**Содержание**

[Введение 4](#_Toc181301723)

[1 Основание разработки 8](#_Toc181301724)

[2 Назначение и область применения программы 9](#_Toc181301725)

[3 Постановка задачи 10](#_Toc181301726)

[3.1 Проектирование архитектуры программного решения 10](#_Toc181301727)

[3.2 Проектирование модели базы данных 11](#_Toc181301728)

[3.3 Проектирование программных модулей решения 19](#_Toc181301729)

[4 Реализация программного решения 26](#_Toc181301730)

[4.1 Анализ средств разработки 26](#_Toc181301731)

[4.2 Обоснование выбора программного средства 29](#_Toc181301732)

[4.3 Разработка пользовательского интерфейса программного решения 31](#_Toc181301733)

[4.4 Разработка программных модулей решения 39](#_Toc181301734)

[4.5 Тестирование разработанного программного решения 46](#_Toc181301735)

[Заключение 50](#_Toc181301736)

[Список использованных источников 51](#_Toc181301737)

Введение

Мессенджер — это программа для мгновенного обмена текстовыми сообщениями, аудиозаписями, фотографиями и другими мультимедиа. Приложения устанавливаются на компьютер, смартфон, планшет и работают через интернет. Этим мессенджеры отличаются от СМС, которые отправляют и получают по мобильной сети.

Виды мессенджеров:

* текстовые мессенджеры: пользователи общаются текстовыми сообщениями;
* голосовые мессенджеры: пользователи общаются голосом, но при этом не звонят друг другу;
* видеомессенджеры: пользователи созваниваются по видео звонкам и видеоконференциям;
* многофункциональные мессенджеры: сочетают функции перечисленных выше видов мессенджеров.

Мессенджеры открывают безграничные возможности для общения и взаимодействия пользователей. С их помощью можно:

* общаться с другими один на один или в группах;
* отправлять текстовые и голосовые сообщения, фото, видео, документы, стикеры и GIF;
* звонить и устраивать видео звонки как с одним человеком, так и в конференциях;
* создавать публичные каналы и подписываться на них;
* пользоваться чат-ботами для напоминаний, скачивания файлов, рассылок и покупок.

Мессенджеры обычно работают в фоновом режиме и мгновенно доставляют сообщения, если устройство получателя подключено к сети Интернет.

Обычно для обеспечения работы мессенджера необходим специальный сервер, который обрабатывает входящие запросы от пользователей, хранит различные данные, связывает двух пользователей и т.д. Под сервером понимается два понятия:

1. Сервер — сетевой компьютер, обрабатывающий запросы от других компьютеров в локальной или глобальной сети. Как правило, сервер выделен из всей группы компьютеров, подключенных к единой сети, работает автономно, без участия человека, за исключением первичной настройки.
2. Программный сервер — это специализированное программное обеспечение, которое предоставляет различные услуги и ресурсы клиентам по сети. Программные серверы могут быть установлены на любом компьютере, подходящем для выполнения серверных задач. Они работают на операционных системах серверов или на обычных операционных системах.

Сервер включает мощный компьютер и программное обеспечение. Компьютер оснащён специальными комплектующими и деталями, рассчитанными на беспрерывную работу под большими нагрузками. Задач, которые можно поручить серверам, очень много. Серверы обрабатывают запросы пользователей и могут, например, отправлять и принимать письма, обеспечивать доступ к сайтам и базам данных, идентифицировать пользователей.

Для структуризации хранимых данных используются так называемые базы данных.

База данных (БД) — это набор данных, который как-то структурирован. В базе данных может храниться что угодно: фотографии на телефоне, расположение игроков на карте в компьютерной игре, номера заказов в сервисе такси или доставки продуктов. Основная работа базы данных — сохранение, изменение и быстрый поиск информации.

Работу с базой данных невозможно представить без систем управления базами данных.

Система управления базами данных (СУБД) — это набор программного обеспечения, который позволяет работать с базами данных: создавать, удалять и редактировать их.

Во время прохождения производственной практики в ООО «РСП-М» было дано задание, заключавшееся в разработке корпоративного мессенджера, который позволит организовать взаимодействие между сотрудниками организации, а также увеличит безопасность хранения данных, связанных с работой предприятия. Проблемы безопасности связаны с тем, что участились случаи утечки различных данных с серверов компаний, что может привести к распространению персональных данных, которыми могут воспользоваться сторонние люди для своих целей. Следовательно, использование внутреннего мессенджера обеспечит безопасность и надежность данных, связанных с организацией.

Исходя из задания, полученного на практике, была сформулирована цель курсового проекта: «Реализация прототипа корпоративного мессенджера для ООО «РСП-М»».

В соответствии с целью работы были выделены следующие задачи курсового проекта:

1. Изучить требования заказчика и провести анализ предметной области – информация, передаваемая между сотрудниками предприятия.
2. Построить модель базы данных.
3. Построить архитектуру прототипа корпоративного мессенджера.
4. Реализовать пользовательский интерфейс мессенджера.
5. Разработать функциональные возможности серверной части.
6. Разработать функциональные возможности клиентской части.
7. Провести тестирование и выполнить отладку разработанного программного решения.
8. Оформить техническую документацию в виде пояснительной записки на курсовой проект в соответствии со всеми нормами ЕСПД.

1 Основание разработки

Основанием для разработки корпоративного мессенджера является задание, полученное на курсовой проект по профессиональному модулю ПМ.02 Осуществление интеграции программных модулей междисциплинарного курса МДК 02.01 Технология разработки программного обеспечения.

Руководителем курсового проекта является преподаватель специальных дисциплин, председатель цикловой методической комиссии ОПД и ПМ специальности «Информационные системы и программирование» Брюханова Яна Олеговна.

Заказчиком на разработку корпоративного мессенджера является организация ООО «РСП-М».

Требования заказчика заключалось в разработке программного приложения, предназначенного для общения сотрудников во время работы, обсуждения рабочих моментов и передачи документов.

Преимуществами разработанного приложения являются:

1. Интуитивно-понятный пользовательский интерфейс.
2. Организация общения между сотрудниками предприятия посредством передачи данных и сообщений.
3. Обеспечение надежности и безопасности передаваемой информации.

2 Назначение и область применения программы

Разработанное программное решение является универсальным и может использоваться в деятельности не только ООО «РСП-М», но и в других организациях.

Область применения программы охватывает средние и крупные компании, где важна защита корпоративных данных и удобный интерфейс для общения. Мессенджер также может быть полезен для государственных и частных организаций, которые нуждаются в централизованной и контролируемой платформе для внутренней коммуникации. Это позволит отделить личное общение от рабочих моментов, сотрудникам не придется создавать сторонние рабочие профили в социальных сетях.

Конечными пользователями приложения могут быть все сотрудники организации.

Использование данного мессенджера позволит выполнить следующую работу:

* организовать обмен сообщениями;
* организовать обмен файлами;
* организовать систему управления задачами
* осуществить мониторинг деятельности сотрудников.

3 Постановка задачи

3.1 Проектирование архитектуры программного решения

На рисунке 1 представлена архитектура корпоративного мессенджера в виде диаграммы прецедентов (UseCase):



Рисунок 1 – Архитектура корпоративного мессенджера

Если на предприятие устроился новый сотрудник, для входа в приложение ему необходимо создать новый профиль для дальнейшего взаимодействия с системой.

После входа в свой профиль пользователь видит список чатов и изображение своего профиля. Выбрав определенный чат, перед ним появляется история сообщений того чата, который был выбран. В специальной текстовой строке пользователь может набрать сообщение и отправить его другому пользователю. Также как и отправлять клиент может соответственно получать сообщения.

Если вернуться в список чатов, то можно увидеть кнопку, позволяющую создавать новый чат.

Нажав на изображения своего профиля, пользователь может изменить данные своего профиля.

Ход реализации корпоративного мессенджера можно разделить на следующие этапы:

1. Определение архитектуры приложения.
2. Создание базы данных в SSMS, создание таблиц для хранения данных пользователей, истории сообщений, списка друзей.
3. Проектирование интерфейса клиентского приложения, различных аспектов отправки и синхронизации данных.
4. Создание серверного приложения для работы всей системы, продумывание всех возможный действий пользователя для предотвращения ошибок серверной части.
5. Тестирование всех разработанных возможностей в различных условиях, устранение выявленных проблем.

3.2 Проектирование модели базы данных

Модель базы данных — это логическая структура, определяющая организацию, хранение и связь данных в системе управления базами данных.

Предметная область — это конкретная область знаний или деятельности, которую необходимо описать и автоматизировать с помощью информационной системы. Она включает все сущности, процессы и правила, характерные для данной области, и определяет, какие данные и функции будут иметь значение в создаваемой системе.

Модель базы данных разработана на основе анализа требований заказчика и изучения предметной области – информация, передаваемая между сотрудниками предприятия.

Модель базы данных представлена в виде ER-диаграммы, которая была создана при помощи Microsoft Visio.

ER-диаграмма — это графическое представление данных и их взаимосвязей в предметной области. Она используется для моделирования структуры базы данных на этапе проектирования и помогает визуализировать, как различные сущности взаимодействуют друг с другом.

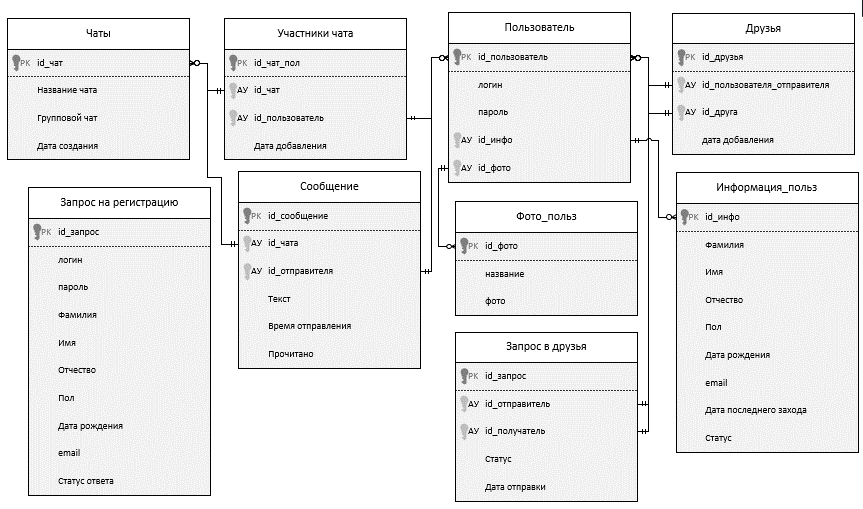


Рисунок 2 – Модель базы данных

В разработанной модели базы данных объектами являются таблицы:

* User (Пользователь) – предназначена для хранения учётных данных зарегистрированных пользователей;
* PhotoUser (Фото пользователя) – предназначена для хранения фото профиля пользователей;
* InfoUser (Информация пользователя) – предназначена для хранения основной информации о зарегистрированных пользователях;
* FriendRequests (Запросы в друзья) – предназначена для хранения списка запросов добавления в друзья;
* Friend (Друзья) – предназначена для хранения списка друзей пользователей;
* RegistrarionRequests (Запросы на регистрацию) – предназначена для хранения запросов на регистрацию новых пользователей;
* ChatParticipants (Участники чата) – предназначена для хранения списка участников чата;
* Chat (Чаты) – предназначена для хранения информации о чате;
* Message (Сообщение) – предназначена для хранении информации об отправленных сообщениях пользователей.

Вышеперечисленные таблицы были созданы в системе управления базами данных Microsoft SQL Server и являются исходными.

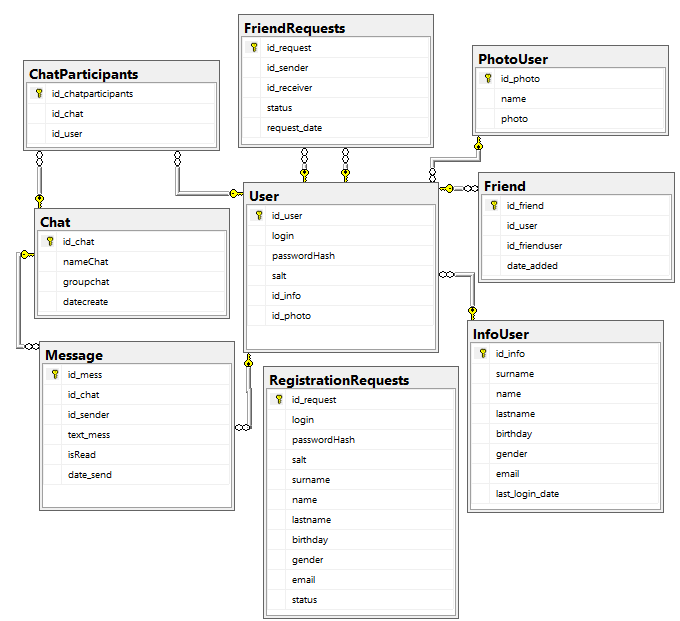


Рисунок 3 – Физическая модель базы данных

Сущность «RegistrationRequests» не имеет связей, так как предназначена для хранения запросов на регистрацию профиля, которые никак не связаны с остальными сущностями.

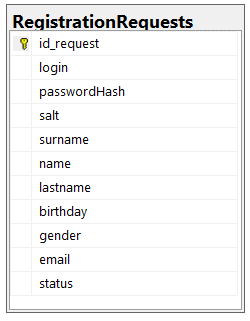


Рисунок 4 – Сущность «RegistrationRequests»

Сущность «PhotoUser» связана с сущностью «User» типом связи «один-к-одному», так как один пользователь может иметь только одну фотографию профиля.

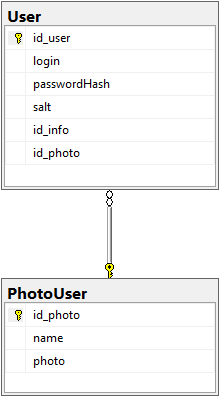


Рисунок 5 – Ассоциация «User» и «PhotoUser»

Сущность «InfoUser» связана с сущностью «User» типом связи «один-к-одному», так как один пользователь может иметь только одну информацию о своём профиле.

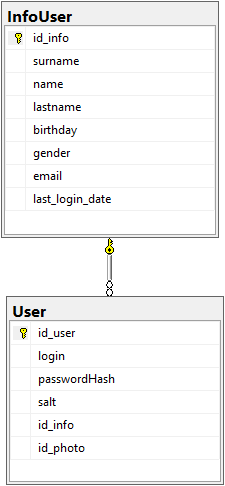


Рисунок 6 – Ассоциация «User» и «InfoUser»

Сущность «Friend» связана с сущностью «User» типом связи «один-ко-многим», так как один пользователь может иметь несколько друзей.

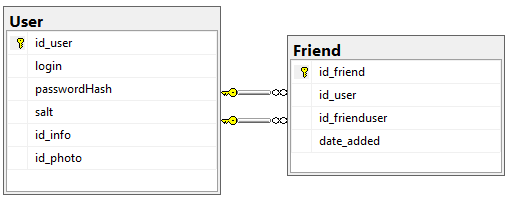


Рисунок 7 – Ассоциация «User» и «Friend»

Сущность «User» связана с сущностью «FriendRequests» типом связи «один-ко-многим», так как один пользователь может иметь несколько запросов в друзья.

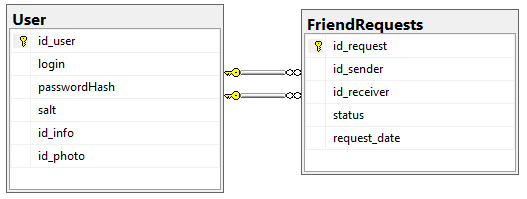


Рисунок 8 – Ассоциация «User» и «FriendRequests»

Сущность «User» связана с сущностью «ChatParticipants» типом связи «один-к-одному», так как в одном чате пользователь может быть участником только один раз.

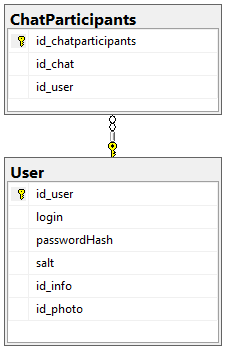


Рисунок 9 – Ассоциация «User» и «ChatParticipants»

Сущность «ChatParticipants» связана с сущностью «Chat» типом связи «один-ко-многим», так как в одном чате несколько участников.

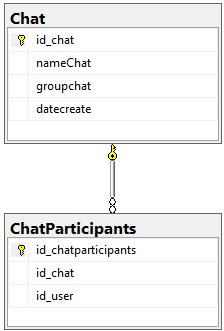


Рисунок 10 – Ассоциация «Chat» и «ChatParticipants»

Сущность «Message» связана с сущностью «User» типом связи «один-к-одному», так как сообщение может принадлежать только одному пользователю.

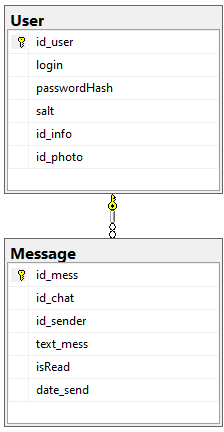


Рисунок 11 – Ассоциация «Message» и «User»

Сущность «Message» связана с сущностью «Chat» типом связи «один-к-одному», так как сообщение может быть привязано только к одному чату.

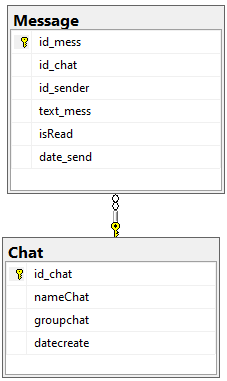


Рисунок 12 – Ассоциация «Message» и «Chat»

В ходе проектирования базы данных была проведена нормализация всех таблиц до третьей нормальной формы (3НФ). Это решение было обусловлено необходимостью обеспечения целостности данных, минимизации избыточности и повышения производительности системы.

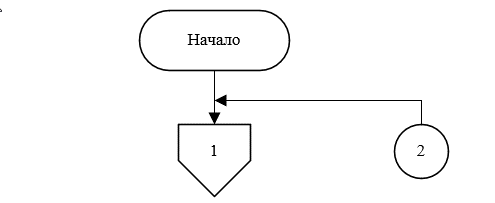
3.3 Проектирование программных модулей решения

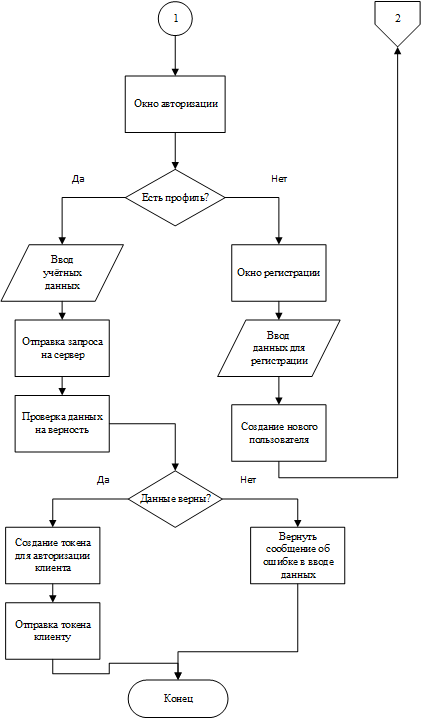
Разработанный корпоративный мессенджер имеет следующие функциональные возможности:

1. Регистрация нового пользователя – данная функция позволяет зарегистрировать нового пользователя.
2. Изменение пользователя – данная функция позволяет пользователям изменять свой профиль.
3. Добавление чата – с помощью данной функции пользователь может создать новый чат с другим пользователем.
4. Добавление нового сообщения – данная функция позволяет хранить историю сообщений пользователей в базе данных.
5. Просмотр чатов – с помощью данной функции пользователь может просмотреть список своих чатов.
6. Просмотр сообщений чата – данная функция позволят пользователю просматривать историю сообщений чата.
7. Добавление заявки в друзья – с помощью данной функции пользователь может отправлять заявку в друзья другому клиенту.
8. Добавление друга – данная функция позволяет принять заявку для добавления другого пользователя в друзья.
9. Удаление заявки в друзья – данная функция позволят отклонить заявку в друзья.
10. Поиск пользователей – с помощью данной функции пользователь может искать других клиентов по фамилии, имени или отчеству.

Алгоритм — это конечная последовательность действий или инструкций, предназначенных для решения определённой задачи или достижения поставленной цели. Алгоритмы являются основой программирования и компьютерных наук, так как они позволяют описать путь решения задачи, который затем можно перевести на язык программирования и выполнить на компьютере. Блок-схема — это графическое представление алгоритма, где каждый шаг обозначается специальной геометрической фигурой (блоком) и соединяется стрелками, показывающими последовательность выполнения шагов. Блок-схемы помогают визуализировать логику алгоритма и упрощают его понимание.

Блок-схема 1 – Авторизация

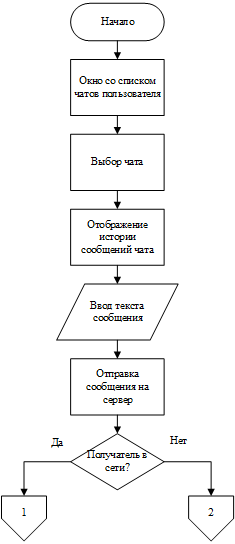


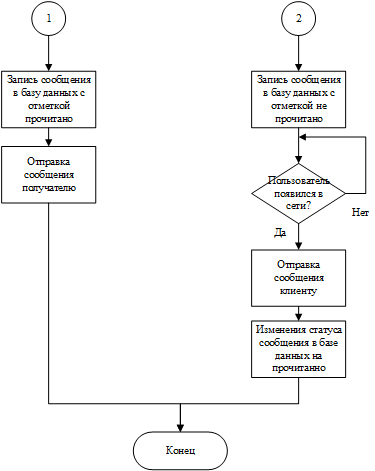


Блок-схема 1 описывает процесс авторизации пользователя при загрузке программы. Как только пользователь запускает приложение перед ним открывается окно авторизации. Тут возможно два варианта развития сценария:

1. У клиента есть зарегистрированный профиль, и он вводит свои учётные данные. Эти данные отправляются на сервер, где сравниваются со сведеньями хранящимися в базе данных. Если они совпадают, то сервер генерирует специальный токен и отправляет его клиенту. Используя этот токен, пользователь может войти в систему. Иначе сервер вернёт запрос с ошибкой.
2. У клиента нет профиля, он нажимает на кнопку регистрации. Заполнив все необходимые данные, создаётся запрос на регистрацию нового профиля, который будет обработан сервером. Гость будет возвращён на окно авторизации.

Блок-схема 2 – Отправка сообщения

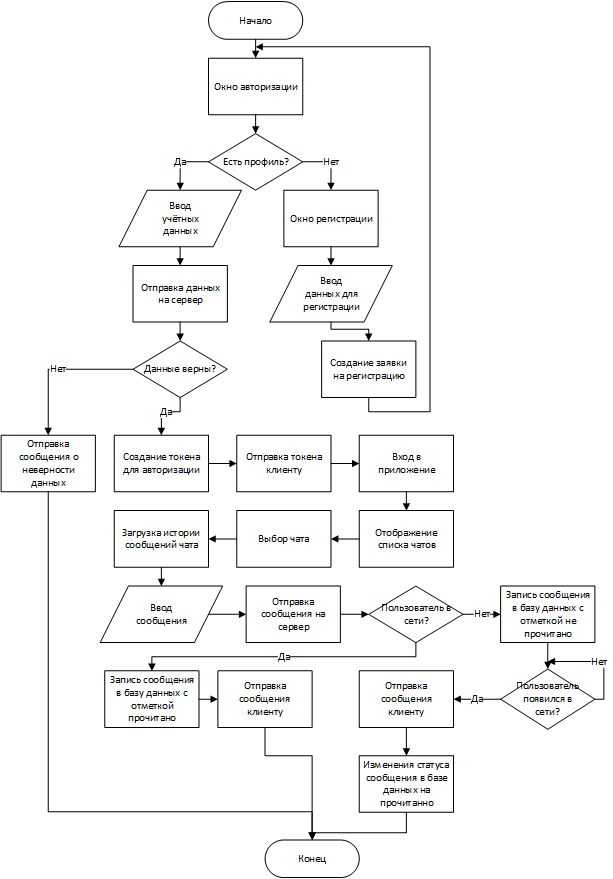




Блок-схема 2 представляет собой процесс отправки уже авторизированным пользователем сообщения другому пользователю. Выбрав чат из списка других чатов, перед клиентом появляется история сообщений. Введя текст в специальное поле, пользователь нажимает кнопку для отправки сообщения. Сообщение передаётся на сервер, где возможны следующие варианты событий:

1. Получатель в сети, тогда сообщение заносится в базу данных с отметкой прочитано, после чего отправляется клиенту
2. Получатель не в сети, тогда сообщение заносится в базу данных с отметкой не прочитано, и как только пользователь появится в сети, ему будет отправлено данное сообщение и статус сообщения в базе данных изменится на прочитано.

Блок-схема 3 – Общая логика приложения



Блок-схема 3 представляет общую логику всего приложения. При запуске приложения гость может как войти в систему, так и отправить заявку на регистрацию нового профиля. После входа в систему пользователь видит список своих чатов. Выбрав необходимый чат, будет прогружена история сообщений. Клиент, набрав текст в специальном поле, может отправить сообщение другому пользователю чата.

4 Реализация программного решения

4.1 Анализ средств разработки

Любое программное решение возможно разработать при помощи следующих технологий и программных средств:

1. Языки программирования:

* Java - поддерживает кроссплатформенную разработку благодаря JVM (Java Virtual Machine). Использует Swing и JavaFX для создания интерфейсов, что позволяет запускать приложения на Windows, macOS и Linux без изменения кода;
* Python - отличается простотой синтаксиса и богатым выбором библиотек, которые позволяют создавать приложения для Windows, macOS и Linux. Часто используется для создания прототипов и небольших приложений с графическим интерфейсом;
* C++ - быстрый и производительный язык с библиотеками, такими как Qt и wxWidgets, для кроссплатформенной разработки. Подходит для ресурсоёмких приложений, таких как графические редакторы и игры, благодаря возможности прямого управления памятью и аппаратными ресурсами;
* Swift - основной язык для разработки приложений под macOS. Поддерживает удобные инструменты для создания приложений с использованием Cocoa и SwiftUI. Предпочтителен для создания нативных приложений для устройств Apple благодаря отличной производительности и интеграции с экосистемой Apple.

1. Системы управления базами данных:

* SQLite - легковесная встраиваемая СУБД, которая не требует отдельного серверного ПО и отлично подходит для небольших десктопных приложений. Она хранит данные в одном файле и широко поддерживается языками программирования, что делает её удобной для быстрого и локального хранения данных;
* MySQL - одна из самых популярных реляционных СУБД с открытым исходным кодом. Часто используется для приложений, где требуется больше функциональности и масштабируемости, чем может предложить SQLite. MySQL можно использовать как локально, так и на сервере для десктопных приложений, требующих подключения к удалённым базам;
* PostgreSQL - продвинутая реляционная СУБД с открытым исходным кодом, известная своей надёжностью и поддержкой сложных операций и транзакций. Отличный выбор для сложных приложений с высокими требованиями к целостности данных и возможностью масштабирования;
* MongoDB - документо-ориентированная NoSQL база данных, которая хранит данные в формате JSON-подобных документов. MongoDB часто используется для приложений, где структура данных может меняться и нужна гибкость, например, для хранения и обработки сложных объектов. Она поддерживает кроссплатформенную разработку и может использоваться как локально, так и на сервере.

1. Технологии доступа к данным:

* Dapper - легковесный ORM для .NET, отличающийся высокой производительностью и простотой. Dapper не генерирует сложных SQL-запросов автоматически, но помогает легко управлять ими, возвращая данные в виде объектов. Это хороший выбор для случаев, когда важны быстродействие и контроль над SQL-запросами;
* JDBC (Java Database Connectivity) - стандартный API для Java, позволяющий подключаться к реляционным базам данных. JDBC обеспечивает прямую связь с базой и поддерживает различные типы драйверов для работы с разными СУБД. Это один из самых универсальных и гибких способов работы с базами данных в приложениях на Java;
* ADO.NET - основная технология доступа к данным в экосистеме .NET. ADO.NET предоставляет низкоуровневый доступ к базам данных и позволяет напрямую работать с SQL-запросами и хранимыми процедурами. Это гибкое и производительное решение, которое подходит для сценариев, где важен полный контроль над взаимодействием с базой данных, и часто используется в корпоративных и высоконагруженных приложениях.

1. Интегрированные среды разработки (IDE)

* JetBrains Rider - многофункциональная IDE для .NET и C#, разработанная компанией JetBrains. Rider сочетает в себе мощные инструменты для разработки и отладки, поддерживает различные платформы и фреймворки, включая .NET Core и ASP.NET. Rider также обеспечивает отличную поддержку рефакторинга и интеграцию с системами контроля версий;
* Eclipse - открытая IDE, популярная среди разработчиков на Java, но также поддерживающая различные языки программирования через плагины. Eclipse имеет множество инструментов для разработки, включая поддержку работы с базами данных и интеграцию с системами контроля версий. Она также предоставляет возможность создания кроссплатформенных приложений;
* NetBeans - открытая IDE, которая поддерживает несколько языков, включая Java, PHP и C/C++. Она включает в себя множество инструментов для разработки десктопных и веб-приложений, таких как интеграция с базами данных и система контроля версий. NetBeans подходит для быстрого прототипирования и разработки приложений с графическим интерфейсом.

4.2 Обоснование выбора программного средства

Для написания кода приложения был выбрал язык программирования C#, так как он является отличным выбором для разработки приложений благодаря своему современному и выразительному синтаксису, что упрощает чтение и написание кода. Также из плюсов можно выделить:

1. Поддержка .NET и WPF: C# тесно интегрирован с .NET Framework и WPF (Windows Presentation Foundation), что позволяет легко создавать современные и отзывчивые графические интерфейсы. Это особенно важно для десктопных приложений, так как вы сможете быстро разрабатывать и внедрять удобные пользовательские интерфейсы.
2. Богатая экосистема библиотек и фреймворков: C# предлагает множество библиотек и фреймворков, таких как Entity Framework для работы с базами данных, SignalR для реализации реального времени (например, для обмена сообщениями), а также различные библиотеки для сетевого взаимодействия. Это значительно ускоряет процесс разработки, так как многие функции уже реализованы.
3. Кроссплатформенность с .NET Core: С переходом на .NET Core (сейчас .NET 5 и выше), C# стал кроссплатформенным, что позволяет разрабатывать приложения, которые могут работать не только на Windows, но и на macOS и Linux. Это открывает больше возможностей для распространения вашего мессенджера и использования на разных устройствах.
4. Отличная производительность: C# компилируется в промежуточный код (IL), который выполняется на .NET Common Language Runtime (CLR). Это обеспечивает хорошую производительность и позволяет использовать различные оптимизации на уровне платформы.
5. Широкая поддержка сообщества и документации: C# имеет большое и активное сообщество разработчиков, что облегчает поиск решений для различных проблем, а также доступ к множеству учебных материалов, библиотек и инструментов.

Для создания интерфейса приложения была выбрана технология WPF. WPF (Windows Presentation Foundation) является отличным выбором для создания пользовательских интерфейсов по нескольким причинам:

1. Разделение логики и интерфейса: WPF поддерживает паттерн MVVM (Model-View-ViewModel), который позволяет отделить бизнес-логику от интерфейса. Это облегчает тестирование, поддержку и развитие приложений, так как изменения в одном компоненте не затрагивают другие.
2. Современные интерфейсы: WPF предоставляет мощные инструменты для создания богатых и отзывчивых пользовательских интерфейсов с использованием векторной графики, анимации и стилей. Это позволяет разработчикам создавать привлекательные и интуитивно понятные приложения, которые соответствуют современным требованиям дизайна.
3. Поддержка данных: WPF обладает встроенной поддержкой привязки данных, что упрощает синхронизацию между пользовательским интерфейсом и источниками данных. Это позволяет автоматизировать обновление интерфейса при изменении данных и значительно упрощает работу с данными.
4. Кроссплатформенные возможности: хотя WPF изначально предназначен для Windows, с выходом .NET Core и проектов, таких как .NET MAUI, у разработчиков появилась возможность использовать WPF в кроссплатформенных приложениях, что расширяет возможности разработки.
5. Интеграция с XAML: WPF использует XAML (Extensible Application Markup Language) для описания пользовательского интерфейса, что позволяет разработчикам удобно и быстро разрабатывать интерфейсы, отделяя визуальное представление от логики. Это также упрощает совместную работу дизайнеров и разработчиков.

При выборе системы управления базами данных (СУБД) для корпоративного мессенджера важно учитывать множество факторов, включая производительность, безопасность и интеграцию с используемыми технологиями. Microsoft SQL Server (MSSQL) выделяется как один из лучших вариантов благодаря своим мощным функциональным возможностям и удобству в работе. Ниже приведены ключевые причины, почему MSSQL является оптимальным выбором для вашего приложения.

1. Производительность и масштабируемость: MSSQL предлагает высокую производительность и возможность масштабирования, что делает его подходящим для приложений, требующих обработки больших объёмов данных и высокой нагрузки. Он поддерживает различные механизмы оптимизации и индексирования, что позволяет ускорять выполнение запросов.
2. Интеграция с .NET и C#: MSSQL хорошо интегрирован с экосистемой .NET, что упрощает разработку приложений на C#. Существуют удобные инструменты и библиотеки, такие как Entity Framework, которые позволяют легко взаимодействовать с базой данных и управлять данными без необходимости написания сложных SQL-запросов.
3. Безопасность и управление доступом: MSSQL предлагает многоуровневую безопасность, включая аутентификацию, авторизацию и шифрование данных. Эти функции позволяют защитить вашу базу данных от несанкционированного доступа и обеспечивают соответствие требованиям безопасности.

4.3 Разработка пользовательского интерфейса программного решения

Работа с приложением начинается с загрузочного окна (рисунок 13), после которого происходит переход на окно авторизации.



Рисунок 13 – Загрузочное окно

В окне авторизации (рисунок 14), пользователь вводит свои учётные данные для входа в приложение. Также здесь он может перейти на форму регистрации нового профиля, нажав на кнопку «Регистрация».

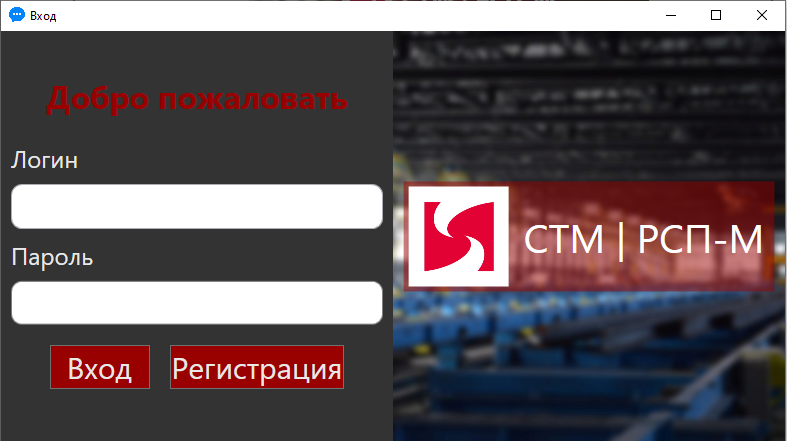


Рисунок 14 – Окно авторизации

Регистрация пользователя разделяется на две формы: в первой пользователю необходимо ввести свои персональные данные (рисунок 15); во второй нужно придумать новые учетные данные (рисунок 16).

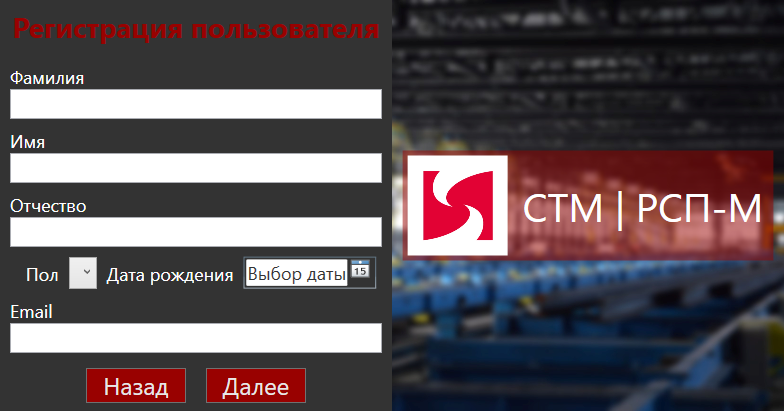


Рисунок 15 – Первая форма регистрации

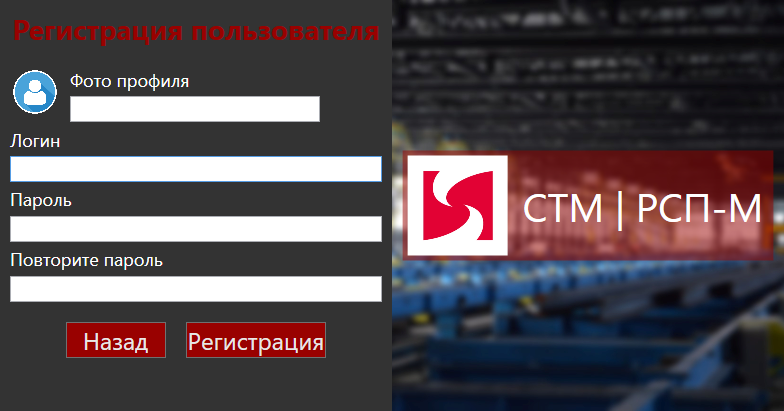


Рисунок 16 – Вторая форма регистрации

Если пользователь не ввёл все необходимые данные, то появится сообщение с ошибкой (рисунок 17), предупреждающее пользователя о некорректности введённой информации.

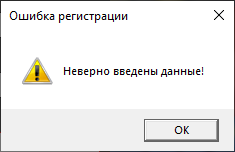


Рисунок 17 – Сообщение об ошибке

Если вовремя регистрации или авторизации пользователя, сервер будет не доступен по некоторым причинам, то появится уведомление с ошибкой подключение к серверу(рисунок 18).

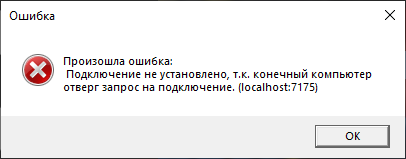


Рисунок 18 – Ошибка подключения к серверу

После авторизации пользователь попадает в главное меню приложения (рисунок 19), где может увидеть список своих чатов, а также краткую информацию о каждом профиле.

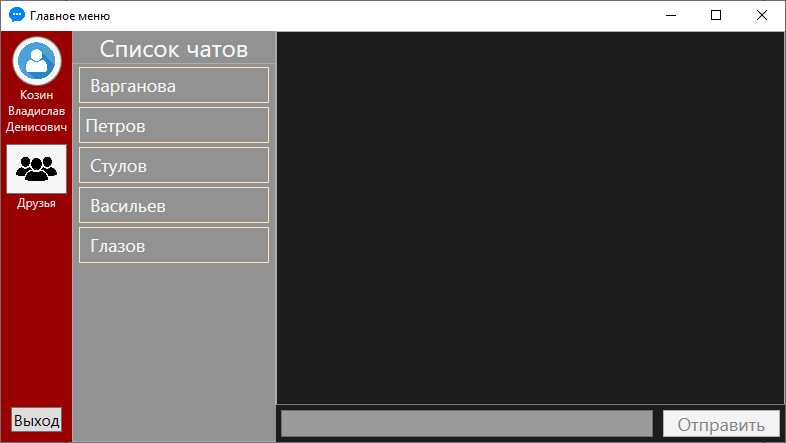


Рисунок 19 – Главное окно приложения

В левой части окна имеется несколько кнопок. Нажимая на первую кнопку, откроется окно с полной информацией о профиле пользователя (рисунок 20).

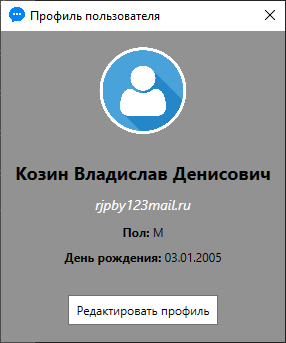


Рисунок 20 – Окно профиля пользователя

В этом окне нажав на кнопку «Редактировать профиль», пользователь может перейти на форму изменения информации о своём профиле (рисунок 21).

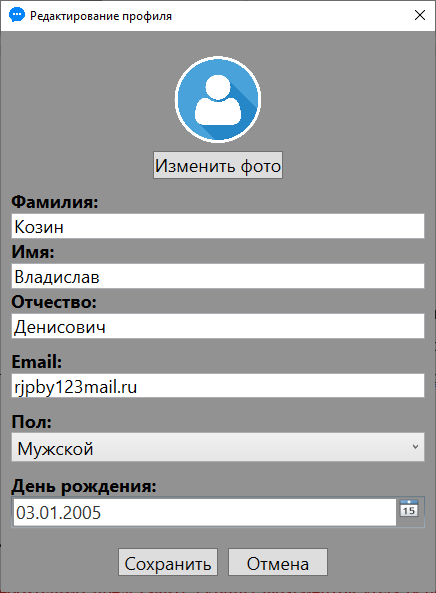


Рисунок 21 – Окно изменения профиля

На главном окне, нажав на кнопку «Друзья», откроется соответствующая форма (рисунок 22), где пользователь может просмотреть своих друзей, а также увидеть список заявок на добавление в друзья (рисунок 23).

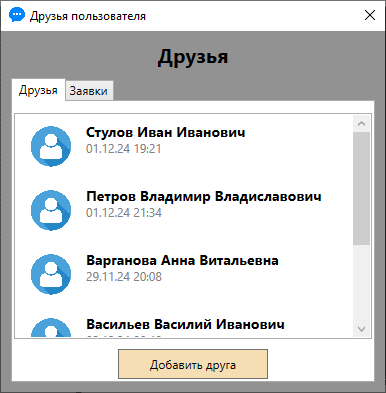


Рисунок 22 – Окно «Друзья»

Во вкладке «Заявки» пользователь видит информацию о клиенте, отправившего заявку в друзья. С этой же вкладки клиент может принять или отклонить заявку на добавление.

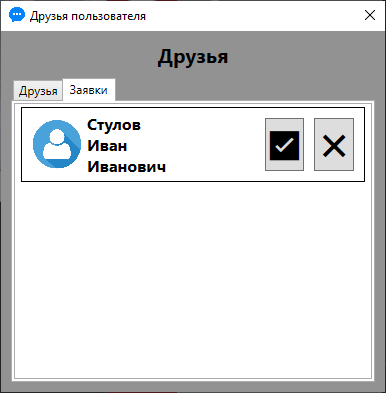


Рисунок 23 – Вкладка «Заявки»

Также, нажав на кнопку во вкладке «Друзья», откроется окно (рисунок 24), где пользователь может путём поиска по фамилии, имени или отчеству, найти других пользователей.

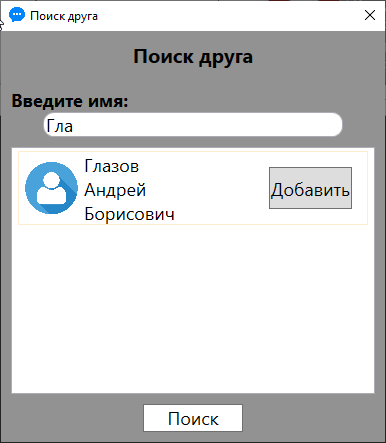


Рисунок 24 – Окно поиска пользователей

После нажатия кнопки «Добавить», выбранному пользователю отправится заявка на добавление в друзья и придет уведомление об успешной отправке (рисунок 25).

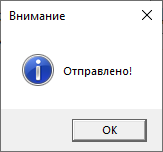


Рисунок 25 – Уведомление об отправке заявки

Вернувшись на главную форму, пользователь может видеть список своих чатов. Нажав на один из них, в правой части окна будут загружены сообщения выбранной беседы.

Если сообщение было отправлено пользователем, то будет отображаться справа.

Если же сообщение было получено пользователем, то будет отображаться слева (рисунок 26).

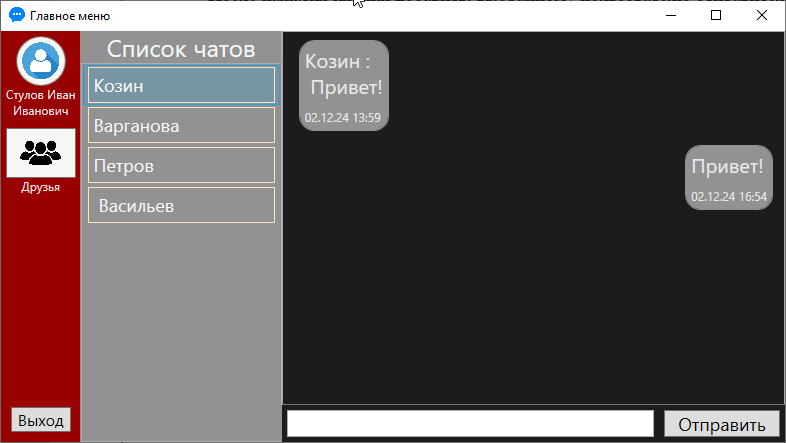


Рисунок 26 – Сообщения выбранного чата

4.4 Разработка программных модулей решения

Модуль — это независимая функциональная часть программы, выполняющая определённые задачи и обладающая четко определёнными границами и интерфейсом для взаимодействия с другими частями системы. Модульность позволяет разделить сложную систему на меньшие и более управляемые части (модули), упрощая разработку, тестирование, поддержку и масштабирование программного обеспечения.

В программном решении были реализованы следующие функции:

1. Авторизация пользователя (рисунки 27, 28).
2. Регистрация новых пользователей (рисунки 29, 30).
3. Обмен сообщениями между пользователями (рисунок 31).
4. Сохранение сообщений в базу данных (рисунок 32).
5. Список друзей (рисунок 33).



Рисунок 27 – Фрагмент кода авторизации (клиентская часть)

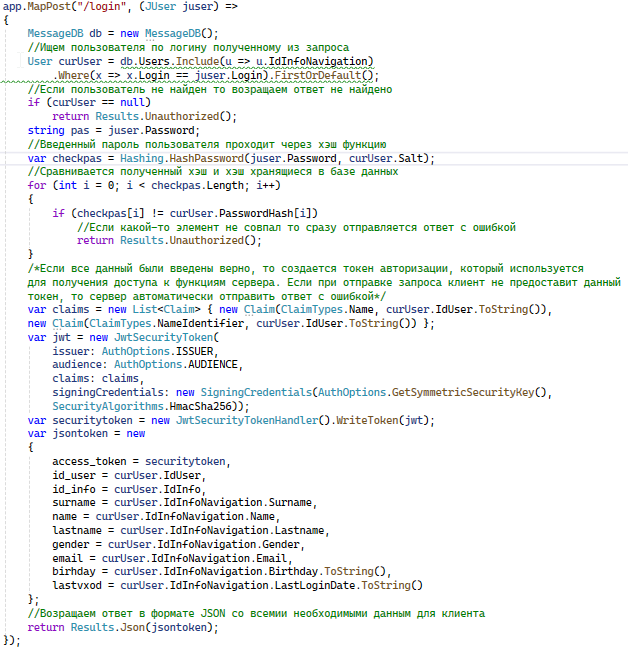


Рисунок 28 – Фрагмент кода авторизации (серверная часть)

В представленных фрагментах кода происходит обмен данными между клиентом и сервером. Клиент отправляет на сервер данные, введенные пользователем, после чего сервер сравнивает значения с тем, что хранятся в базе данных. Если данные верны, сервер формирует специальный токен, с помощью которого клиент может получать доступ к закрытым методам сервера, так как они требуют данный токен для работы с ними.

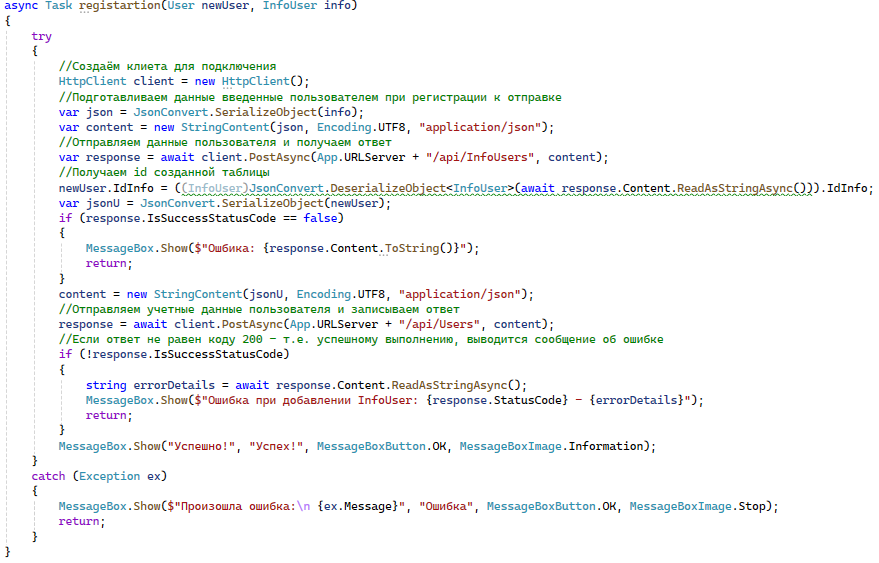


Рисунок 29 – Фрагмент кода регистрации (клиентская часть)

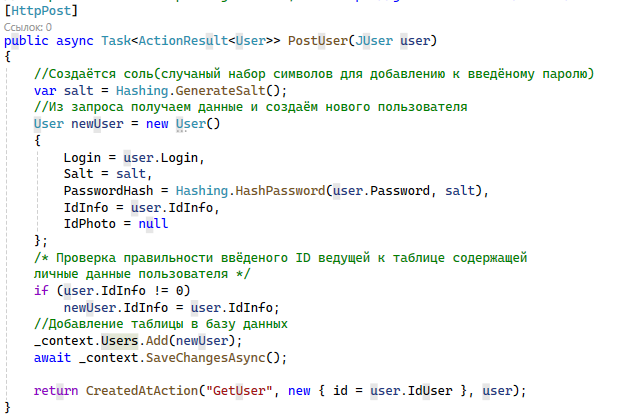


Рисунок 30 – Фрагмент кода регистрации (серверная часть)

В данном фрагменте кода происходит регистрация клиента. Клиент вводит все необходимые данные после чего они отправляются на сервер. Здесь полученные данные записываются в базу данных. Перед тем как записать пароль в базу данных он хэшируется с помощью специального метода (рисунок 31) для большей безопасности.



Рисунок 30 – Фрагмент кода метода хеширования пароля

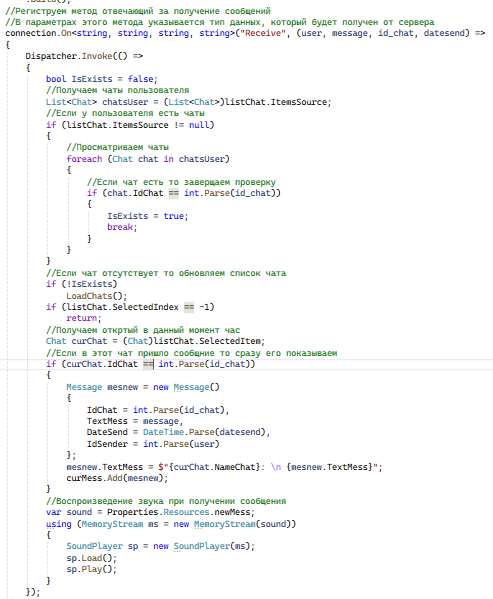


Рисунок 31 – Фрагмент кода, отвечающего за получение сообщений

В данном фрагменте при инициализации окна происходит регистрация метода, который отвечает за получение сообщения пользователем. В это методе происходит несколько проверок. Например, открыт сейчас чат или нет. Это сделано для того, чтобы сообщения не путались между чатами.

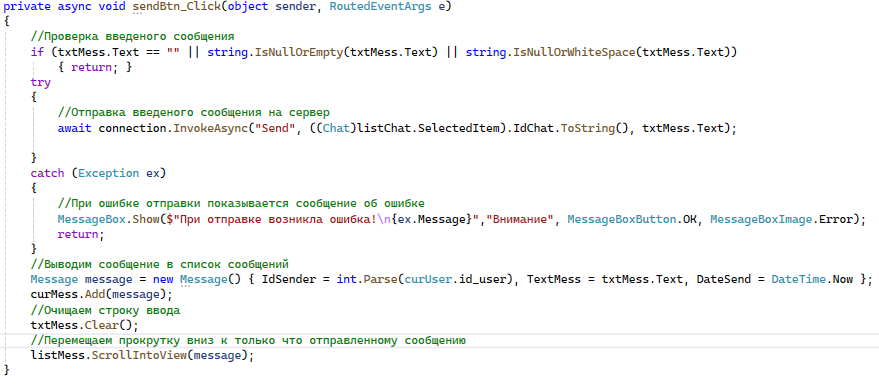


Рисунок 32 – Фрагмент кода, отвечающего за отправку сообщений

В данном фрагменте кода происходит обработка отправки пользователем сообщения другому клиенту. В этом методе происходит поиск отправителя сообщения в базе данных и получателя. После чего сообщение записывается в базу данных и отправляется конечному пользователю.



Рисунок 33 – Фрагмент кода, получение списка друзей пользователя

4.5 Тестирование разработанного программного решения

Для тестирования корпоративного мессенджера использовались следующие методы:

1. Модульное тестирование

Модульное тестирование применялось для проверки функциональности основных модулей приложения, включая модули авторизации, отправки сообщений и сохранения истории чатов. Особое внимание уделялось проверке корректности обмена сообщениями между двумя пользователями, поскольку на данном этапе приложение поддерживает только личные чаты. Основной целью было убедиться в надежной работе этих функций и исключить ошибки, которые могли бы повлиять на безопасность или стабильность общения.

1. Тестирование методом «черного ящика»

Этот метод использовался для проверки взаимодействия с приложением с точки зрения пользователя. Проверялись сценарии авторизации, отправки текстовых сообщений в личных чатах и обработка случаев, когда вводятся некорректные данные (например, пустые сообщения). Также оценивалась стабильность работы при активной переписке между двумя пользователями.

1. Тестирование методом «белого ящика»

Метод «белого ящика» применялся для анализа исходного кода модулей, отвечающих за обмен сообщениями, авторизацию и сохранение истории. Были протестированы граничные случаи, такие как одновременная отправка сообщений двумя пользователями, и выявлены потенциальные проблемы в алгоритмах обработки данных.

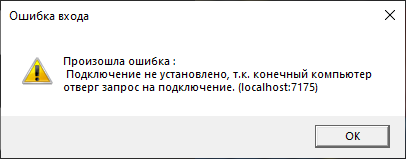


Рисунок 34 – Ошибка подключения к серверу

Данная ошибка появляется, когда пользователь не может подключится к серверу.

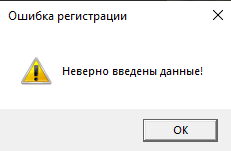


Рисунок 35 – Ошибка ввода данных

Данная ошибка появляется тогда, когда пользователь при регистрации некорректно вводит некоторые данные. Чтобы пользователю легче было понять, где была допущена ошибка – использовалось подсвечивание элементов (рисунок 36).

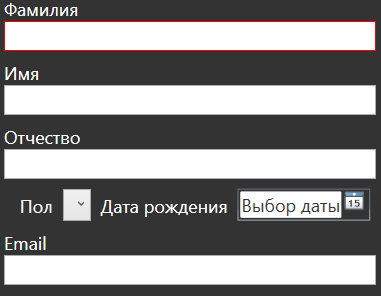


Рисунок 36 – Подсвеченный элемент

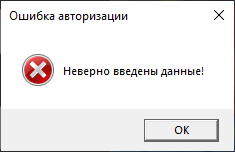


Рисунок 37 – Ошибка при авторизации пользователя

Данная ошибка появляется, когда пользователь неверно вводит учётные данные для входа в профиль.

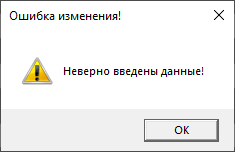


Рисунок 38 – Ошибка при изменении данных профиля

Данная ошибка появляется, когда пользователь вводит некорректные данные для изменения своего профиля.

Заключение

В соответствии с полученным заданием, была достигнута цель курсового проекта: был реализован корпоративный мессенджер для ООО «РСП-М».

В ходе работы над курсовым проектом были выполнены следующие задачи:

1. Были изучены требования заказчика и проведён анализ предметной области.
2. Построена модель базы данных.
3. Построена архитектура прототипа корпоративного мессенджера.
4. Реализован пользовательский интерфейс мессенджера.
5. Разработаны функциональные возможности серверной части.
6. Разработаны функциональные возможности клиентской части.
7. Проведено тестирование и выполнена отладка разработанного программного решения.
8. Оформлена техническая документация в виде пояснительной записки на курсовой проект в соответствии со всеми нормами ЕСПД.

Список использованных источников

1. Болье, А. Изучаем SQL. Генерация, выборка и обработка данных: учебное пособие / Алан Болье. — М.: Диалектика-Вильямс, 2021. — 512 с.: ил. — ISBN 978-5-907365-54-4.
2. Евдокимов, П. В. C# на примерах. Практика, практика и только практика: практическое руководство / П. В. Евдокимов. — Минск: Наука и техника, 2022. — 320 с.: ил. — ISBN 978-5-94387-411-6.
3. Литвиненко, Н. А. Декларативное программирование на языке XAML + C#. WPF проекты: специализированная литература / Н. А. Литвиненко. — М.: Горячая Линия - Телеком, 2023. — 320 с.: ил. — ISBN 978-5-9912-0971-7.
4. Молинаро, Э., де Грааф, Р. SQL. Сборник рецептов: справочник / Э. Молинаро, Р. де Грааф. — СПб.: БХВ-Петербург, 2019. — 592 с.: ил. — ISBN 978-5-9775-6759-6.
5. Натан, А. WPF 4. Подробное руководство: учебное пособие / Адам Натан. — М.: Символ-Плюс, 2023. — 1024 с.: ил. — ISBN 978-5-93286-196-7.
6. Прайс, М. C# 10 и .NET 6. Современная кросс-платформенная разработка: практическое руководство / М. Прайс. — СПб.: БХВ-Петербург, 2022. — 1019 с.: ил. — ISBN 978-5-4461-2249-.
7. Ризо, Дж. ASP.NET Core MVC 2 с примерами на C# для профессионалов: учебное пособие / Джон Ризо. — СПб.: БХВ-Петербург, 2019. — 720 с.: ил. — ISBN 978-5-6041394-3-1.
8. Сергеев, А. Основы локальных компьютерных сетей: учебное пособие / А. Сергеев. — СПб.: Питер, 2019. — 272 с.: ил. — ISBN 978-5-507-44766-4.
9. Таненбаум, Э., Уэзеролл, Д. Компьютерные сети: 5-е изд.: учебное пособие / Э. Таненбаум, Д. Уэзеролл. — СПб.: Питер, 2019. — 960 с.: ил. — ISBN 978-5-4461-0445-9.
10. Тейлор, А. SQL для чайников: вводный курс / Аллен Тейлор. — М.: Вильямс, 2020. — 320 с.: ил. — ISBN 978-5-907144-81-1.
11. Троелсен, Э., Джепикс, Ф. Язык программирования C# 7 и платформы .NET и .NET Core: справочник / Э. Троелсен, Ф. Джепикс. — СПб.: Питер, 2019. — 1488 с.: ил. — ISBN 978-5-4461-0121-3.
12. Фленов, М. Библия C#. 6-е изд.: справочник / М. Фленов. — СПб.: Питер, 2023. — 704 с.: ил. — ISBN 978-5-4461-0373-0.
13. Фридл, Д. П., Нельсон, Дж. К. ASP.NET Core в действии: специализированная литература / Д. П. Фридл, Дж. К. Нельсон. — СПб.: БХВ-Петербург, 2020. — 416 с.: ил. — ISBN 978-5-9775-0758-
14. Metanit. Язык C#: учебные материалы [Электронный ресурс]. — URL: https://metanit.com/sharp/tutorial/ (дата обращения: 17.10.2024).
15. Metanit. ASP.NET 6: учебные материалы [Электронный ресурс]. — URL: https://metanit.com/sharp/aspnet6/ (дата обращения: 23.10.2024).
16. Metanit. SignalR: учебные материалы [Электронный ресурс]. — URL: https://metanit.com/sharp/signalr/ (дата обращения: 10.10.2024).
17. Habr. Программирование C#: статья [Электронный ресурс]. — URL: https://habr.com/ru/articles/215117/ (дата обращения: 10.10.2024).
18. Habr. Современные подходы к ASP.NET: статья [Электронный ресурс]. — URL: https://habr.com/ru/articles/590679/ (дата обращения: 05.11.2024).
19. Microsoft Learn. Учебное пособие: Создание первого веб-API [Электронный ресурс]. — URL: https://learn.microsoft.com/ru-ru/aspnet/core/tutorials/first-web-api?view=aspnetcore-8.0&tabs=visual-studio (дата обращения: 10.10.2024).
20. Microsoft Learn. Введение в SignalR [Электронный ресурс]. — URL: https://learn.microsoft.com/ru-ru/aspnet/signalr/overview/getting-started/introduction-to-signalr (дата обращения: 04.11.2024).

**Приложение А**