Министерство образования Республики Беларусь

Учреждение образования «Белорусский государственный университет информатики и радиоэлектроники»

Факультет компьютерных систем и сетей

Кафедра программного обеспечения информационных технологий

Дисциплина: Компьютерные системы и сети (КСиС)

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

к курсовой работе

на тему

TCP-ЧАТ

БГУИР КР 1-40 01 01 24 ПЗ

Студент: гр. 751002 Батышев А. Д.

Руководитель: Шимко И.В.

Минск 2019

**СОДЕРЖАНИЕ**

[Введение 5](#_Toc10469251)

[1 Анализ прототипов, литературных источников и формирование требований к проектируемому программному средству 6](#_Toc10469252)

[1.1 Анализ прототипов 6](#_Toc10469253)

[1.2 Анализ литературных источников 9](#_Toc10469254)

[1.3 Формирование требований к проектируемому программному средству 10](#_Toc10469255)

[1.4 Требования к надежности 11](#_Toc10469256)

[1.5 Отказы из-за некорректных действий оператора 12](#_Toc10469257)

[2 Анализ требований к пс и разработка функциональных требований 13](#_Toc10469258)

[2.1 Описание функциональности ПС 13](#_Toc10469259)

[2.2 Архитектурное решение 13](#_Toc10469260)

[3 Создание программного средства 15](#_Toc10469261)

[3.1 Разработка алгоритмов ПС и программирование отдельных модулей 15](#_Toc10469262)

[3.2 Описание классов и методов 19](#_Toc10469263)

[4 Тестирование и обеспечение качества 21](#_Toc10469264)

[4.1 Тестирование и анализ полученных данных 21](#_Toc10469265)

[5 Руководство по установке и использованию 22](#_Toc10469268)

[Заключение 24](#_Toc10469269)

[Список используемых источников 25](#_Toc10469270)

[Приложение А 26](#_Toc10469271)

# 

# ВВЕДЕНИЕ

Чат – средство обмена сообщениями по компьютерной сети в режиме реального времени, а также [программное обеспечение](https://ru.wikipedia.org/wiki/%D0%9F%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%BD%D0%BE%D0%B5_%D0%BE%D0%B1%D0%B5%D1%81%D0%BF%D0%B5%D1%87%D0%B5%D0%BD%D0%B8%D0%B5), позволяющее организовывать такое общение. Характерной особенностью является коммуникация именно в реальном времени или близкая к этому, что отличает чат от [форумов](https://ru.wikipedia.org/wiki/%D0%92%D0%B5%D0%B1-%D1%84%D0%BE%D1%80%D1%83%D0%BC) и других «медленных» средств. То есть, если на форуме можно написать вопрос и ждать, пока кто-нибудь посчитает нужным на него ответить (в то же время можно получить и несколько супов от разных пользователей), то в чате общение происходит только с теми, кто присутствует в нём в настоящий момент, а результаты обмена сообщениями могут и не сохраняться. В последнее время чаты значительно расширили свою функциональность за счёт красивого радующего глаз дизайна, а также большого числа улучшений. Например, появились возможности помещать одного или нескольких пользователей в черный список, сообщения которых после этого перестают быть видимыми тому, кто поместил их в черный список, при том для данной операции необязательно быть модератором или администратором чата. Такое бывает необходимо, когда пользователь не нарушает правила чата, но в то же время неприятен, либо пользователь по какой-то причине не наказывается администрацией чата за своё поведение. Также есть частные комнаты, где можно вести беседу с другим пользователем, при этом администрация не видит и не контролирует их общение. В некоторых чатах стало появляться несколько комнат для общения с разными правилами.

При реализации программного средства должны быть достигнуты следующие характеристики качества ПО: корректность – отсутствие/наличие дефектов в спецификации, проекте и реализации системы, практичность – легкость изучения и использования системы, надежность – способность системы выполнять необходимые функции в предопределенных условиях, адаптируемость – возможность использования системы без ее изменения в тех областях, на которые она не была ориентирована, устойчивость – способность системы продолжать работу при вводе недопустимых данных.

Требуется реализовать программное средство, которое позволит вести обмен текстовыми сообщениями в реальном времени между клиентами. Сервер в данном случае представляет собой программное средство, транслирующее сообщения всем клиентам. Клиент – приложение, позволяющее осуществлять подключение к предпочтительному серверу и отправляющее на него сообщения клиентов.

1. **АНАЛИЗ ПРОТОТИПОВ, ЛИТЕРАТУРНЫХ ИСТОЧНИКОВ И ФОРМИРОВАНИЕ ТРЕБОВАНИЙ К ПРОЕКТИРУЕМОМУ ПРОГРАММНОМУ СРЕДСТВУ**

## **1.1 Анализ прототипов**

Для успешной реализации программного средства необходимо ознакомиться с существующими аналогами в данной сфере. Анализ их достоинств и недостатков позволяет сформировать требования к проектируемому средству, учитывающие опыт существующих разработок, и внести в них улучшения или изменения. Были выбраны следующие аналоги: Vypress chat, Intranet chat, Mychat, ICQ.

Vypress Chat – [чат](https://ru.wikipedia.org/wiki/%D0%A7%D0%B0%D1%82_(%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B0)) для [локальных сетей](https://ru.wikipedia.org/wiki/%D0%9B%D0%BE%D0%BA%D0%B0%D0%BB%D1%8C%D0%BD%D0%B0%D1%8F_%D1%81%D0%B5%D1%82%D1%8C), разработанная компанией VyPRESS Research. Программа работает только в [локальных сетях](https://ru.wikipedia.org/wiki/%D0%9B%D0%BE%D0%BA%D0%B0%D0%BB%D1%8C%D0%BD%D0%B0%D1%8F_%D1%81%D0%B5%D1%82%D1%8C), возможности работы в сети [Интернет](https://ru.wikipedia.org/wiki/%D0%98%D0%BD%D1%82%D0%B5%D1%80%D0%BD%D0%B5%D1%82) у неё нет. Программа не требует [выделенного сервера](https://ru.wikipedia.org/wiki/%D0%92%D1%8B%D0%B4%D0%B5%D0%BB%D0%B5%D0%BD%D0%BD%D1%8B%D0%B9_%D1%81%D0%B5%D1%80%D0%B2%D0%B5%D1%80) – каждая запущенная копия программы является и сервером и клиентом одновременно, тем самым делая равнозначными всех участников.

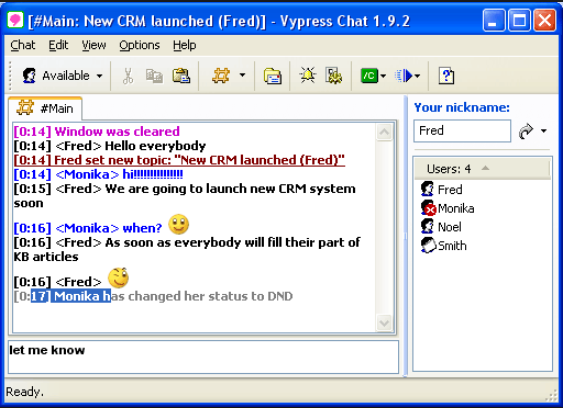


Рисунок 1.1.1 – Интерфейс Vypress Chat

Достоинства: не требует выделенного сервера, минималистичный [интерфейс](https://ru.wikipedia.org/wiki/%D0%9F%D0%BE%D0%BB%D1%8C%D0%B7%D0%BE%D0%B2%D0%B0%D1%82%D0%B5%D0%BB%D1%8C%D1%81%D0%BA%D0%B8%D0%B9_%D0%B8%D0%BD%D1%82%D0%B5%D1%80%D1%84%D0%B5%D0%B9%D1%81), встроенная поддержка русского языка, заполнение личной информации и получение подобной о другом пользователе, фильтрация нежелательных сообщений, защита от [флуда](https://ru.wikipedia.org/wiki/%D0%A4%D0%BB%D1%83%D0%B4), поддержка [Юникода](https://ru.wikipedia.org/wiki/%D0%AE%D0%BD%D0%B8%D0%BA%D0%BE%D0%B4).

Недостатки: теряет сообщения (нет гарантированной доставки), достаточно сильно засоряет сеть многоадресными сообщениями, особенно при большом количестве пользователей, существуют средства перехвата приватных сообщений для старых версий, общая нестабильная сетевая поддержка.

Intranet Chat – компьютерная [программа](https://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D0%BC%D0%BF%D1%8C%D1%8E%D1%82%D0%B5%D1%80%D0%BD%D0%B0%D1%8F_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B0), предназначенная для обмена сообщениями между компьютерами, объединёнными в локальную сеть, в среде [Windows](https://ru.wikipedia.org/wiki/Windows" \o "Windows).

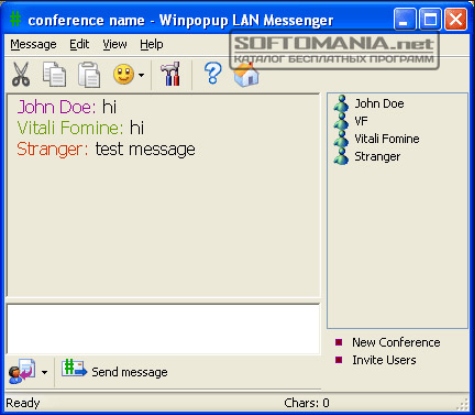


Рисунок 1.1.2 – Интерфейса программы Intranet Chat

Достоинства: обширный функционал, работа с многочисленными клиентами, множественная локализация, масштабируемость.

Недостатки: Непривлекательный пользовательский интерфейс, шифрование данных при передаче слабое, что делает возможным подмену никнеймов в сообщениях, смену чужих никнеймов , спам. Также многие пользователи высказывали недовольство ограниченным набором «украшений», особенно графических [смайликов](https://ru.wikipedia.org/wiki/%D0%A1%D0%BC%D0%B0%D0%B9%D0%BB%D0%B8%D0%BA).

MyChat – [клиент-серверное](https://ru.wikipedia.org/wiki/%D0%9A%D0%BB%D0%B8%D0%B5%D0%BD%D1%82-%D1%81%D0%B5%D1%80%D0%B2%D0%B5%D1%80) [программное обеспечение](https://ru.wikipedia.org/wiki/%D0%9F%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%BD%D0%BE%D0%B5_%D0%BE%D0%B1%D0%B5%D1%81%D0%BF%D0%B5%D1%87%D0%B5%D0%BD%D0%B8%D0%B5) для передачи текстовых сообщений с гарантированной доставкой.

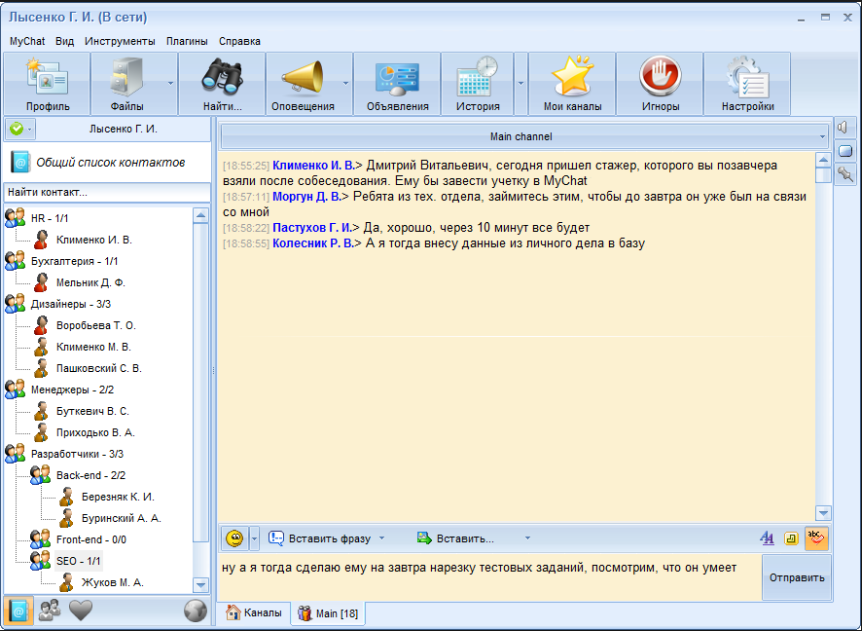


Рисунок 1.1.3 – Скриншот интерфейса программы MyChat

Достоинства: MyChat построен по технологии клиент-сервер, что позволяет ему работать как в обычной [локальной сети](https://ru.wikipedia.org/wiki/%D0%9B%D0%BE%D0%BA%D0%B0%D0%BB%D1%8C%D0%BD%D0%B0%D1%8F_%D0%B2%D1%8B%D1%87%D0%B8%D1%81%D0%BB%D0%B8%D1%82%D0%B5%D0%BB%D1%8C%D0%BD%D0%B0%D1%8F_%D1%81%D0%B5%D1%82%D1%8C), так и в сети со сложной многоуровневой конфигурацией, а также через [Интернет](https://ru.wikipedia.org/wiki/%D0%98%D0%BD%D1%82%D0%B5%D1%80%D0%BD%D0%B5%D1%82), использование [SSL](https://ru.wikipedia.org/wiki/SSL) для шифрования передаваемых сообщений, [авторизация](https://ru.wikipedia.org/wiki/%D0%90%D0%B2%D1%82%D0%BE%D1%80%D0%B8%D0%B7%D0%B0%D1%86%D0%B8%D1%8F) пользователей на сервере и централизованное хранение личных данных, базовое управление сервером с консоли клиента, чат полностью поддерживает [Unicode](https://ru.wikipedia.org/wiki/Unicode" \o "Unicode) в интерфейсе и отправляемых сообщениях, открытый протокол обмена данными, основанный на [JSON](https://ru.wikipedia.org/wiki/JSON), [веб-чат](https://ru.wikipedia.org/wiki/%D0%92%D0%B5%D0%B1-%D1%87%D0%B0%D1%82) для общения пользователей без установки приложения, используя [браузер](https://ru.wikipedia.org/wiki/%D0%91%D1%80%D0%B0%D1%83%D0%B7%D0%B5%D1%80), встроенные антимат и [антифлуд](https://ru.wikipedia.org/wiki/%D0%A4%D0%BB%D1%83%D0%B4" \o "Флуд) фильтры.

Недостатки: графический интерфейс на сервере, нет возможности объединять сервера, ОС до Windows 2000 не поддерживаются.

ICQ – бесплатная [система мгновенного обмена текстовыми сообщениями](https://ru.wikipedia.org/wiki/%D0%A1%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D0%B0_%D0%BC%D0%B3%D0%BD%D0%BE%D0%B2%D0%B5%D0%BD%D0%BD%D0%BE%D0%B3%D0%BE_%D0%BE%D0%B1%D0%BC%D0%B5%D0%BD%D0%B0_%D1%81%D0%BE%D0%BE%D0%B1%D1%89%D0%B5%D0%BD%D0%B8%D1%8F%D0%BC%D0%B8) для мобильных и иных платформ с поддержкой голосовой и видеосвязи. Позволяет пересылать текстовые сообщения, изображения, видео и аудио через Интернет. Клиент работает на платформах [Android](https://ru.wikipedia.org/wiki/Android" \o "Android), [iOS](https://ru.wikipedia.org/wiki/IOS" \o "IOS) (iPhone), [Windows Phone](https://ru.wikipedia.org/wiki/Windows_Phone" \o "Windows Phone), [Nokia Symbian](https://ru.wikipedia.org/wiki/Symbian_OS" \o "Symbian OS), [Nokia S40](https://ru.wikipedia.org/wiki/S40" \o "S40), а также ОС [Windows](https://ru.wikipedia.org/wiki/Windows" \o "Windows). Через сервер осуществляется поиск и связь с другими клиентами, а обмен служебными данными, сообщениями между пользователями может осуществляться как через сервер, так и без его участия.

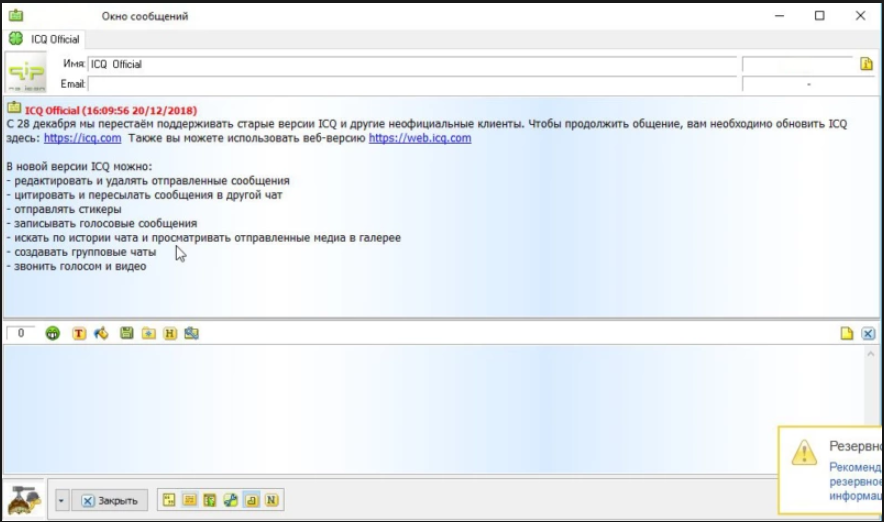


Рисунок 1.1.4 – Скриншот интерфейса программы ICQ

Достоинства: кроссплатформенность, удобное управление, простой интерфейс, бесплатное приложение, небольшая требовательность к аппаратным ресурсам компьютера, отправка текста любой длины путём его разбивки на отдельные сообщения самим клиентом, шифрование передаваемых сообщений, если оба собеседника используют совместимый клиент, запрет отправки собеседнику уведомлений о наборе сообщения.

Недостатки: обильное наличие рекламы, официальный клиент не поддерживает ни шифрование связи с сервером, ни между клиентами - т.е., Ваши пароли и разговоры могут быть легко доступны посторонним лицам. Кроме того, через сеть ICQ рассылается множество вирусов, которые могут многое, от кражи UIN'а и рассылки через него спама до полного захвата управления компьютером, проблемы при использовании в коммерческих целях, множество ограничений.

Анализ показал, что большая часть существующих аналогов программного средства обладают удобным для пользователя интерфейсом и широким спектром функциональных возможностей. Целью курсового проектирования является создание ПС, которое обладает достоинствами рассмотренных программ.

## **1.2 Анализ литературных источников**

При написании программного средства была выбрана платформа .NET Framework и язык программирования C#. В печатном издании Герберта Шилдта “C# 4.0 Полное руководство” детально рассмотрены все основные средства языка [1]. В книге подробно описываются возможности C#, даются профессиональные рекомендации и приводятся сотни примеров программ, охватывающие все аспекты программирования на C#. К недостаткам этой книги можно отнести часто встречающиеся синтаксические ошибки, связанные с проблемами перевода оригинального издания, как в коде программ, так и в объяснениях автора.

Книга Бориса Пахомова “C# для начинающих” является руководством для начинающих по разработке приложений на языке C# [2]. Основным достоинством данного издания является большой набор практического материала, с помощью которого можно отработать теоретические знания, выполняя различные задания. Автор кратко, но в полной мере описывает основные элементы и конструкции языка, концепции их использования, а также демонстрирует как правильное, так и неправильное использование тех или иных ключевых слов. Из недостатков можно отметить отсутствие сложных, однако важных для успешной реализации данного программного средства конструкций языка C#.

В книге Э.Джонса и Д.Оланда “Программирование в сетях Windows” [3] детально описан интерфейс прикладного программирования Winsock, рассмотрены семейства адресов, включая протоколы IP, IPX, NetBIOS, AppleTalk и ATM. Основным достоинством данной книги является подробное описание сетевой модели OSI и клиент-серверной архитектуры.

Веб-сайт msdn.microsoft.com представляет собой документацию для разработчиков, использующих программные продукты компании Microsoft. В разделе .NET есть подробное описание всех инструкций и особенностей их поведения [4]. Преимущества данного ресурса: наличие большого количества примеров, посвященных каждой теме, все описание излагается на английском языке, что не приводит к искажению оригинальной информации, удобная навигационная панель, позволяющая, например, выбрав какой-либо класс, просмотреть все его доступные методы и свойства.

Печатное издание “Совершенный код” Стива Макконнелла является практическим руководством по разработке программного обеспечения [5]. В

данной книге обсуждаются такие детали конструирования, как этапы создания классов, использование данных и управляющих структур, отладка, рефакторинг, написание самодокументирующегося кода, методики оптимизации и характеристики качества программы. В конце каждого раздела представлен список контрольных вопросов, позволяющий оценить качество архитектуры ПС, подход к проектированию, качество классов и методов, имена переменных и многое другое.

## **1.3 Формирование требований к проектируемому программному средству**

**1.3.1 Назначение разработки**

Функциональным назначением программного средства является предоставление возможности передачи текстовых сообщений между пользователями по протоколу TCP.

**1.3.2 Состав выполняемых функций**

Программное средство должно обеспечивать возможность выполнения следующих функций:

* функция ввода данных;
* функция вывода данных;
* функция передачи данных;
* функция запуска сервера;
* функция присоединения клиентов;
* функция передачи сообщения;
* функция получения сообщения;

**1.3.3 Входные данные**

Входные данные для программного средства должны быть представлены в виде: никнейм пользователя и печать сообщений.

**1.3.4 Выходные данные**

В качестве выходных данных выступают письма, отправленные пользователями.

**1.3.5 Обоснование выбора языка и сред разработки**

Язык программирования C# лежит в основе разработки программного средства, целевой платформой которой является .NET Framework 4.6.1. Достоинствами C# являются простота использования, объектная ориентированность, “сборка мусора”, поддержка совместимости версий и многое другое. Унифицирование системы типов позволяет рассматривать каждый тип как объект. Таким образом, используя класс, структуру, массив или встроенный тип, обращаться к ним можно как к объекту. Все объекты собраны в пространства имен (namespaces). Это позволяет вместо списка подключаемых файлов указывать, какие пространства имен, для доступа к объектам и классам внутри них, будут использованы в программе.

## **1.4 Требования к надежности**

Надежное функционирование программы должно быть обеспечено выполнением совокупности организационно-технических мероприятий, перечень которых приведен ниже:

1.4.1 Организацией бесперебойного питания технических средств;

1.4.2 Выполнением требований ГОСТ 31078-2002. Испытания

программных средств на наличие компьютерных вирусов.

## 1.5 Отказы из-за некорректных действий оператора

## Отказы программы возможны вследствие некорректных действий пользователя при взаимодействии с операционной системой. Во избежание возникновения отказов программы по указанной выше причине следует обеспечить работу конечного пользователя без предоставления ему ограниченного доступа и максимально безотказную работу программы при любых входных данных, а также при любых действиях пользователя.

**2 АНАЛИЗ ТРЕБОВАНИЙ К ПС И РАЗРАБОТКА ФУНКЦИОНАЛЬНЫХ ТРЕБОВАНИЙ**

## **2.1 Описание функциональности ПС**

Пользователь имеет следующие возможности:

1. Запуск сервера.
2. Подключение клиента к серверу.
3. Отправка сообщений.
4. Получение сообщений.
5. Видеть какой пользователь зашел в чат.
6. Видеть какой пользователь покинул чат.

## **2.2 Архитектурное решение**

Данное программное средство будет состоять из двух частей: серверной и клиентской.

Отправка и приём сообщений

Пересылка сообщений

Клиентская часть

Серверная часть

Клиент-серверное приложение

Рисунок 2.2.1 – Структура приложения Чат

Серверное приложение предназначено для управления системой. После запуска сервер сразу же проинициализирует начальный набор данных и перейдет в режим «прослушивания», и будет ждать подключений. Как только будет зафиксировано подключение, автоматически создастся новый сокет для обмена данными с новым клиентом.

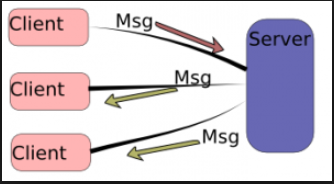


Рисунок 2.2.2 – Структура отправки сообщений

Для получения сообщений необходимо через клиентское приложение подключиться к серверу. Если подключение пройдет удачно, то пользователь получит возможность принимать и отправлять сообщения. Список пользователей будет обновляться сразу же, как только в нем появятся какие-либо изменения.

# 3 СОЗДАНИЕ ПРОГРАММНОГО СРЕДСТВА

## **3.1 Разработка алгоритмов ПС и программирование отдельных модулей**

На рисунке 3.1.1 представлена схема алгоритма работы программного средства.



Рисунок 3.1.1 – Схема работы программы

На рисунке 3.1.2 представлена схема алгоритма работы прослушивания подключений к серверу.



Рисунок 3.1.2 – Схема работы сервера

Запускается поток tcpListener для прослушки соединений по любому IP и порту 8888. Далее в бесконечном цикле при подключении клиента создается новый поток для каждого отдельно.

На рисунке 3.1.3 представлена схема алгоритма получения сообщения клиентом.



Рисунок 3.1.3 – Схема получения сообщения клиентом

Первоначально создается буффер для получения данных, затем создается динамическая строка builder. Пока доступны данные по данному соединению мы считываем их в builder. В конце метода строка выводится в консоль.

На рисунке 3.1.4 представлена схема алгоритма получения сообщения сервером от клиента.



Рисунок 3.1.4 – Схема получения сообщения сервером

В начале получаем никнейм клиента и оповещаем об этом всех подключенных пользователей. Далее в бесконечном цикле получаем сообщения от этого клиента и рассылаем их другим.

## **3.2 Описание классов и методов**

Все реализованные в программе классы обладают хорошей инкапсуляцией, которая, в отличие от абстракции, не позволяет узнать детали реализации. Однако удачная инкапсуляция помогает минимизировать доступ к классам и снизить сопряжение между ними.

Методы, реализованные в классах программы, позволяют повысить удобочитаемость кода, его надежность, а также облегчить изменение. Имена методов полностью описывают все выходные данные и побочные эффекты.

Класс ClientObject содержит следующие методы и свойства:

* protected internal string Id { get; private set; } – свойство, которое

хранит ID клиента.

* protected internal NetworkStream SslStream { get; private set; } –

свойство, которое предоставляет поток, используемый для связи клиент-сервер.

* string userName – свойство, хранящее никнейм пользователя.
* TcpClient client – класс, нужный для создания программы,

работающей с протоколом TCP.

* ServerObject server – объект сервера.
* public void Process() – метод, предназначенный для получения

информации о клиенте.

* private string GetMessage() – метод, предназначенный для чтения

входящего сообщения из потока и преобразование его в строку.

* protected internal void Close() – метод, предназначенный для закрытия

подключения к серверу.

Класс ServerObject содержит следующие методы и свойства:

* static TcpListener tcpListener – сервер для прослушивания.
* List<ClientObject> clients = new List<ClientObject>() – список

клиентов на сервере.

* protected internal void AddConnection(ClientObject clientObject) –

метод, предназначенный для добавления клиента в список подключений.

* protected internal void RemoveConnection(string id) – метод,

предназначенный для удаления клиента из списка подключений.

* protected internal void Listen() – метод, предназначенный для

прослушивания входящих подключений.

* protected internal void BroadcastMessage(string message, string id) –

метод, предназначенный для трансляции сообщения подключенным клиентам.

* Close() – метод, предназначенный для закрытия подключений.
* Discoconnect() – метод, предназначенный для отключения всех

клиентов.

# 4 ТЕСТИРОВАНИЕ И ОБЕСПЕЧЕНИЕ КАЧЕСТВА

## **4.1 Тестирование и анализ полученных данных**

Тестирование является самой популярной методикой повышения качества ПО. Существует множество видов тестирования, однако работа данного программного средства проверялась на корректность с помощью функционального метода. Данный подход используется с целью проверки реализуемости функциональных требований, то есть способности ПО в определённых условиях решать задачи, нужные пользователям.

Таблица 1 – Тестирование функциональных требований

|  |  |  |  |
| --- | --- | --- | --- |
| № тест-кейса | Тестируемая  функция | Последовательность действий | Ожидаемый результат |
| 1 | Запуск сервера | Запустить сервер и ждать подключений | Сервер запущен |
| 2 | Подключение клиента к серверу | Запустить программу ChatClient и подключить нового клиента к запущенному серверу | Клиент подключен к серверу |
| 3 | Отправка сообщений | Отправление сообщений на сервер и другим клиентам | Сообщение получено сервером и другими пользователями |
| 4 | Получение сообщений | Отправление сообщений на сервер и другим клиентам | Сообщение получено сервером и другими пользователями |
| 5 | Видеть, какой пользователь зашел в чат | Подключение клиента к серверу | Пользователи увидели нового клиента |
| 6 | Видеть, какой пользователь покинул чат | Отключение клиента от сервера | Пользователи увидели отключение клиента |

Успешное прохождение тестов доказывает устойчивость программного средства к любым действиям со стороны пользователя. Программа уведомит клиента при отсутствии подключения к интернету, а также при попытке ввода некорректных входных данных.

# 5 РУКОВОДСТВО ПО УСТАНОВКЕ И ИСПОЛЬЗОВАНИЮ

При запуске сервера появляется окно, представленное на рисунке 5.1. Если появилась надпись “Сервер запущен. Ожидание подключений…”, то это означает, что сервер запустился без ошибок.

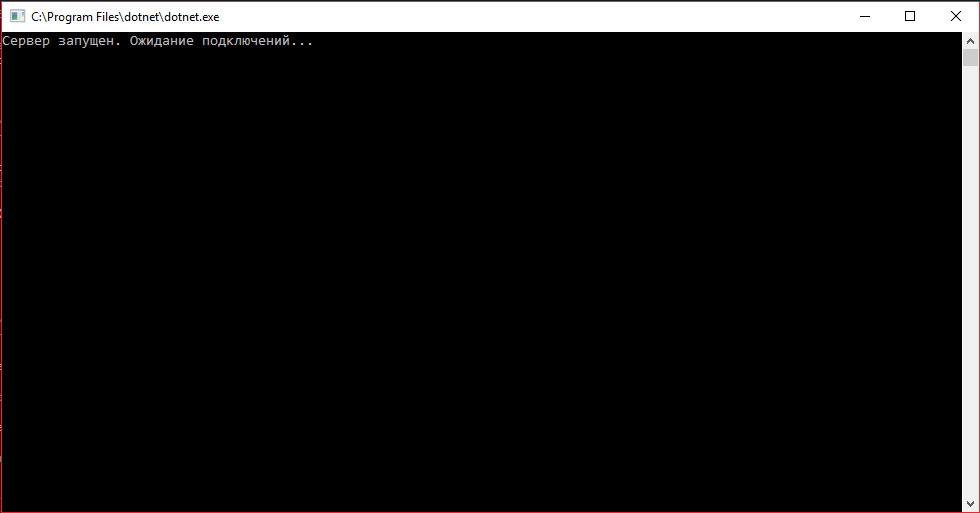


Рисунок 5.1 – Сервер

Далее пользователь должен запустить программу ChatClient. При ее запуске появляется окно, представленное на рисунке 5.2. Пользователь должен придумать себе имя для чата.

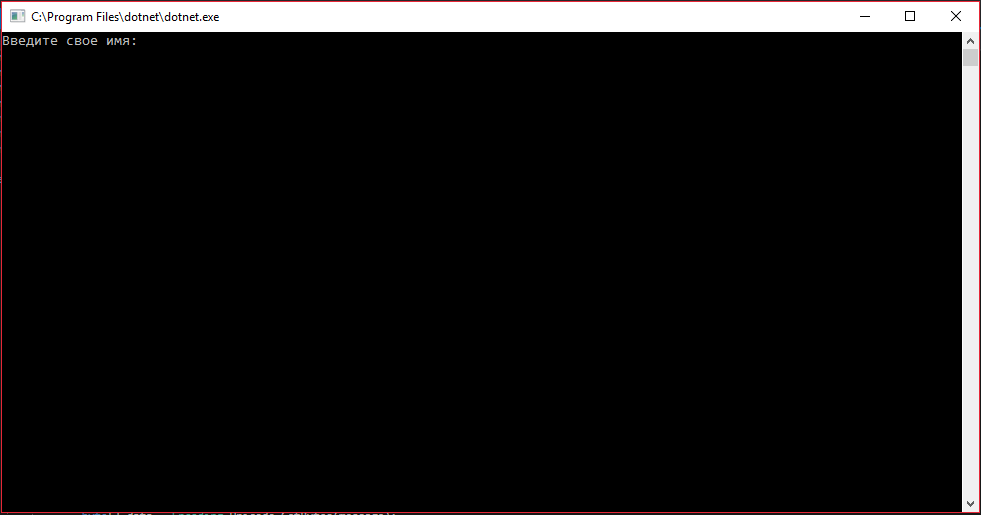


Рисунок 5.2 – Клиент

После того, как пользователь придумает себе имя он подключится к серверу. Затем на сервер придет уведомление о данном пользователе.



Рисунок 5.3 – Пользователь подключен к серверу

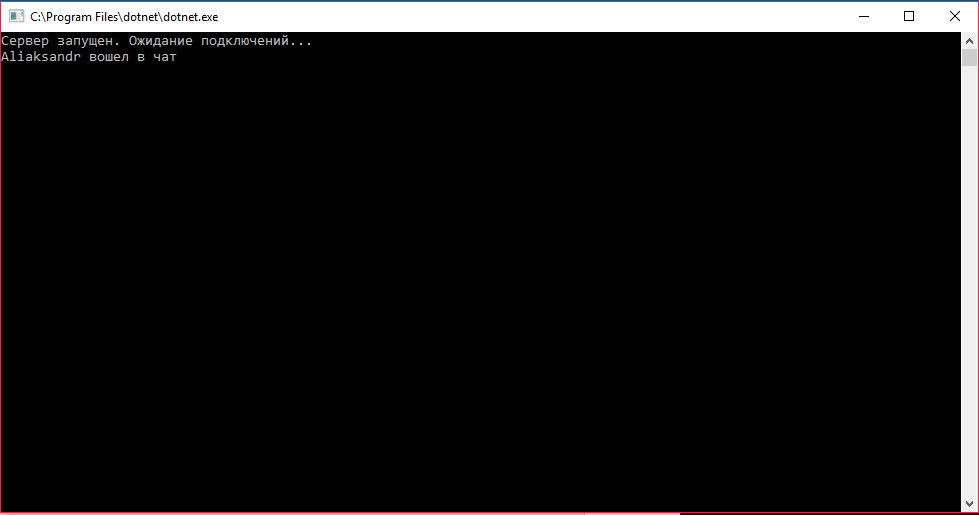


Рисунок 5.4 – Уведомление сервера о подключении

Затем можно подключить еще несколько пользователей и вести диалог.



Рисунок 5.5 – Пример использования чата

# ЗАКЛЮЧЕНИЕ

В ходе работы над курсовым проектом были проанализированы методы для работы с сокетами. Также были рассмотрены различные способы взаимодействия по сети, архитектурные принципы создания клиент-серверных приложений, изучены различные сетевые протоколы. Проведен анализ предметной области, исследованы существующие аналоги.

Результатом анализа явилось обобщение достоинств и недостатков существующих решений, которые учтены при разработке функциональных требований к разработанному программному средству.

На основе функциональных требований была разработана архитектура ПС, которая соответствует всем основным характеристикам качества. Описаны основные подходы, которые были использованы для создания высококачественных интерфейсов классов и методов. Произведено проектирование программного средства, представлена разработка алгоритмов отдельных модулей. Произведена оптимизация кода для улучшения быстродействия ПС. Выполнено функциональное тестирование, успешное прохождение которого доказало отказоустойчивость программы. Система спроектирована таким образом, что некоторые ее части можно повторно использовать в других проектах.

# СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ

[1] Шилдт, Г. C# 4.0 – Полное руководство / Г. Шилдт. – Вильямс, 2019. – 1056 c.

[2] Пахомов, Б.И. C# для начинающих / Б.И. Пахомов. – БХВ, 2014. – 432с.

[3] Оланд, Д. Программирование в сетях Microsoft Windows / Д. Оланд, Э. Джонс. – Питер, 2002. – 594.

[4] metanit.com [Электронный ресурс]. – Электронные данные. – Режим доступа: <https://metanit.com/sharp/net>.

[5] Макконнелл, С. Совершенный код / С. Макконнелл. – БХВ, 2018. – 861с.

[6] msdn.microsoft.com [Электронный ресурс]. – Электронные данные. – Режим доступа: [https://docs.microsoft.com/en-us/dotnet/csharp/language-reference](https://docs.microsoft.com/en-us/dotnet/csharp/language-reference/)

[7] Wikipedia.org [Электронный ресурс]. – Электронные данные. – Режим доступа: <https://ru.wikipedia.org/wiki/Online_chat>.

# ПРИЛОЖЕНИЕ А

# (обязательное)

# Исходный код

ChatClient:

using System;

using System.Threading;

using System.Net.Sockets;

using System.Text;

using System.Net.Security;

using System.Security.Authentication;

using System.IO;

namespace ChatClient

{

class Program

{

static string userName;

private const string host = "127.0.0.1";

private const int port = 8888;

static TcpClient client;

static NetworkStream Sslstream;

//static SslStream Sslstream;

static void Main(string[] args)

{

Console.Write("Введите свое имя: ");

userName = Console.ReadLine();

client = new TcpClient();

try

{

client.Connect(host, port); //подключение клиента

//SslStream Sslstream = new SslStream(client.GetStream());

 Sslstream = client.GetStream(); // получаем поток

string message = userName;

byte[] data = Encoding.Unicode.GetBytes(message);

Sslstream.Write(data, 0, data.Length);

// запускаем новый поток для получения данных

Thread receiveThread = new Thread(new ThreadStart(ReceiveMessage));

receiveThread.Start(); //старт потока

                Console.WriteLine("Добро пожаловать, {0}", userName);

SendMessage();

}

catch (Exception ex)

{

Console.WriteLine(ex.Message);

}

finally

{

Disconnect();

}

}

        // отправка сообщений

        static void SendMessage()

{

Console.WriteLine("Введите сообщение: ");

while (true)

{

string message = Console.ReadLine();

byte[] data = Encoding.Unicode.GetBytes(message);

Sslstream.Write(data, 0, data.Length);

}

}

        // получение сообщений

        static void ReceiveMessage()

{

while (true)

{

try

{

byte[] data = new byte[64]; // буфер для получаемых данных

                    StringBuilder builder = new StringBuilder();

int bytes = 0;

do

{

bytes = Sslstream.Read(data, 0, data.Length);

builder.Append(Encoding.Unicode.GetString(data, 0, bytes));

}

while (Sslstream.DataAvailable);

string message = builder.ToString();

Console.WriteLine(message);//вывод сообщения

                }

catch

{

Console.WriteLine("Подключение прервано!"); //соединение было прервано

                    Console.ReadLine();

Disconnect();

}

}

}

static void Disconnect()

{

if (Sslstream != null)

Sslstream.Close();//отключение потока

            if (client != null)

client.Close();//отключение клиента

            Environment.Exit(0); //завершение процесса

        }

}

}

ChatServer:

- ServerObject:

using System;

using System.Collections.Generic;

using System.Linq;

using System.Net.Sockets;

using System.Net.Security;

using System.Net;

using System.Text;

using System.Threading;

namespace ChatServer

{

public class ServerObject

{

static TcpListener tcpListener; // сервер для прослушивания

List<ClientObject> clients = new List<ClientObject>(); // все подключения

protected internal void AddConnection(ClientObject clientObject)

{

clients.Add(clientObject);

}

protected internal void RemoveConnection(string id)

{

// получаем по id закрытое подключение

ClientObject client = clients.FirstOrDefault(c => c.Id == id);

// и удаляем его из списка подключений

if (client != null)

clients.Remove(client);

}

// прослушивание входящих подключений

protected internal void Listen()

{

try

{

tcpListener = new TcpListener(IPAddress.Any, 8888);

tcpListener.Start();

Console.WriteLine("Сервер запущен. Ожидание подключений...");

while (true)

{

TcpClient tcpClient = tcpListener.AcceptTcpClient();

ClientObject clientObject = new ClientObject(tcpClient, this);

Thread clientThread = new Thread(new ThreadStart(clientObject.Process));

clientThread.Start();

}

}

catch (Exception ex)

{

Console.WriteLine(ex.Message);

Disconnect();

}

}

// трансляция сообщения подключенным клиентам

protected internal void BroadcastMessage(string message, string id)

{

byte[] data = Encoding.Unicode.GetBytes(message);

for (int i = 0; i < clients.Count; i++)

{

if (clients[i].Id != id) // если id клиента не равно id отправляющего

{

//передача данных

clients[i].SslStream.Write(data, 0, data.Length);

}

}

}

// отключение всех клиентов

protected internal void Disconnect()

{

tcpListener.Stop(); //остановка сервера

for (int i = 0; i < clients.Count; i++)

{

clients[i].Close(); //отключение клиента

}

Environment.Exit(0); //завершение процесса

}

}

}

- ClientObject:

using System;

using System.Collections.Generic;

using System.Text;

using System.Net.Sockets;

using System.Net.Security;

namespace ChatServer

{

public class ClientObject

{

protected internal string Id { get; private set; }

protected internal NetworkStream SslStream { get; private set; }

string userName;

TcpClient client;

ServerObject server; // объект сервера

public ClientObject(TcpClient tcpClient, ServerObject serverObject)

{

Id = Guid.NewGuid().ToString();

client = tcpClient;

server = serverObject;

serverObject.AddConnection(this);

}

public void Process()

{

try

{

SslStream = client.GetStream();

// получаем имя пользователя

string message = GetMessage();

userName = message;

message = userName + " вошел в чат";

// посылаем сообщение о входе в чат всем подключенным пользователям

server.BroadcastMessage(message, this.Id);

Console.WriteLine(message);

// в бесконечном цикле получаем сообщения от клиента

while (true)

{

try

{

message = GetMessage();

message = String.Format("{0}: {1}", userName, message);

Console.WriteLine(message);

server.BroadcastMessage(message, this.Id);

}

catch

{

message = String.Format("{0}: покинул чат", userName);

Console.WriteLine(message);

server.BroadcastMessage(message, this.Id);

break;

}

}

}

catch (Exception e)

{

Console.WriteLine(e.Message);

}

finally

{

// в случае выхода из цикла закрываем ресурсы

server.RemoveConnection(this.Id);

Close();

}

}

// чтение входящего сообщения и преобразование в строку

private string GetMessage()

{

byte[] data = new byte[64]; // буфер для получаемых данных

StringBuilder builder = new StringBuilder();

int bytes = 0;

do

{

bytes = SslStream.Read(data, 0, data.Length);

builder.Append(Encoding.Unicode.GetString(data, 0, bytes));

}

while (SslStream.DataAvailable);

return builder.ToString();

}

// закрытие подключения

protected internal void Close()

{

if (SslStream != null)

SslStream.Close();

if (client != null)

client.Close();

}

}

}

- Program:

using System;

using System.Threading;

namespace ChatServer

{

class Program

{

static ServerObject server; // сервер

        static Thread listenThread; // потока для прослушивания

        static void Main(string[] args)

{

try

{

server = new ServerObject();

listenThread = new Thread(new ThreadStart(server.Listen));

listenThread.Start(); //старт потока

            }

catch (Exception ex)

{

server.Disconnect();

Console.WriteLine(ex.Message);

}

}

}

}