Федеральное государственное образовательное бюджетное

учреждение высшего образования

**«Финансовый университет при Правительстве Российской Федерации»**

**(Финансовый университет)**

Колледж информатики и программирования

Специальность 09.02.03 Программирование в компьютерных системах

**ОТЧЕТ**

**ПО ПРОИЗВОДСТВЕННОЙ ПРАКТИКЕ**

**(преддипломной)**

Выполнил:

обучающийся учебной группы № 4ПКС-120

Плюснин М.Ю.

*(И.О. Фамилия)*

Проверил:

руководитель практики от колледжа:

Сибирев И.В.

*(И.О. Фамилия)*

**Москва**

**20­­­­24**

# **СОДЕРЖАНИЕ**

[**ВВЕДЕНИЕ** 2](#_Toc165991857)

[**Глава 1. Предпроектное исследование** 5](#_Toc165991858)

[**1.1. Обоснование выбора и анализ предметной области** 5](#_Toc165991859)

[**1.2. Сравнительный анализ программ-аналогов** 6](#_Toc165991860)

[**1.3. Постановка задачи и обоснование выбора программного обеспечения** 8](#_Toc165991861)

[**Глава 2. Проектирование и реализация программы** 10](#_Toc165991862)

[**2.1. Анализ требований и разработка спецификаций программного обеспечения** 10](#_Toc165991863)

[**2.2. Проектирование программного обеспечения** 12](#_Toc165991864)

[**2.3. Разработка программного обеспечения** 13](#_Toc165991865)

[**2.4. Отладка и тестирование программного обеспечения** 23](#_Toc165991866)

[**2.5. Руководство по использованию программы** 33](#_Toc165991867)

[**ЗАКЛЮЧЕНИЕ** 41](#_Toc165991868)

[**СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ И ИНТЕРНЕТ-РЕСУРСОВ** 43](#_Toc165991869)

[**ПРИЛОЖЕНИЕ** 46](#_Toc165991870)

# **ВВЕДЕНИЕ**

Современный мир невозможно представить без социальных сетей и мессенджеров.Они стали неотъемлемой частью нашей жизни и используются практически всеми людьми по всему миру, помогая поддерживать связь с друзьями и коллегами, делиться новостями и идеями, а также решать различные задачи. Буквально в пару кликов можно отправить любую информацию любому человеку, где бы он не был, за несколько секунд. В данной работе рассматривается тема создания и использования мессенджера, его основные функции и возможности.

Цель данного дипломного проекта заключается в проектировании и разработке программного обеспечения «Мессенджер». Главной задачей проекта является разработка удобного и интуитивно понятного приложения и его веб-версии для обмена сообщениями между пользователями.

Для достижения поставленной цели необходимо решить следующие задачи:

1. проанализировать особенности передачи данных, а также ведущих представителей подобного программного обеспечения;
2. проанализировать и выбрать инструментальные средства для разработки приложения «Мессенджер»;
3. выделить функциональные и эксплуатационные требования к разрабатываемому программному обеспечению;
4. спроектировать интерфейс приложения;
5. спроектировать базу данных;
6. разработать серверное приложение;
7. разработать клиентское приложение;
8. провести тестирование разработанного приложения;
9. составить и оформить пользовательскую документацию.

Актуальность предлагаемого программного решения обусловлена тем, что всё больше и больше людей начали переходить на формат удаленной работы, которая требует коммуникации между друг другом на расстоянии. Разработка собственного мессенджера может предоставить уникальный опыт создания и проектирования приложения, которое передаёт и хранит данные.

Основное функциональное назначение разрабатываемого программного средства будет заключаться в предоставлении пользователю возможности общения с другими пользователями, управлении чатами и также редактировании своего профиля.

Объектом исследования является технологии обмена данными.

Предметом исследования является приложения для обмена сообщениями.

Основными методами исследования выступают анализ и сравнение уже существующих мессенджеров, анализ документации по языку программирования C# и графическому интерфейсу WPF, а также проведение экспериментов разрабатываемого программного продукта.

Источниковой базой исследования являются учебные пособия по технологии программирования, по разработке программного обеспечения в графическом интерфейсе WPF, интернет-ресурсы по предметной области, существующим программным решениям, средствам разработки, а также государственные стандарты по оформлению документации.

В данном дипломном проекте будет описано следующее:

1. проектирование интерфейса приложения;
2. проектиирование базы данных;
3. разработка клиентской части приложения;
4. разработка модели взаимодействия с серверной частью приложения.

Также для разработки требуется определиться с инструментальными средствами и языками проектирования и разработки программного обеспечения. Для разработки приложения был выбран язык программирования C#. Он является одним из популярных языков программирования для разработки приложений под операционные системы Windows и Android. Средой разработки была выбрана Visual Studio, т.к в ней имеется удобный графический интерфейс WPF. Для построения диаграмм и схем используются онлайн инструменты Draw.io, а также PlantUML. При создании инсталлятора было задействовано расширение из Visual Studio «Microsoft Visual Studio Installer Projects». Для написания и оформления пояснительной записки был использован Microsoft Word, а также для создания презентации к защите - Microsoft Powerpoint.

# **Глава 1. Предпроектное исследование**

## **1.1. Обоснование выбора и анализ предметной области**

В настоящее время мессенджеры являются неотъемлемой частью нашей жизни и используются практически всеми людьми по всему миру. Они позволяют нам быстро и удобно общаться с друзьями и коллегами, обмениваться сообщениями, фотографиями и видео, а также делиться своими мыслями и идеями.Буквально в пару кликов можно отправить любую информацию любому человеку, где бы он не был, за несколько секунд.

Для более точного определения что такое мессенджер, проясним следующие понятия: «Пользователь – это человек или организация, которые используют мессенджер для общения с другими людьми», «Мессенджер – это программа или сервис, который позволяет пользователям обмениваться сообщениями и файлами между собой», «Сообщения – это текстовые сообщения, которые пользователи отправляют друг другу через мессенджер», «Файлы – это документы, фотографии, видео или другие файлы, которые пользователи загружают в мессенджер».

Цель работы - разработать функциональный десктоп мессенджер для обмена защищенных шифрованием сообщений между пользователями, которые хранятся на сервере. Сообщение может быть в виде текста с прикреплёнными файлами. Программа подойдет для использования любой организацией, в которой происходит общение сотрудников с использованием каких-либо корпоративных данных. В других мессенджерах зачастую сообщения хранятся на их серверах. К этим сообщениям может иметь доступ кто угодно. В случае если мессенджер является корпоративным, то все сообщения хранятся на сервере компании и доступ к этому хранилищу есть только у сотрудников, что позволит максимально избежать утечек каких-то важных данных.

Мессенджер имеет понятный для любых пользователей интерфейс, что позволит пользоваться программой любому пользователю.

## **1.2. Сравнительный анализ программ-аналогов**

Прежде чем приступать к реализации, проведем анализ и сравним существующие мессенджеры в таблице 1. Для сравнения выберем самые популярные мессенджеры в странах СНГ: Вконтакте [14], Telegram [11], Whatsapp [13], Viber [12].

Таблица 1. Сравнительный анализ

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Вконтакте | Telegram | Whatsapp | Viber |
| Функционал | | | | |
| Отправка сообщений | + | + | + | + |
| Прикрепление файлов | + | + | + | + |
| Видеосообщения | + | + | - | - |
| Звонки | + | + | + | + |
| Хранение переписки | На серверах компании | На серверах компании | На устройстве | На серверах компании |
| Тематические сообщества | + | + | - | - |
| Интерфейс | | | | |
| Наличие нескольких цветовых тем | + | + | + | + |
| Наличие смены языков | + | + | + | + |

Продолжение таблицы 1

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Вконтакте | Telegram | Whatsapp | Viber |
| Стоимость | | | | |
| Платная установка | - | - | - | - |
| Платная подписка, дающая преимущества | + | + | - | - |
| Кроссплатформенность | | | | |
| Мобильное приложение | + | + | + | + |
| Десктоп приложение | + | + | + | + |
| Веб-версия приложения | + | + | + | - |

При разработке собственного мессенджера примется во внимание лучшие качества сравниваемых продуктов: бесплатная установка клиентского приложения без платных подписок, отправка сообщений со всевозможными приложениями файлов, наличие цветовой и языковой смены интерфейса и хранилище переписки на сервере компании.

## **1.3. Постановка задачи и обоснование выбора программного обеспечения**

Для программной реализации необходимо определиться с входными и выходными данными мессенджера. Входными данными мессенджера являются сообщения и файлы, которые пользователь отправляет другому пользователю, а также логин и пароль при регистрации или авторизации. Выходными данными же является то, что пользователь получает от другого пользователя и его панель чатов.

Программа должна иметь следующие функции:

1. регистрация пользователей;
2. авторизация пользователей;
3. поиск других пользователей;
4. просмотр профиля других пользователей;
5. просмотр и редактирование профиля;
6. отправка сообщений пользователям;
7. отправка пользователям медиафайлов, таких как: видео, аудио и фото.

Архитектура ПО должна быть многопользовательской. Данные должны храниться в реляционной базе данных. Интерфейс должен быть интуитивно понятен пользователю, не быть перегруженым ненужной информацией, а также иметь приятную палитру цветов.

К реализации программы предъявляются следующие требования:

1. операционная система Windows 7 и выше;
2. объём памяти на жёстком диске – 100мб;
3. объём оперативного запоминающего устройства(ОЗУ) - 4гб;
4. качественное интернет соединение.

Ниже приведены требования к надёжности программного обеспечения:

1. данные при регистрации пользователя должны шифроваться;
2. пароли и личная информация должны храниться в зашифрованном виде;
3. при обмене между сервером и клиентом сообщения должны шифроваться открытым ключом шифрования, а в базе данных - закрытым ключом;
4. обмен данными между сервером и клиентом должен происходить по безопасному протоколу передачи данных – TCP.

Для разработки приложения был выбран язык программирования C#. Он является одним из популярных языков программирования для разработки приложений под операционные системы Windows и Android. Средой разработки была выбрана Visual Studio, т.к в ней имеется удобный графический интерфейс WPF. Для построения диаграмм и схем используются онлайн инструменты Draw.io, а также PlantUML. При создании инсталлятора было задействовано программное обеспечение Actual Installer. Для написания и оформления пояснительной записки был использован Microsoft Word, а также для создания презентации к защите - Microsoft Powerpoint.

# **Глава 2. Проектирование и реализация программы**

## **2.1. Анализ требований и разработка спецификаций программного обеспечения**

Чтобы было больше понимания над структурой программного обеспечения были спроектированы и показаны ниже несколько диаграмм, которые представляют функции данного приложения.

Все диаграммы были спроектированы с помощью ресурса PlantText.

На рисунке 1 представлены диаграмма деятельности.

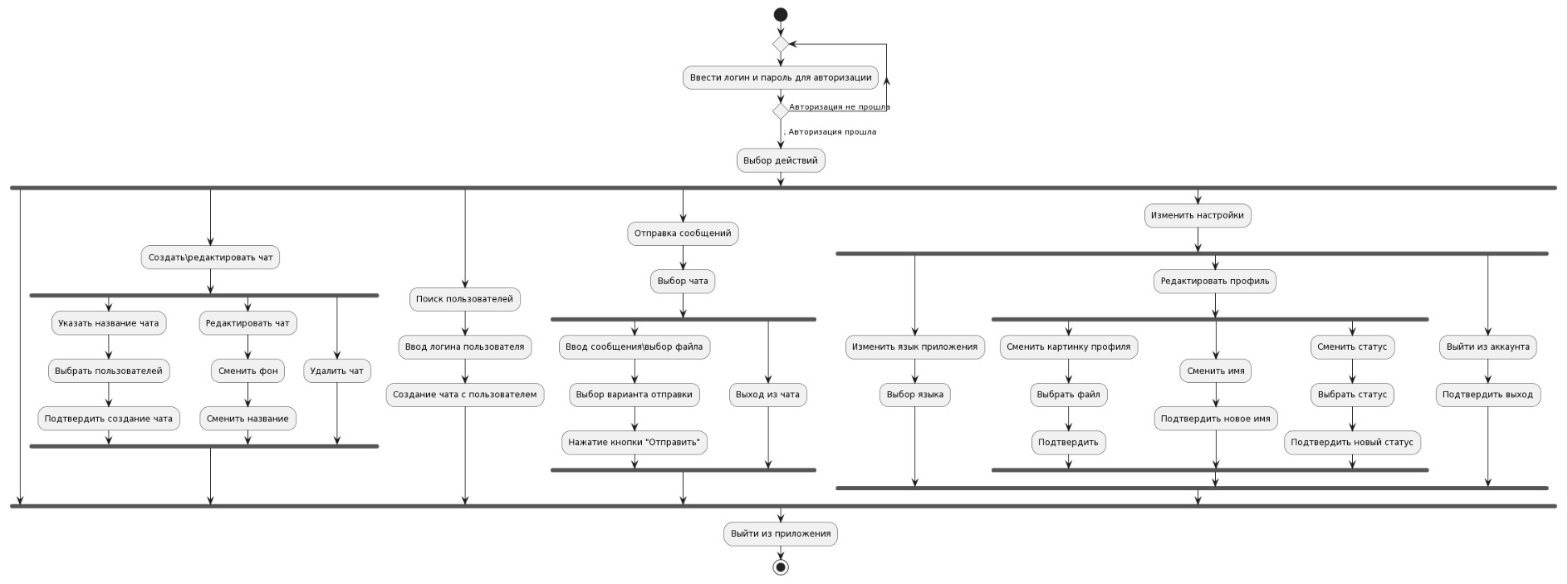


Рисунок 1. Диаграмма деятельности

Диаграмма деятельности представляет подробную информацию о функционале программы. В приложении имеется 4 главных функции, разделенные на подфункции:

1. Авторизация\Регистрация пользователя
2. Изменение настроек
3. Поиск пользователей
4. Отправка сообщений

Далее на рисунке 2 изображена диаграмма вариантов использования.

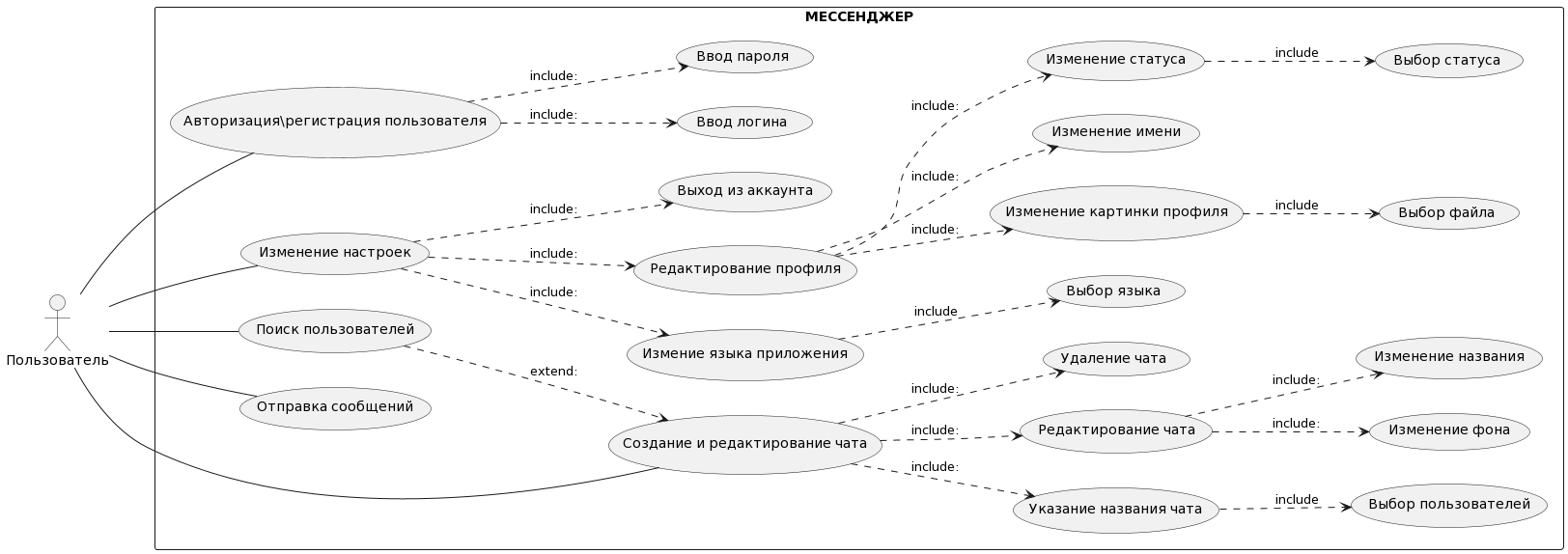


Рисунок 2. Диаграмма вариантов использования

Диаграмма вариантов использования представляет возможности программного обеспечения, а также показывает как пользователь может взаимодействовать с ним.

Также представлена диаграмма последовательности на рисунке 3.

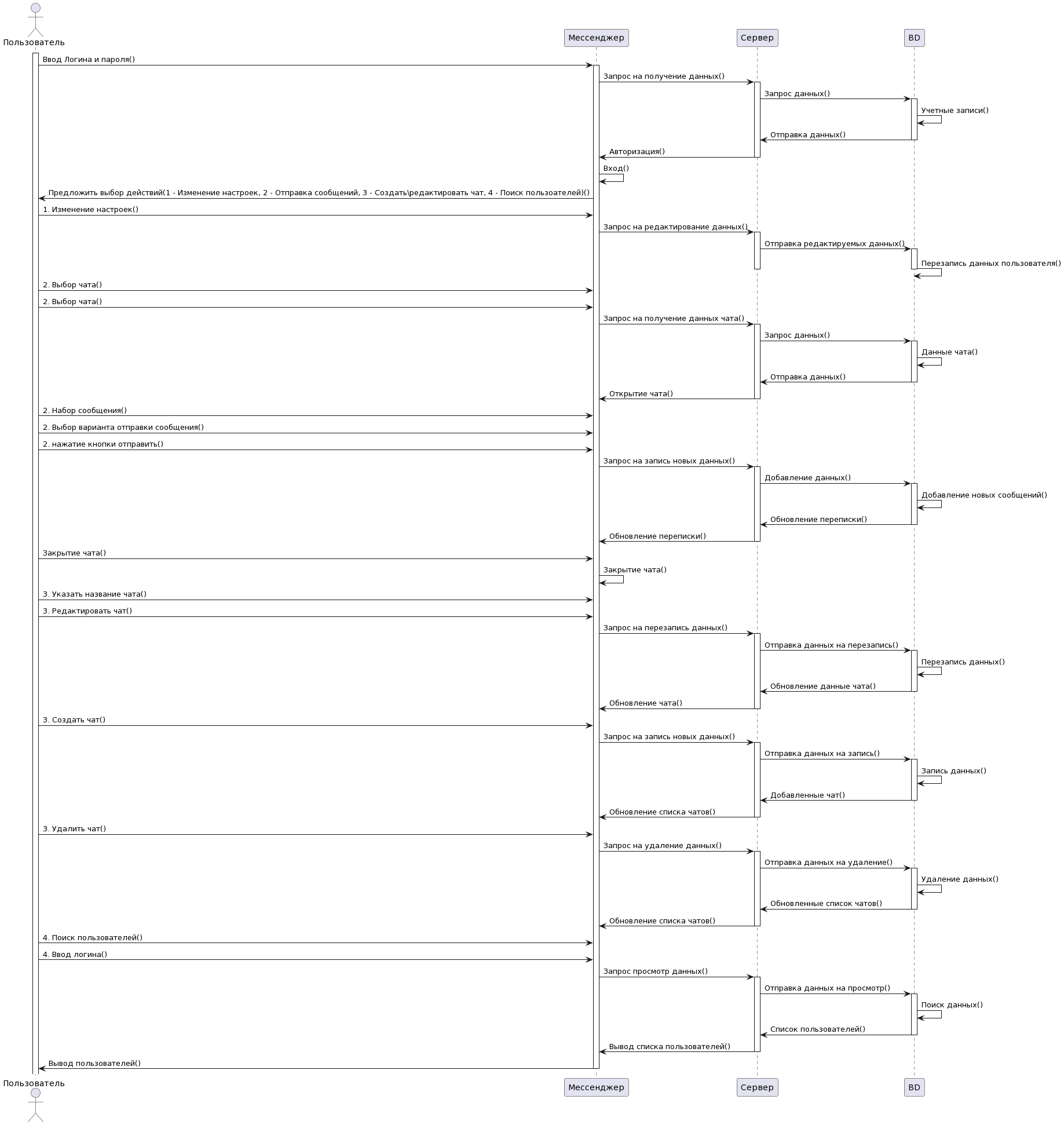


Рисунок 3. Диаграмма последовательности

Эта диаграмма показывает как пользователь взаимодействует с приложением, а приложение с сервером, который обращается к базе данных, а также последовательность действий при выборе функции.

## **2.2. Проектирование программного обеспечения**

Для получения полного понимания работы мессенджера необходимо создать две схемы: структурную и функциональную.

Структурная схема используется для описания структуры программы и организации выполнения ее компонентов. Она помогает читать и понимать код программы, показывая последовательность выполнения блоков кода и взаимосвязи между ними.

Функциональная схема программы отображает взаимодействие различных функций и подпрограмм, а также поток данных в программе.

Структурная схема изображена на рисунке 4.



Рисунок 4. Структурная схема

На рисунке 5 представлена функциональная схема.

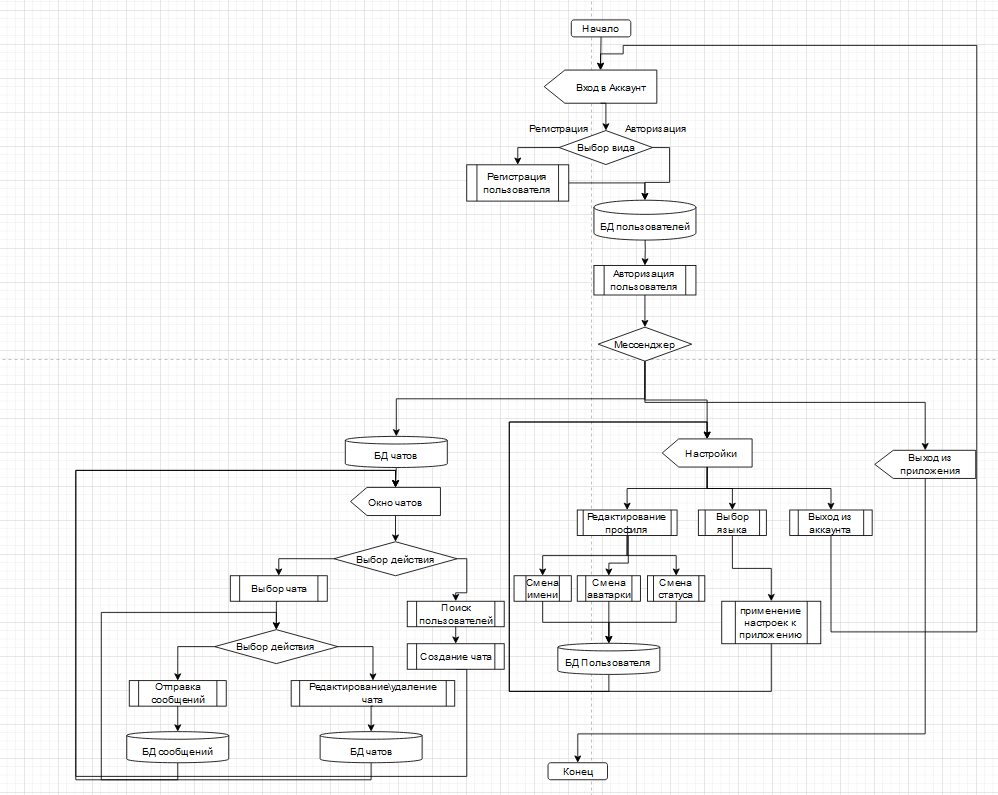


Рисунок 5. Функциональная схема

## **2.3. Разработка программного обеспечения**

На данном этапе будет подробнее описан процесс проектирования и программирования, освещая основные этапы разработки, выбор используемых технологий, архитектуру приложения и важные аспекты кодирования. Этот раздел предоставит подробный обзор шагов, предпринятых в процессе разработки программного обеспечения, с акцентом на создание функционального, эффективного и масштабируемого мессенджера для успешного завершения курсового проекта.

Самой важной частью клиентской части мессенджера является отправка сообщений и вывод на экран чатов. Эта функция производится благодаря тому, что в стиле поля ввода нажатие клавиши Enter или соответствующей кнопки отправки ассоциируется с коммандой для сервера о передаче сообщения. Ниже, в листинге 1 представлен фрагмент стиля поля ввода.

Листинг 1. Фрагмент стиля поля ввода

<TextBox VerticalAlignment="Stretch" VerticalContentAlignment="Center"

HorizontalAlignment="Stretch"

x:Name="MessageBox"

Background="Transparent"

TextWrapping="Wrap"

BorderThickness="0"

Foreground="LightGray"

Text="{Binding MessageText, Mode=TwoWay, UpdateSourceTrigger=PropertyChanged}">

<TextBox.InputBindings>

<KeyBinding Command="{Binding SendCommand}" Key="Return"/>

</TextBox.InputBindings>

Соответственно после нажатия выполняется команда указаная в листинге 2, которая выполняет функцию SendMessageUs, которая в свою очередь отправляет запрос серверу на отправку сообщения другому пользователю.

Листинг 2. Команда SendCommand

SendCommand = new RelayCommand(o =>

{

App.server.Message = MessageText;

App.server.SendMessageUs.Execute(this);

this.AddMessage();

});

На листинге 3 представлен запрос к серверу, который исполняет команда. Этот запрос отправляет объект сообщения с данными: содержание, время отправки и чат, в котором пользователь находится.

Листинг 3. Запрос к серверу SendMessageUs

public AsyncCommand SendMessageUs

{

get

{

return new AsyncCommand(() =>

{

return Task.Run(() =>

{

if (!String.IsNullOrEmpty(Message))

{

\_writer.WriteLine("SendMessage");

var a = new { MessageText = Message, Time = DateTime.Now, SelectedChat = ChatID };

\_writer.WriteLine(JsonConvert.SerializeObject(a));

}

});

}, () => \_client?.Connected == true);

}

}

Далее команда SendCommand вызывает публичную функцию AddMessage(), показанную в листинге 4, которая проверяет пустое ли сообщение, и добавляет в массив сообщений написаный текст, автора и время написания, ниже представлен код. Соответсвенно в окне с сообщениями добавляется “облако” с текстом, оно продемонстрированно на рисунке 6.

Листинг 4. Функция AddMessage()

public void AddMessage()

{

if (!String.IsNullOrEmpty(MessageText))

{

Messages.Add(new MessageModel

{

Message = MessageText,

MessageAutor = App.server.Nick,

dateTime = DateTime.Now

});

MessageText = "";

App.chatPage.messagesList.ScrollIntoView(Messages[Messages.Count() - 1]);

}

}

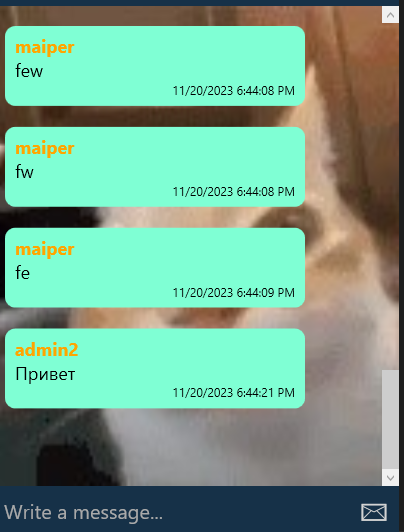


Рисунок 6. Окно с сообщениями и “Облака” с тектом

На рисунке 7 изображена панель со списком чатов.

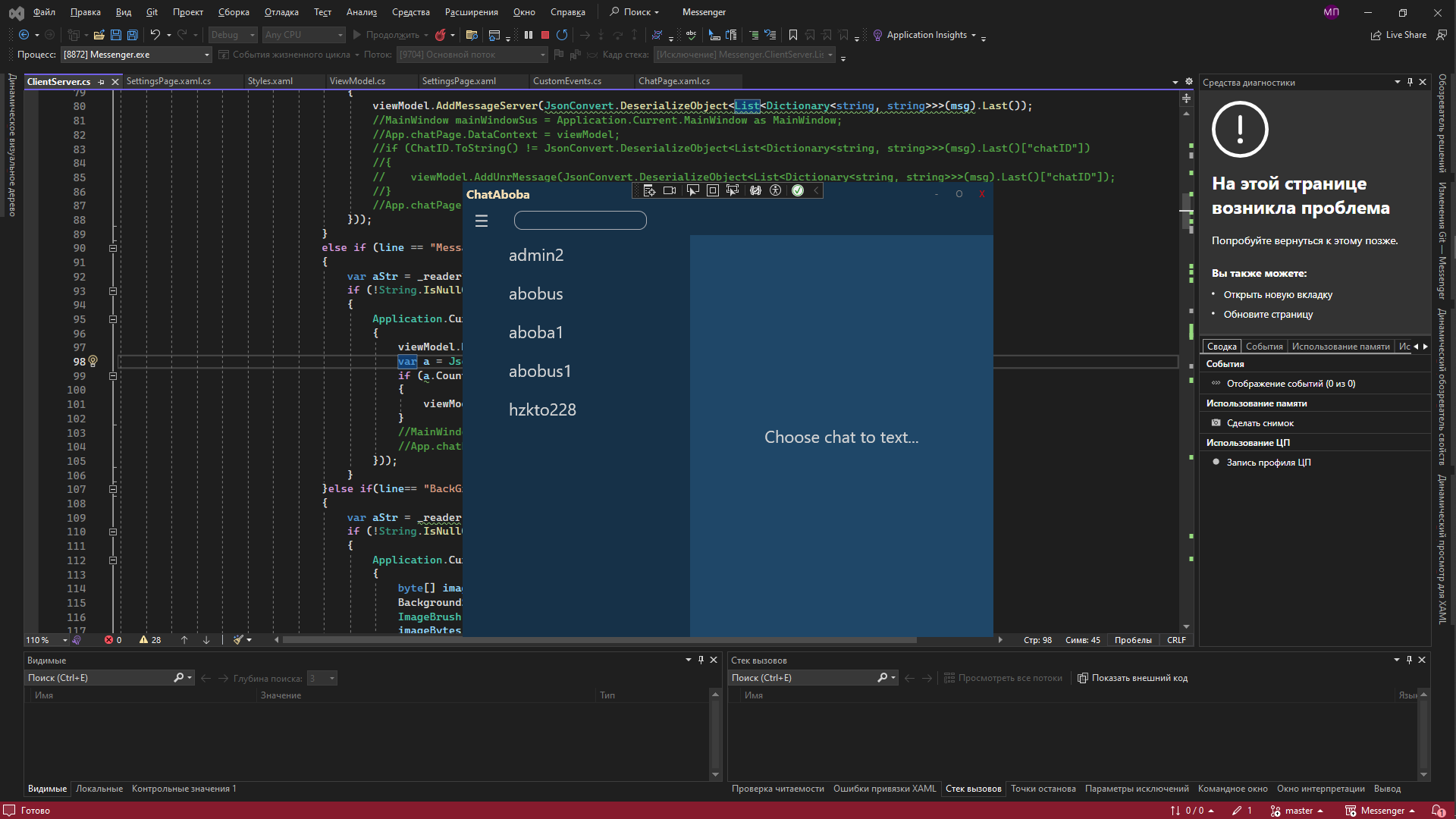


Рисунок 7. Список чатов

Список чатов также хранится в массиве, который содержит такие данные как: название чата, id чата. При авторизации пользователя клиент запрашивает у сервера список чатов по ID авторизовавшегося, далее представлен фрагмент кода. Более подробная часть запроса на авторизацию представлена в Приложении 1.

Листинг 5. Запрос у сервера на получение списка чатов

\_writer.WriteLine("GetChats");

viewModel = new ViewModel();

string aStr = \_reader.ReadLine();

viewModel.AddChatList(JsonConvert.DeserializeObject<List<Dictionary<string,string>>>(aStr));

Application.Current.Dispatcher.BeginInvoke((Action)(() =>

{

MainWindow mainWindowSus = Application.Current.MainWindow as MainWindow;

connected = true;

App.chatPage = new ChatPage(usId);

App.chatPage.DataContext = viewModel;

mainWindowSus.frameMenu.Navigate(App.chatPage);

}));

При успешной авторизации клиент получает список в виде Json, который конвертируется в нужный массив, вызывая функцию добавления в массив чатов информации, такой как: название чата и его ID. В листинге 6 указан код функции AddChatList().

Листинг 6. Функция AddChatList()

public void AddChatList(List<Dictionary<string, string>> chats)

{

if (chats!=null)

{

foreach (var item in chats)

{

int a = String.IsNullOrEmpty(item["UnReMessage"]) ? 0 : Convert.ToInt32(item["UnReMessage"]);

Users.Add(new UserListModel { ChatName = item["ChatName"], unrMessages = a, ChatID = item["ChatID"] });

}

}

}

Также не менее важными функциями мессенджера является поиск пользователей и создание с ними чатов. На главной странице имеется поле, по нажатию на которое выводится список всех пользователей, которые есть в базе данных. Список пользователей представлен на рисунке 8.

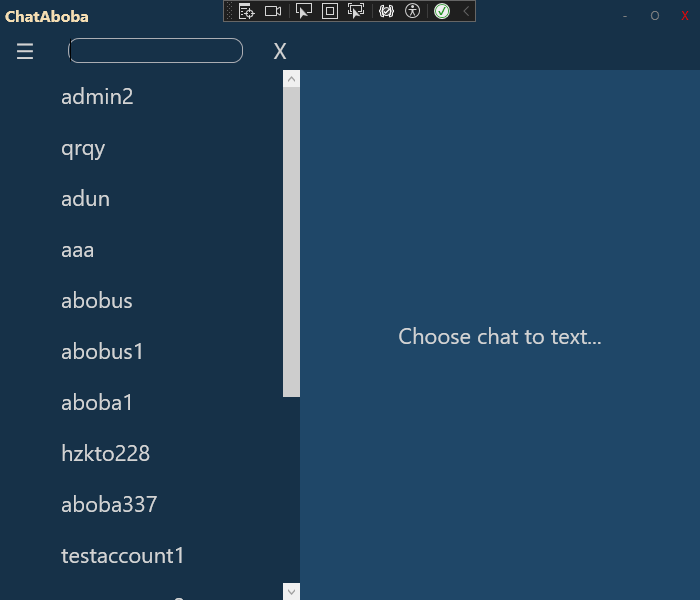


Рисунок 8. Список пользователей

При нажатии на поле поиска вызывается функция \_searchDown(), которая и выводит список пользователей, в листинге 7 указан код.

Листинг 7. Код функции \_searchDown()

private void \_searchDown(object sender, MouseButtonEventArgs e)

{

App.server.viewModel.searchRes.Clear();

Task.Run(()=>App.server.GetUsersAndChatsIDsWithUser.Execute(this)).Wait();

bord.Visibility = Visibility.Collapsed;

bord2.Visibility = Visibility.Visible;

closeSearch.Visibility = Visibility.Visible;

}

Эта функция вызывает запрос к серверу, на получение пользователей, который как раз, после выполнения заполняет массив с поиском. В листингах 8 и 9 представлены фрагменты кода.

Листинг 8. Запрос к серверу на получение пользователей

public AsyncCommand GetUsersAndChatsIDsWithUser

{

get

{

return new AsyncCommand(() =>

{

return Task.Run(() =>

{

if (ChatID >= 0)

{

\_writer?.WriteLine("GetUsersAndChatsIDsWithUser");

}

});

}, () => \_client?.Connected == true);

}

}

Следующий фрагмент этого запроса вызывает функцию на добавление в массив пользователей, которая схожа с функцией на добавление чатов.

Листинг 9. Вызов функции на добавление пользователей в массив

else if (line == "GetUsersAndChatsIDsWithUser")

{

Application.Current.Dispatcher.BeginInvoke((Action)(() =>

{ viewModel.AddUserSearchList(JsonConvert.DeserializeObject<List<Dictionary<string, string>>>(\_reader.ReadLine()));

}

));

Поиск пользователей происходит при изменении текста в поисковой строке далее показаны листинги 10 и 11, первый вызывает функцию поиска, второй является этой функцией.

Листинг 10. Вызов функции поиска

private void TextBox\_TextChanged(object sender, TextChangedEventArgs e)

{

App.server.viewModel.SearchUsers(search.Text);

}

Листинг 11. Функция поиска пользователей

public void SearchUsers(string SearchText)

{

App.chatPage.searchList.ItemsSource = searchRes.Where(x => x.ChatName.Contains(SearchText)).Select(a => a).ToObservableCollection();

}

Создание чата вызывается при нажатии на пользователя. В этой функции производится проверка на совпадение созданного чата с выбранным, и если его нет, то создаёт чат, а если есть, то открывает существующий. Код функции указан в Приложении 2. Также на рисунке 9 изображено окно создания чата.

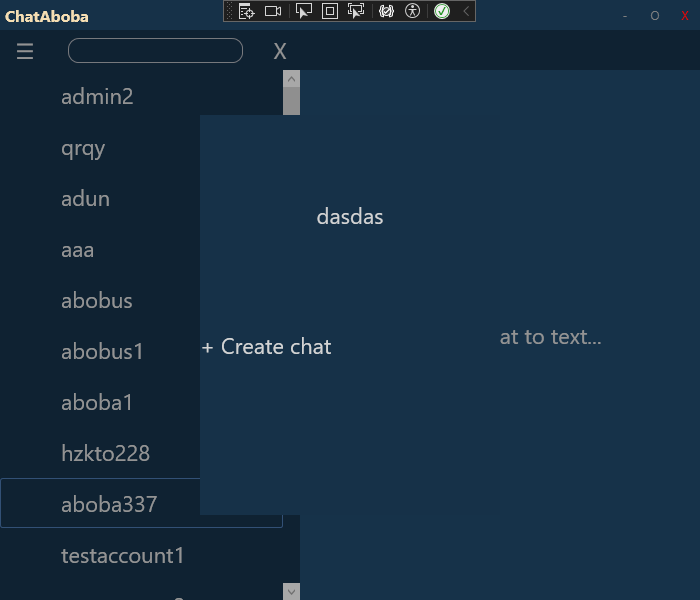


Рисунок 9. Окно создания чата

Заодно не опущу возможность смены языка приложения. На рисунке 10 представлен английский язык интерфейса, а на 11 рисунке русский язык.

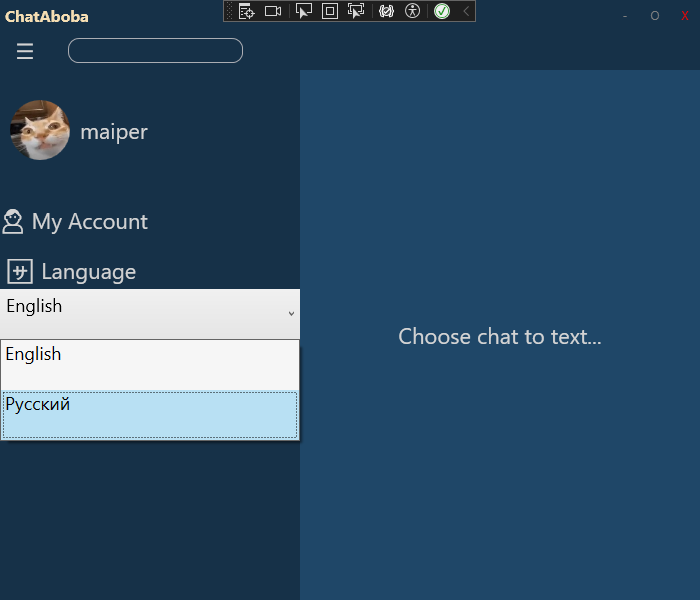


Рисунок 10. Английкая версия

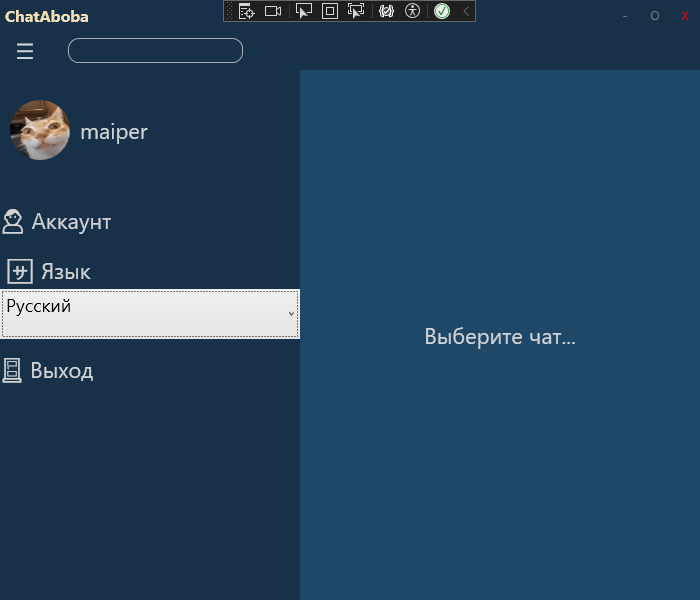


Рисунок 11. Русская версия

Это было достигнуто путём создания двух словарей, в которых хранится один и тот же текст для кнопок, но на разных языках, а также элемент ComboBox, в котором при смене элемента меняется и язык. В приложении 3 указан код данной функции.

## **2.4. Отладка и тестирование программного обеспечения**

В данном параграфе внимание сфокусировано на ключевых этапах, направленных на обеспечение стабильной и безошибочной работы разработанного мессенджера. Отладка и тестирование представляют собой неотъемлемые составляющие процесса разработки, направленные на выявление, анализ и устранение возможных ошибок, а также проверку соответствия функционала заявленным требованиям.

Для оценки качества проекта была разработана программа проведения тестирования, включающая набор тестовых сценариев.

Рассмотрим тестирование модуля авторизации(см. табл. 2-7).

Таблица 2. Тестовый пример 1

|  |  |
| --- | --- |
| Тестовый пример 1 | Messenger\_LoginPage |
| Приоритет тестирования | Высокий |
| Заголовок\название теста | TC\_Login\_1 |
| Краткое изложение теста | Тест должен проверить модуль на пустые поля при заполнении. |
| Этапы теста | 1. Не ввести логин  2. Не ввести пароль  3. Нажать на кнопку “Авторизоваться” |
| Тестовые данные |  |
| Ожидаемый результат | Программа должна вывести ошибку при незаполненных полях |
| Фактический результат | Функция вернула false |
| Статус | Успешно |
| Предварительное условие | Кнопка “Авторизоваться” доступна для клика. |
| Постусловие | Выведена ошибка. |

Таблица 3. Тестовый пример 2

|  |  |
| --- | --- |
| Тестовый пример 2 | Messenger\_LoginPage |
| Приоритет тестирования | Высокий |
| Заголовок\название теста | TC\_Login\_2 |
| Краткое изложение теста | Тест должен проверить модуль на неправильный ввод пароля |
| Этапы теста | 1. Ввести логин  2. Ввести пароль  3. Нажать на кнопку “Авторизоваться” |
| Тестовые данные | admin - Логин  djasndaj - Пароль |
| Ожидаемый результат | Программа должна вывести ошибку, что пароль неверный. |
| Фактический результат | Функция вернула false |
| Статус | Успешно |
| Предварительное условие | Кнопка “Авторизоваться” доступна для клика.  Введенный логин должен быть в базе данных. |
| Постусловие | Выведена ошибка. |
| Примечание\комментарий | Нет комментариев |

Таблица 4. Тестовый пример 3

|  |  |
| --- | --- |
| Тестовый пример 3 | Messenger\_LoginPage |
| Приоритет тестирования | Высокий |
| Заголовок\название теста | TC\_Login\_3 |
| Краткое изложение теста | Тест должен проверить модуль на неправильный ввод логина. |
| Этапы теста | 1. Ввести логин  2. Ввести пароль  3. Нажать на кнопку “Авторизоваться” |
| Тестовые данные | dsaw- Логин  adminpass- Пароль |
| Ожидаемый результат | Программа должна вывести ошибку, что данного логина не существует. |
| Фактический результат | Функция вернула false |
| Статус | Успешно |

Продолжение таблицы 4

|  |  |
| --- | --- |
| Предварительное условие | Кнопка “Авторизоваться” доступна для клика.  Логина не должно быть в базе данных. |
| Постусловие | Выведена ошибка. |
| Примечание\комментарий | Нет комментариев |

Таблица 5. Тестовый пример 4

|  |  |
| --- | --- |
| Тестовый пример 4 | Messenger\_LoginPage |
| Приоритет тестирования | Высокий |
| Заголовок\название теста | TC\_Login\_4 |
| Краткое изложение теста | Тест должен проверить модуль на правильный ввод логина и пароля |
| Этапы теста | 1. Ввести логин  2. Ввести пароль  3. Нажать на кнопку “Авторизоваться” |
| Тестовые данные | admin - Логин  adminpass- Пароль |
| Ожидаемый результат | Программа должна перенаправить пользователя на экран с чатами |
| Фактический результат | Функция вернула true |
| Статус | Успешно |
| Предварительное условие | Кнопка “Авторизоваться” доступна для клика.  Введенный логин и пароль существуют в базе данных. |
| Постусловие | Перевод на экран с чатами пользователя. |
| Примечание\комментарий | Нет комментариев |

Таблица 6. Тестовый пример 5

|  |  |
| --- | --- |
| Тестовый пример 5 | Messenger\_LoginPage |
| Приоритет тестирования | Высокий |
| Заголовок\название теста | TC\_Login\_5 |
| Краткое изложение теста | Тест должен проверить модуль на правильный ввод логина и пароля |

Продолжение таблицы 6

|  |  |
| --- | --- |
| Этапы теста | 1. Ввести логин  2. Ввести пароль  3. Нажать на кнопку “Авторизоваться” |
| Тестовые данные | maiper- Логин  maiperpass - Пароль |
| Ожидаемый результат | Программа должна перенаправить пользователя на экран с чатами |
| Фактический результат | Функция вернула true |
| Статус | Успешно |
| Предварительное условие | Кнопка “Авторизоваться” доступна для клика.  Введенный логин и пароль существуют в базе данных. |
| Постусловие | Перевод на экран с чатами пользователя. |

Таблица 7. Тестовый пример 6

|  |  |
| --- | --- |
| Тестовый пример 6 | Messenger\_LoginPage |
| Приоритет тестирования | Высокий |
| Заголовок\название теста | TC\_Login\_6 |
| Краткое изложение теста | Тест должен проверить модуль пустую строку Логина. |
| Этапы теста | 1. Не ввести логин  2. Ввести пароль  3. Нажать на кнопку “Авторизоваться” |
| Тестовые данные | adminpass- Пароль |
| Ожидаемый результат | Программа должна вывести ошибку, что поле логина пустое. |
| Фактический результат | Функция вернула false |
| Статус | Успешно |
| Предварительное условие | Кнопка “Авторизоваться” доступна для клика. |
| Постусловие | Вывод ошибки. |
| Примечание\комментарий | Нет комментариев |

Рассмотрим тестирование модуля регистрации (см. табл. 8-14).

Таблица 8. Тестовый пример 7

|  |  |
| --- | --- |
| Тестовый пример 7 | Messenger\_AuthPage |
| Приоритет тестирования | Высокий |
| Заголовок\название теста | TC\_Auth\_1 |
| Краткое изложение теста | Тест должен проверить модуль на пустые строки логина пароля и подтверждения пароля. |
|  |  |
| Этапы теста | 1. Не ввести логин  2. Не ввести пароль  3. Не ввести подтверждение пароля  3. Нажать на кнопку “Зарегистрироваться” |
| Тестовые данные |  |
| Ожидаемый результат | Программа должна вывести ошибку, что поля не заполнены |
| Фактический результат | Функция вернула false |
| Статус | Успешно |
| Предварительное условие | Кнопка “Зарегистрироваться” доступна для клика. |
| Постусловие | Вывод ошибки. |
| Примечание\комментарий | Нет комментариев |

Таблица 9. Тестовый пример 8

|  |  |
| --- | --- |
| Тестовый пример 8 | Messenger\_AuthPage |
| Приоритет тестирования | Высокий |
| Заголовок\название теста | TC\_Auth\_2 |
| Краткое изложение теста | Тест должен проверить модуль на существование логина в базе данных. |
| Этапы теста | 1. Ввести логин  2. Ввести пароль  3. Ввести подтверждение пароля  3. Нажать на кнопку “Зарегистрироваться” |
| Тестовые данные | admin - Логин  12345qrqy - Пароль  12345qrqy - Подтверждение пароля |

Продолжение таблицы 9

|  |  |
| --- | --- |
| Ожидаемый результат | Программа должна вывести ошибку, что логин сущестует |
| Фактический результат | Функция вернула false |
| Статус | Успешно |
| Предварительное условие | Кнопка “Зарегистрироваться” доступна для клика.  Логин существует в базе данных. |
| Постусловие | Вывод ошибки. |
| Примечание\комментарий | Нет комментариев |

Таблица 10. Тестовый пример 9

|  |  |
| --- | --- |
| Тестовый пример 9 | Messenger\_AuthPage |
| Приоритет тестирования | Высокий |
| Заголовок\название теста | TC\_Auth\_3 |
| Краткое изложение теста | Тест должен проверить модуль на несовпадение паролей. |
| Этапы теста | 1. Ввести логин  2. Ввести пароль  3. Ввести подтверждение пароля  3. Нажать на кнопку “Зарегистрироваться” |
| Тестовые данные | admin - Логин  12345qrqy - Пароль  54321yqrq - Подтверждение пароля |
| Ожидаемый результат | Программа должна вывести ошибку, что логин сущестует |
| Фактический результат | Функция вернула false |
| Статус | Успешно |
| Предварительное условие | Кнопка “Зарегистрироваться” доступна для клика. |
| Постусловие | Вывод ошибки. |

Таблица 11. Тестовый пример 10

|  |  |
| --- | --- |
| Тестовый пример 10 | Messenger\_AuthPage |
| Приоритет тестирования | Высокий |
| Заголовок\название теста | TC\_Auth\_4 |
| Краткое изложение теста | Тест должен проверить модуль на совпадение паролей. |

Продолжение таблицы 11

|  |  |
| --- | --- |
| Этапы теста | 1. Ввести логин  2. Ввести пароль  3. Ввести подтверждение пароля  3. Нажать на кнопку “Зарегистрироваться” |
| Тестовые данные | kratosDlogan - Логин  12345qrqy - Пароль  12345qrqy - Подтверждение пароля |
| Ожидаемый результат | Программа должна перенести пользователя на окно авторизации |
| Фактический результат | Функция вернула true |
| Статус | Успешно |
| Предварительное условие | Кнопка “Зарегистрироваться” доступна для клика.  Логина не существует в базе данных. |
| Постусловие | Выведено окно авторизации |
| Примечание\комментарий | Нет комментариев |

Таблица 12. Тестовый пример 11

|  |  |
| --- | --- |
| Тестовый пример 11 | Messenger\_AuthPage |
| Приоритет тестирования | Высокий |
| Заголовок\название теста | TC\_Auth\_5 |
| Краткое изложение теста | Тест должен проверить модуль на правильность введённого пароля. |
| Этапы теста | 1. Ввести логин  2. Ввести пароль  3. Ввести подтверждение пароля  3. Нажать на кнопку “Зарегистрироваться” |
| Тестовые данные | kratos\_logan - Логин  12345 - Пароль  12345 - Подтверждение пароля |
| Ожидаемый результат | Программа должна вывести ошибку, что пароль состоит менее чем из 6 символов |
| Фактический результат | Функция вернула false |
| Статус | Успешно |

Продолжение таблицы 12

|  |  |
| --- | --- |
| Предварительное условие | Кнопка “Зарегистрироваться” доступна для клика.  Логина не существует в базе данных. |
| Постусловие | Выведена ошибка, что пароль состоит менее чем из 6 символов. |
| Примечание\комментарий | Нет комментариев |

Таблица 13. Тестовый пример 12

|  |  |
| --- | --- |
| Тестовый пример 12 | Messenger\_AuthPage |
| Приоритет тестирования | Высокий |
| Заголовок\название теста | TC\_Auth\_6 |
| Краткое изложение теста | Тест должен проверить модуль на правильность введённого пароля. |
| Этапы теста | 1. Ввести логин  2. Ввести пароль  3. Ввести подтверждение пароля  3. Нажать на кнопку “Зарегистрироваться” |
| Тестовые данные | kratos\_logan - Логин  12345йцук - Пароль  12345йцук - Подтверждение пароля |
| Ожидаемый результат | Программа должна вывести ошибку, что пароль содержит кириллицу. |
| Фактический результат | Функция вернула false |
| Статус | Успешно |
| Предварительное условие | Кнопка “Зарегистрироваться” доступна для клика.  Логина не существует в базе данных. |
| Постусловие | Выведена ошибка, что пароль содержит кириллицу. |
| Примечание\комментарий | Нет комментариев |

Таблица 14. Тестовый пример 13

|  |  |
| --- | --- |
| Тестовый пример 13 | Messenger\_AuthPage |
| Приоритет тестирования | Высокий |
| Заголовок\название теста | TC\_Auth\_7 |

Продожение таблицы 14

|  |  |
| --- | --- |
| Краткое изложение теста | Тест должен проверить модуль на правильность введённого пароля. |
| Этапы теста | 1. Ввести логин  2. Ввести пароль  3. Ввести подтверждение пароля  3. Нажать на кнопку “Зарегистрироваться” |
| Тестовые данные | kratos\_logan - Логин  qrqypog - Пароль  qrqypog - Подтверждение пароля |
| Ожидаемый результат | Программа должна вывести ошибку, что пароль не содержит минимум 1 цифру. |
| Фактический результат | Функция вернула false |
| Статус | Успешно |
| Предварительное условие | Кнопка “Зарегистрироваться” доступна для клика.  Логина не существует в базе данных. |
| Постусловие | Выведена ошибка, что пароль не содержит минимум 1 цифру. |

Все написанные автоматические тесты были успешно выполнены, результаты тестирования отображены на рисунке 12, из этого следует соответствие модуля регистрации и авторизации требованиям.

**Изображение выглядит как текст, снимок экрана, программное обеспечение, Мультимедийное программное обеспечение

Автоматически созданное описание**

Рисунок 12. Результаты тестирования

На рисунке 13 продемонтрирована ошибка авторизации.

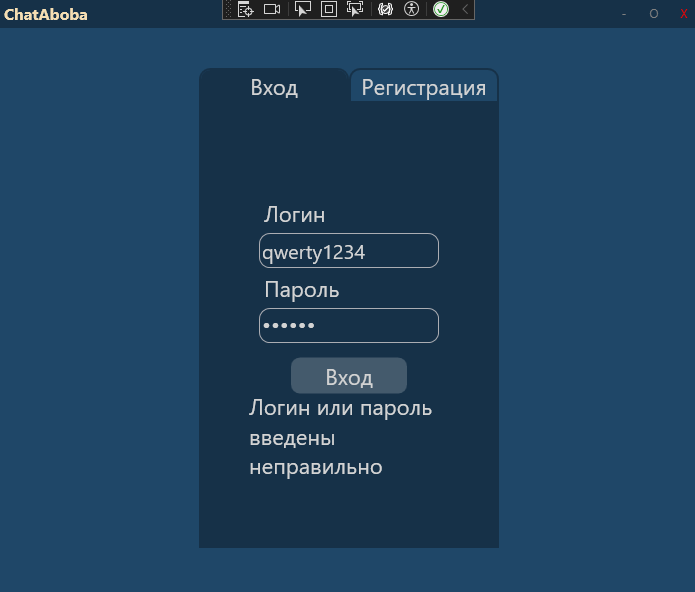


Рисунок 13. Ошибка авторизации

Так же на рисунке 14 изображена ошибка регистрации пользователя.

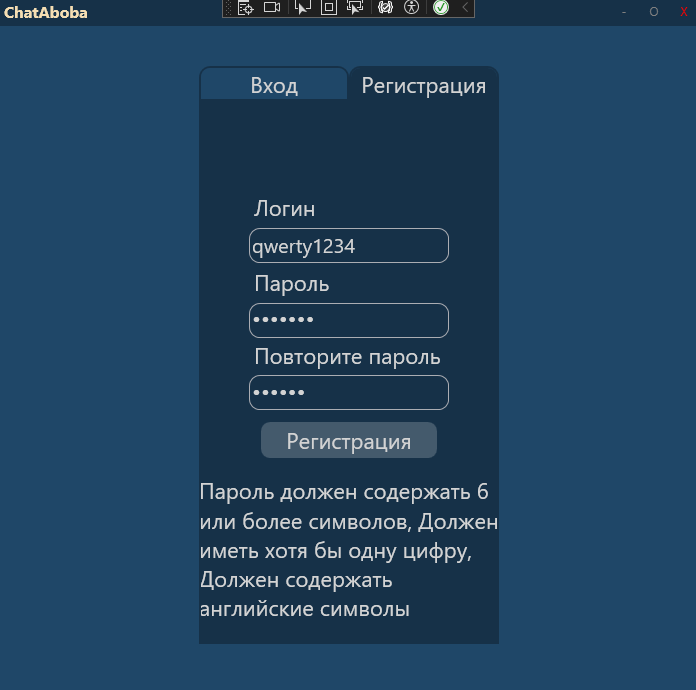


Рисунок 14. Ошибка регистрации

## **2.5. Руководство по использованию программы**

Руководство по использованию программы представлено отдельно для системного программиста, программиста и оператора.

Руководство системного программиста для мессенджера составлено по следующему плану:

- общие сведения о программе;

- структура программы;

- настройка программы;

- проверка программы;

- сообщения системному программисту.

Общие сведения о программе. Клиентское приложение отправляет запросы серверу через TCP подключение. Для работы программы необходиммы:

- стабильное интернет соединение;

- установленный .NET 7.

Общие сведения о программе. Клиентская часть состоит из окна, в котором много разнообразных страниц. Чтобы переключаться между ними надо нажимать на кнопки. Также стили всех объектов находятся в подключённом словаре стилей, как и языки программы.

Настройка программы. Для настройки программы необходимо выполнить несколько действий:

- установить .NET 7;

- запустить \*.sln файл и открыть в нём ClientServer.cs;

- поменять IP и Port на те, которые указаны на сервере.

Чтобы установить программу из инсталлятора необходимо:

- запустить setup.exe;

- выполнить необходимые действия и указать место установки;

- если не установлен .NET 7 инсталлятор перенаправит на сайт, где его можно установить;

- запустить ехе файл.

Проверка программы. Для проверки приложения небходимо открыть ехе файл и попробовать авторизоваться/зарегистрироваться, также если не получается перепроверить правильность введённых данных на примерах, которые указаны в параграфе 2.4, либо проверить правильность указаных IP и Port в ClientServer.cs.

Сообщение системному администратору. Большинство ошибок не связаных с авторизацией выводятся в серверной части приложенпя. Во время подключения людей к серверу в консоль будет выводиться, кто подключился, это означает что человек успешно использовал клиент и свои данные для авторизации. Любые ошибки будут писаться в консоль, программа не будет закрываться аварийно.

Руководство программиста написано по следующему плану:

- назначение и условия применения программы;

- требования программы;

- обращение к программе;

- входные и выходные данные;

- сообщения.

Назначение и условия применения программы. Клиентское приложение отправляет запросы серверу через TCP подключение. Для работы программы необходиммы:

- стабильное интернет соединение;

- установленный .NET 7.

Требования программы:

- операционная система Windows 7 и выше;

- объём памяти на жёстком диске – 100 мб;

- объём оперативного запоминающего устройства (ОЗУ) ­­­­­­­­­­­­­­­­­­­­­­­– 4 гб;

- доступ к стабильному интернету.

Обращение к программе. Для запуска программы необходимо:

- запустить setup.exe;

- выполнить необходимые действия и указать место установки;

- если не установлен .NET 7 инсталлятор перенаправит на сайт, где его можно установить;

- запустить ехе файл.

Входные и выходные данные. Входными данными мессенджера являются сообщения и файлы, которые пользователь отправляет другому пользователю, а также логин и пароль при регистрации или авторизации. Выходными данными же является то, что пользователь получает от другого пользователя и его панель чатов.

Сообщения. По ходу работы с программой при не правильном вводе данных во время регистрации или авторизации будут выводиться сообщения об ошибках, если они появились, то надо проверить правильность введённых данных.

Руководство оператора построено по следующим пунктам:

- назначение программы;

- условия выполнения программы;

- выполнение программы;

- сообщения оператору.

Назначение программы. Клиентское приложение отправляет запросы серверу через TCP подключение. Такие запросы как: получения списка сообщений чата, чаты, картинка профиля пользователя, список пользователей, картинка фона чата, список данных пользователей.

Условия выполнения программы. Для работы программы необходиммы:

- стабильное интернет соединение;

- установленный .NET 7.

Выполнение программы. При запуске программы оператора встречает окно регистрации и авторизации, оно продемонстрировано на рисунке 15.

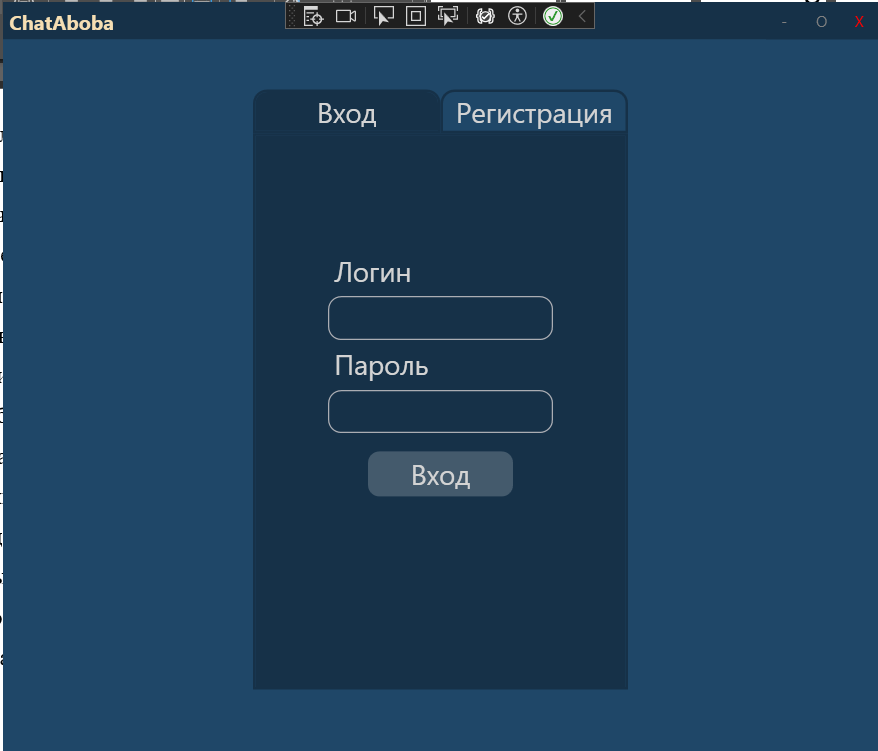


Рисунок 15. Окно авторизации и регистрации

Если пользователь не зарегистрирован, то ему надо это сделать и ввести корректные данные. Если данные введены неправильно, то выведется ошибка в зависимости от того, что введено не так:

- ошибка о несовпадающих паролях;

- ошибка о невозможности создания логина;

- ошибка о том, что пароль введён не верно.

После регистрации пользователю надо авторизоваться, тут уже другие ошибки:

- ошибка о том, что логина не существует;

- ошибка о не правильном пароле.

При авторизации в приложение пользователь увидит главную страницу приложения, она изображена на рисунке 16.

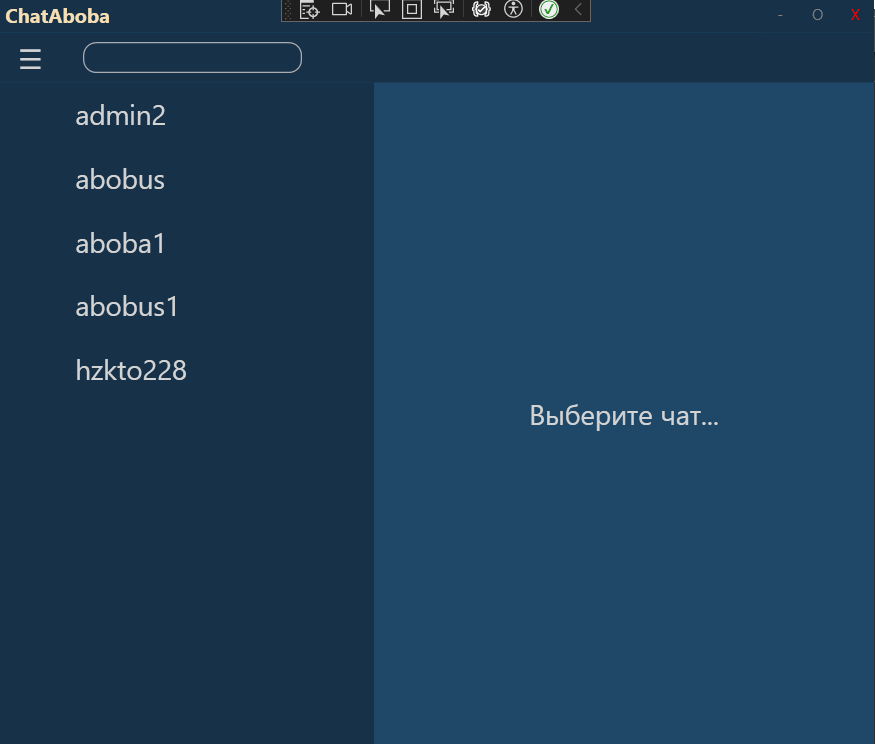


Рисунок 16. Главная страница

Чтобы выбрать чат, пользователю надо нажать на чат, который указан слева, если чатов нет, то надо найти пользователя в строке поиска справа от трё полосок. После поиска нужного пользователя требуется нажать на него и появится окно создания чата, показано на рисунке 17. Если же чат уже создан с пользователем, то по нажатию на него в поиске откроется существующий чат.

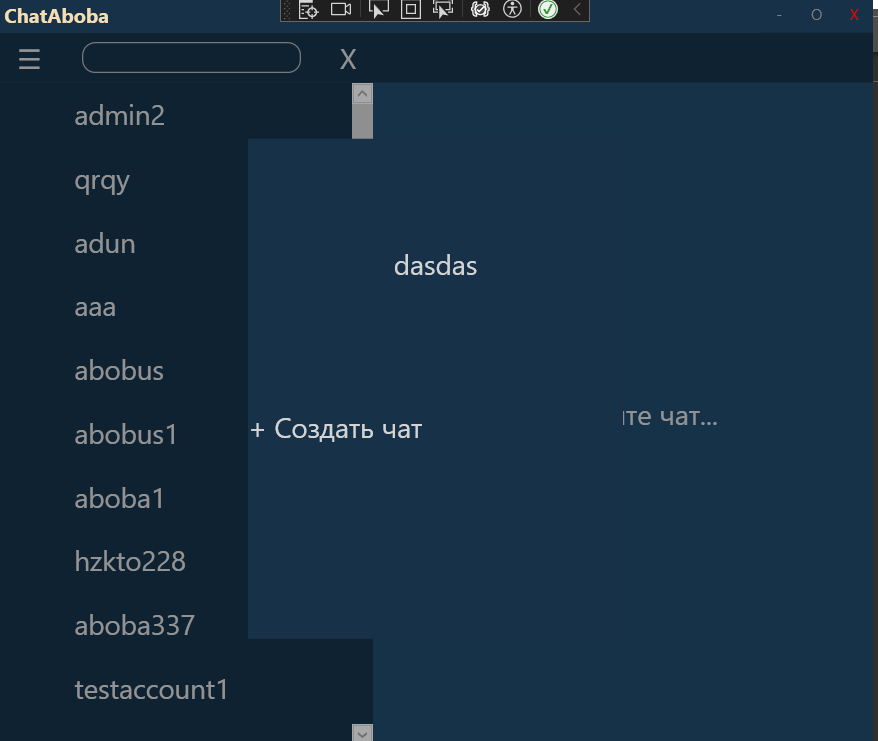


Рисунок 17. Окно создания чата

Чтобы поменять картинку профиля, необходимо нажать на три полоски, где откроется панель с профилем, сменой языка приложения и выходом из аккаунта. Далее нажать на кнопку Аккаунт, где откроется окно, где можно изменять информацию профиля. На рисунке 18 показана выдвигающаяся панель, а на рисунке 19 окно профиля.

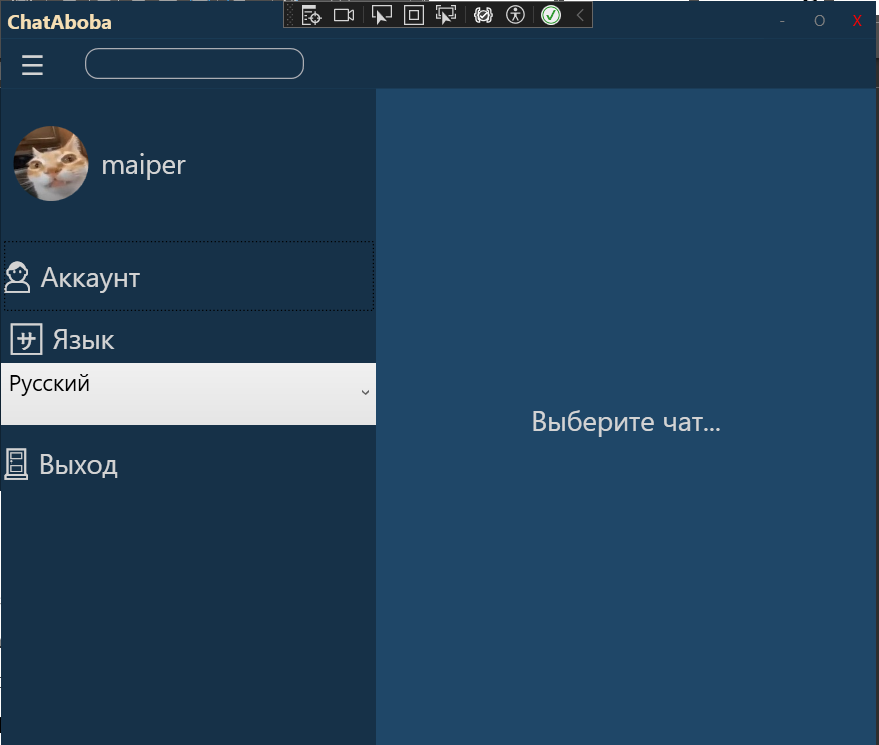


Рисунок 18. Выдвижная панель

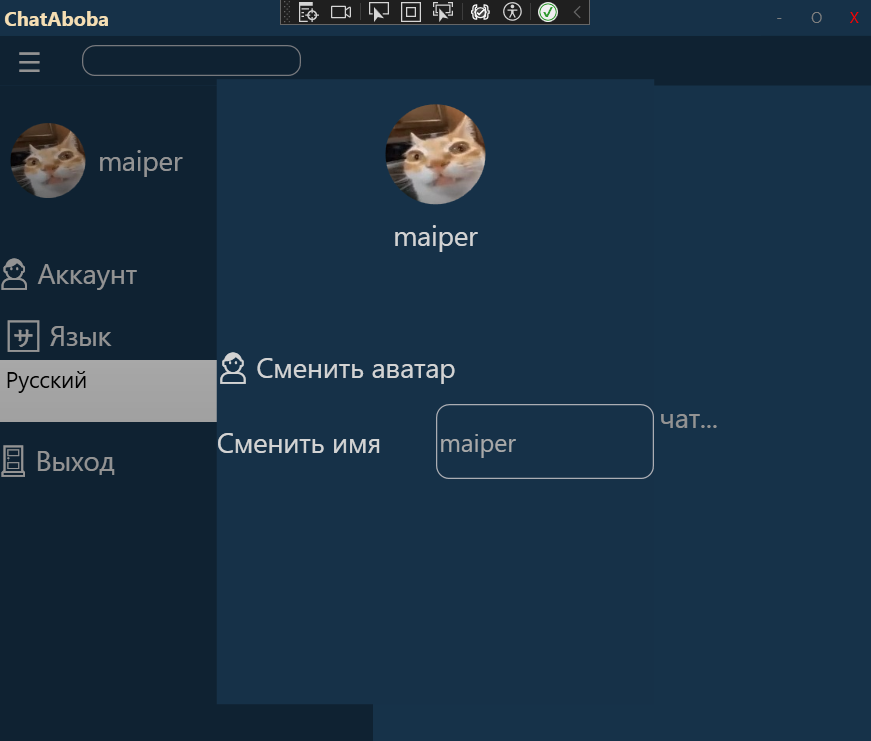


Рисунок 19. Окно профиля

Чтобы выйти из приложения, надо нажать на крестик в верхнем углу окна.

Сообщение оператору. По ходу работы с программой при не правильном вводе данных во время регистрации или авторизации будут выводиться сообщения об ошибках, если оно появилось, то надо проверить правильность введённых данных.

# **ЗАКЛЮЧЕНИЕ**

В процессе работы над дипломным проектом на тему «Разработка кроссплатформенного приложения «Мессенджер»» было предпринято исследование и разработка мессенджера, который представляет собой инструмент для обмена сообщениями, обеспечивая таким образом коммуникацию между пользователями.

В рамках проекта успешно достигнута поставленная цель - проектирование и разработка программного обеспечения «Мессенджер» с использованием языка программирования C# и графической подсистемы WPF, чему способствовало выполнение ряда задач.

Анализ особенностей передачи данных и ведущих представителей подобного программного обеспечения, осуществленный в ходе проекта, предоставил ценную информацию для выбора инструментальных средств и определения функциональных и эксплуатационных требований к разрабатываемому программному обеспечению.

Спроектированный интерфейс приложения стал сердцем разработки, а разработанное клиентское приложение претерпело успешную реализацию. Проведение этапа тестирования разработанного приложения позволило убедиться в качестве разработки и готовности продукта к использованию.

В рамках дипломного проектирования успешно выполнена разработка клиентской части мессенджера, предоставляющая широкий набор функциональных возможностей для пользователей. Программа реализована с учетом следующих функций:

1) регистрация пользователей: был разработан и внедрен механизм регистрации новых пользователей в системе, обеспечивающий безопасность и простоту процесса регистрации;

2) авторизация пользователей: был создан функционал, позволяющий пользователям авторизоваться в программе с использованием уникальных учетных данных;

3) поиск других пользователей: в мессенджере была внедрена функция поиска других пользователей, обеспечивающая возможность быстрого нахождения нужных контактов;

4) просмотр и редактирование профиля: пользователи имеют возможность просматривать свой профиль, а также вносить в него изменения с учетом своих предпочтений;

5) отправка сообщений пользователям: реализована возможность отправки текстовых сообщений между пользователями, обеспечивая связь и обмен информацией.

Таким образом, разработанная клиентская часть мессенджера предоставляет полный набор функций, необходимых для комфортного общения и обмена данными между пользователями, что делает проект востребованным и удобным для использования в повседневной деятельности.

# **СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ И ИНТЕРНЕТ-РЕСУРСОВ**

*Законодательные и нормативные акты:*

1. ГОСТ 7.1-2003 Библиографическая запись. Библиографическое описание. Общие требования и правила составления – 92 с.
2. ГОСТ 7.32-2017 Отчет о научно-исследовательской работе. Структура и правила оформления – 47 с.
3. ГОСТ 7.82-2001 Библиографическая запись. Библиографическое описание электронных ресурсов. Общие требования и правила составления – 39 с.
4. ГОСТ Р 7.0.100-2018 Библиографическая запись. Библиографическое описание. Общие требования и правила составления – 122 с.
5. ГОСТ Р 7.0.12-2011 Библиографическая запись. Сокращение слов и словосочетаний на русском языке. Общие требования и правила составления. – М.: ИПК Издательство стандартов, 2004. – 169 с.
6. ГОСТ Р 7.0.5-2008 Библиографическая ссылка. Общие требования и правила составления – 32 с.
7. Единая система программной документации. – М.: Стандартинформ, 2005. – 128 с.

*Учебная и научная литература:*

1. Агальцов, В.П. Математические методы в программировании: учебник / В.П. Агальцов. – 2-е изд., перераб. и доп. – Москва: ИД «ФОРУМ»: ИНФРА-М, 2023. – 240 с.
2. Мюллер Д. П. Си Шарп для чайников – Москва, Санкт-Петербург 2019 ООО "Диалектика" – Текст: электронный. – Режим доступа: <https://vk.com/doc161644846_634870821?hash=y8mbUO5UhKRlCR4tJkTnQsTg7BV4cQQhbNC6FkByWTP&dl=f4zdcFgzbAIpG5Qn0BCz1QINJJ49cQORMlAhF4bX8js> (дата обращения: 03.10.2023).

*Интернет документы:*

1. Аксенова Т.Г. Онлайн-курс по технологии разработки программного обеспечения [Электронный ресурс]. – Режим доступа:<https://classroom.google.com/u/2/w/NTc2MzExNTI0MTY0/t/all> (дата обращения: 03.10.2023).
2. Ресурс для сравнительного анализа мессенджер Telegram. [Электронный ресурс]. – Режим доступа: <https://web.telegram.org/> (дата обращения: 20.10.2023).
3. Ресурс для сравнительного анализа мессенджер Viber. [Электронный ресурс]. – Режим доступа: <https://www.viber.com/ru/> (дата обращения: 20.10.2023).
4. Ресурс для сравнительного анализа мессенджер WhatsApp. [Электронный ресурс]. – Режим доступа: <https://www.whatsapp.com/> (дата обращения: 20.10.2023).
5. Ресурс для сравнительного анализа мессенджер Вконтакте. [Электронный ресурс]. – Режим доступа: <https://vk.com> (дата обращения: 20.10.2023).
6. Руководство по работе в среде Visual Studio. – [Электронный ресурс]. – Режим доступа:<https://docs.microsoft.com/ru-ru/visualstudio/> (дата обращения: 28.09.2023).
7. Форум программистов и сисадминов - [Электронный ресурс]. – Режим доступа: <https://www.cyberforum.ru/unity/thread1230990.html> (дата обращения: 03.10.2023).
8. C# WPF UI | How to Design Chat Messaging App in WPF. [Электронный ресурс]. – Режим доступа: <https://youtu.be/tUSCm_t6Ypw?si=6kq_yLs2WYn6MAt> (дата обращения: 01.11.2023).
9. C# WPF UI Tutorials: 21 - Custom Dialog System Message Box Popup. [Электронный ресурс]. – Режим доступа: <https://youtu.be/jrgT-fbV2tM?si=rpP3KNq5OeZyygz4> (дата обращения: 25.10.2023).
10. WPF C# Professional Modern Chat App UI Tutorial. [Электронный ресурс]. – Режим доступа: <https://youtu.be/V9DkvcT27WI?si=IZqGPf56ZE-hPwxy> (дата обращения: 27.10.2023).
11. Полное руководство по языку программирования C# 6.0 и платформе .NET 4.6. – [Электронный ресурс]. – Режим доступа:<http://metanit.com/sharp/tutorial/> (дата обращения: 03.10.2023).

# **ПРИЛОЖЕНИЕ**

**Приложение №1**

**Авторизация пользователей**

Ниже представлен код запроса на авторизацию пользователя. При ошибке на клиенте выводится сообщение об ошибке.

public AsyncCommand ConnectCommand

{

get

{

return new AsyncCommand(() =>

{

return Task.Run(() =>

{

//try

//{

\_client = new TcpClient();

\_client.Connect(IP, Port);

\_reader = new StreamReader(\_client.GetStream());

\_writer = new StreamWriter(\_client.GetStream());

\_writer.AutoFlush = true;

\_writer.WriteLine("auth");

\_writer.WriteLine($"Login: {Nick}");

\_writer.WriteLine($"Password: {Password}");

//\_writer.WriteLine(Nick);

//\_writer.WriteLine(Password);

var result =\_reader?.ReadLine();

if (result == "OK")

{

var usId = \_reader?.ReadLine();

var IsAvatar = \_reader?.ReadLine()=="true";

if(IsAvatar){

AvatarString=\_reader?.ReadLine();

}

\_writer.WriteLine("GetChats");

viewModel = new ViewModel();

string aStr = \_reader.ReadLine();

viewModel.AddChatList(JsonConvert.DeserializeObject<List<Dictionary<string,string>>>(aStr));

Application.Current.Dispatcher.BeginInvoke((Action)(() =>

{

MainWindow mainWindowSus = Application.Current.MainWindow as MainWindow;

connected = true;

App.chatPage = new ChatPage(usId);

App.chatPage.DataContext = viewModel;

mainWindowSus.frameMenu.Navigate(App.chatPage);

}));

Listener();

}

else {

Application.Current.Dispatcher.BeginInvoke((Action)(() =>

{

App.loginPage.ErrorLogin.Visibility = Visibility.Visible;

}));

}

//}

//catch (Exception ex)

//{

// MessageBox.Show(ex.Message);

//}

});

}, () => \_client is null || \_client?.Connected == false);

}

}

**Приложение №2**

**Создание чатов**

Ниже указан код функции для создания чата с указаным пользователем.

public void CreateChat(int ChatID, string ChatName, int UserID, dynamic a, ListViewItem item1, ListViewItem item2)

{

if (App.server.viewModel.Users.Where(x=>x.ChatID == ChatID.ToString()).Count() > 0)

{

App.server.ChatID = Convert.ToInt32(ChatID);

Task.Run(() => App.server.GetBackground.Execute(this)).Wait();

Task.Run(() => App.server.GetMessages.Execute(this)).Wait();

item1.Background = new SolidColorBrush(Color.FromRgb(34, 76, 112));

if (item2 != null)

{

item2.Background = new SolidColorBrush(Color.FromRgb(22, 49, 72));

}

var c = App.server.viewModel.Users.Where(x => x.ChatID == ChatID.ToString()).Select(a => a.ChatName);

MessangerName.Content = c.First();

messagesView.Visibility = Visibility.Visible;

chatBox.Visibility = Visibility.Visible;

stub.Visibility = Visibility.Collapsed;

EditChatButton.Visibility = Visibility.Visible;

closeSearch.Visibility = Visibility.Collapsed;

bord.Visibility = Visibility.Visible;

bord2.Visibility = Visibility.Collapsed;

}

else

{

App.createChat = new CreateChat(ChatID, ChatName, null, 0, UserID);

mainWindow.frameMenu2.Visibility = Visibility.Visible;

mainWindow.frameMenu2.Navigate(App.createChat);

App.createChat.Focus();}}

**Приложение №3**

**Смена языка приложения**

Ниже представлен код функции по смене языка приложения.

private void LanguageChoose\_SelectionChanged(object sender, SelectionChangedEventArgs e)

{

App.save = new SaveData();

if (LanguageChoose.SelectedIndex == 0)

{

App.save.Save(App.save.data = new SaveData.MyData() { Language = 0 });

System.Windows.Application.Current.Resources.MergedDictionaries[1].Clear();

System.Windows.Application.Current.Resources.MergedDictionaries.Add(new ResourceDictionary() { Source = new Uri("pack://application:,,,/Res/Languages.en-US.xaml") });

}

else if (LanguageChoose.SelectedIndex == 1)

{

App.save.Save(App.save.data = new SaveData.MyData() { Language = 1 });

System.Windows.Application.Current.Resources.MergedDictionaries[1].Clear();

System.Windows.Application.Current.Resources.MergedDictionaries.Add(new ResourceDictionary() { Source = new Uri("pack://application:,,,/Res/Languages.ru-RU.xaml") });

}}