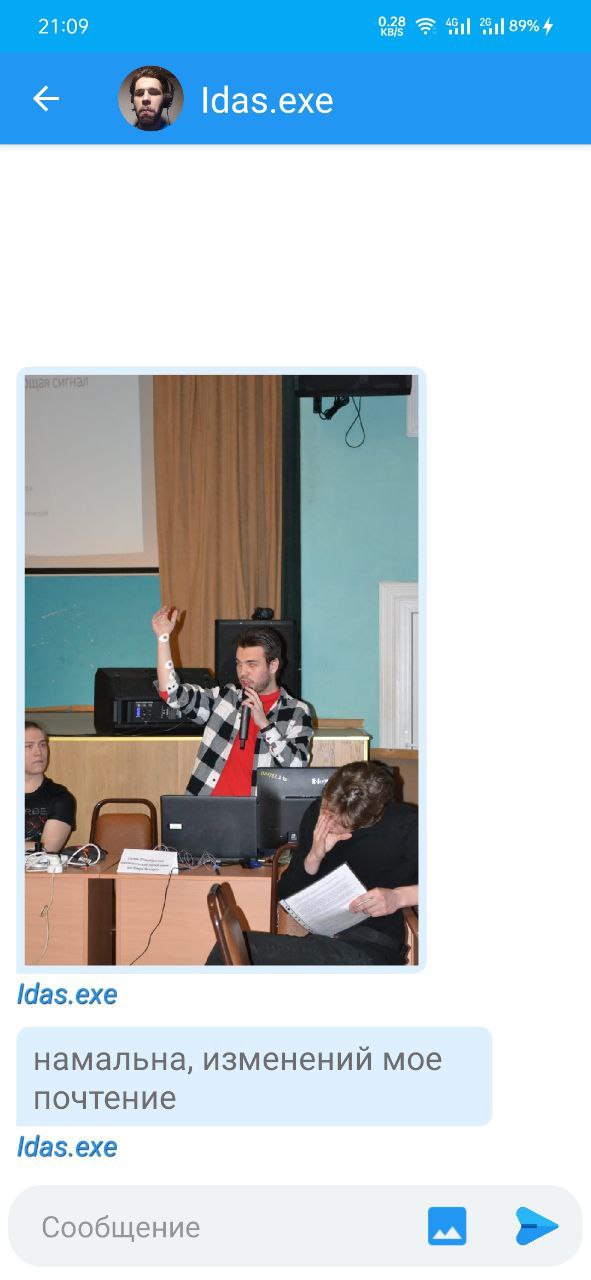


Рисунок 27 – Личный диалог

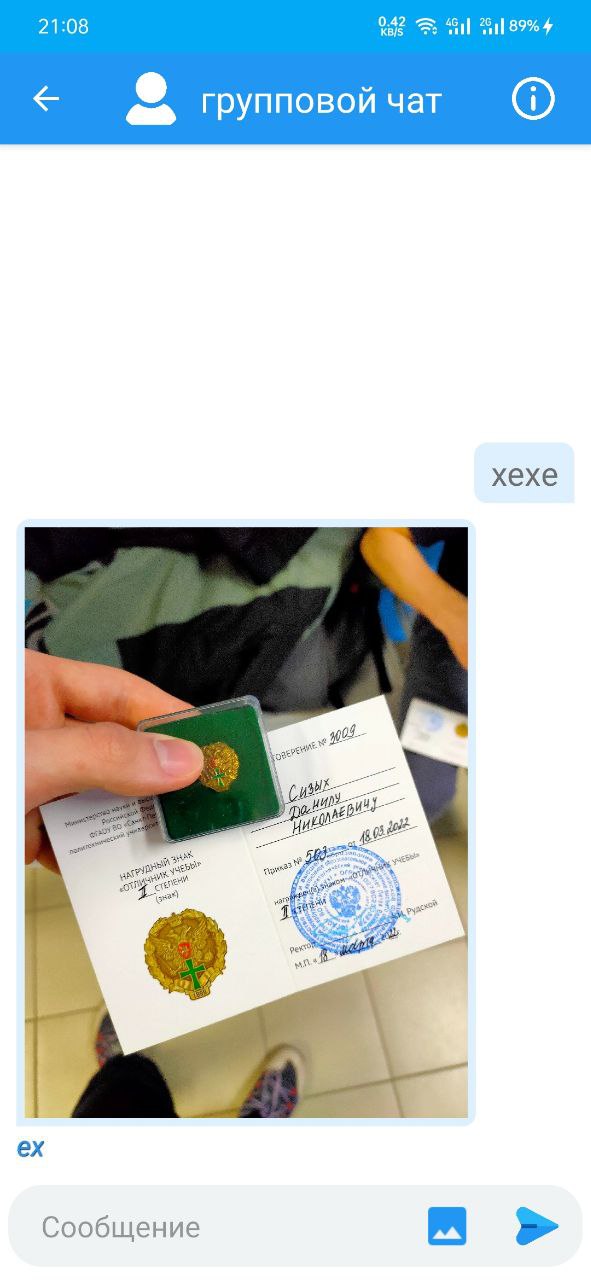


Рисунок 28 – Групповой диалог

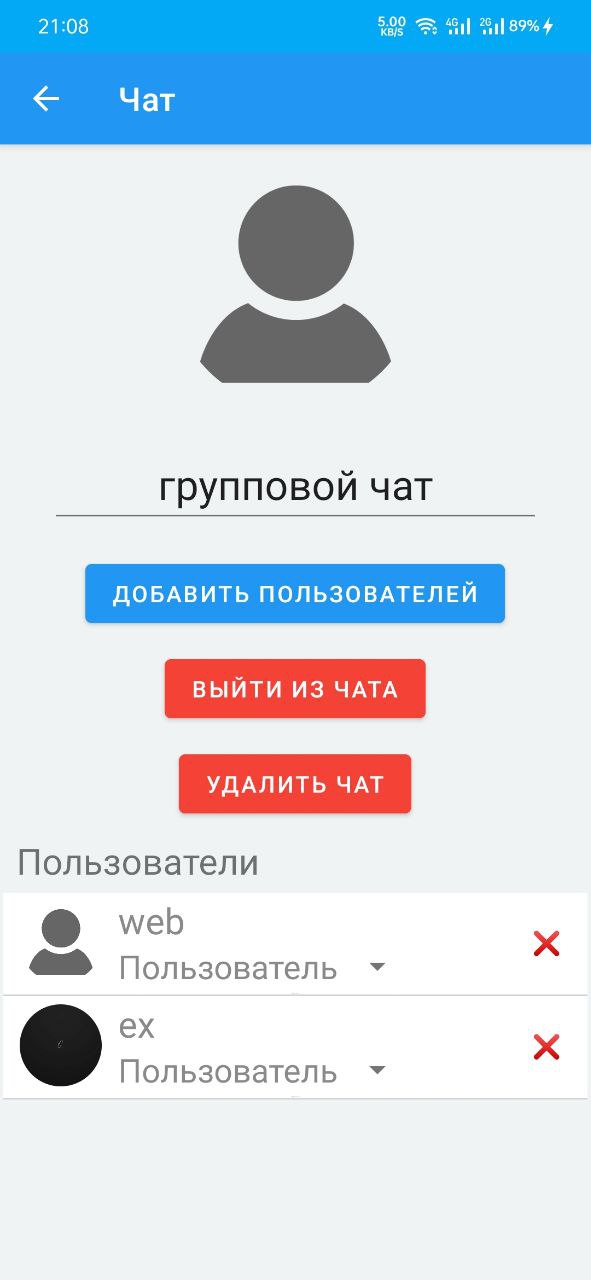


Рисунок 29 – Настройки группового чата

На данном окне настроек чата доступны:

1. смена заглавного изображения чата;
2. смена названия чата;
3. возможность добавить пользователей;
4. возможность выйти из чата;
5. возможность удалить чат для всех.

Любые изменения отображаются у всех пользователей в режиме реального времени. Запрос на изменение посылается на сервер, после успешной операции сервер начинает рассылать всем участникам чата данные об изменениях.

В нижней части экрана отображается список участников с рангом каждого. Каждый ранг имеет определенные права доступа.

Всего в системе существует четыре типа рангов:

1. Читатель – имеет возможность только чтения сообщений;
2. Пользователь – имеет полноценный функционал участника, к которому относится чтение и написание сообщений, возможность приглашать других пользователей в данную групповую беседу.
3. Модератор – участник с привилегиями, который включает права прошлого типа и надбавку над ними в виде возможности смены заглавного изображения чата, редактировать название, выгонять из чата и менять ранг участников, имеющих прошлый тип прав
4. Администратор – создатель групповой беседы, имеет возможность смены ранга любого участника, имеет возможность удалить данный чат полностью со всеми данными о нем для всех участников. Включает права всех других рангов.

Независимо от набора прав любой участник может добровольно покинуть чат.

При приглашении других пользователей в беседу откроется фрагмент-активность на переднем плане, изображено на рисунке 30. На данном экране будет отображен список друзей, которых еще нет в данном чате (добавление новых участников, как и создание группового чата возможно только с пользователями из списка друзей). Для ускорения операции присутствует возможность выполнить поиск нужного пользователя по имени среди списка друзей.

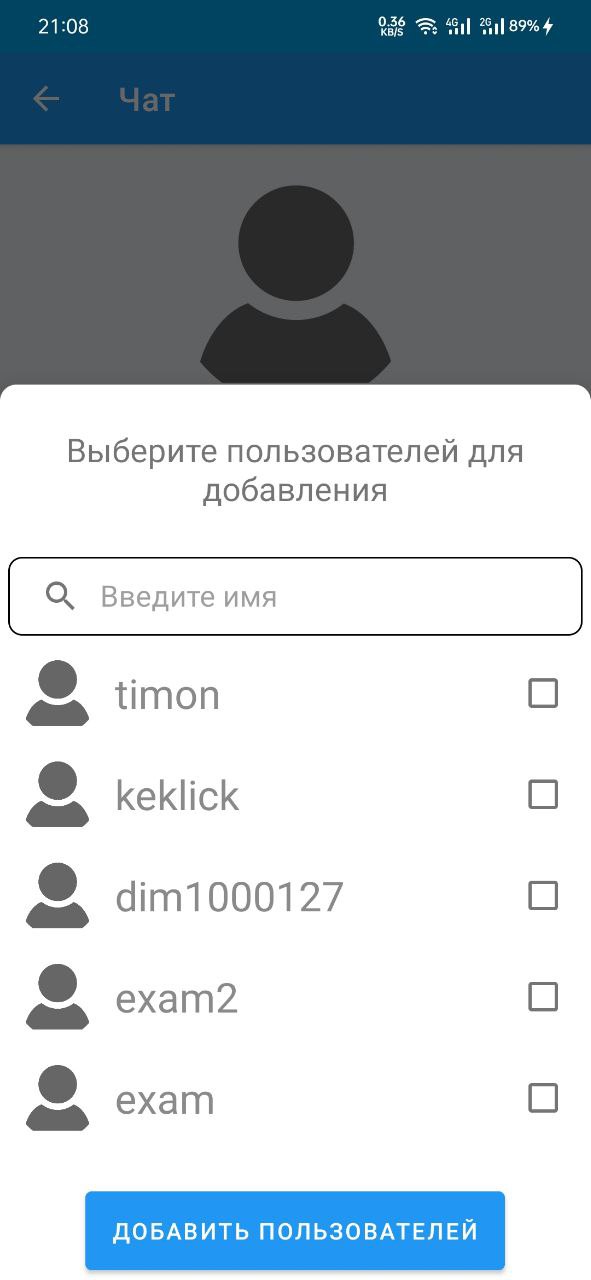


Рисунок 30 – Добавление новых участников в групповой чат

Для отправки сообщения в нижней части экрана существует надлежащий инструментарий для этого. В нижнем поле вводится текст и посылается нажатием кнопки отправки. Помимо текстового содержания сообщение может состоять из изображения. Для отправки изображения его необходимо выбрать из локального хранилища, обозреватель которого открывается нажатием кнопки правее поля для ввода текста. Систематика отправки сообщений никоим образом не выбивается из системы взаимодействия клиента и сервера: формируется запрос, отправляется на сервер, сервер обрабатывает этот запрос, отвечает клиенту и делает рассылки по другим клиентам, которым было направлено сообщение. Сообщение на экране отправителя появляется одновременно с получателем, так как разрешение на добавление он получит лишь после одобрения сервером. Чтобы система могла определять, куда и кем было отправлено сообщение, каждый чат при создании наделяется уникальным идентифицирующим номером, указывающийся при составлении запроса совместно с тегом отправителя.

* + - 1. **Пользователи**

Третьим основным разделом является окно «Люди». На данном экране отображаются: друзья, пользователи, находящиеся онлайн и запросы в друзья. Все перечисленное представляется в виде фрагментов в компоненте RecyclerView, за счет которого реализовано смена вкладок горизонтальными свайпами по экрану, что существенно положительно сказывается на опыте использования приложения.

Вкладка «Друзья» отображает список друзей, изображено на рисунке 31, а на вкладке «Пользователи» находится список всех подключенных к серверу пользователей при условии, что они включили отображение себя на страничке своего профиля, изображено на рисунке 32. На обеих этих вкладках на списочных элементах подле имени пользователя находится кнопка создания личного диалога. Создается именно личный диалог, который будет отличаться по функционалу с групповым диалогом с одним участником. Создать такой диалог можно с любым пользователем, который вышел в статус онлайн, каких-либо ограничений по статусу взаимоотношений нет.



Рисунок 31 – Вкладка «Друзья»

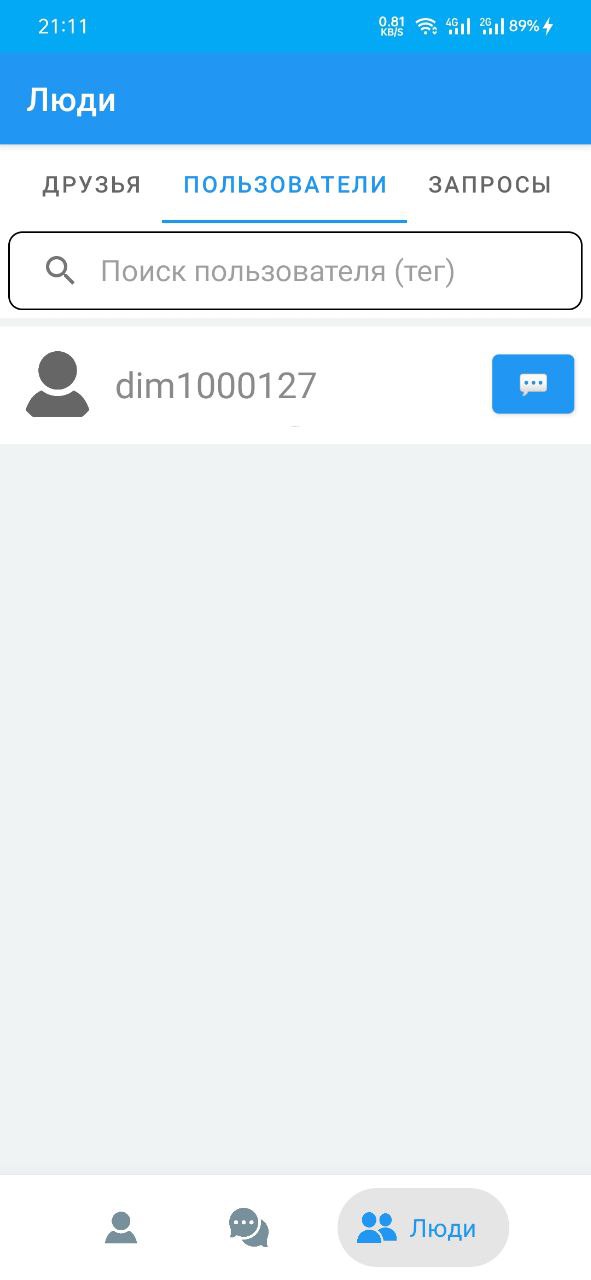


Рисунок 32 – Вкладка «Пользователи»

Вкладка «Запросы» выводит список пользователей, которые отправили запрос на добавление в друзья, изображено на рисунке 33. Имеется возможность просмотреть профиль пользователя, принять запрос или отклонить его. Вся система функционирует следующим образом: имеется три состояния взаимоотношений. Пользователь получил запрос в друзья – код 0, пользователь отправил запрос в друзья – код 1, пользователь находится в друзьях – код 2. При отправке запроса в друзья создаются две записи в базе данных, в которых отражается отношение одного пользователя к другому и наоборот. Отправивший имеет код 1, получивший – код 0. При принятии запроса в обеих записях код сменяется на 2, означающий, что пользователи находятся в друзьях. При отклонении запроса происходит удаление записей из базы данных. Статус взаимоотношений с другим пользователем также отображается в его профиле над главным изображением при открытии, изображено на рисунке 34. Там же имеется возможность сменить его.

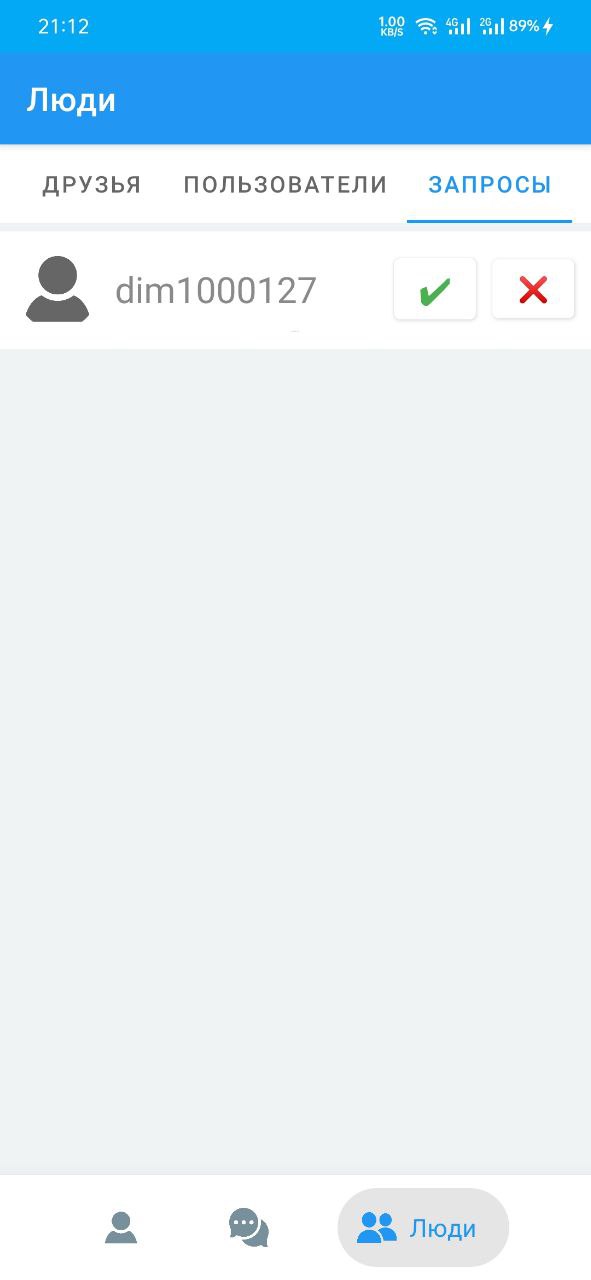


Рисунок 33 – Вкладка «Запросы»



Рисунок 34 – Индикатор взаимоотношений с пользователем

* 1. **Тестирование полученного программного обеспечения**
     1. **Функциональное тестирование**

Функциональное тестирование – это тестирование программного обеспечения в целях проверки реализуемости функциональных требований, то есть способности программного обеспечения в определенных условиях решать задачи, нужные пользователям. Функциональные требования определяют, что именно делает программное обеспечение, какие задачи оно решает. Набор тестов на функциональность представлен в таблице 2.

Таблица 2 – Тестирование клиентского приложения

| **№** | **Название** | **Шаги** | **Ожидаемый результат** | **Результат теста** |
| --- | --- | --- | --- | --- |
| 1 | Регистрация | На странице входа нажать кнопку «Регистрация». Ввести требуемые данные. Согласиться на обработку. Нажать кнопку «Зарегистрироваться». | В базе данных сервера появился новый пользователь,  на экране отобразилось сообщение об успешном результате | Пройден |
| 2 | Авторизация | На странице входа нажать кнопку «Войти». Ввести электронный адрес и пароль. Нажать кнопку «Войти». | Пользователь получил уведомление об успешной авторизации, открылось окно профиля. | Пройден |
| 3 | Автоматическая авторизация | Пройти авторизацию. Закрыть приложение и запустить его заново. | Пользователь автоматически подключится к системе с ранее авторизованным профилем. | Пройден |
| 4 | Смена данных профиля | На странице профиля выбрать и изменить любые данные доступные для редактирования при помощи окна редактуры. | Данные будут успешно изменены. Пользователь получит оповещение об успешной смене данных. | Пройден |
| 5 | Создание диалога | На странице пользователей создать личный чат с любым пользователем путем нажатия на кнопку создания личного чата. | В базе данных сервера появится новый диалог. На экране в разделе чатов отобразится новый личный чат. | Пройден |
| 6 | Создание группового диалога | Открыть окно создания групповых диалогов. Выбрать будущих участников и задать название чата. Нажать кнопку «Создать диалог». | В базе данных сервера появится новый диалог. На экране в разделе чатов отобразится новый групповой чат. | Пройден |
| 7 | Отправка сообщения | Выбрать чат. Написать в поле для ввода сообщение. Нажать кнопку отправки сообщения. Проверить наличие сообщения у другого пользователя. | На экране появится отправленное сообщение. На экране получателя появится полученной сообщение | Пройден |
| 8 | Редактирование данных созданного группового чата | Открыть групповой чат. Нажать на кнопку для открытия информации по чату. Выбрать поле с названием чата. Отредактировать название. Отправить изменения нажатием кнопки для отправки. | В базу данных сервера внесутся изменения. На экране отобразится новое название чата. | Пройден |
| 9 | Открытие чужого профиля | На странице с пользователями или с друзьями нажать на любого человека. | Откроется окно с профилем выбранного пользователя. | Пройден |
| 10 | Добавление в друзья | На странице профиля другого пользователя отправить запрос. Дождаться принятия запроса. | Выбранный пользователь появится в списке друзей в разделе «Люди». | Пройден |

Программа была проверена на наличие функционала, описанного в задачах. В ходе испытаний программа справилась с возложенными на нее задачами и испытания завершились успехом. В каждом испытании приложение корректно реагировало на совершаемые над ним действия и выдавало корректный результат.

* + 1. **Нагрузочное тестирование**

Нагрузочное тестирование - подвид тестирования производительности, сбор показателей и определение производительности и времени отклика программно-технической системы или устройства в ответ на внешний запрос с целью установления соответствия требованиям, предъявляемым к данной системе (устройству).

Нагрузочное тестирование предполагает одновременный запуск большого числа тестовых клиентов, для чего обычно используется много тестовых стендов (компьютеров). Это позволит на базовом уровне определить возможности сервера, информацию о производительности и пользовательском опыте.

Для создания ситуации полноценного использования программного обеспечения были созданы тестовые стенды и приглашены участники для тестирования. Общее количество одновременно подключенных клиентов достигало десяти единиц.

Тестеры проводили полноценное использование созданного программного обеспечения, длившееся в течение дня. В ходе этих действий продукт оставался стабильным и испытания завершились успехом проверки системы.

* 1. **Оценка качества разработанного программного обеспечения**
     1. **Реализуемость и принципиальная осуществимость**

В ходе выполнения данной работы были выполнены все поставленные задачи и реализованы все необходимые функции.

* + 1. **Конкурентоспособность**

Спроектированное и разработанное программное обеспечение в ходе данной работы, относительно имеющегося реализованного функционала и других различных требований, а также возможных затрат имеет полную конкурентную способность с другими подобными продуктами, представленными на рынке.

* + 1. **Жизнеспособность**

Спроектированное и разработанное программное обеспечение в ходе данной работы имеет малые возможности очень крупного тиражирования на текущем этапе проектирования всей системы «ReChat», но эти возможности существенно возрастают при продолжении разработки проекта.

Отсутствие финансирования проекта не сказывается на его дальнейшем развитии из-за первоначальной идейности создания на основе собственной инициативы и других оснований для разработки.

* + 1. **Востребованность проекта**

Спроектированное и разработанное программное обеспечение в ходе данной работы имеет востребованный характер среди аудитории, которая является заказчиком, вследствие текущих растущих тенденций общества по использованию средств виртуального общения.

* + 1. **Устойчивость проекта**

Спроектированное и разработанное программное обеспечение в ходе данной работы имеет в наличии поддержку проекта на местном уровне, а также возможность продолжения деятельности и дальнейшего развития проекта с опорой на местные ресурсы.

* 1. **Текст программы**

Текст программы представляет собой запись программы на исходном языке программирования с необходимыми комментариями. Текст программы представляет собой документ, выполненный машинным способом, и приведен в приложении В.

* 1. **Руководство оператора**

Руководство оператора представляет собой документ, выполненный машинным способом, и приведен в приложении Г.

1. **Экономическая часть**
   1. **Область применения программного продукта и его преимущества перед аналогичным программным продуктом**

Система предназначена для пользования сотрудниками компании ООО «Омега», которые являются Заказчиком, а также другим заинтересованным кругом лиц для пользования программным обеспечением по созданию виртуального пространства в личных целях.

Достоинствами системы являются возможности создания неограниченного количества профилей пользователей без принуждения подтверждения каких-либо данных и высокие показатели анонимности их личности.

* 1. **Трудоемкость разработки программного продукта, квалификация исполнителя и его оклад**

Трудоемкость разработки можно определить в таблице 3. Строка «Всего» отображает общую трудоемкость разработки.

Таблица 3 – Трудоемкость разработки программного продукта

| **Наименование этапа** | **Условное обозначение** | **Трудоемкость выполнения этапа, час** |
| --- | --- | --- |
| Описание задания | То | 15 |
| Разработка UML - диаграмм | Тд | 30 |
| Разработка Базы данных | Тбд | 50 |
| Проектирование программы | Тпп | 250 |
| Отладка программы | Топ | 20 |
| Тестирование программы | Тп | 20 |
| Оценка качества программы | Тоц | 17 |
| Оформление документации | Тд | 100 |
| Всего | Тобщ | 502 |

Разработчики программного продукта представлены в таблице 4.

Таблица 4 – Разработчики программного продукта

|  |  |  |  |
| --- | --- | --- | --- |
| **Исполнители** | **Оклад, руб.** | **Часовая тарифная ставка, руб./час** | **Количество сотрудников** |
| Разработчик-программист | 60 000 | 340,90 | 1 |
| Руководитель проекта | 80 000 | 454,54 | 1 |

Часовая тарифная ставка ЧТС, руб./час., определяется исходя из месячного оклада, количества рабочих дней в месяце и продолжительности рабочего по формуле:

(1)

где Ом – оклад исполнителя в месяц, руб./мес.;

Д – количество рабочих дней в месяце (22 раб. дня);

Тс – продолжительность рабочего дня (8часов).

В таблице 5 показана стоимость технических средств для разработки системы.

Таблица 5 – Стоимость технических средств разработки

|  |  |  |  |
| --- | --- | --- | --- |
| **Наименование компонента** | **Цена, руб.** | **Количество, шт.** | **Стоимость, руб.** |
| Ноутбук Lenovo IdeaPad L3 15IML05 | 53990 | 1 | 53990 |
| МФУ лазерное HP DeskJet 2320 | 3500 | 1 | 3500 |
| Мышь MotoSpeed v70 | 1500 | 1 | 1500 |
| Итого: | | | 58990 |

В таблице 6 представлены затраты на расходные материалы.

Таблица 6 – Планируемые затраты на расходные материалы (Р)

| **Затраты** | **Стоимость** | **Количество** | **Сумма, руб.** |
| --- | --- | --- | --- |
| Интернет | 550 руб./месяц | 3 месяца | 1650 |
| Электричество | 4,82 руб./КВт\*ч | 1850 КВт\*ч | 8917 |
| Бумага | 500 руб. | 1 упаковка | 500 |
| Ручка | 50 руб. | 2 штуки | 100 |
| Заправка МФУ | 3 000 руб. | 2 раз | 6000 |
| Итого: | | | 17167 |

* 1. **Расчет затрат на разработку**

Исходные данные связанные с разработкой программного продукта приведены в таблице 7.

Таблица 7 – Исходные данные

| **Наименование** | **Обозначение** | **Значение** |
| --- | --- | --- |
| Оклад разработчика | Ор | 140 000 руб. |
| Время разработки | Трп | 3 месяца |
| Машинное время разработки | Тмч | 2,5 месяца |
| Коэффициент дополнительной заработной платы | Кд | 0,16 |
| Коэффициент страховых взносов | Кст | 0,3 |
| Количество единиц техники | Q | 1 шт. |
| Себестоимость содержания техники | См/ч | 18 руб./час |
| Коэффициент готовности техники | Кгт | 0,95 |
| Число рабочих дней в месяце | ЧРД | 22 дня |
| Продолжительность смены | Тст | 8 часов |
| Коэффициент сменности | Ксм | 1 |
| Коэффициент транспортных расходов | Кт | 0 |
| Коэффициент накладных расходов | Кнр | 0,54 |

Расчет полных затрат на разработку проектного решения (программного продукта) осуществляется по формуле:

Зрп = Зот + Зст + Зэвм + Зсп + Зрм + Знр, (2)

где Зот – затраты на оплату труда разработчика (разработчиков), руб.;

Зст – страховые взносы по оплате труда во внебюджетные фонды, руб.;

Зэвм – затраты, связанные с содержанием вычислительной техники, руб.;

Зсп – затраты на технические средства, руб.;

Зрм – затраты на расходные материалы, необходимые при разработке программного продукта, руб.;

Знр – затраты по накладным расходам, приходящиеся на разработку программного продукта, руб.

Зрп = 487200 + 146160 + 7524 + 58990 + 17167 + 263088 = 980129 руб.

* + 1. **Затраты на оплату труда разработчиков (Зот), руб.**

Размер фонда оплаты труда разработчиков (Зот) рассчитывается по формуле:

Зот = Ор Трп (1 + Кд) , (3)

где Ор - месячный оклад разработчика проектного решения, руб./мес.;

Трп - время разработки проектного решения разработчиком, мес.(час.);

Кд – коэффициент дополнительной заработной платы разработчика.

Зот = 140000 3 (1 + 0,16) = 487200 руб.

* + 1. **Затраты по страховым вносам (Зст), руб.**

Сумма страховых взносов определяется по формуле:

Зст = Кст Зот , (4)

где Кст - коэффициент страховых взносов для расчета отчислений во внебюджетные фонды.

Зст = 0,3 487200 = 146160 руб.

* + 1. **Затраты по содержанию ЭВМ (Зэвм), руб.**

Затраты, связанные с эксплуатацией и содержанием ЭВМ, определяются по формуле:

Зэвм = Тмрп Кгт Q Cм/ч , (5)

где Тмрп - машинное время на разработку проектного решения, час;

Кгт - коэффициент готовности техники;

Q - количество условных единиц, используемой техники;

Cм/ч – стоимость машино-часа, эксплуатации оборудования, руб. в час.

Зэвм = 440 0,95 1 18 = 7524 руб.

Так как машинное время может измеряться в месяцах, а себестоимость машино-часа за один час, то машинное время необходимо перевести в часы.

Перевод рабочего времени в часы осуществляется по формуле:

Тмрп = Тмч Чрд Тсм Ксм , (6)

где Тмч – рабочее время в месяцах;

Чрд – число рабочих дней в месяце;

Тсм – продолжительность рабочей смены;

Ксм – количество рабочих смен.

Тмрп = 2,5 22 8 1 = 440 час.

* + 1. **Затраты на расходные материалы (Зрм), руб.**

Затраты на расходные материалы необходимые для разработки проектного решения определяются по формуле:

Зрм = , (7)

затраты на расходные материалы, руб.;

Кт - коэффициент транспортных расходов.

Зрм = = 17167 руб.

* + 1. **Затраты по накладным расходам (Знр), руб.**

Затраты по накладным расходам определяются по формуле:

Знр = Кнр Зот , (8)

где Кнр - коэффициент накладных расходов, принимается для расчета по данным предприятия;

Зот- затраты по оплате труда, руб.

Знр = 0,54 487200 = 263088 руб.

Произведя вычисления было выявлено, что полные затраты на разработку составляют 980129 рублей. Промежуточные результаты вычислений представлены в таблице 8.

Таблица 8 – Результаты вычислений

| **Наименование** | **Обозначение** | **Значение** |
| --- | --- | --- |
| Оплата труда | Зот | 487200 руб.; |
| Страховые взносы | Зст | 146160 руб.; |
| Содержание ЭВМ | Зэвм | 7524 руб.; |
| Затраты на специальные программы | Зсп | 58990 руб.; |
| Расходные материалы | Зрм | 17167 руб.; |
| Накладные расходы | Знр | 263088 руб.; |
| Итого затрат на разработку | Зрп | 980129 руб. |

* 1. **Расчет цены и прибыли**

Исходные данные связанные с разработкой программного продукта приведены в таблице 9.

Таблица 9 – Исходные данные

| **Наименование** | **Обозначение** | **Значение** |
| --- | --- | --- |
| Коэффициент рентабельности | Кр | 0,3 |
| Коэффициент налога на добавочную стоимость | Кндс | 0,2 |
| Ставка налога | Кнп | 0,2 |
| Стоимость затрат на разработку | Квпр | 980129 руб. |

Цена программного продукта, который разработан одной организаций по заказу другой и не предназначен для тиражирования, определяется по формуле:

Цпп = Квпр + Ппл + НДС , (9)

где Ппл - планируемая прибыль, руб.;

НДС - налог на добавленную стоимость, руб.

Ппл = Квпр Кр (10)

Ппл = 980129 0,3 = 294038,70 руб.

НДС = (Квпр + Ппл) Кндс (11)

НДС = (980129 +294038,70) 0,2 = 254833,54 руб.

Цпп = 980129 + 294038,70 + 254833,54 = 1529001,24 руб.

Каждое предприятие с полученной прибыли перечисляет государству налог на прибыль. На сегодня ставка налога 20% от полученной прибыли, и определяется по формуле:

НП = Ппл Кнп (12)

НП = 294038,70 0,2 = 58807,74руб.

Чистая прибыль, остающаяся в распоряжении предприятия

ЧП = Ппл – НП (13)

ЧП = 294038,70 –58807,74 = 235230,96 руб.

Поступление в бюджет складываются из налога на прибыль и НДС

ПБ = НП + НДС (14)

ПБ = 58807,74 + 254833,54= 313641,28 руб.

Таблица 10 – Результаты вычислений

|  |  |  |
| --- | --- | --- |
| **Наименование** | **Обозначение** | **Значение** |
| Плановая прибыль | Ппл | 294038,70 руб. |
| Налог на добавочную стоимость | НДС | 254833,54 руб. |
| Цена программного продукта | Цпп | 1529001,24 руб. |
| Налог на прибыль | НП | 58807,74 руб. |
| Чистая прибыль | ЧП | 235230,96 руб. |
| Поступление в бюджет | ПБ | 313641,28 руб. |

Произведя вычисления было выявлено, что чистая прибыль будет составлять 235230,96 рублей. Промежуточные результаты вычислений представлены в таблице 10.

ВЫВОД: На основании произведенных расчетов, можно сделать вывод, что на разработку продукта будет затрачено 980129 рублей. Цена готового продукта будет равна 1529001,24 рубля, что в свою очередь позволит получить чистую прибыль в размере от 235230,96 рублей. Цена аналогичного проекта составляет от 1700000 до 2000000, что делает проект экономически выгодным и способным конкурировать на данном сегменте рынка.

1. **Техника безопасности и охрана труда**
   1. **Анализ условий труда программиста, работающего в компании ООО «Омега»**

Рабочее место программиста — это его рабочий кабинет, либо часть помещения, в котором он проводит большую часть работы. Совершенно очевидно, что чем лучше приспособлено рабочее место для выполнения функций программиста, тем производительнее и эффективнее будет его трудовая деятельность.

Программист имеет собственное рабочее место, оборудованное ПЭВМ. Рабочий стул (кресло) должен быть подъемно-поворотным и регулируемым по высоте и углам наклона сиденья и спинки, а также – расстоянию спинки до переднего края сиденья.

Рабочая поверхность стола сделана из дерева. На столе размещены ЭВМ, монитор и принтер. Площадь поверхности стола предполагает размещение монитора, устройств ввода/вывода и рабочей зоны с местом расположения оперативной документации.

Кабинет, в котором расположено рабочее место, представляет собой помещение площадью 12 м2, длиной 4 м, шириной 3 м и высотой 2,8 м.

Для создания благоприятных условий для зрительного восприятия кабинет оформлен в светло-бежевом цвете.

Для создания и поддержания независимо от наружных условий оптимальных значений температуры, влажности, чистоты воздуха, в холодное время используется водяное отопление, в теплое время года применяется кондиционирование воздуха.

Режим работы программиста - односменный с пятидневной рабочей неделей и нормированным рабочим днем. Продолжительность ежегодного отпуска - 28 календарных дней.

При идентификации вредных производственных факторов было установлено, что опасными могут являться:

* шум и вибрация, источником которого является оргтехника;
* электромагнитные поля и излучения от дисплея ПК;
* статическое электричество, накапливаемое на клавиатуре, дисплее, корпусе системного блока;
* электрический ток в электрических сетях;
* запыленность рабочей зоны, так влажная уборка рабочей поверхности проводиться не ежедневно, а только по мере загрязнения;
* статические перегрузки, так как работа программиста связана с неизменной статической позой, иногда неудобной позе (работа за ПК), что может привести к искривлению позвоночника, остеохондрозу, застою в органах малого таза;
* перенапряжение анализаторов, в большей степени зрительного, что может привести к различным нарушениям зрения (снижение остроты, спазм и уменьшение запаса аккомодации), помимо этого повышение зрительной нагрузки может приводить к общему утомлению, возникновению головной боли, ухудшению самочувствия;
* умственное перенапряжение, так как в процессе деятельности программист анализирует и перерабатывает огромное количество административных сетей, разрабатывает программные продукты, поддерживает технику в рабочем состоянии.

В целях выявления вредных и (или) опасных производственных факторов и осуществления мероприятий по приведению условий труда в соответствие государственными нормативными требованиями охраны трудаежегодно проводится аттестация рабочих мест по условиям труда - оценка условий труда на рабочих местах. Аттестации подлежат все имеющиеся в организации рабочие места.

При аттестации рабочих мест проводят оценку условий труда, гигиеническую оценку и оценку травмоопасности рабочих мест.

При этом учитывают наличие средств коллективной защиты, обеспеченность работников средствами индивидуальной защиты и определяют эффективность этих средств.

Согласно санитарным правилам СП 2.2.3670-20 "Санитарно-эпидемиологические требования к условиям труда", дисплей должен располагаться на расстоянии 60-70 см, но не менее 50 см от глаз. Между боковыми поверхностями мониторов должно быть не менее 1,2 м. При использовании жидкокристаллических дисплеев на каждый компьютер должно приходиться не менее 4,5 м2 площади. На дисплей ПК не должен попадать прямой солнечный свет во избежание бликов и повышенной нагрузки на зрение.

Поэтому в кабинете, где работает программист, стоит только один компьютер, а комплектующая оргтехника расположена достаточно далеко непосредственно от самого специалиста. Это обусловлено тем, что все копировальные аппараты, принтеры и факс в периоды своей работы выделяют не только вредные токсичные газы, но и шумы и вибрацию. Данные факторы оказывают гораздо меньшее негативное влияние, нежели излучение от компьютера, но также являются вредными и могут нанести вред здоровью сотрудника.

Для снижения напряжения с глаз, а также для снятия мышечного напряжения во время рабочего дня проводится проветривание в течении 10 – 15 минут каждые 1,5 часа.

* 1. **Расчет искусственного освещения рабочего места программиста на предприятии ООО «Омега»**

Через зрительный анализатор человек получает около 80% из общего объема информации. Качество поступающей информации во многом зависит от освещения: неудовлетворительное количественно или качественно, оно не только утомляет зрение, но и вызывает утомление организма в целом.

Кроме того, нерациональное освещение может явиться причиной травматизма: плохо освещенные опасные зоны, слепящие источники света и блики от них, резкие тени ухудшают видимость настолько, что вызывают полную потерю ориентировки работающих, снижают производительность труда и увеличивают брак продукции. Поэтому необходимо достаточное искусственное освещение для рабочего места.

Условия работы для расчета показателя освещенности рабочего места программиста являются:

* помещение размером 12 квадратных метров;
* высота помещения 2,8 метра, длина 4 метра, ширина 3 метров;
* лампы люминесцентные (дневного света) в количестве трех штук, новые;
* крупногабаритная мебель отсутствует.

Расчет общего равномерного искусственного освещения горизонтальной рабочей поверхности выполняется методом коэффициента использования светового потока. Световой поток (лм) одной лампы рассчитывается по следующей формуле:



где Ен - нормируемая минимальная допустимая освещенность по СП 52.13330.2016, для помещения 2000 лк;

*S -* площадь освещаемого помещения, м2;

z - коэффициент неравномерности освещения, *z* = 1,1;

K - коэффициент запаса, учитывающий снижение освещенности в процессе эксплуатации вследствие загрязнения и старения ламп и светильников, а также снижения отражающих свойств поверхностей помещения для кабинета будет равен 1,3;

Nc - число светильников в помещении;

*-* коэффициент затенения, обычно 1;

= i - коэффициент использования светового потока

Коэффициент использования светового потока, давший название методу расчета, определяется по индексу помещения i в зависимости от типа светильника и коэффициентов отражения света от потолка, стен и пола:



где А, В - длина и ширина помещения в плане, м;

Нс - высота подвеса светильников над рабочей поверхностью, м.

Согласно СП 52.13330.2016 величина полученного светового потока достаточно для данного помещения.

* 1. **Электробезопасность на предприятии ООО «Омега»**

При поступлении на работу сотрудник должен пройти инструктаж по технике безопасности и электробезопасности. Работника знакомят с основными правилами по технике безопасности, предлагают внимательно прочитать действующие на предприятии инструкции, поясняя при этом отдельные правила и требования.

Инструктаж по технике безопасности при выполнении конкретной работы проводит руководитель соответствующего производственного участка, показывая инструктируемому правильные безопасные приемы работы.

Повторный инструктаж проводится для рабочих независимо от их квалификации, стажа и опыта работы, не реже 1 раза в 6 месяцев по программе инструктажа на рабочем месте.

При нарушении работающими правил и инструкций по технике безопасности, технологической и производственной дисциплины, а также в случаях изменения технологического процесса или вида работы проводят дополнительные инструктажи.

Все инструктажи оформляются записями в специальном журнале с указанием номеров или шифров инструкций. Журнал о проведении инструктажа хранится у руководителя подразделения.

Работник на рабочем месте должен соблюдать общие меры электробезопасности. При этом запрещается:

* прикасаться к задней панели системного блока при включенном питании;
* допускать попадание влаги на поверхность системного блока, монитора, рабочую поверхность клавиатуры, дисководов, принтеров и др. устройств;
* производить самостоятельно вскрытие и ремонт оборудования;
* пользоваться неисправными электроприборами и электропроводкой;
* ремонтировать электроприборы самостоятельно;
* подвешивать электропровода на гвоздях, металлических и деревянных предметах, перекручивать провод, закладывать провод и шнуры на водопроводные трубы и батареи отопления, вешать что-либо на провода, вытягивать за шнур вилку из розетки;
* прикасаться одновременно к персональному компьютеру и к устройствам, имеющим соединение с землей (радиаторы отопления, водопроводные краны, трубы и т.п.);
* пользоваться самодельными электронагревательными приборами и электроприборами с открытой спиралью;
* наступать на переносимые электрические провода, лежащие на полу.

При работе с электроприборами и оргтехникой (персональные компьютеры, принтеры, сканеры, копировальные аппараты, факсы, бытовые электроприборы, приборы освещения):

* автоматические выключатели и электрические предохранители должны быть всегда исправны;
* изоляция электропроводки, электроприборов, выключателей, штепсельных розеток, ламповых патронов и светильников, а также шнуров, с помощью которых включаются в электросеть электроприборы, должны быть в исправном состоянии;
* электроприборы необходимо хранить в сухом месте, избегать резких колебаний температуры, вибрации, сотрясений;
* для подогрева воды пользоваться сертифицированными электроприборами с закрытой спиралью и устройством автоматического отключения, с применением несгораемых подставок.
  1. **Пожарная безопасность на предприятии ООО «Омега»**

Инструктаж по пожарной безопасности проводится по программе, разработанной инженером по охране труда организации ООО «Омега», с учетом требований стандартов, правил, норм и инструкций о мерах пожарной безопасности. Продолжительность инструктажа устанавливается в соответствии с утвержденной программой. Инструктаж по пожарной безопасности, как правило, проводится совместно с инструктажем по технике безопасности и в те же сроки.

Инструктаж по пожарной безопасности проходят все работники организации, независимо от их образования, стажа работы по данной профессии или должности, временные работники, командированные, обучающиеся и студенты, прибывшие на практику.

О проведении инструктажа по пожарной безопасности работник, проводивший инструктаж, делает запись в журнале учета проведения инструктажей по пожарной безопасности, с обязательной подписью инструктируемого и инструктирующего. При регистрации внепланового инструктажа по пожарной безопасности указывают причину его проведения.

Все работники организаций должны допускаться к работе только после прохождения противопожарного инструктажа, а при изменении специфики работы проходить дополнительное обучение предупреждению и тушению возможных пожаров в порядке, установленном руководителем.

На проведение первичного противопожарного инструктажа необходимо отводить не менее 1 ч. Инструктируемые должны ознакомиться:

* с действующими на объекте правилами пожарной безопасности и инструкциями;
* с производственными участками, наиболее опасными в пожарном отношении, где запрещается курить, применять открытый огонь;
* с возможными причинами возникновения пожара и мерами его предупреждения;
* с практическими действиями в случае возникновения пожара - вызов пожарной помощи, использование первичных средств пожаротушения, место расположения ближайшего телефона и ознакомление с правилами поведения в случае возникновения пожара, эвакуации людей и материальных ценностей.

При первичном инструктаже инструктирующий обязан рассказать о производственных установках с повышенной пожарной опасностью, мерах предотвращения пожаров и загораний, указать место курения, ознакомить вновь поступившего с имеющимися на объекте средствами пожаротушения, показать ближайший телефон и объяснить правила поведения в случае возникновения пожара.

Проведение противопожарного инструктажа в обязательном порядке должно сопровождаться практическим показом способов использования имеющихся на объекте средств пожаротушения (противогазы, респираторы, огнетушители и т.д.).

В итоге, на предприятии ООО «Омега» техника пожарной безопасности соблюдена согласно СП 4.13130.2013. Соблюдение рассмотренных в данном разделе мероприятий по охране труда и технике безопасности на предприятии ООО «Омега» позволяет снизить утомляемость и травматизм, повысить производительность труда, обеспечить комфортные условия трудовой деятельности специалиста, работающего в должности программиста.

# **ЗАКЛЮЧЕНИЕ**

Результатом проделанной работы в рамках выпускной квалификационной работы является программный продукт, состоящий из программы клиента, сервера и сервера базы данных. Все это образует целостную полноценно функционирующую систему «ReChat».

В ходе выполнения данной работы была проанализирована предметная область, рассматривающая виртуальное пространство, как основное средство общения в настоящее время, также были рассмотрены аналогичные проекты в сети Интернет. Анализ послужил основой для проектирования системы «ReChat»: основных ключевых моментов, особенностей взаимодействия частей программного обеспечения, правил функционирования клиент-серверных приложений. Также для получения удовлетворительного продукта были изучены требования конечного пользователя по функциональности будущего приложения, которые были успешно реализованы. Полученный продукт прошел тестирование, в ходе которого не выявилось проблем, препятствующих его функционированию. Таким образом, поставленные задачи были решены, а цель - достигнута.

Преимуществами разработанного программного продукта являются:

1. возможность создания различных профилей одним пользователем без принудительной верификации, что ведет к высокой анонимности;
2. не уступающая конкурентам функциональность в соответствии с предполагаемыми экономическими показателями;
3. скорость и гибкость работы, последняя из которых достигается за счет ориентированности на разные клиентские платформы;
4. лаконичный и удобный дизайн, соответствующий современным меркам, и ведущий к легкости использования.

Использование программы позволит сотрудникам компании ООО «Омега» иметь вариативность в средствах общения между собой, и организовывать собственные личные закрытые пространства для общения.

Также в рамках данного проекта выпускной квалификационной работы была выполнена оценка экономической эффективности продукта и разработаны мероприятия по охране труда.

Ключевым моментом выполнения этой работы стал новый полученный опыт из множества сфер, начиная от разработки приложений и заканчивая оценкой экономической эффективности. Также были закреплены знания о большом стеке языков программирования, полученные в ходе обучения, навыки по проектированию проекта, моделированию требований, проведения тестирования и составления документации.

# **СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ**

Окулов, С.М. Основы программирования [Текст]: Учебник для вузов / С.М. Окулов. – 10-е изд. – М.: Лаборатория знаний, 2020. – 339с.

Андрианова, А. А. Алгоритмизация и программирование [Текст]: Учебное пособие / А.А. Андрианова, Л. Н. Исмагилов, Т. М. Мухтарова. – СПб.: Лань, 2019. – 240с.

Тюкачев, Н. А. C#. Основы программирования [Текст]: Учебное пособие / Н. А. Тюкачев, В. Г. Хлебостроев. – 4-е изд. – СПб.: Лань, 2021. – 272с.

Документация C#. Начало работы, руководства, справочные материалы [Электронный ресурс]. – Проверено: 20.04.2022. – Режим доступа : https://docs.microsoft.com/ru-ru/dotnet/csharp, свободный. – Загл. с экрана.

Документация C++. Начало работы, руководства, справочные материалы [Электронный ресурс]. – Проверено: 15.04.2022. – Режим доступа : https://docs.microsoft.com/ru-ru/cpp/cpp/?view=msvc-160, свободный. – Загл. с экрана.

Kotlin Docs [Электронный ресурс]. – Проверено: 25.04.2022. – Режим доступа : https://kotlinlang.org/docs/home.html, свободный. – Загл. с экрана.

Алексеев, А. П. Введение в Web-дизайн [Текст]: Учебное пособие / А. П. Алексеев. – М.: СОЛОН-ПРЕСС, 2019. – 192с.

Изучение веб-разработки [Электронный ресурс]. – Проверено: 30.04.2022. – Режим доступа : https://developer.mozilla.org/ru/docs/Learn, свободный. – Загл. с экрана.

Диков, А. В. Клиентские технологии веб-программирования: JavaScript и DOM [Текст]: учебное пособие для вузов / А. В. Диков. — СПб.: Лань, 2020. — 124 с.

Гвоздева, Т. В. Проектирование информационных систем: технология автоматизированного проектирования. Лабораторный практикум [Текст]: учебное пособие для вузов / Т. В. Гвоздева, Б. А. Баллод. — 2-е изд. — СПб.: Лань, 2020. — 156 с.

Коннолли, Т. Базы данных: проектирование, реализация и сопровождение. Теория и практика [Текст]: учебник / Т. Коннолли, К. Бегг. — 3-е изд. — М.: Вильямс, 2018. — 1439 с.

Васильев, Н. П. Введение в гибридные технологии разработки мобильных приложений [Текст]: учебное пособие для вузов / Н. П. Васильев, А. М. Заяц. — 2-е изд., стер. — СПб.: Лань, 2021. — 160 с.

JSON Web Tokens [Электронный ресурс]. – Проверено: 05.05.2022. – Режим доступа : https://jwt.io, свободный. – Загл. с экрана.

Полное руководство по языку программирования С# 6.0 и платформе .NET 4.6 [Электронный ресурс]. — Проверено: 18.04.2022. – Режим доступа: http://metanit.com/ sharp/tutorial/, свободный. – Загл. с экрана.

Руководство по языку Kotlin [Электронный ресурс]. — Проверено: 27.04.2022. – Режим доступа: https://kotlinlang.ru/, свободный. – Загл. с экрана.

# Приложение А

(справочное)

**Описание таблиц баз данных**

В таблицах А.1 - А.11 представлены подробные виды таблиц баз данных клиентской и серверной части.

Таблица А.1 – Таблица DlgInfo серверной части

|  |  |  |  |
| --- | --- | --- | --- |
| **№** | **Атрибут** | **Семантика** | **Тип** |
| 1 | dialog\_id | Уникальный идентификатор диалога | nvarchar(25) |
| 2 | lastTimeMsg | Время последнего сообщения в диалоге (Unix-время) | int |
| 3 | nameOfChat | Название чата для отображения пользователям | nvarchar(25) |
| 4 | typeOfDlg | Тип диалога (личный или групповой) | int |

Таблица А.2 – Таблица FriendsTable серверной части

| **№** | **Атрибут** | **Семантика** | **Тип** |
| --- | --- | --- | --- |
| 1 | tagSenderFrnd | Уникальный идентификатор (тег) инициатора добавления в друзья | nvarchar(8) |
| 2 | tagReceiverFrnd | Уникальный идентификатор (тег) ответчика добавления в друзья | nvarchar(8) |
| 3 | nameFrnd | Имя пользователя, что отвечает на запрос | nvarchar(20) |
| 4 | status | Тип взаимоотношений | int |

Таблица А.3 – Таблица GenderInfo серверной части

|  |  |  |  |
| --- | --- | --- | --- |
| **№** | **Атрибут** | **Семантика** | **Тип** |
| 1 | idGender | Уникальный идентификатор типа гендера | int |
| 2 | nameGender | Название типа гендера | nvarchar(10) |

Таблица 4 – Таблица InfoUsers серверной части

| **№** | **Атрибут** | **Семантика** | **Тип** |
| --- | --- | --- | --- |
| 1 | tagUser | Уникальный идентификатор (тег) пользователя | nvarchar(8) |
| 2 | nickUser | Имя пользователя | nvarchar(20) |
| 3 | isVisible | Статус онлайна пользователя | bit |
| 4 | isAvatar | Статус наличия аватара у пользователя | bit |
| 5 | isVisionData | Тип видимости данных (приватность) пользователя | int |
| 6 | gender | Тип гендера пользователя | int |
| 7 | birthday | Дата рождения пользователя | nvarchar(10) |
| 8 | socialStatus | Социальный статус пользователя | nvarchar(15) |
| 9 | country | Страна пользователя | nvarchar(15) |
| 10 | dateRegistration | Дата регистрации пользователя | date |
| 11 | aboutMe | Информация о пользователя | nvarchar(MAX) |

Таблица А.4 – Таблица MsgDlgTable серверной части

| **№** | **Атрибут** | **Семантика** | **Тип** |
| --- | --- | --- | --- |
| 1 | id | Уникальный идентификатор сообщения | int |
| 2 | dialog\_id | Идентификатор диалога | nvarchar(25) |
| 3 | sender | Идентификатор отправителя сообщения | nvarchar(8) |
| 4 | typeMsg | Тип сообщения | nvarhcar(10) |
| 5 | textMsg | Данные сообщения (текст или идентификатор изображения) | nvarhcar(MAX) |
| 6 | timeCreated | Время отправки сообщения (Unix-время) | int |

Таблица А.5 – Таблица TypesOfDialogs серверной части

| **№** | **Атрибут** | **Семантика** | **Тип** |
| --- | --- | --- | --- |
| 1 | id\_type | Уникальный идентификатор типажа сообщения | int |
| 2 | name\_type | Название типа сообщения | nvarchar(10) |

Таблица А.6 – Таблица UserDlgTable серверной части

| **№** | **Атрибут** | **Семантика** | **Тип** |
| --- | --- | --- | --- |
| 1 | id | Уникальный идентификатор записи участника диалога | int |
| 2 | dialog\_id | Идентификатор диалога | nvarchar(25) |
| 3 | tagUser | Идентификатор участника | nvarchar(8) |
| 4 | enteredTime | Время присоединения к диалогу (Unix-время) | int |
| 5 | countMsg | Количество прочитанных сообщений в диалоге | int |
| 6 | rang | Ранг участника | int |

Таблица А.7 – Таблица UsersData серверной части

| **№** | **Атрибут** | **Семантика** | **Тип** |
| --- | --- | --- | --- |
| 1 | loginUser | Логин авторизации | nvarchar(20) |
| 2 | passUser | Пароль | nvarchar(32) |
| 3 | tagUser | Уникальный дентификатор пользователя | nvarchar(8) |
| 4 | deviceToken | Токен авторизованного устройства | nvarchar(255) |

Таблица А.8 – Таблица DataGroupInfo клиентской части

| **№** | **Атрибут** | **Семантика** | **Тип** |
| --- | --- | --- | --- |
| 1 | Tag\_Of\_User | Уникальный идентификатор участника диалога | TEXT |
| 2 | rang | Ранг участника диалога | INTEGER |

Таблица А.9 – Таблица FriendsTable клиентской части

| **№** | **Атрибут** | **Семантика** | **Тип** |
| --- | --- | --- | --- |
| 1 | tagSenderFrnd | Уникальный идентификатор (тег) инициатора добавления в друзья | TEXT |
| 2 | tagReceiverFrnd | Уникальный идентификатор (тег) ответчика добавления в друзья | TEXT |
| 3 | friendName | Имя пользователя, что отвечает на запрос | TEXT |
| 4 | status | Тип взаимоотношений | INTEGER |

Таблица А.10 – Таблица ListUsersChat клиентской части

| **№** | **Атрибут** | **Семантика** | **Тип** |
| --- | --- | --- | --- |
| 1 | Tag\_Of\_User | Уникальный идентификатор диалога | TEXT |
| 2 | Name\_Of\_User | Название диалога (имя пользователя, с которым диалог) | TEXT |

Таблица А.11 – Таблица MsgDlgTable клиентской части

| **№** | **Атрибут** | **Семантика** | **Тип** |
| --- | --- | --- | --- |
| 1 | id | Уникальный идентификатор сообщения | INTEGER |
| 2 | dialog\_id | Идентификатор диалога | TEXT |
| 3 | sender | Идентификатор отправителя сообщения | TEXT |
| 4 | typeMsg | Тип сообщения | TEXT |
| 5 | textMsg | Данные сообщения (текст или идентификатор изображения) | TEXT |
| 6 | timeCreated | Время отправки сообщения (Unix-время) | INTEGER |

Таблица А.12 – Таблица OnlineUsers клиентской части

| **№** | **Атрибут** | **Семантика** | **Тип** |
| --- | --- | --- | --- |
| 1 | Tag\_Of\_User | Уникальный идентификатор пользователя | TEXT |
| 2 | Name\_Of\_User | Имя пользователя | TEXT |

Таблица А.13 – Таблица UserDlgTable клиентской части

| **№** | **Атрибут** | **Семантика** | **Тип** |
| --- | --- | --- | --- |
| 1 | id | Уникальный идентификатор записи участника диалога | INTEGER |
| 2 | dialog\_id | Идентификатор диалога | TEXT |
| 3 | Tag\_Of\_User | Идентификатор участника | TEXT |
| 4 | EnteredTime | Время присоединения к диалогу (Unix-время) | INTEGER |
| 5 | countMsg | Количество прочитанных сообщений в диалоге | INTEGER |
| 6 | lastTimeMsg | Время последнего сообщения в диалоге | INTEGER |
| 7 | typeOfDialog | Тип диалога | INTEGER |
| 8 | rang | Ранг участника диалога | INTEGER |

Таблица А.14 – Таблица allUserInfoTable клиентской части

| **№** | **Атрибут** | **Семантика** | **Тип** |
| --- | --- | --- | --- |
| 1 | Tag\_Of\_User | Идентификатор пользователя | TEXT |
| 2 | genderUser | Пол пользователя | INTEGER |
| 3 | birthdayUser | Дата рождения пользователя | TEXT |
| 4 | socStatusUser | Социальный статус пользователя | INTEGER |
| 5 | countryUser | Страна пользователя | TEXT |
| 6 | dateRegUser | Дата регистрации пользователя | TEXT |
| 7 | aboutMeUser | Данные о пользователе | TEXT |

# Приложение Б

(справочное)

**Интерфейс браузерного клиента**

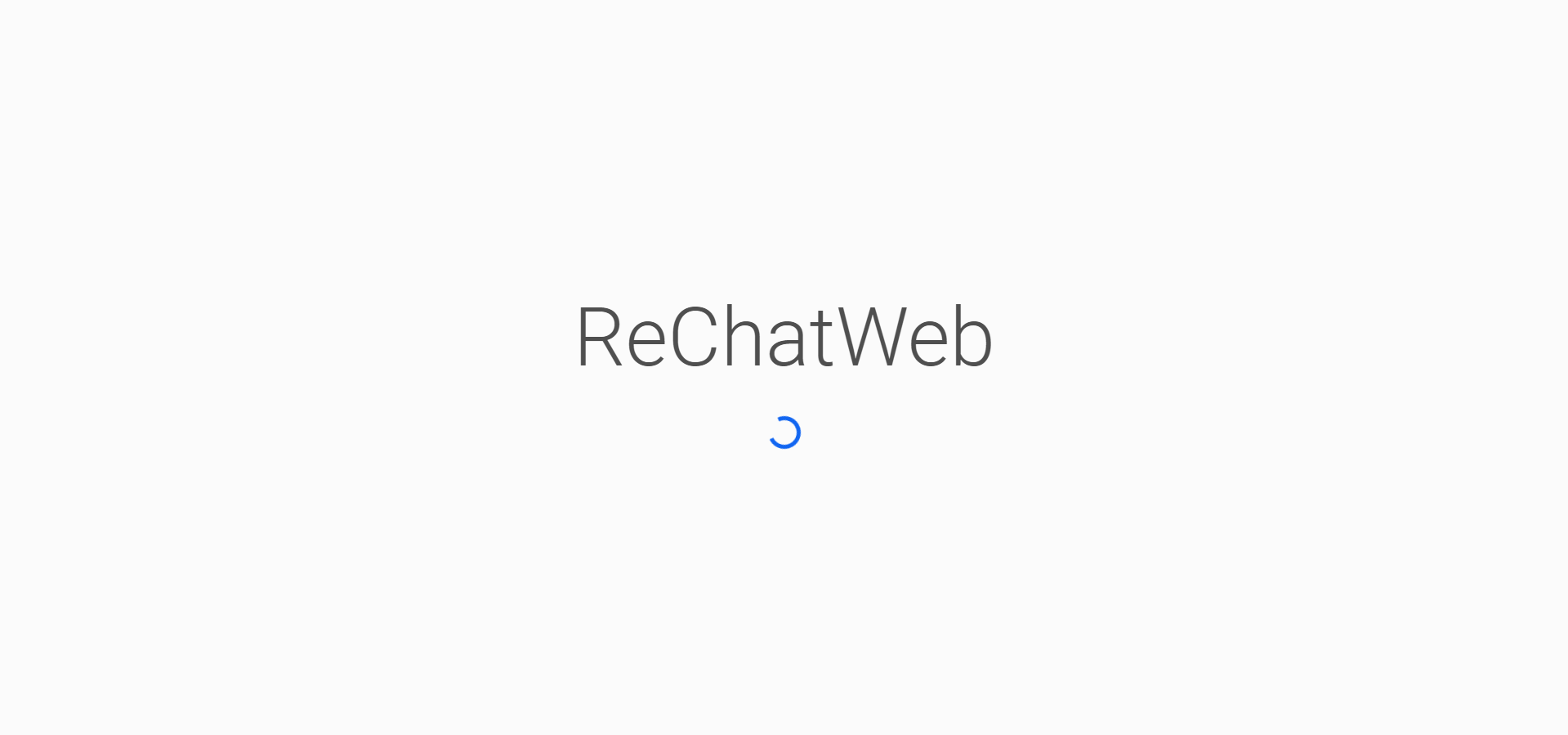


Рисунок Б.1 – Экран загрузки

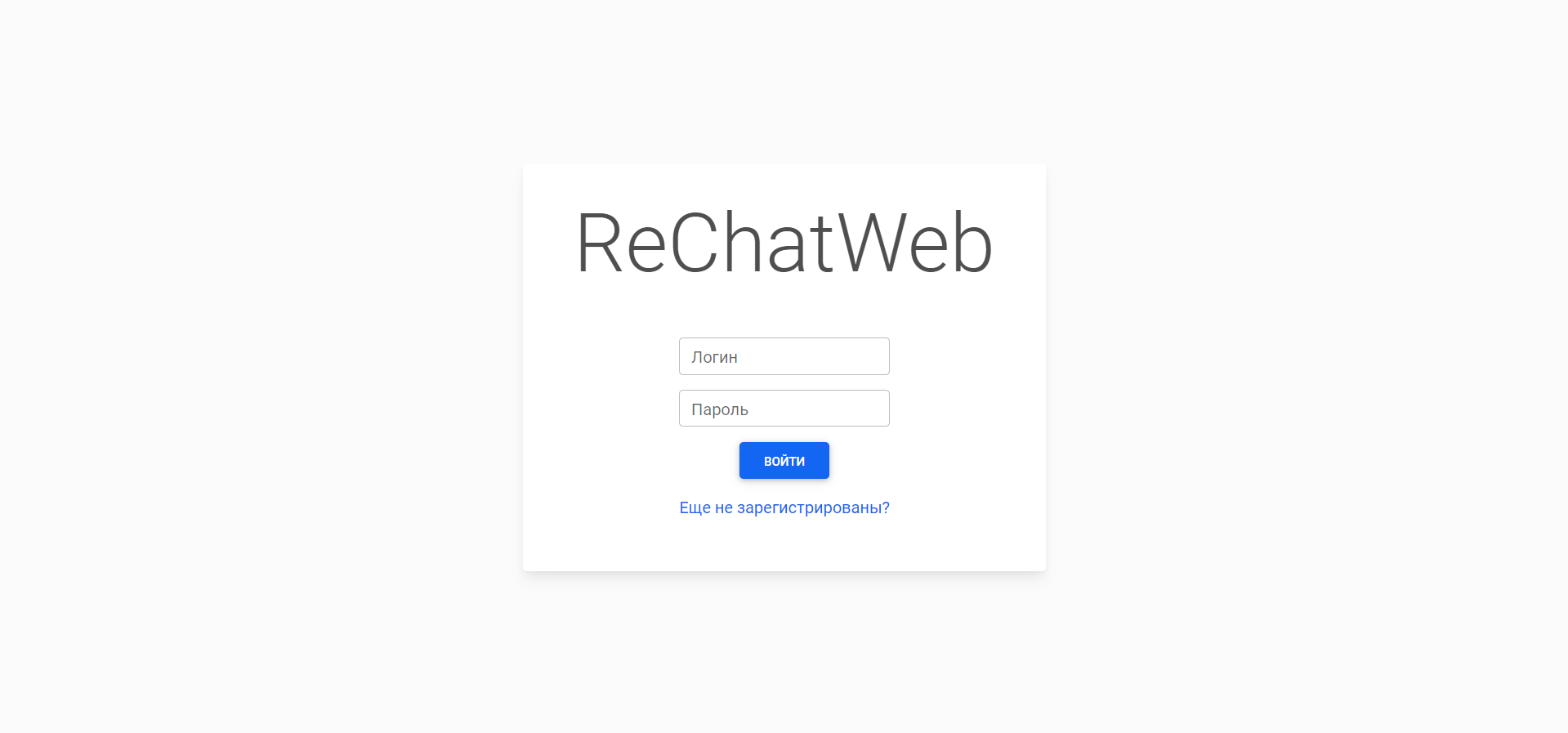


Рисунок Б.2 – Экран авторизации



Рисунок Б.3 – Экран регистрации

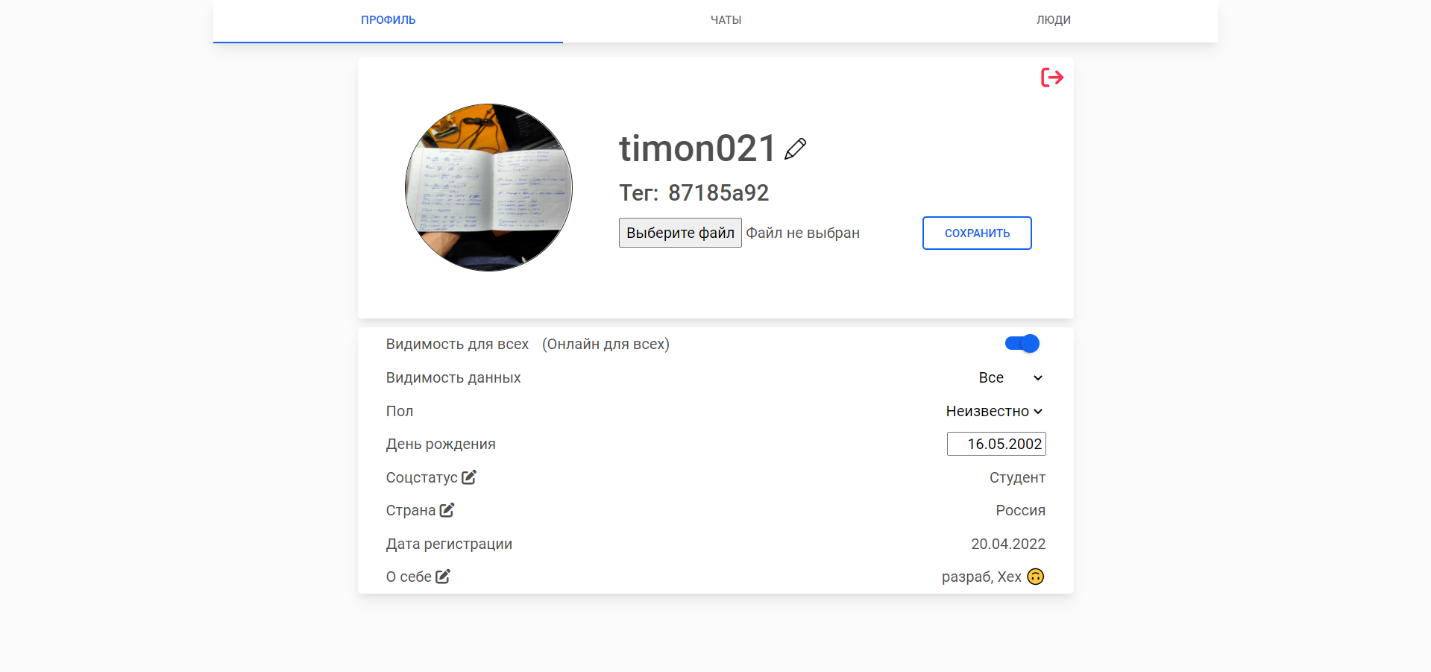


Рисунок Б.4 – Экран профиля пользователя

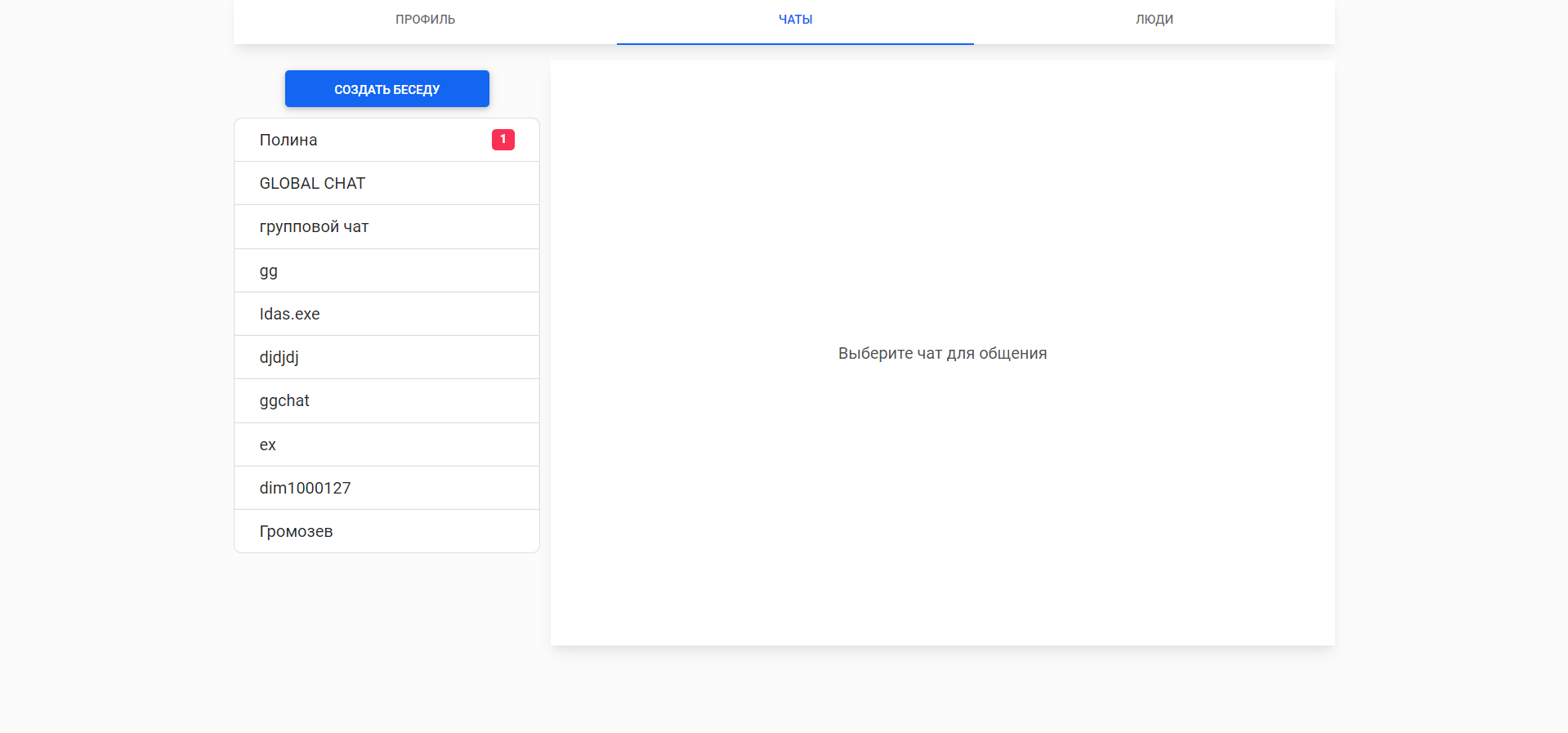


Рисунок Б.5 – Экран диалогов

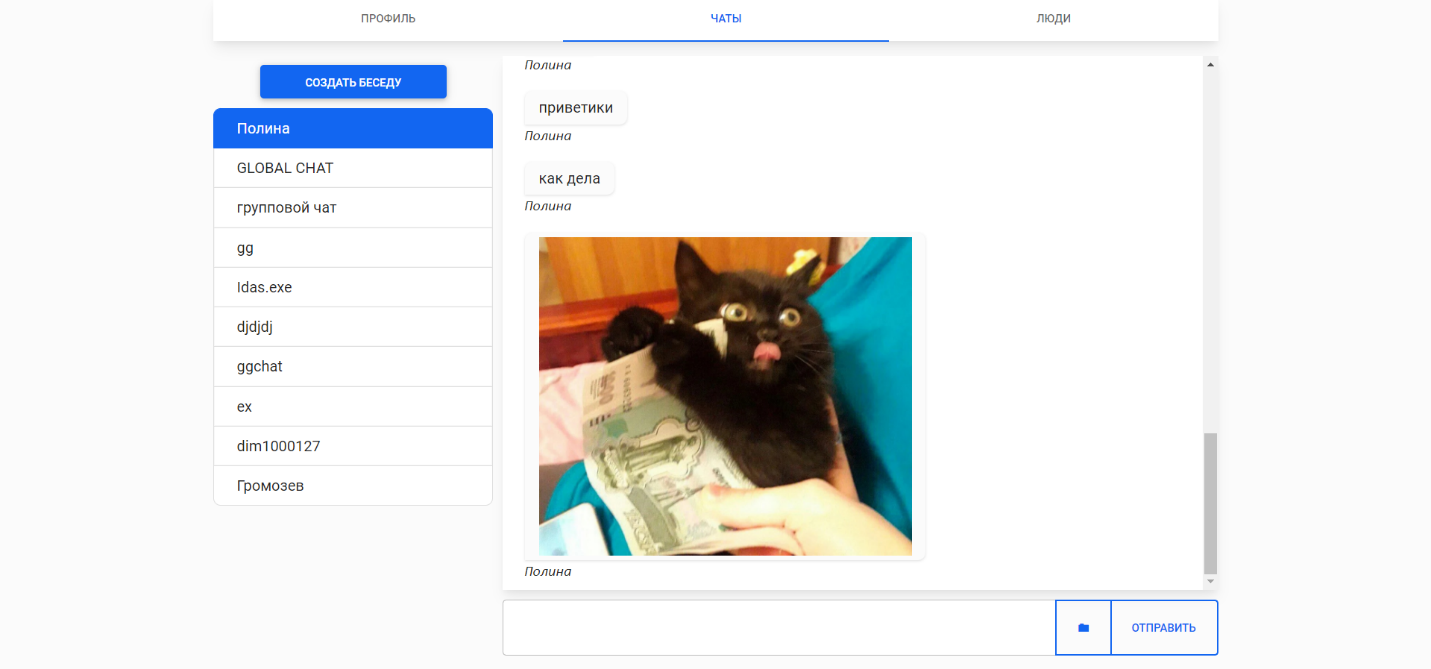


Рисунок Б.6 – Экран открытого чата с пользователем

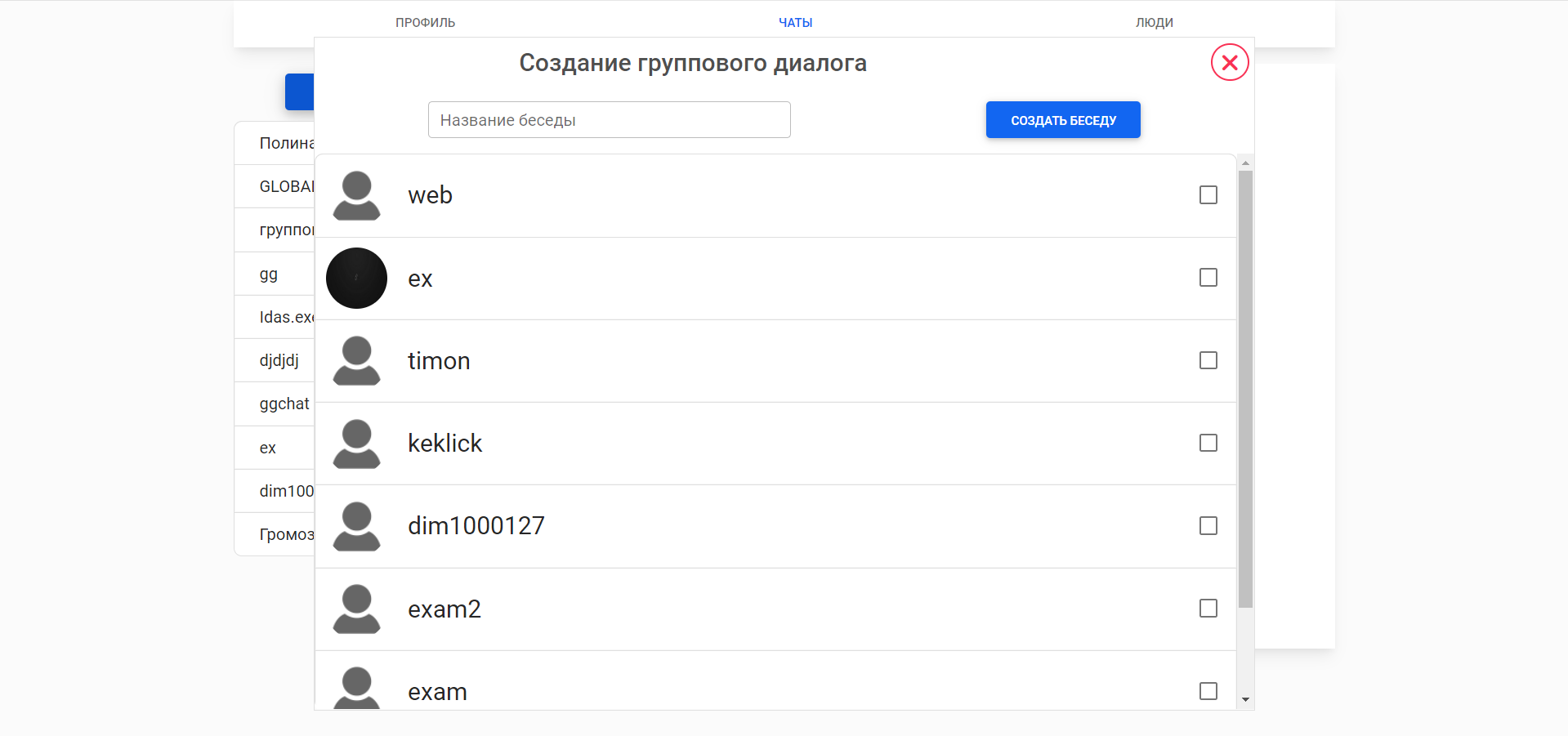


Рисунок Б.7 – Экран создания группового диалога

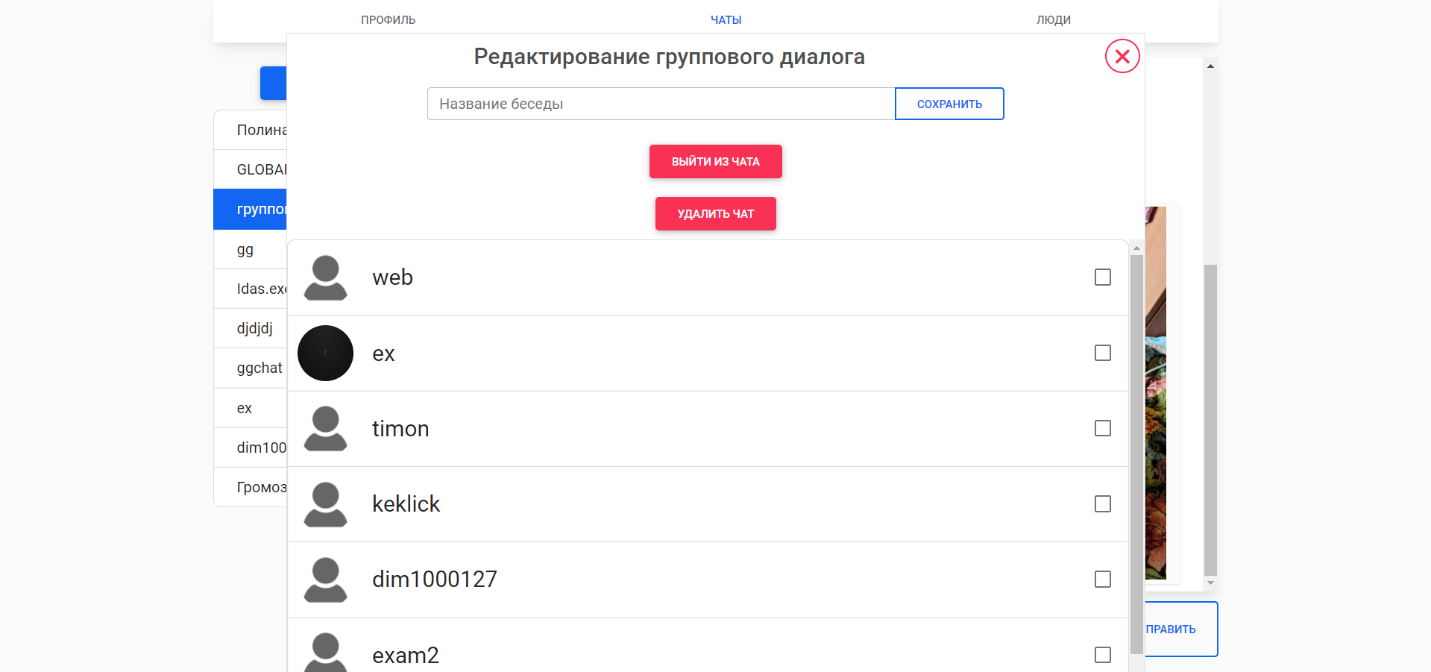


Рисунок Б.8 – Экран редактирования группового диалога

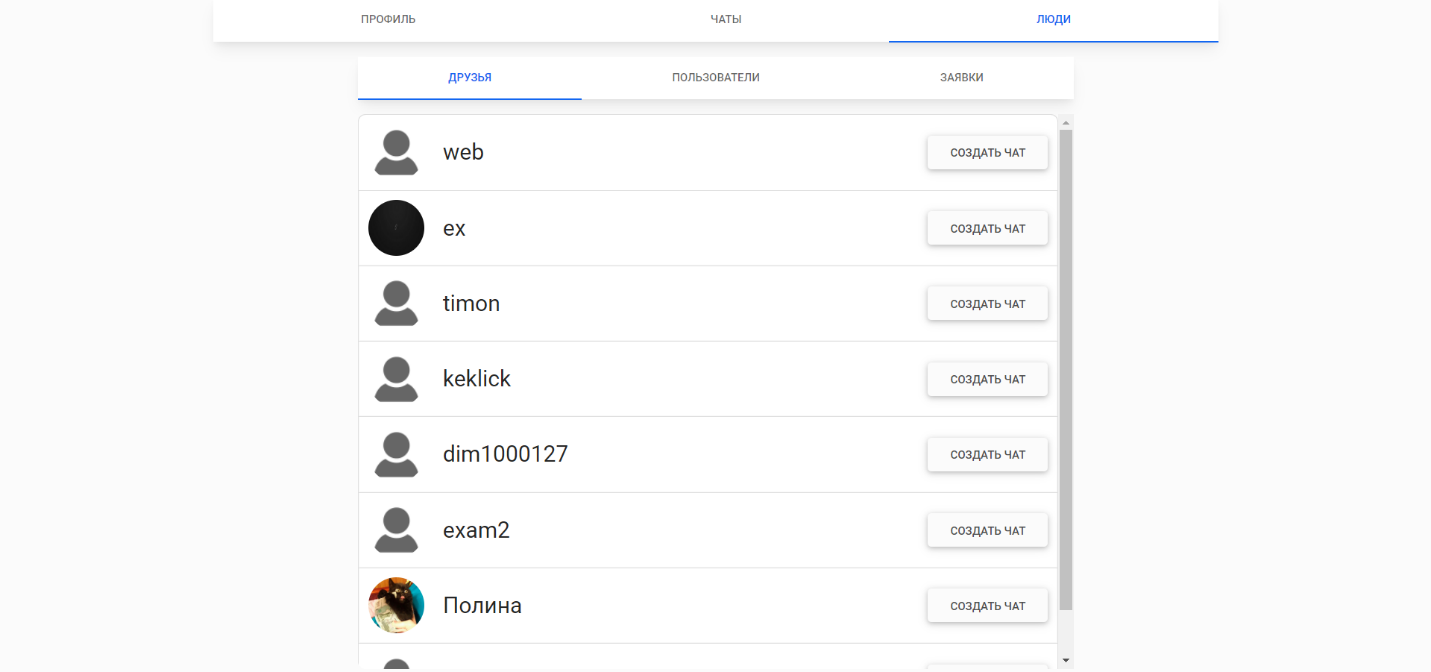


Рисунок Б.9 – Экран вкладки «Друзья» раздела «Люди»

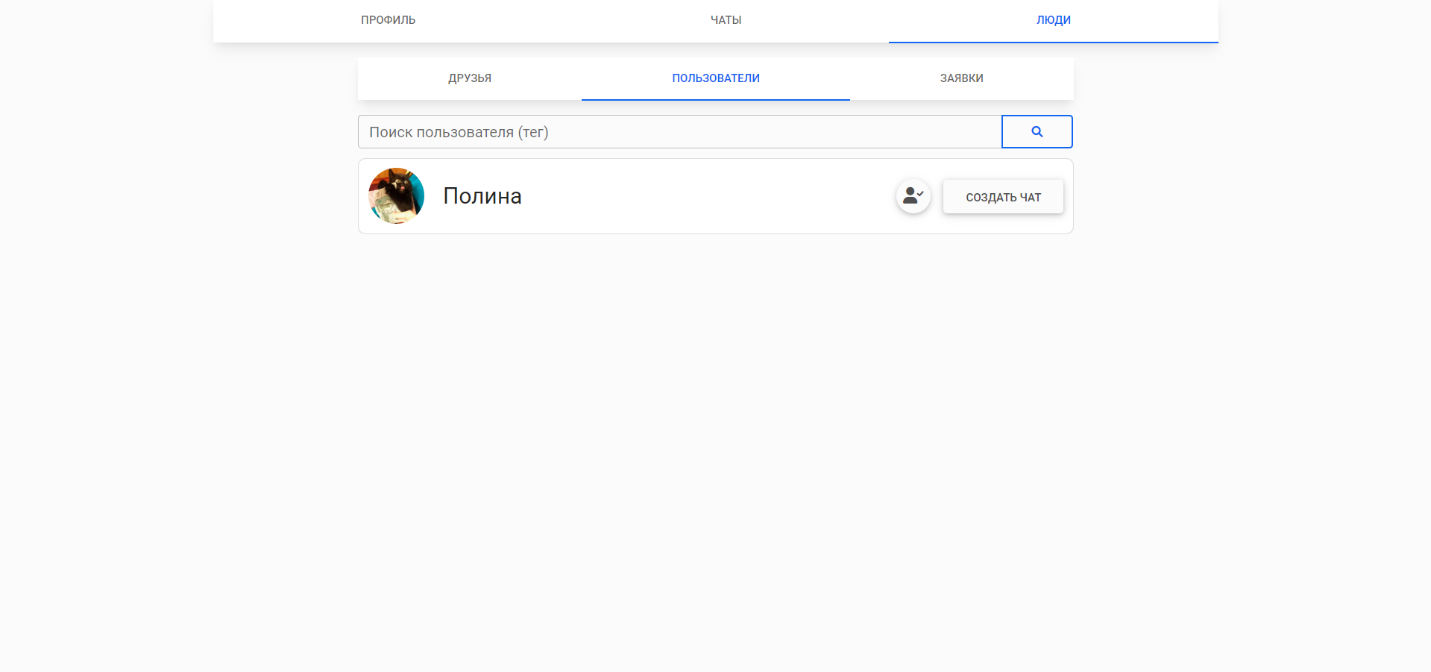


Рисунок Б.10 – Экран с пользователями онлайн

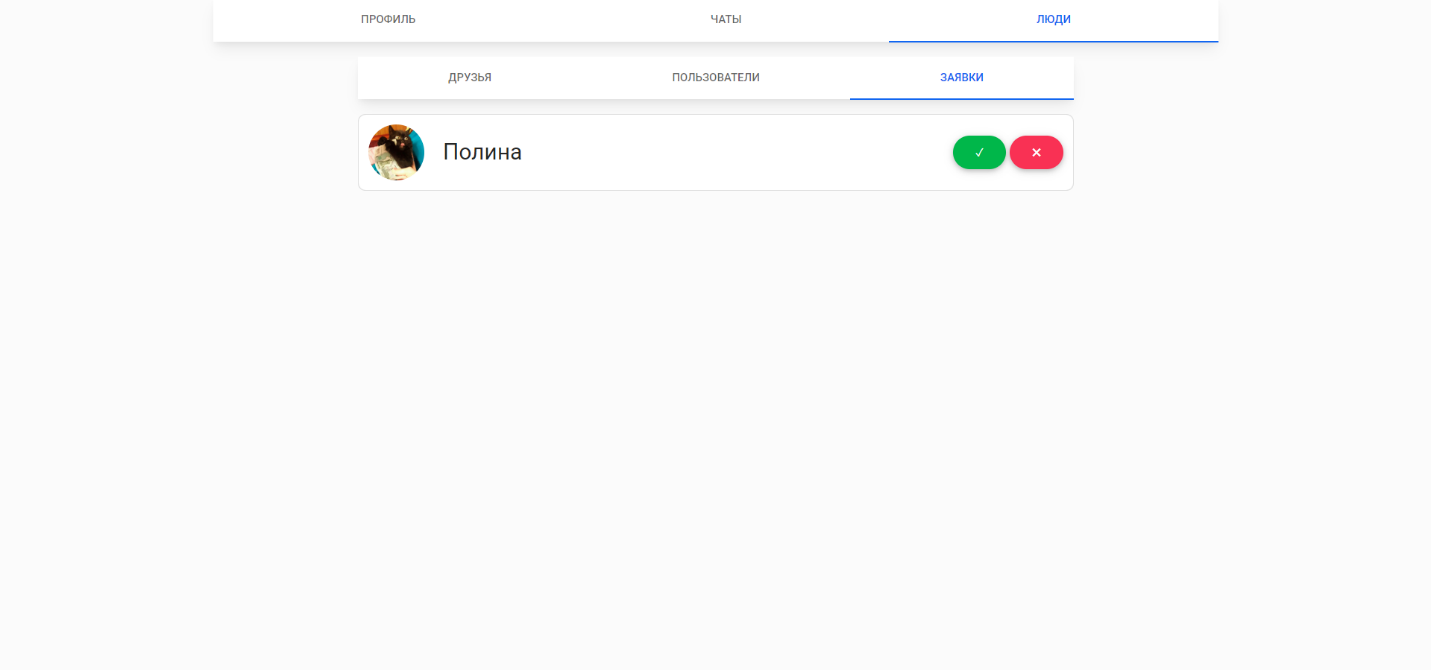


Рисунок Б.11 – Экран запросов в друзья

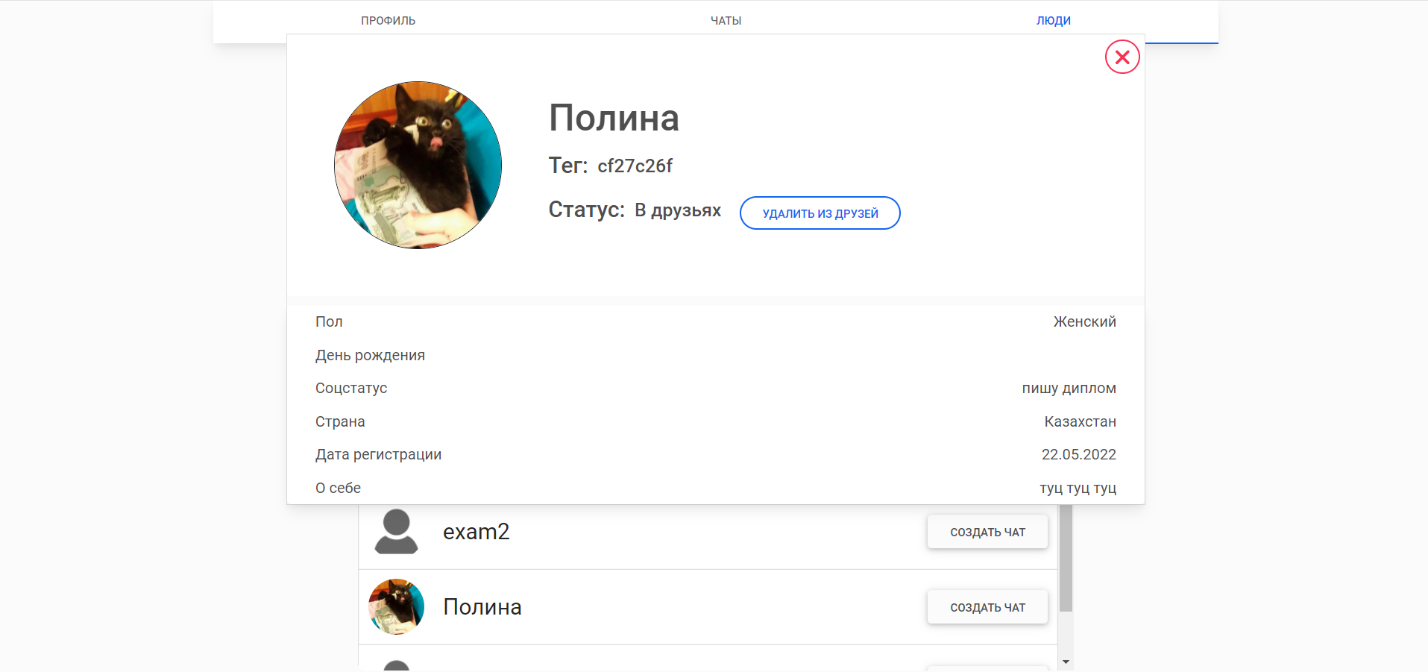


Рисунок Б.12 – Экран профиля другого пользователя

# Приложение В

(справочное)

**Исходный код**

Код сервера (Файл SkillChatServer.cpp):

#include <iostream>

#include <uwebsockets/App.h>

#include <map>

#include <nlohmann/json.hpp>

// [UUID] для создания уникального uuid

#include <boost/uuid/uuid.hpp>

#include <boost/uuid/random\_generator.hpp>

#include <boost/uuid/uuid\_io.hpp>

// …[SERVER] - константы для сервера…

using namespace std;

using json = nlohmann::json;

map<string, string> userNames;

map<string, string> ghostUserNames;

// Какую информацию о пользователе мы храним

struct PerSocketData {

string name; // имя юзерра

string uId; // уникальный идентификатор

};

// …Внешние функии для обработки данных…

int main() {

// unsigned int last\_user\_id = 10; // последний идентификатор пользователя

// сделать UUID / GUID

int userID = 1;

// Настраиваем сервер

uWS::SSLApp(). // Создаем простое приложение без шифрования

ws<PerSocketData>("/\*", {

// для каждого пользователя мы храним данные в виде PerSocketData

/\* Settings \*/

.idleTimeout = 1200, // таймаут в секундах (отключает пользователя)

.open = [&userID](auto\* ws) {

// функция open (лямбда функция)

// вызывается при открытии соединения

// 0. получить структуру PerSocketData

PerSocketData\* userData = (PerSocketData\*) ws->getUserData();

// 1. назначить пользователю уникальный идентификатор

userData->name = "UNNAMED";

userData->uId = generateUUID();

; for (auto entry : userNames) {

if (ghostUserNames.find(entry.first) != ghostUserNames.end()) break;

ws->send(online(entry.first), uWS::OpCode::TEXT);}

updateName(userData);

cout << "New user connected, id = " << userData->uId << endl;

cout << "Users connected: " << userNames.size() << endl;

string userChannel = "user#" + userData->uId;

ws->subscribe(userChannel); // укаждого юзера есть личка

ws->subscribe(BROADCAST\_CHANNEL); // подписка юзера на общий канал

// todo Сообщить всем пользователям, что кто-то онлайн

ws->publish(userChannel, "FC::" + userData->uId, uWS::OpCode::TEXT, false);},

.message = [](auto\* ws, string\_view message, uWS::OpCode opCode) {

string strMessage = string(message);

auto jsonData = json::parse(strMessage);

PerSocketData\* userData = (PerSocketData\*)ws->getUserData();

string authorId = userData->uId;

// …Обработка полученных данных…

if (isMessageTo(jsonData["type"])) {

// подготовить данные и отправить их

string receiverId = jsonData["id"];

string text = jsonData["text"];

string dialog\_id = jsonData["dialog\_id"];

string typeMsg = jsonData["typeMsg"];

json jsonOut = {

{"dialog\_id", dialog\_id},

{"sender", authorId},

{"typeMsg", typeMsg},

{"text", text},

{"receiverId", receiverId}};

string outgoingMessage = FORDB + SQL + INSERT + NEWMSGDLG + (string)jsonOut.dump();

ws->publish("user#999", outgoingMessage, uWS::OpCode::TEXT, false);}

if (isSetName(jsonData["type"])) {…}

if (isDeleteAvatar(jsonData["type"])) {…}

if (isSetAvatar(jsonData["type"])) {…}

if (isSignNewUser(jsonData["type"])) {…}

if (isAuthUser(jsonData["type"])) {…}

if (isAuthToken(jsonData["type"])) {…}

if (isUpdateVisible(jsonData["type"])) {…}

if (isUpdate(jsonData["type"])) {…}

if (isConnectionServerDB(jsonData["type"])) {…}

if (isCreateDlg(jsonData["type"])) {…}

if (isFrnd(jsonData["type"])) {…}

if (isAddUserDlg(jsonData["type"]) {…}

if (isDownLoadData(jsonData["type"])) {…}

if (isResultFromDB(jsonData["type"])) {…}},

.close = [](auto\* ws , int /\*code\*/, string\_view /\*message\*/) {

// вызывается при отключении от сервера

PerSocketData\* userData = (PerSocketData\*)ws->getUserData();

ws->publish(BROADCAST\_CHANNEL, offline(userData->uId));

deleteName(userData);

cout << "Users connected: " << userNames.size() << endl;}})

.listen(9001, [](auto\* listen\_socket) {

if (listen\_socket) {

// если все ок, вывести сообщение

cout << "Listening on port " << 9001 << std::endl;}

}).run();} // запуск

Код сервера базы данных (Файл Prgram.cs):

using System;

using WebSocket4Net;

using System.Data.SqlClient;

using Newtonsoft.Json;

using System.Collections.Generic;

using JWT.Builder;

using JWT.Algorithms;

using WSClientDB.dataClasses;

using System.Security.Cryptography;

using System.Text;

namespace WSClientDB

{

class Program{

// …[SERVER] - константы для сервера…

static WebSocket webSocket;

static SqlConnection sqlConnection;

static SqlCommand sqlCommand;

static void Main(string[] args){

webSocket = new WebSocket("ws://servchat.ddns.net:9001/");

webSocket.Opened += WebSocket\_Opened;

webSocket.Error += WebSocket\_Error;

webSocket.Closed += WebSocket\_Closed;

webSocket.MessageReceived += WebSocket\_MessageReceived;

webSocket.Open();

Console.ReadKey();}

// < внешние функции для обработки данных >

public static string GetHash(string input) {…}

private static void InsertDataSignUp(string loginUser, string passUser, string nickUser, string tagUser) {…}

private static void InsertDataNewMsgDLG(string dialog\_id, string sender, string typeMsg, string text, string receiverId) {…}

private static void InsertDataNewUserDLG(List<string> userCompanion, string userManager, string nameOfChat) {…}

private static string getToken(string tagUser) {…}

private static void SelectDataForAuth(string authorUser, string loginUser, string passUser) {…}

private static List<string> GetInfoAboutDialogs(string tagUser) {…}

private static void SelectDataForAllDlg(string tagUser, string token) {…}

private static List<DataOfMessage> SlctAllMsgHelper(string dialog\_id) {…}

private static void SelectDataForAllMsg(List<string> dialog\_ids, string authorId, string token) {…}

private static void SelectDataForAllTagName(List<string> dialog\_ids, string authorId, string token) {…}

private static void UpdateNameOfUser(string tagId, string newName) {…}

private static void UpdateVisibleOfUser(string tagUser, bool isVisible) {…}

private static void UpdateAvatarOfUser(string tagUser) {…}

private static void DeleteAvatarOfUser(string tagUser) {…}

private static void AddInGlobalChat(string userTag) {…}

private static void SelectDeviceForAuth(string userTag, string token) {…}

private static void WorkWithFrnd(ActionsWithFrnd actionsWithFrnd) {…}

private static void UpdateFriend(UpdateFriend updateFriend) {…}

private static void DeleteFriend(DeleteFriend deleteFriend) {…}

private static void SelectDataForAllFriends(string tagUser, string token) {…}

private static void SelectDataForAllInfoUsers(string tagUser, string needTagUser, bool isFriend) {…}

private static void FindFriend(UpdateFriend updateFriend) {…}

private static void UpdateCountMessage(UpdateCountMsg updateCountMsg) {…}

private static void UpdateVisionData(UpdateVandG updateVandG) {…}

private static void UpdateGenderData(UpdateVandG updateVandG) {…}

private static void UpdateBirthday(UpdateDataString updateDataString) {…}

private static void UpdateSocStatus(UpdateDataString updateDataString) {…}

private static void UpdateCountry(UpdateDataString updateDataString) {…}

private static void UpdateAboutMe(UpdateDataString updateDataString) {…}

private static void UpdateTitleDialog(UpdateTitleDialog updateTitleDialog) {…}

private static void SelectDataAllGroupInfo(DataAllGroupInfo dataAllGroupInfo) {…}

private static void AddUserInDlg(AddUserDlg addUserDlg) {…}

private static void DeleteUserFromDialog(DeleteUserFromDlg deleteUserFromDlg) {…}

private static void DeleteDialog(DeleteDialog deleteDialog) {…}

private static void UpdateRang(UpdateRangUser updateRangUser) {…}

private static void WebSocket\_MessageReceived(object sender, MessageReceivedEventArgs e){

if (e.Message.IndexOf(FORDB) == -1) return;

string message = e.Message.Substring(FORDB.Length);

if (message.IndexOf(INFO) != -1){

message = message.Substring(INFO.Length);

Console.WriteLine($"[MSG] -> {message}");}

if (message.IndexOf(SQL) != -1){

message = message.Substring(SQL.Length);

if (message.IndexOf(INSERT) != -1){…}

if (message.IndexOf(SELECT) != -1){…}

if (message.IndexOf(UPDATE) != -1){…}

if (message.IndexOf(DELETE) != -1){…}}}

private static void WebSocket\_Closed(object sender, EventArgs e){

Console.WriteLine("Connection closed");}

private static void WebSocket\_Error(object sender, SuperSocket.ClientEngine.ErrorEventArgs e){

Console.WriteLine($"Error: {e.Exception}");}

private static void WebSocket\_Opened(object sender, EventArgs e){

Console.WriteLine("Successful connection");

SecureConnection secureConnection = new SecureConnection();

secureConnection.type = "DBSERVER";

secureConnection.key = "2FA";

string jsonMsg = JsonConvert.SerializeObject(secureConnection);

webSocket.Send(jsonMsg);

sqlConnection = new SqlConnection("server=HOME-PC;database=ReChat;integrated security=true");

sqlCommand = new SqlCommand();}}}

Код приложения Android (файл ActivityMain.kt):

package com.example.myapplication.activity

import …

class ActivityMain :  
 AppCompatActivity(),  
 UserFragment.OnFragmentSendDataListener,  
 ChatFragment.OnFragmentSendDataListener,  
 FriendsFragment.OnFragmentSendDataListener,  
 SharedPreferences.OnSharedPreferenceChangeListener,  
 FirstDisplayFragment.OnFragmentSendDataListener,  
 AuthFragment.OnFragmentSendDataListener,  
 FriendListFragment.OnFragmentSendDataListener,  
 UserListFragment.OnFragmentSendDataListener,  
 FrndListRequestFragment.OnFragmentSendDataListener{  
 companion object {  
 const val WEB\_SOCKET\_URL = "ws://servchat.ddns.net:9001"  
 const val IMAGE\_REQUEST = 1  
 const val VERSION\_APP = "0.7"  
 lateinit var webSocketClient : WebSocketClient  
 lateinit var sqliteHelper: SqliteHelper}  
 private lateinit var sp : SharedPreferences  
 private lateinit var tagUser : String  
 private lateinit var bubbleNav : BubbleNavigationLinearView  
 private var needLoad : Boolean = false  
 override fun onCreate(savedInstanceState: Bundle?) {  
 super.onCreate(savedInstanceState)  
 setContentView(R.layout.bottom\_nav\_bar)  
 sqliteHelper = SqliteHelper(this)  
 supportActionBar?.title = resources.getString(R.string.rechat)  
 sp = getSharedPreferences("OURINFO", Context.MODE\_PRIVATE)  
 sp.registerOnSharedPreferenceChangeListener(this)  
 bubbleNav = findViewById(R.id.bottom\_navigation\_view\_linear)  
 val ed = sp.edit()  
 ed.putString("queryImg", LocalDateTime.now().toString())  
 ed.apply()  
 isActVersion()  
 loadFragment(-1)  
 bubbleNav.setNavigationChangeListener { view, position ->  
 loadFragment(position)}}

private fun isActVersion(){…}

override fun onPostResume(){…}

override fun onRestart() {…}

private fun loadFragment(position : Int){  
 val toolbar = supportActionBar  
 val fragment: Fragment  
 when (position) {  
 0 -> {  
 toolbar?.title = resources.getString(R.string.profile)  
 fragment = UserFragment()  
 bubbleNav.visibility = View.VISIBLE}  
 1 -> {  
 toolbar?.title = resources.getString(R.string.chat)  
 fragment = ChatFragment()  
 bubbleNav.visibility = View.VISIBLE}  
 2 -> {  
 toolbar?.title = resources.getString(R.string.people)  
 fragment = FriendsFragment()  
 bubbleNav.visibility = View.VISIBLE}  
 3 ->{  
 bubbleNav.visibility = View.GONE  
 fragment = AuthFragment()  
 toolbar?.title = resources.getString(R.string.rechat)}  
 else -> {  
 bubbleNav.visibility = View.GONE  
 fragment = FirstDisplayFragment()  
 toolbar?.title = resources.getString(R.string.rechat)}}  
 supportFragmentManager.beginTransaction()  
 .replace(R.id.host\_fragment, fragment)  
 .commit()}

override fun onUserLoadView() {…}

override fun exitFromAccount(){…}

override fun onChatLoadView() {…}

override fun onCreateNewDialog(){…}

override fun onFriendsLoadView() {}  
 override fun onFirstDisplayLoadView() {}  
 override fun onAuthLoadView() {}

override fun onFriendsListLoadView(){…}

override fun onUserListLoadView(){…}

override fun onFrndListRequestLoadView(){…}

fun changePhotoClick(view : View) {…}

override fun onActivityResult(requestCode: Int, resultCode: Int, data: Intent?) {…}

override fun onSharedPreferenceChanged(sharedPreferences: SharedPreferences?, key: String?) {…}

private fun initWebSocket(){  
 val chatservUri = URI(WEB\_SOCKET\_URL)  
 createWebSocketClient(chatservUri)   
 webSocketClient.connect()}  
 private fun createWebSocketClient(chatservURI: URI?){  
 webSocketClient = object : WebSocketClient(chatservURI){  
 override fun onOpen(handshakedata: ServerHandshake?) {  
 Log.i("\_\_CHAT\_\_", "onOpen")  
 sqliteHelper.clearTable()  
 val tokenAuth = sp.getString("token", "")  
 if (tokenAuth != null) {  
 if (tokenAuth.isEmpty()) {  
 runOnUiThread {  
 loadFragment(3)}  
 } else{  
 val successAuthToken = SuccessAuthToken("AUTHTOKEN::", tokenAuth)  
 val msg = Json.encodeToString(successAuthToken)  
 if(!webSocketClient.connection.isClosed){  
 webSocketClient.send(msg)}}}}  
 override fun onClose(code: Int, reason: String?, remote: Boolean) {  
 Log.i("\_\_CHAT\_\_", "onClose")  
 this@ActivityMain.runOnUiThread{  
 loadFragment(4)  
 webSocketClient.reconnect()}}  
 override fun onMessage(message: String?) {  
 Log.i("\_\_CHAT\_\_", "onMessage: $message");  
 this@ActivityMain.runOnUiThread{  
 parseMessage(message)}}  
 override fun onError(ex: Exception?) {  
 Log.i("\_\_CHAT\_\_", "onException: $ex");}}}  
fun parseMessage(message: String?){  
 when (message?.substringBefore("::")) {  
 "DBNOTACTIVE" -> {…}

"RESULTDB" -> {…}

"ONLINE" -> {…}

"OFFLINE" -> {…}

"MESSAGE\_FROM" -> {…}}}

private fun messageToUser(jsonData: String) {…}

private fun messagePrint(jsonData: String) {…}

private fun deviceAuth(data: String) {…}

private fun authorization(data: String) {…}

private fun getJwt(tagUser : String) : String {…}

private fun checkVerifyJWT(token : String) : Boolean {…}}

Код веб-приложения (Файл main.js):

let webSocket;  
let db;  
let activeChat;  
let addedUserForDlg = [];  
let dialog\_id = "";  
$(window).on('click', function (e){  
 const obj = e.target

if(obj.classList.contains('openSignField')) {…}

// <обработки нажатий мыши>

});  
$(document).ready(function (){  
 if (typeof webSocket === "undefined" ){  
 connectToServer();}  
 let f = $('#fileImgMsg');  
 f.change(function(){  
 if (f[0].value.length){  
 $('.msgWindow').css({"height" : "74vh"});  
 $('.blockImgMsg').show();  
 $('.textAreaMsg').prop('disabled', true);  
 }else{clearAndHideInputImgMsg();}});});  
function connectToServer(){  
 try{  
 webSocket = new WebSocket("ws:servchat.ddns.net:9001");  
 webSocket.onopen = () => onOpen();  
 webSocket.onmessage = ({data}) => onMessage(data);  
 webSocket.onclose = (event) => onClose(event);  
 webSocket.onerror = () => onError(error);  
 workWithDB();}  
 catch (e){  
 alert(e.message);}}  
function onMessage(data){  
 parseMessage(data);}  
function onOpen(){  
 $('.startScreen').hide();  
 let myToken = localStorage.getItem('token');  
 if(myToken === null || myToken.trim() === '') {  
 $('.authUser').show();  
 }else{  
 let data = {  
 type : "AUTHTOKEN::",  
 token : myToken};  
 let jsonString = JSON.stringify(data);  
 if(webSocket.readyState === WebSocket.OPEN){  
 webSocket.send(jsonString);}}}  
function onClose(event){  
 if (event.wasClean){  
 alert('Соединение закрыто чисто');  
 }else{  
 alert('Обрыв соединения'); // например, "убит" процесс сервера}  
 $('.startScreen').hide();  
 $('.authUser').hide();  
 $('.mainContent').hide();  
 $('.userProfile').hide();  
 $('.errorScreen').show();}  
function onError(error){  
 alert("Ошибка " + error.message);  
 $('.startScreen').hide();  
 $('.authUser').hide();  
 $('.mainContent').hide();  
 $('.userProfile').hide();  
 $('.errorScreen').show();}  
function parseMessage(data) {  
 let posTypeMsg = data.indexOf('::');  
 let typeMsg = data.substring(0,posTypeMsg);  
 switch (typeMsg){

case 'RESULTDB': {…}

case 'ONLINE':

case 'OFFLINE':

case 'MESSAGE\_FROM':

default:{break;}}}

function funcRegisterUser(){…}

function funcAuthUser(){…}

function deviceAuth(dataFS) {…}

function authorization(dataFS) {…}

function saveNewAvatar (){…}

function changeName(){…}

function changeVisible(obj) {…}

function workWithDB(){…}

function downloadChat(){…}

function addChatInWindow(){…}

function updatePositionChat(){…}

function getAllData(){…}

function recoveryMsg(obj) {…}

function sendMsg(){…}

function messageTo(msg) {…}

function messagePrint(msg) {…}

function clearAndHideInputImgMsg(){…}

function autoScrollDown(){…}

function printFriendsList(){…}

function addUserInOnline(id, name) {…}

function updateNameInUserChat(id, name) {…}

function updateNameInFriends(id, name) {…}

function deleteUserFromOnline(id) {…}

function updateAddUsersList(id,name) {…}

function updateDeleteUsersList(id) {…}

function addRequestsUser(tagUser, nameOfUser) {…}

function acceptAddFriend(obj) {…}

function denyAddFriend(obj) {…}

function addUserInFriend(obj) {…}

function deleteUserFromFriend(obj) {…}

function updateUserFriends(obj) {…}

function addFriend(obj) {…}

function hoverAvatar(obj) {…}

function unHoverAvatar(obj) {…}

function deleteAvatar(){…}

function addChat(obj) {…}

function onUserClick(obj) {…}

function closeUserClick(){…}

function actWithFriends(obj) {…}

function searchPeople(){…}

function addSearchedPeople(obj) {…}

function onTextChangedSearched(obj) {…}

function exitFromAcc(){…}

function editSocStatus(){…}

function editCountry(){…}

function editAboutMe(){…}

function saveSocStatus(){…}

function saveCountry(){…}

function saveAboutMe(){…}

function closeNewDlgClick(){…}

function closeEditDlgClick(){…}

function printAddFriends(){…}

function chooseUserNewDlg(obj) {…}

function createDialog(){…}

function aboutDlgClick(obj) {…}

function logOutChat(){…}

function deleteChat(){…}

function renameChat() {…}

# Приложение Г

(справочное)

**Руководство оператора**

1. **Назначение программы**

Данное программное обеспечение предназначено для решения задачи предоставления пользователям возможности использования виртуального пространства с поддержкой чата. Эксплуатация программы будет осуществляться в личных целях заинтересованного круга лиц, которым выступает заказчик разработки.

Функциональные возможности:

1. Создание профиля пользователя с различными данными, управление этими данными;
2. Создание личных и групповых диалогов с другими пользователями;
3. Общение с другими пользователями внутри диалогов;
4. Использование текстовых сообщений и сообщений в виде изображений;
5. Добавление пользователей в список избранных представляемый списком друзей;
6. Просмотр профилей других пользователей и взаимодействие с ними.
7. **Условия выполнения программы**
   1. **Климатические условия**

Климатические условия эксплуатации, при которых должны обеспечиваться заданные характеристики, должны удовлетворять требованиям, предъявляемым к техническим средствам в части условий их эксплуатации.

* 1. **Минимальный состав аппаратных средств**

Требования, предъявляемые к составу технических средств, обслуживающих программное обеспечение должно соответствовать следующим минимальным параметрам:

1. Для мобильного решения:
   1. процессор x86 с тактовой частотой, не менее 1 ГГц;
   2. оперативную память объемом, не менее 1 Гб;
   3. размер встроенной памяти от 4 Гб;
   4. подключение к сети Интернет.
2. Для настольного решения;
   1. Архитектура процессора – x86;
   2. Частота процессора – 2,5 ГГц;
   3. Объем оперативной памяти – 4 Гб;
   4. Объем постоянного накопителя – 50 Гб;
   5. Сетевая карта с подключением к сети Интернет;
   6. Монитор, клавиатура, мышь;
   7. **Минимальный состав программных средств**

Требования, предъявляемые к составу программных средств, обслуживающих программное обеспечение должно соответствовать следующим минимальным параметрам: для настольного решения необходимо наличие операционной системы, которая может предоставлять пользователю свой GUI и иметь браузер, или возможность его инсталлирования, для мобильной платформы необходимо иметь операционную систему Android 10 и выше.

1. **Выполнение программы**
   1. **Загрузка и запуск программы**

Для запуска мобильного приложения «ReChat» на устройстве Оператора необходимо найти его в списке приложений и открыть нажатием на иконку.

Появится стартовый экран (рисунок Г.1), на котором нужно открыть диалоговое окно нажатием на кнопку «Войти». Необходимо ввести логин в поле «Логин», а также пароль в поле «Пароль». После ввода авторизационных данных необходимо нажать на кнопку «Войти».

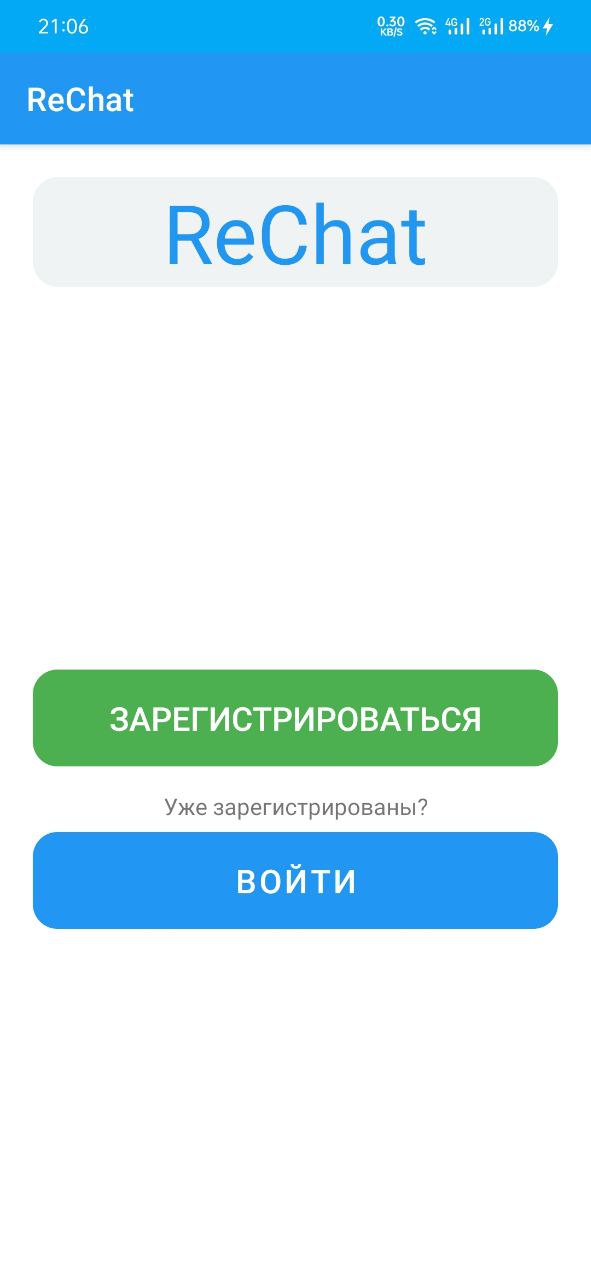


Рисунок Г.1 – Стартовый экран



Рисунок Г.2 – Диалоговое окно авторизации

После загрузки приложения произойдет открытие основной страницы программы – раздел «Профиль» (рисунок Г.3).

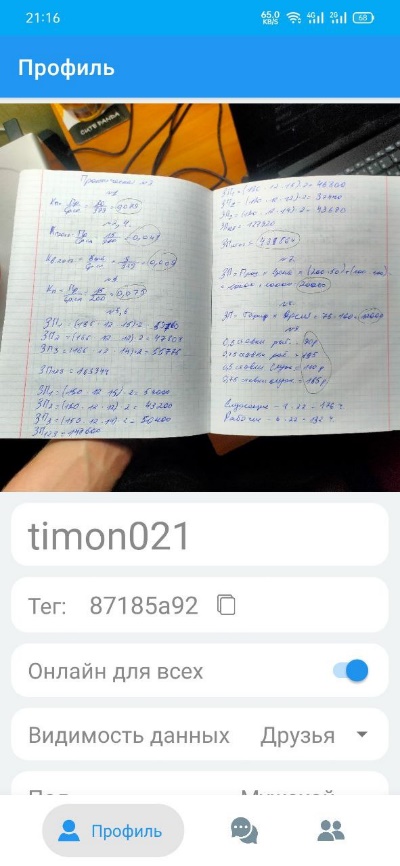


Рисунок Г.3 – Раздел «Профиль»

* 1. **Выполнение программы**
     1. **Выполнение функции редактирования данных профиля**

Для редактирования данных выберите желаемое поле и нажмите на него. Откроется диалоговое окно с полем для ввода новых данных (рисунок Г.4). В случае редактирования изображения профиля появится диалоговое окно с выбором конкретного действия (рисунок Г.5).

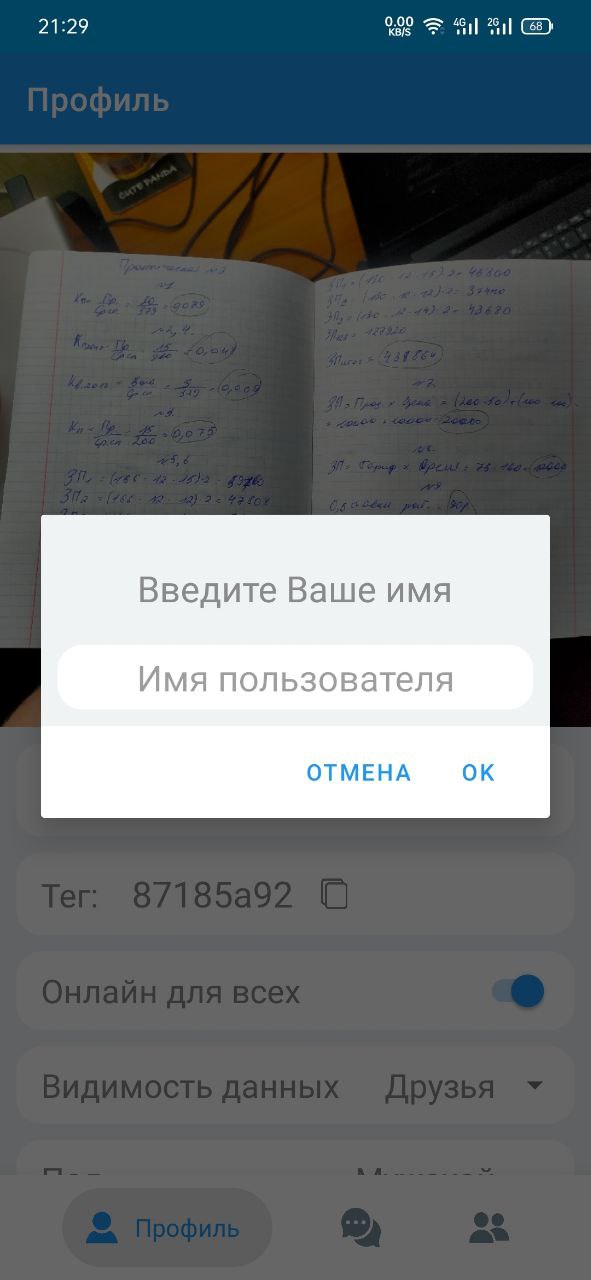


Рисунок Г.4 – Диалоговое окно с полем ввода

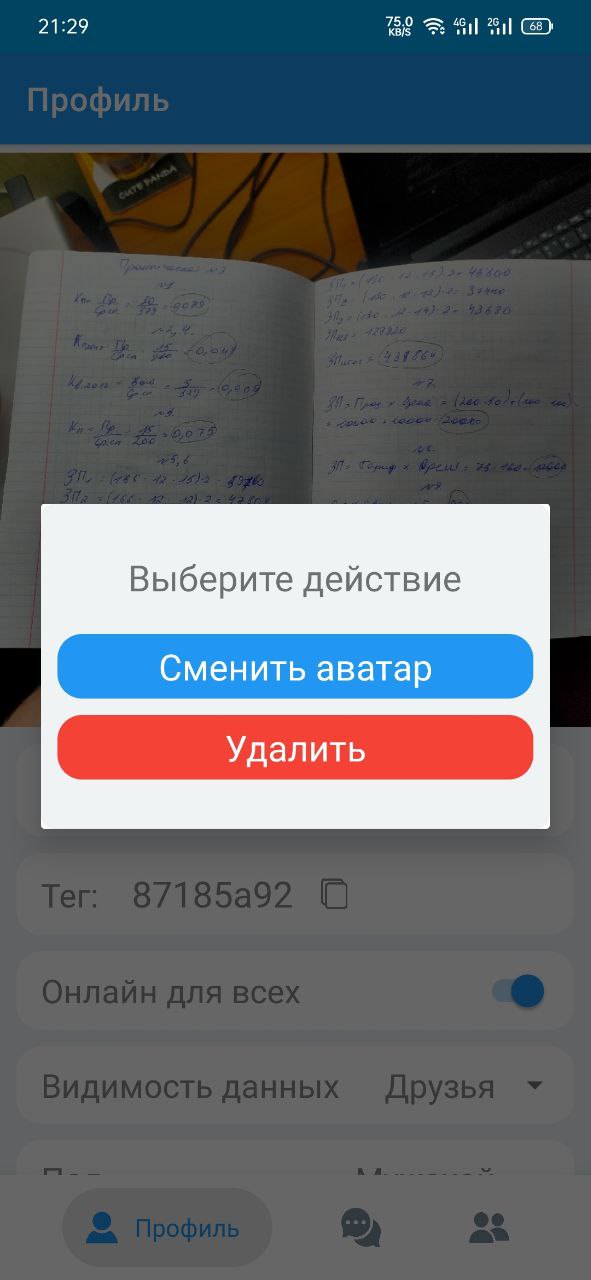


Рисунок Г.5 – Диалоговое окно с выбором действия над изображением

* + 1. **Выполнение функции выхода их профиля**

Необходимо спуститься в нижнюю часть раздела «Профиль» и нажать кнопку «Выход», чтобы выйти из профиля (рисунок Г.6).

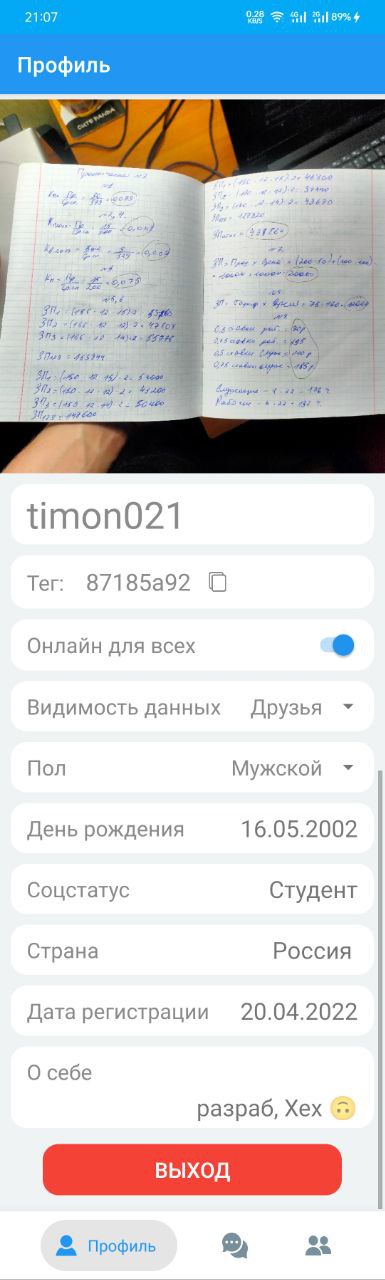


Рисунок Г.6 – Кнопка выхода из профиля

* + 1. **Выполнение функции создания группового чата**

Необходимо в разделе «Чат» нажать на кнопку открытия активности создания группового диалога (рисунок Г.7).

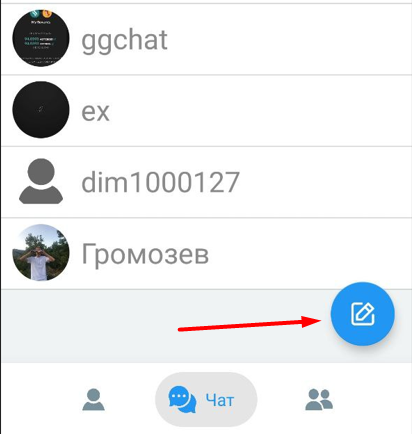


Рисунок Г.7 – Кнопка открытия активности создания группового диалога

После открытия активности необходимо выбрать участников будущего диалога, задать название в поле ввода и нажать кнопку «Создать диалог» (рисунок Г.8).

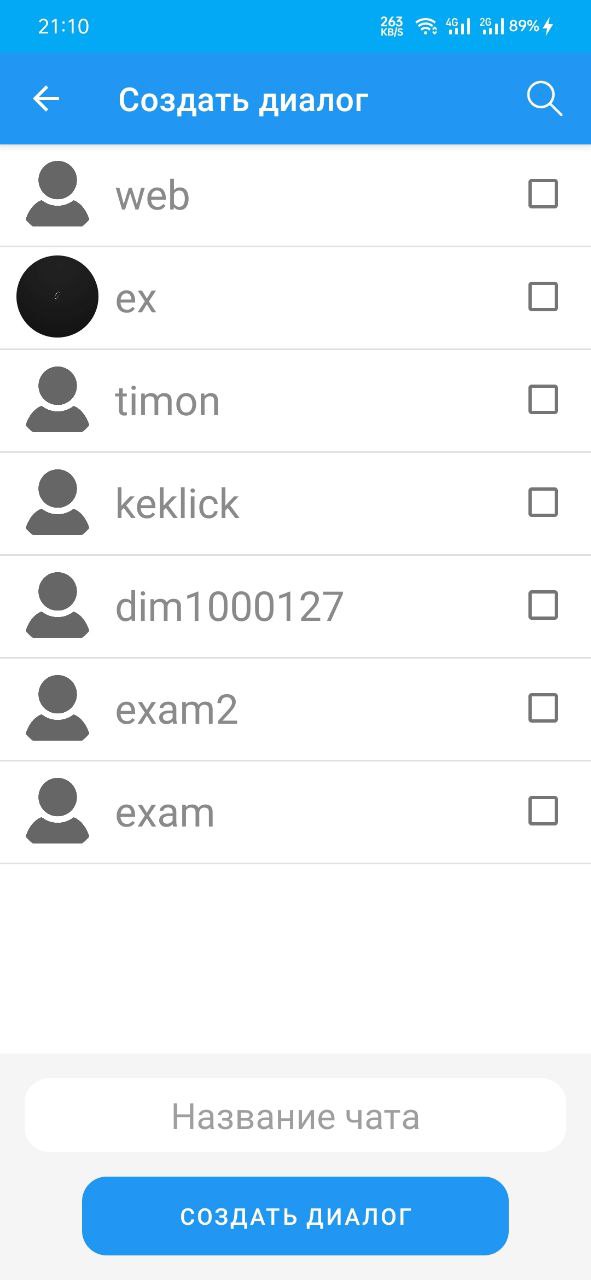


Рисунок Г.8 – Активность создания группового диалога

* + 1. **Выполнение функции отправки сообщений**

Для отправки сообщения необходимо написать текст в поле для ввода (рисунок Г.9), либо же выбрать изображение для отправки (рисунок Г.10) с помощью диалогового окна. После этого нужно нажать на кнопку отправки сообщения (рисунок Г.11).

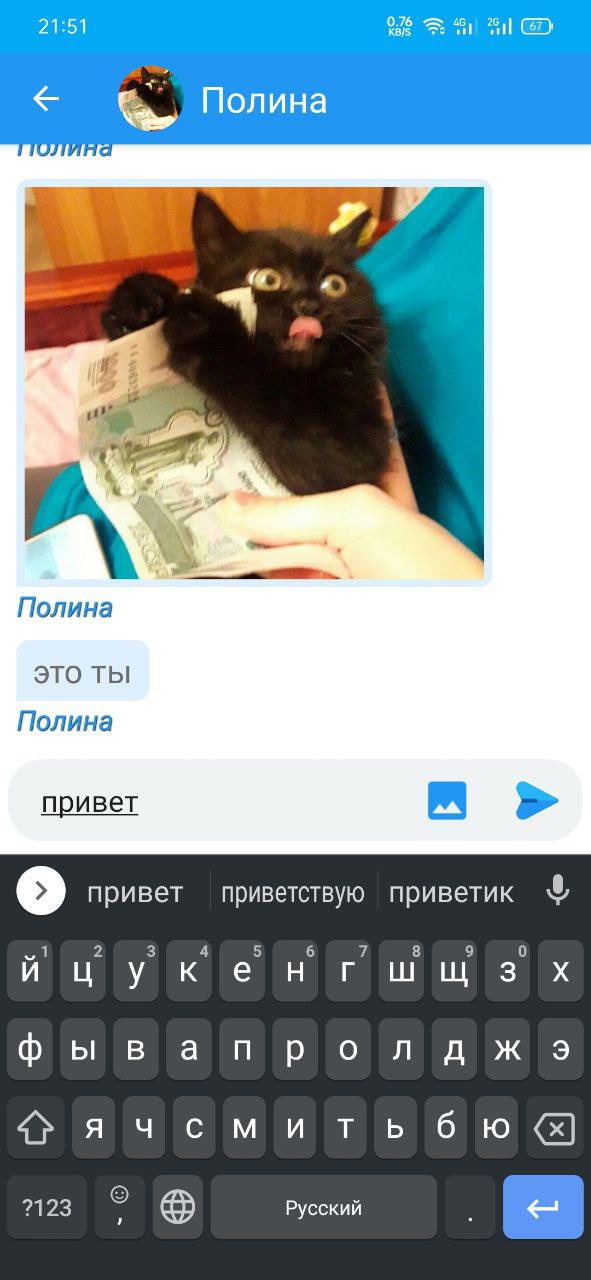


Рисунок Г.9 – Поле для ввода текста

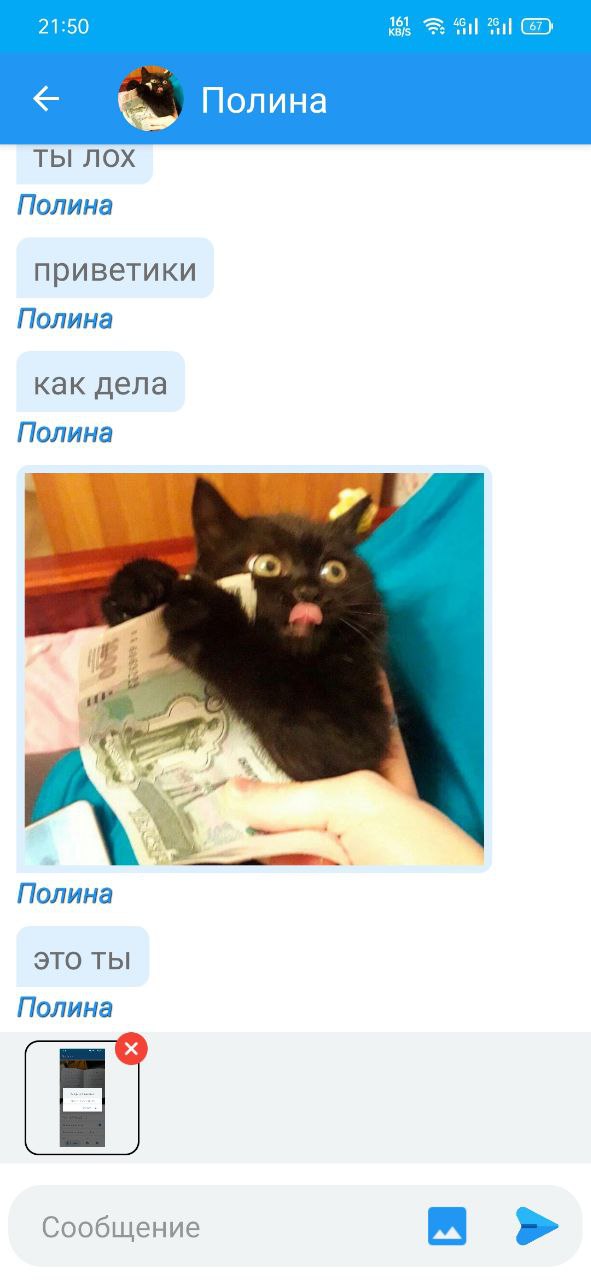


Рисунок Г.10 – Прикрепленное сообщение



Рисунок Г.11 – Кнопка для отправки сообщения

* + 1. **Выполнение функции редактирования группового диалога**

Для открытия окна редактирования группового диалога нужно нажать на кнопку информационного знака в ActionBar (рисунок Г.12).

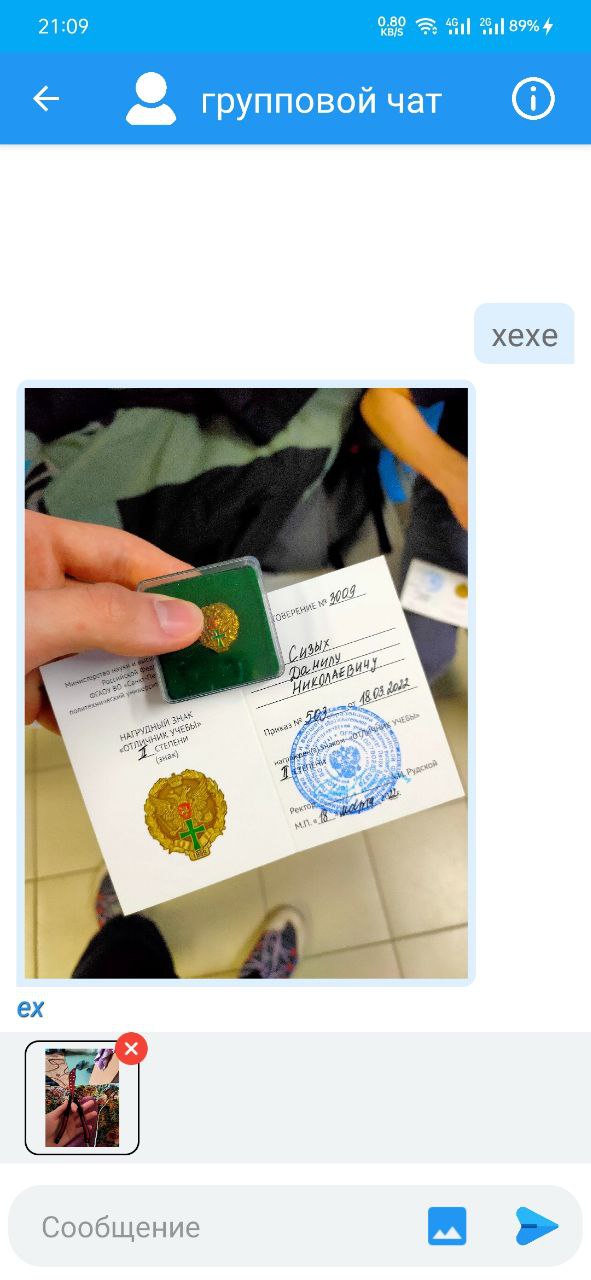


Рисунок Г.12 – ActionBar группового диалога

Откроется окно (рисунок Г.13), в котором можно:

1. сменить изображение группового диалога нажатием на него;
2. сменить название, нажав на него и вызвав диалоговое окно с полем ввода нового названия;
3. добавить нового участника, вызвав диалоговое окно (рисунок Г.14);
4. выйти из группового диалога, нажав на кнопку «Выйти из чата»;
5. удалить групповой диалог, нажав на кнопку «Удалить чат», при наличии прав администратора;
6. удалить участника группового диалога, нажав на кнопку в виде креста подле имени пользователя участника, при наличии прав администратора.

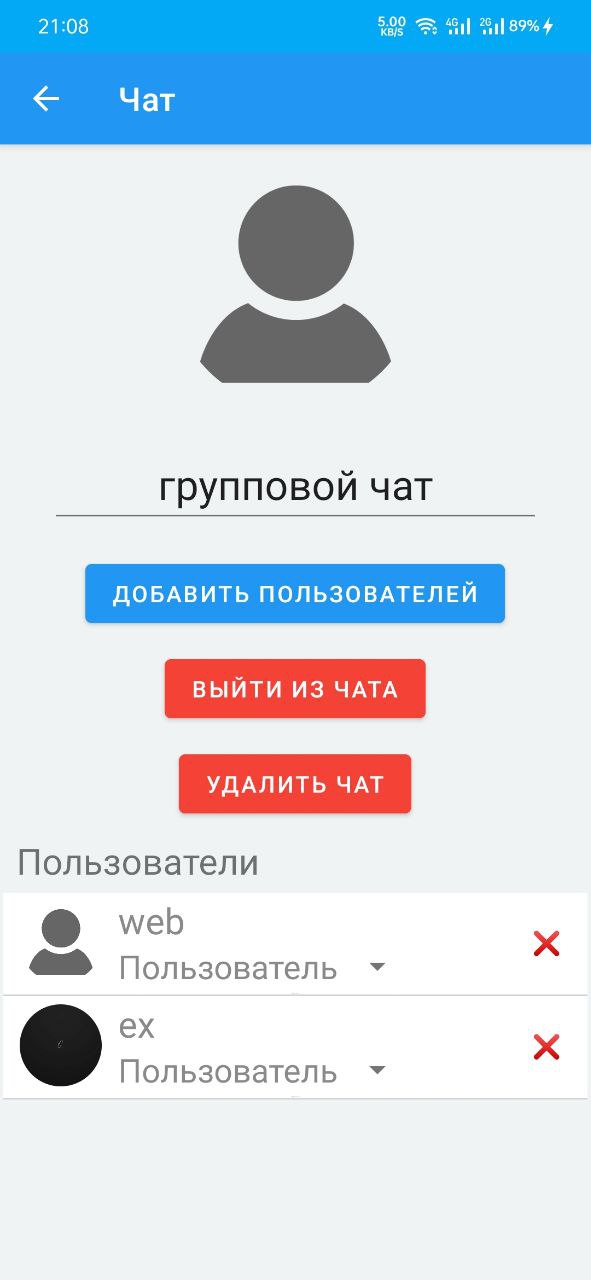


Рисунок Г.13 – Окно редактирования группового диалога

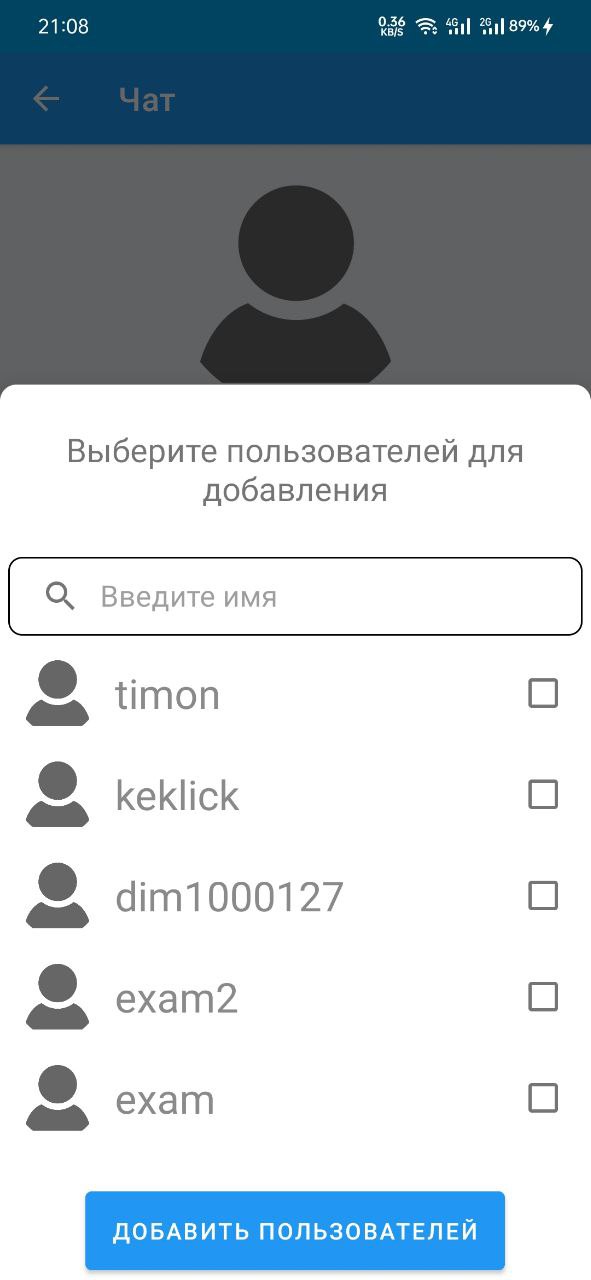


Рисунок Г.14 – Окно добавления нового участника в групповой диалог

* + 1. **Выполнение функции открытий профиля другого пользователя**

Чтобы открыть профиль другого пользователя необходимо нажать на его миниатюру в списке пользователей (рисунок Г.15). После нажатия откроется новое окно с профилем (рисунок Г.16).



Рисунок Г.15 – Список пользователей

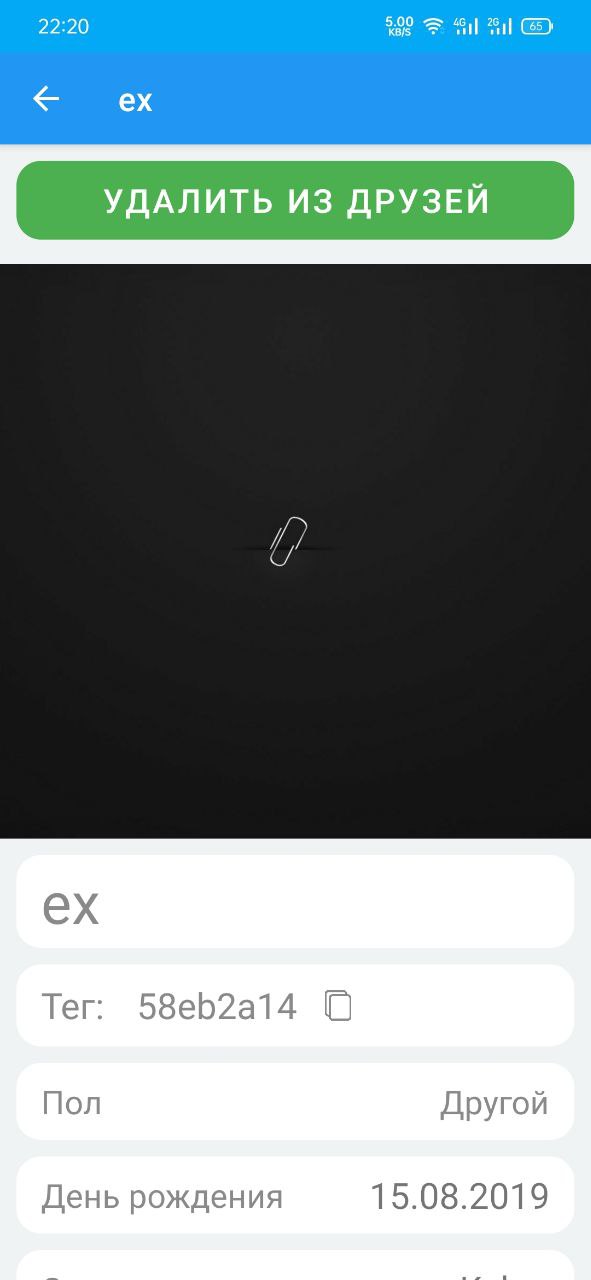


Рисунок Г.16 – Профиль другого пользователя

* 1. **Выполнение функции добавления в друзья**

Необходимо открыть профиль другого пользователя и наверху нажать на кнопку «Добавить в друзья» (рисунок Г.17). Запрос будет отправлен, после этого нужно ожидать принятия.

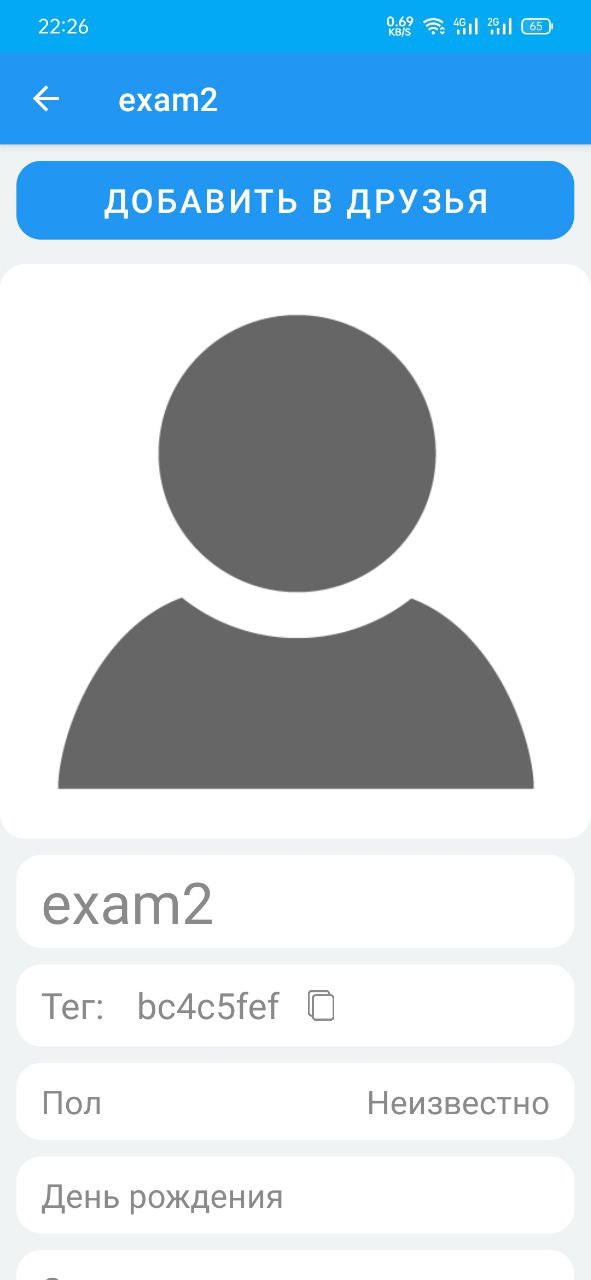


Рисунок Г.17 – Кнопка для добавления в список друзей

* 1. **Завершение работы программы**

Для завершения работы приложения необходимо выйти из приложения используя системные функции для выполнения действия «Назад» или «Домой».

1. **Сообщение оператору**

При совершении различных действий или же при разрыве соединения с сервером приложение оповещает об этом Оператора путем вывода на экран всплывающих сообщений (рисунок Г.18).

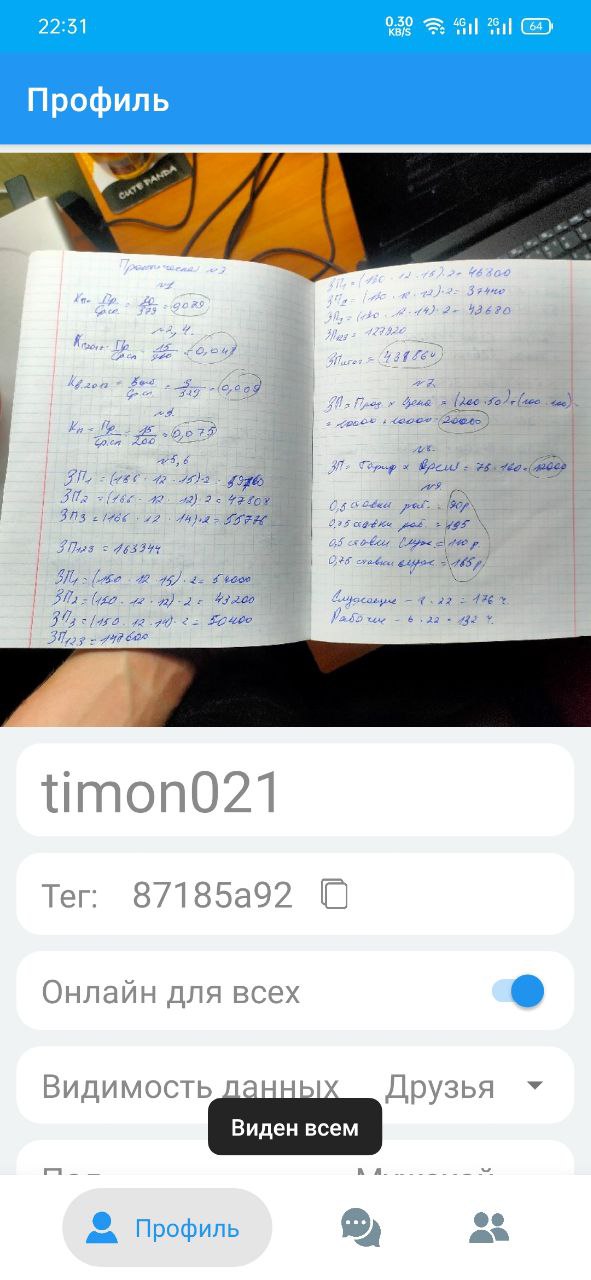


Рисунок Г.18 – Пример всплывающего сообщения о статусе онлайна пользователя