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 курсовому проекту на тему

«Социальный фоторедактор»

БГУИР КП 1–40 01 01 23 ПЗ

Выполнил

студент: гр. 551004 Евтушенко Т.С.

Проверил: Красковский П.Н.

Минск 2017

Содержание

[Введение 5](#_Toc482867460)

[1 Обзор аналогов 6](#_Toc482867461)

[2 Постановка задачи 7](#_Toc482867462)

[3 Технологии, методы, структуры, принципы, положенные в основу алгоритма 8](#_Toc482867463)

[3.1 Язык программирования 8](#_Toc482867464)

[3.2 Среда разработки 8](#_Toc482867465)

[3.3 Работа с сетью 9](#_Toc482867466)

[3.4 Создание пользовательского интерфейса 10](#_Toc482867467)

[4 Разработка сетевого взаимодействия 12](#_Toc482867468)

[5 Разработка программного средства 16](#_Toc482867469)

[5.1 Составные части приложения «Сервер» 16](#_Toc482867470)

[5.2 Составные части приложения «Клиент» 20](#_Toc482867471)

[5.3 Составные части динамически подключаемой библиотеки 24](#_Toc482867472)

[6 Тестирование программного средства 26](#_Toc482867473)

[7 Руководство пользователя 30](#_Toc482867474)

[7.1 Работа с приложением «Cервер» 30](#_Toc482867475)

[7.2 Работа с приложением «Клиент» 32](#_Toc482867476)

[Заключение 35](#_Toc482867477)

[Список использованных источников 36](#_Toc482867478)

[Приложение А 37](#_Toc482867479)

Введение

В современном мире уже трудно представить свою жизнь без всемирной паутины, хотя всего 15-20 лет назад никто не мог и предположить, что жизнь наша окажется завязана на Интернете, о котором на то время никто ничего и не знал. Очень быстро Интернет-сфера распространилась из сугубо специального использования государственными службами на все население планеты, и теперь представить без нее свое полноценное существование просто невозможно. Сейчас по почти любому вопросу человек может обратиться к всемирной паутине. [1]

Интернет является отличным инструментом для различного рода коммуникаций между людьми. Для кого-то он служит просто средством общения между родными и близкими людьми, находящимися вдали друг от друга. Другие же используют Интернет для связи с коллегами, начальством, заказичками. Третии используют интернет в развлекательных целях. В любом случае, благодаря интернету коммуникации между людьми сегодня достигли высочайшего уровня разнообразия и технологичности.

Огромную популярность среди пользователей Интернета завоевали различного рода социальные сети. Одни подходят для личного обмена сообщениями, например, FaceBook, в других можно с легкостью совершать видеозвонки – Skype. Интересными мыслями можно делиться с друзьями в Twitter, а для ведения своего блога идеально подойдёт LiveJournal. Однако особенную популярность в последнее время приобрели социальные сети, в которых главной функциональностью является возможность делиться удачными фотографиями с пользователями. Зачастую такие сервисы имеют базовые возможности фоторедактора, что также сказывается на их популярности. Пользователи интернета, посредством публикации фотографий в таких социальных сетях, охотно делятся событиями из своей жизни, приятными моментами и любыми своими впечатлениями.

# Обзор аналогов

В сети Интернет можно найти большое количество сайтов и приложений, реализующих функции социальных сетей для обмена фотографиями. Рассмотрим некоторые из них.

## Instagram

Это могут быть целые сайты, предоставляющие возможность общения в реальном времени. Такого рода чаты обычно имеют так называемые «комнаты». То есть пользователи разделяются на группы в зависимости от темы разговора. Вход в комнату может быть свободным или ограниченным её создателем. При общении в таком чате все видят сообщения друг друга. Также веб-чаты могут быть частью какого-либо сайта. Например, чат для общения клиента с технической поддержкой или с менеджером по продажам. Подобные мини-чаты могут быть как групповыми, так и индивидуальными [2].

1.2

Не имеют возможности работать в глобальной сети и не требуют выделения сервера. Каждая запущенная копия такой программы одновременно является и сервером, и клиентом. Примером таких приложений могут служить Vypress Chat, Intranet Chat [2].

* **Чаты, реализующие протокол IRC**

IRC (Internet Relay Chat) – протокол прикладного уровня для обмена сообщениями в режиме реального времени. Разработан в основном для группового общения, также позволяет общаться через личные сообщения и обмениваться файлами. IRC использует транспортный протокол TCP и криптографический TLS (опционально). Примеры IRC-клиентов: mIRC, Miranda IM, Trillian, XChat, Pidgin [3].

* **Чаты, работающие в одноранговых сетях**

Одноранговая сеть – это компьютерная сеть, основанная на равноправии участников. Взаимодействие клиентов в такой сети идёт напрямую, без выделения сервера. Сети такого вида могут сохранять работоспособность при любом количестве и сочетании составляющих её узлов. Пример такого чата: Tox. Не требует регистрации для использования, идетификатор пользователя создаётся локально. Для пользователя создаётся пара ключей: открытый и секретный. Открытый можно передавать кому угодно: он служит для поиска собеседников. Секретный ключ хранится у пользователя и служит для подтверждения его личности без раскрытия персональных данных [4].

# Постановка задачи

Назначение программного средства заключается в предоставлении возможности обмена сообщениями групы пользователей в локальной сети. Для сетевого взаимодействия используется протокол TCP и архитекутура «клиент-сервер».

Серверная часть предоставляет следующие возможности:

* возможность указания IP-адреса и номера порта, на которых будет работать сервер;
* возможность подключить и обслуживать до N пользователей. N указывается при запуске сервера;
* перед подключением нового пользователя проверка наличия свободных мест, коллизии имён уже подключённых пользователей с именем нового пользователя;
* поддержка групповых сообщений;
* поддержка личных сообщений;
* поддержка «чёрных списков» для каждого пользователя;
* возможность отключить/включить у заданного пользователя функцию отправки групповых сообщений;
* возможность отключения заданного пользователя;
* возможность отключения всех пользователей;
* возможность временно закрыть сервер для новых клиентов;
* обеспечение возможности отображения сообщений различными цветами в зависимости от отправителя;
* возможность самостоятельного выбора цветов, которые будут использованы для отображения сообщений;
* проверка всех вводимых пользователем данных на корректность.

Клиентская часть предоставляет следующие возможности:

* возможность ввода имени пользователя и подключения к серверу с заданными IP-адресом и номером порта;
* возможнось отправки заданного сообщения всем подключённым к серверу пользователям;
* возможность отправки личного сообщения указанному пользователю;
* возможность добавления указанного пользователя в «чёрный список»;
* возможность удаления указанного пользователя из «чёрного списка»;
* поддержка в сообщениях смайликов;
* проверка всех вводимых пользователем данных на корректность.

# Технологии, методы, структуры, принципы, положенные в основу алгоритма

## Язык программирования

При разаработке программного средства использовался язык программирования C#.

C# – [объектно-ориентированный](https://ru.wikipedia.org/wiki/%D0%9E%D0%B1%D1%8A%D0%B5%D0%BA%D1%82%D0%BD%D0%BE-%D0%BE%D1%80%D0%B8%D0%B5%D0%BD%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%BD%D0%BE%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5) [язык программирования](https://ru.wikipedia.org/wiki/%D0%AF%D0%B7%D1%8B%D0%BA_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F). Разработан в [1998](https://ru.wikipedia.org/wiki/1998) – [2001 годах](https://ru.wikipedia.org/wiki/2001_%D0%B3%D0%BE%D0%B4) группой инженеров под руководством [Андерса Хейлсберга](https://ru.wikipedia.org/wiki/%D0%A5%D0%B5%D0%B9%D0%BB%D1%81%D0%B1%D0%B5%D1%80%D0%B3,_%D0%90%D0%BD%D0%B4%D0%B5%D1%80%D1%81) в компании [Microsoft](https://ru.wikipedia.org/wiki/Microsoft) как язык разработки приложений для платформы [Microsoft .NET Framework](https://ru.wikipedia.org/wiki/.NET_Framework). C# относится к семье языков с [C-подобным синтаксисом](https://ru.wikipedia.org/wiki/C-%D0%BF%D0%BE%D0%B4%D0%BE%D0%B1%D0%BD%D1%8B%D0%B9_%D1%81%D0%B8%D0%BD%D1%82%D0%B0%D0%BA%D1%81%D0%B8%D1%81). Из них его синтаксис наиболее близок к [C++](https://ru.wikipedia.org/wiki/C%2B%2B) и [Java](https://ru.wikipedia.org/wiki/Java). Язык имеет [статическую типизацию](https://ru.wikipedia.org/wiki/%D0%A1%D1%82%D0%B0%D1%82%D0%B8%D1%87%D0%B5%D1%81%D0%BA%D0%B0%D1%8F_%D1%82%D0%B8%D0%BF%D0%B8%D0%B7%D0%B0%D1%86%D0%B8%D1%8F), поддерживает [полиморфизм](https://ru.wikipedia.org/wiki/%D0%9F%D0%BE%D0%BB%D0%B8%D0%BC%D0%BE%D1%80%D1%84%D0%B8%D0%B7%D0%BC_(%D0%B8%D0%BD%D1%84%D0%BE%D1%80%D0%BC%D0%B0%D1%82%D0%B8%D0%BA%D0%B0)), [перегрузку операторов](https://ru.wikipedia.org/wiki/%D0%9F%D0%B5%D1%80%D0%B5%D0%B3%D1%80%D1%83%D0%B7%D0%BA%D0%B0_%D0%BE%D0%BF%D0%B5%D1%80%D0%B0%D1%82%D0%BE%D1%80%D0%BE%D0%B2), [делегаты](https://ru.wikipedia.org/wiki/%D0%94%D0%B5%D0%BB%D0%B5%D0%B3%D0%B0%D1%82_(%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5)), атрибуты, [события](https://ru.wikipedia.org/wiki/%D0%A1%D0%BE%D0%B1%D1%8B%D1%82%D0%B8%D0%B9%D0%BD%D0%BE-%D0%BE%D1%80%D0%B8%D0%B5%D0%BD%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%BD%D0%BE%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5), [свойства](https://ru.wikipedia.org/wiki/%D0%A1%D0%B2%D0%BE%D0%B9%D1%81%D1%82%D0%B2%D0%BE_(%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5)), [обобщённые](https://ru.wikipedia.org/wiki/%D0%9E%D0%B1%D0%BE%D0%B1%D1%89%D1%91%D0%BD%D0%BD%D0%BE%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5) типы и методы, [итераторы](https://ru.wikipedia.org/wiki/%D0%98%D1%82%D0%B5%D1%80%D0%B0%D1%82%D0%BE%D1%80_(%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5)#.D0.93.D0.B5.D0.BD.D0.B5.D1.80.D0.B0.D1.82.D0.BE.D1.80.D1.8B), [анонимные функции](https://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%BE%D0%BD%D0%B8%D0%BC%D0%BD%D0%B0%D1%8F_%D1%84%D1%83%D0%BD%D0%BA%D1%86%D0%B8%D1%8F).

Переняв многое от своих предшественников – языков C++, Pascal и, в особенности, Java – C#, опираясь на практику их использования, исключает некоторые модели, зарекомендовавшие себя как проблематичные при разаработке программных средств. Например, C# не поддерживает, в отличие от C++, множественное наследование классов.

Результатом компиляции программы на C# является не исполняемый код, а файл, содержащий особого рода псевдокод, называемый *Microsoft Intermediate Language (MSIL).* Данный код определяет набор переносимых инструкций, не зависящий от конкретного процессора. Псевдокод MSIL преобразуется в исполняемый код с помощью JIT-компилятора (Just-in-time).

Помимо псевдокода MSIL, при компиляции программы на C# получаются также метаданные, которые служат для описания данных, используемых в программе, а также обеспечивают простое взаимодействие одного кода с другим. Метаданные содержатся в том же файле, что и псевдокод MSIL [1].

На сегодняшний день язык программирования C# является одним из самых мощных, быстро развивающихся и востребованных языков в ИТ-отрасли. Сегодня на нём пишутся самые различные приложения: от небольших десктопных программ до крупных веб-порталов и сервисов, обслуживающих ежедневно миллионы пользователей.

## Среда разработки

Для создания программного средства была использована интегрированная среда разработки Microsoft Visual Studio.

Интегрированная среда разработки (англ. IDE – Integrated Development Invironment) – комплекс программных средств, используемый для разработки программного обеспечения (ПО).

Microsoft Visual Studio – набор инструментов для создания ПО: от планирования до разработки пользовательского интерфейса, написания кода, тестирования, отладки, анализа качества кода и производительности. Visual Studio можно использовать для создания различных типов приложений: от простых приложений для магазина и игр для мобильных устройств до больших и сложных систем, обслуживающих предприятия и центры обработки данных. Основные виды программных средств, которые можно создавать с использованием Visual Studio:

* приложения и игры, которые выполняются не только на платформе Windows, но и на Android и iOS;
* веб-сайты и веб-службы на основе ASP.NET, JQuery, AngularJS и других популярных платформ;
* приложения для самых разных устройств и платформ, включая

Office, Sharepoint, Hololens, Kinect, Internet of Things;

* игры и графические приложения для разных устройств Windows, включая Xbox, с поддержкой DirectX.

По умолчанию Visual Studio обеспечивает поддержку C#, C, C++, JavaScript, F#, Visual Basic. Хорошо интегрируется и работает со сторонними приложениями, например, Unity и Apache Cordova. Имеется также возможность самостоятельно расширять Visual Studio путём создания плагинов для выполнения специализированных задач [5].

## Работа с сетью

В основе сетевого взаимодействия лежит понятие сетевого гнезда (англ. socket; русск. сокет) — это пара IP-адреса (логический адрес устройства в сети) и номера порта (16-битное число в диапазоне от 1 до 65535, позволяющее разграничить несколько приложений на одном устройстве).

На платформе .NET сокеты представлены классом Socket из пространства имён System.NET.Sockets. При работе с сокетами наиболее часто используются следующие методы класса Socket:

* Accept : извлекает из очереди входящее подключение и создаёт для него объект класса Socket. Если подключений нет, блокирует вызывающий поток до появления входящих подключений;
* Bind : связывает объект класса Soсket с IP-адресом и портом;
* Listen : переводит сокет в режим прослушивания входящих подключений;
* Connect : устанавливает соединение с удалённым хостом по его IP-адресу и номеру порта;
* Receive/ReceiveFrom : получение данных;
* Send/SendTo : отправление данных;
* Shutdown : блокирует приём и/или передачу данных;
* Close : закрывает сокет.

В зависимости от применяемого протокола передачи данных общий принцип работы с сокетами будет немного различаться.

При использовании протокола, который требует установления соединения, например, TCP, сервер должен вызвать метод Bind для установки точки прослушивания входящих подключений, а затем перевести сокет в режим прослушивания методом Listen. Далее, с помощью метода Accept можно получить входящие запросы на подключение в виде объекта класса Socket, который используется для взаимодействия с удалённым узлом. У полученного объекта класса Socket вызываются методы Receive и Send соответственно для получения и отправки данных. Если необходимо подключиться к серверу, то вызывается метод Connect. Для обмена данными с сервером также применяются методы Receive и Send [6].

Если применятеся протокол, для которого не требуется установка соединения, например, UDP, то после вызова метода Bind вызывать метод

Listen не нужно. Для приёма и отправки сообщений используются, соответственно, методы ReceiveFrom и SendTo [6].

## Создание пользовательского интерфейса

Для разработки графического пользовательского интерфеса используется технология WPF (Windows Presentation Foundation), которая является частью платформы .NET.

Если при создании традиционных приложений на основе WinForms за отрисовку элементов управления и графики отвечали такие части ОС

Windows, как User32 и GDI+, то приложения WPF основаны на DirectX. В этом и состоит ключевая особенность рендеринга графики в WPF: значительная часть работы по отрисовке как простых кнопок, так и сложных 3D-моделей ложится на графический процессор видеокарты, что позволяет использовать аппаратное ускорение графики.

Ещё одной важной особенностью WPF является использование языка декларативной разметки интерфейса XAML, основанного на XML. Можно создавать насыщенный графический интерфейс или с использованием XAML, или в коде на языке C#, или совмещать сразу оба способа.

Главными преимуществами WPF по сравнению с другими библиотеками создания графического пользовательского интерфейса являются:

* использование традиционных языков платформы .NET, например, C#, для написания логики приложения;
* возможность декларативного определения графического интерфейса с помощью специального языка разметки XAML;
* независимость от разрешения экрана: все элементы в WPF измеряются в независимых от устройства единицах, поэтому приложения WPF легко масштабируются под разные разрешения экранов;
* хорошее взаимодействие с WinForms, благодаря чему, например, в приложениях WPF можно использовать традиционные элементы управления из WinForms;
* богатые возможности по созданию различных приложений: это и мультимедиа, и двухмерная и трехмерная графика, и богатый набор встроенных элементов управления, а также возможность самим создавать новые элементы, создание анимаций, привязка данных, стили, шаблоны, темы и многое другое;
* аппаратное ускорение графики;
* создание приложений под множество ОС семейства Windows.

В тоже время WPF имеет определенные ограничения. Несмотря на поддержку трёхмерной визуализации, для создания приложений с большим количеством трёхмерных изображений, прежде всего игр, лучше использовать другие средства, например, DirectX или специальные фреймворки, такие как Monogame или Unity.

Также стоит учитывать, что приложения WPF потребляют больше системных ресурсов в процессе работы чем, например, приложения на Windows Forms. Однако это компенсируется более широкими графическими возможностями и повышенной производительностью при отрисовке графики [7].

# Разработка сетевого взаимодействия

Для реализации всех вышеописанных функций для разрабатываемого программного средства была выбрана клиент-серверная архитектура и разработан протокол передачи информации между клиентом и сервером. В качестве транспортного протокола используется TCP (Transmission Control

Protocol).

Передача данных осуществляется лишь между клиентом и сервером, прямое взаимодействие клиентов отсутствует. Все передаваемые данные соответствуют определённому формату:

<флаг\_сообщения>[<дополнительные\_данные>]

*Флаг сообщения*: 8-битное беззнаковое целое число, по которому определяется смысл и содержимое *сообщения.*

*Дополнительные данные* (могут отсутствовать): информация, необходимая для корректной обработки запроса от клиента или реакции на ответ от сервера.

Единственное сообщение, которое нарушает этот формат, это подключение клиента к серверу: клиент высылает серверу своё имя, после чего сервер либо подтверждает подключение, либо отклоняет его, посылая соответствующий формату ответ. Имя клиента используется только для отображения сообщений пользователю, при сетевом взаимодействии клиенты именуются при помощи уникальных идентификаторов (ID). Имена клиентов передаются по сети только при подключении нового клиента: всем уже подключённым клиентам высылается имя и ID нового клиента, новому клиенту – имена и ID всех ранее подключённых клиентов.

ID-клиента – 8-битное знаковое целое число, которое выдаётся клиенту при подключении к серверу. Клиентам выдаются неотрицательные идентификаторы, а отрицательные используются для информационных сообщений от сервера.

Каждому идентификатору соответствует свой цвет. Это обеспечивает возможность отображения сообщений разными цветами в зависимости от отправителя. Таблица соответствия формируется на сервере при его запуске и высылается каждому пользователю при подключении.

При взаимодействии клиента и сервера используются следующие флаги сообщений:

* ***PublicMessage***

8-битное значение: 0.

Формат дополнительных данных при отправке от клиента к серверу:

<сообщение>

Формат дополнительных данных при отправке от сервера к клиенту:

<ID\_отправителя><сообщение>

Описание: при получении данного сообщения от клиента, сервер должен отправить *сообщение* с флагом *PublicMessage* всем подключённым клиентам, кроме того, от которого было получено данное сообщение. При получении данного сообщения от сервера, клиент должен отобразить *сообщение* как групповое и указать в качестве отправителя имя пользователя, которое соответствует *ID\_отправителя*.

* ***PrivateMessage***

8-битное значение: 1.

Формат дополнительных данных при отправке от клиента к серверу:

<ID\_получателя><сообщение>

Формат дополнительных данных при отправке от сервера к клиенту:

<ID\_отправителя><сообщение>

Описание: при получении данного сообщения от клиента, сервер должен отправить *сообщение* с флагом *PrivateMessage* пользователю с идентификатором *ID\_получателя*. При получении данного сообщения от сервера, клиент должен отобразить *сообщение* как личное от пользователя с идентификатором *ID\_отправителя*.

* ***ConnectionConfirmed***

8-битное значение: 2.

Формат дополнительных данных при отправке от сервера к клиенту:

<ID><N><R\_1><G\_1><B\_1>…<R\_N><G\_N><B\_N><M><ID\_1><имя\_1>…

<ID\_M><имя\_M>

Описание: высылается сервером новому пользователю для подтверждения его подключения. *ID* – идентификатор, выданный данному пользователю. *N* (число, 32 бита, знаковое) – количество цветов в таблице, которая задаёт соответствие идентификаторов пользователей и цветов, которыми будут отображаться их сообщения. *R\_i* (число, 8 бит, беззнаковое) – красный компонент i-го цвета. *G\_i* (число, 8 бит, беззнаковое) – зелёный компонент i-го цвета. *B\_i* (число, 8 бит, беззнаковое) – синий компонент i-го цвета.

*M* (число, 8 бит, беззнаковое) – количество уже подключённых к серверу клиентов. *ID\_i* – идентификатор i-го пользователя. *Имя\_i* – имя i-го пользователя.

* ***NewUser***

8-битное значение: 3.

Формат дополнительных данных при отправке от сервера к клиенту:

<имя\_пользователя><ID\_пользователя>

Описание: при подключении нового клиента сервер высылает всем уже подключённым пользователям идентификатор и имя только что подключившегося пользователя.

* ***UserLeave***

8-битное значение: 4.

Формат дополнительных данных при отправке от сервера к клиенту:

<ID\_пользователя>

Описание: при отключении какого-либо клиента сервер высылает всем оставшимся пользователям идентификатор только что отключившегося пользователя.

* ***ServerOverloaded***

8-битное значение: 5.

Дополнительных данных нет.

Описание: ошибка подключения. Отправляется клиенту, если сервер не может подтвердить подключение из-за того, что на сервере достигнуто максимальное количество одновременно работающих клиентов.

* ***NameConflict***

8-битное значение: 6.

Дополнительных данных нет.

Описание: ошибка подключения. Отправляется клиенту, если сервер не может подтвердить подключение из-за того, что на сервере уже есть пользователь с именем, которое указал новый пользователь.

* ***ServerClosed***

8-битное значение: 7.

Дополнительных данных нет.

Описание: ошибка подключения. Отправляется клиенту, если сервер не может подтвердить подключение из-за того, что сервер временно закрыт для новых клиентов.

* ***ConnectionClosed***

8-битное значение: 8.

Дополнительных данных нет.

Описание: отправляется от сервера клиенту. Указание клиенту того, что он отключён от сервера.

* ***ClientDisconnected***

8-битное значение: 9.

Дополнительных данных нет.

Описание: отправляется от клиента к серверу. Указание серверу того, что клиент отключился от сервера.

* ***ClientBanned***

8-битное значение: 10.

Дополнительных данных нет.

Описание: отправляется сервером клиенту. Указывает на то, что клиенту заблокирована возможность отправки групповых сообщений. При отправке групповых сообщений клиент самостоятельно проверяет, был ли он ранее заблокирован сервером.

* ***ClientАctivated***

8-битное значение: 11.

Дополнительных данных нет.

Описание: отправляется сервером клиенту. Указывает на то, что клиенту разаблокирована возможность отправки групповых сообщений.

* ***BanUser***

8-битное значение: 12.

Формат дополнительных данных при отправке от клиента к серверу:

<ID\_пользователя>

Описание: запрос от клиента серверу на внесение пользователя с идентификатором *ID\_пользователя* в свой «чёрный список». Пользователю с идентификатором *ID\_пользователя* сервер отправляет сообщение с флагом *BannedByUser*.

* ***ActivateUser***

8-битное значение: 13.

Формат дополнительных данных при отправке от клиента к серверу:

<ID\_пользователя>

Описание: запрос от клиента серверу на удаление пользователя с идентификатором *ID\_пользователя* из «чёрного списка» пользователя, от которого было получено данное сообщение. Пользователю с идентификатором *ID\_пользователя* сервер отправляет сообщение с флагом *ActivatedByUser*.

* ***BannedByUser***

8-битное значение: 14.

Формат дополнительных данных при отправке от сервера клиенту:

<ID\_пользователя>

Описание: отправляется сервером клиенту. Указывает на то, что данный клиент был добавлен в «чёрный список» пользователя с идентификатором *ID\_пользователя*. При отправке личных сообщений клиент самостоятельно проверяет, был ли он заблокирован получателем сообщения.

* ***ActivatedByUser***

8-битное значение: 15;

Формат дополнительных данных при отправке от сервера клиенту:

<ID\_пользователя>

Описание: отправляется сервером клиенту. Указывает на то, что данный клиент был удалён из «чёрного списка» пользователя с идентификатором *ID\_пользователя*.

# Разработка программного средства

Программное средство состоит из трёх основных частей:

* приложение, реализующее работу сервера;
* приложение, реализующее работу клиента;
* динамически подключаемая библиотека, используемая как сервером, так и клиентом в процессе своей работы.

## Составные части приложения «Сервер»

* **Класс CServer**

Данный класс реализует управление клиентами: прослушивает входящие соединения, подключает при наличии возможности клиентов, выдаёт каждому клиенту свой идентификатор, осуществляет отправку клиентам необходимых сообщений.

Таблица 1 – Методы класса CServer

|  |  |  |  |
| --- | --- | --- | --- |
| №  п.п. | Наименование  метода | Назначение  метода | Формальные  Параметры |
| 1 | Start | Запускает прослушивание входящих подлкючений в отдельном потоке | Отсутствуют |
| 2 | ListenIncoming  Connections | Переводит сокет сервера в режим прослушивания и начинает ожидание входящих подключений. При появлении нового соединения получает для него объект класса Socket и вызывает метод для подключения нового клиента. Выполняется в отдельном потоке | Отсутствуют |
| 3 | TryToConnectNew  Client | Для полученного соединения создаёт объект класса CClient, у которого получает имя нового пользователя и проверяет возможность подтверждения данного соединения. | newClientSocket (объект класса Socket для соединения нового клиента) |

Продолжение таблицы 1

|  |  |  |  |
| --- | --- | --- | --- |
| №  п.п. | Наименование  метода | Назначение  метода | Формальные  Параметры |
|  |  | Если такая возможность есть, подтверждает подключение и добавлеят клиента в список подключённых, иначе высылает клиенту сообщение с ошибкой подключения |  |
| 4 | IsPossibleTo  ConnectClient | Проверяет возможность подключения клиента newClient: открыт ли сервер для новых клиентов, есть ли места на сервере, нет ли пользователя на сервере с таким же именем, как у нового клиента | newClient (объект нового клиента);  errorType (возвращаемое значение: флаг сообщения, который нужно будет отправить клиенту после завершнеия проверки);  result (возвращаемое значение: подтверждено ли подключение) |
| 5 | IsNameConflict | Проверяет, имеется ли на сервере пользователь с заданным именем. Если да, то возвращается true, если нет – false | name (имя пользователя, которое нужно проверить);  result (возвращаемое значение: было ли найдено заданное имя среди пользователей) |
| 6 | SendClientsList | Высылает клиенту  receiver список всех пользователей, подключённых к серверу | receiver (клиент-получатель) |
| 7 | SendPalette | Высылает клиенту  receiver таблицу цветов | receiver (клиент-получатель) |
| 8 | DisconnectClient | Отключает клиента с заданным идентификатором от сервера и уведомляет об этом всех оставшихся клиентов | clientID (идентификатор клиента) |

Продолжение таблицы 1

|  |  |  |  |
| --- | --- | --- | --- |
| №  п.п. | Наименование  метода | Назначение  метода | Формальные  Параметры |
| 9 | RemoveAllConnections | Отключает всех клиентов от сервера. Если булевый флаг равен true, то при этом выводятся соответствующие сообщения на экран | isNeedMessage (булевый флаг) |
| 10 | IsClientBanned | Если клиент с заданным идентификатором заблокирован сервером, то возвращается true, иначе – false | ID (идентификатор пользователя);  result (возвращаемое значение) |
| 11 | BanClient | Блокирует у заданного клиента возможность отправления групповых сообщений | clientID (идентификатор пользователя) |
| 12 | ActivateClient | Разблокирует у заданного клиента возможность отправления групповых сообщений | clientID (идентификатор пользователя) |
| 13 | RemoveConnection | Убирает заданного клиента из списка подключённых | ID (идентификатор пользователя) |
| 14 | GetClientIndByID | Возвращает индекс заданного пользователя из списка подключённых к серверу | ID (идентификатор пользователя) |
| 15 | Stop | Останавливает работу сервера | Отсутствуют |
| 16 | SendPublicMessage | Отсылает заданное сообщения от заданного пользователя всем клиентам, кроме отправителя | message (сообщение);  senderID (идентификатор отправителя) |
| 17 | SendPrivateMessage | Отсылает заданное сообщения от указанного пользователя заданному получателю | message (сообщение);  senderID (идентификатор отправителя);  receiverID (идентификатор получателя) |

Окончание таблицы 1

|  |  |  |  |
| --- | --- | --- | --- |
| №  п.п. | Наименование  метода | Назначение  метода | Формальные  Параметры |
| 18 | BanUserByAnotherUser | Отправляет пользователю с идентификатором receiverID уведомление о том, что он был заблокирован пользователем с идентификатором senderID | senderID (идентификатор отправителя);  receiverID (идентификатор получателя) |
| 19 | ActivateUserBy  AnotherUser | Отправляет пользователю с идентификатором receiverID уведомление о том, что он был разблокирован пользователем с идентификатором senderID | senderID (идентификатор отправителя);  receiverID (идентификатор получателя) |
| 20 | NotifyAllClientsAbout  NewUser | Уведомляет всех клиентов о том, что к серверу подключился пользователь с именем username и идентификатором userID | username (имя пользователя);  userID (идентификатор пользователя) |
| 21 | NotifyAllClientsAbout  UserLeft | Уведомляет всех клиентов о том, что пользователь с именем username и идентификатором userID отключён от сервера | username (имя пользователя);  userID (идентификатор пользователя) |
| 22 | CloseForIncoming  Connections | Закрывает сервер для подключения новых клиентов | Отсутсвуют |
| 23 | OpenForIncoming  Connections | Открывает сервер для подключения новых клиентов | Отсутсвуют |

* **Класс CClient**

Данный класс реализует взаимодействие сервера с конкретным клиентом: обрабатывает его запросы, высылает сообщения, сгенерированные сервером.

* **Класс CIDManager**

Данный класс реализует управление идентификаторами клиентов: при необходимости выделяет пользователю случайный идентификатор, а после отключения клиента освобождает его.

* **Класс MainWindow**

Данный класс реализует работу графического интерфейса приложения.

* **Класс CView**

Данный класс реализует взаимодействие всех классов программы с пользовательским интерфейсом (класс MainWindow). Если происходит попытка обращения к пользовательскому интерфейсу из другого потока, то вызывается метод Invoke у объекта Dispatcher для выполнения этого действия в нужном потоке.

## Составные части приложения «Клиент»

* **Класс CUser**

Данный класс реализует взаимодействие клиента с сервером: первоначально производит попытку подключения к указанному серверу (рисунок 5.1) отправляет нужные запросы на сервер в зависимости от действий пользователя, обрабатывает ответы сервера и выводит на экран соответсвующие результаты.

Таблица 2 – Методы класса CUser

|  |  |  |  |
| --- | --- | --- | --- |
| №  п.п. | Наименование  метода | Назначение  метода | Формальные  Параметры |
| 1 | ConnectToServer | Производит попытку подключения к заданному серверу. Если сервер подтвердил подключение, то запускает в отдельном потоке прослушивание входящих сообщений от сервера | Отсутсвуют |
| 2 | IsConnectionConfirmed | Ожидает ответа от сервера на запрос подключения. Если сервер подтвердил подключение, возвращает true, иначе – false | result (возвращаемое значение) |
| 3 | DisconnectFromServer | Отключает клиента от сервера: отправляет серверу сообщение с флагом *Client*  *Disconnected* и закрывает с ним соединение | Отсутствуют |

Продолжение таблицы 2

|  |  |  |  |
| --- | --- | --- | --- |
| №  п.п. | Наименование  метода | Назначение  метода | Формальные  Параметры |
| 4 | CloseConnection | Закрывает физическое соединения с сервером | Отсутствуют |
| 5 | AnalyseMessageType | Анализирует флаг сообщения и в зависимости от него вызывает соответствующий метод-обработчик | curType (флаг полученного сообщения) |
| 6 | GetInitialDataFrom  Server | Получает от сервера всю необходимую для работы информацию: таблицу цветов и список подключённых к серверу клиентов | Отсутствуют |
| 7 | ReceiveMessages | Выполняет прослушивание входящих сообщений от сервера. При поступлении сообщения считывает его флаг и вызывает метод для анализа сообщения. Выполняется в отдельном потоке | Отсутствуют |
| 8 | SendPublicMessage | Если пользователь не заблокирован сервером, то отправляет заданное сообщение как групповое | message (сообщение) |
| 9 | SendPrivateMessage | Получает текущего собеседника и отправляет ему заданное сообщение как личное, если это возможно | message (сообщение) |
| 10 | BanUser | Отправляет серверу запрос на внесение пользователя с заданным идентификатором в «чёрный список» | ID (идентификатор пользователя) |
| 11 | ActivateUser | Отправляет серверу запрос на удаление пользователя с | ID (идентификатор пользователя) |

Продолжение таблицы 2

|  |  |  |  |
| --- | --- | --- | --- |
| №  п.п. | Наименование  метода | Назначение  метода | Формальные  Параметры |
|  |  | заданным идентификатором из «чёрного списка» |  |
| 12 | IsUserBannedFor  CurrentDialog | Если пользователь был заблокирован текущим собеседником, то возвращает true, иначе – false | result (возвращаемое значение) |
| 13 | ReadMessageFromUser | Читает из потока данных идентификатор отправителя и текст сообщения | ID (возвращаемое значение: идентификатор отправителя);  message (возвращаемое значение: текстовое сообщение) |
| 14 | ReceivePrivateMessage | Обработчик сообщения от сервера. Считывает из потока данных сообщение и отображает его как личное сообщение от заданного отправителя | Отсутствуют |
| 15 | ReceivePublicMessage | Обработчик сообщения от сервера. Считывает из потока данных сообщение и отображает его как групповое сообщение от заданного отправителя | Отсутствуют |
| 16 | BannedByAnotherUser | Обработчик сообщения от сервера. Считывает из потока данных идетификатор отправителя и помечает себя в качестве заблокированного данным пользователем | Отсутствуют |
| 17 | ActivatedBy  AnotherUser | Обработчик сообщения от сервера. Считывает из потока данных идетификатор | Отсутствуют |

Окончание таблицы 2

|  |  |  |  |
| --- | --- | --- | --- |
| №  п.п. | Наименование  метода | Назначение  метода | Формальные  Параметры |
|  |  | отправителя и помечает себя в качестве разблокированного данным пользователем |  |
| 18 | NewUserConnectedTo  Server | Обработчик сообщения от сервера. Считывает из потока данных имя и идентификатор нового пользователя и добавляет его в список контактов | Отсутствуют |
| 19 | SomeUserLeftServer | Обработчик сообщения от сервера. Считывает из потока данных идентификатор отключившегося пользователя и удаляет его из списка контактов |  |

* **Класс CUsersManager**

Данный класс осуществляет управление всеми контактами текущего пользовталя. Отслеживает, диалог с каким собеседником сейчас отображается на экране, хранит список клиентов, которые были заблокированы данным пользователем и которыми был заблокирован сам пользователь. Ведёт счётчик непрочитанных сообщений от каждого из клиентов. Осуществляет переключение диалогов на экране.

* **Класс CSmiles**

Данный класс осуществляет работу со смайликами. При загрузке приложения сканируется определённый каталог и из него загружаются смайлики. При получении текстового сообщения от другого пользователя производится парсинг данного сообщения (рисунок 5.2). В результате оно разбивается на отдельные участки, между которыми необходимо вставить изображение определённого смайлика. Вся эта информация передаётся методу вывода сообщения на экран.

* **Класс MainWindow**

Данный класс реализует работу графического интерфейса приложения.

* **Класс CView**

Назначение и принцип работы класса аналогичны одноимённому классу приложения «Сервер»: взаимодействие всех классов программы с пользовательским интерфейсом.

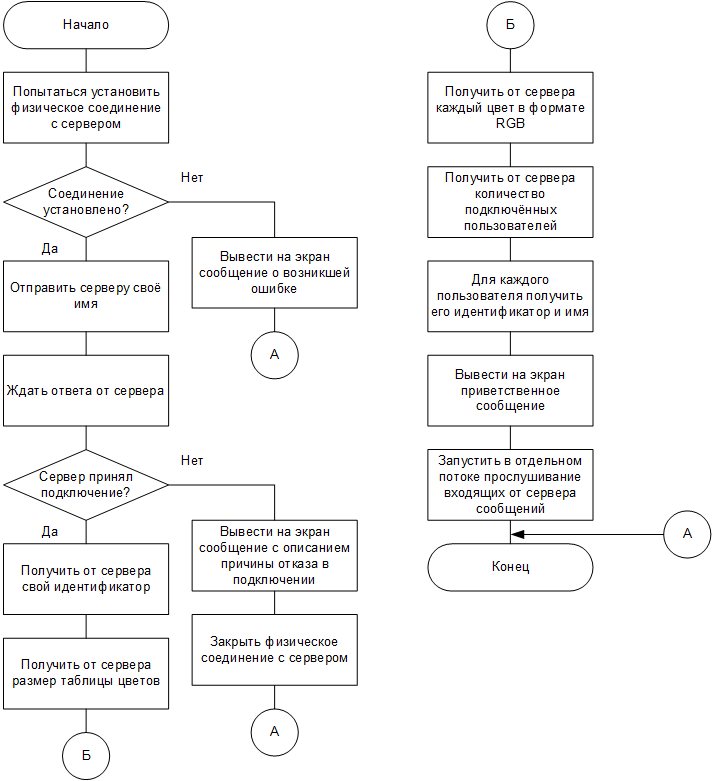


Рисунок 5.1 – Схема алгоритма подключения клиента к серверу

## Составные части динамически подключаемой библиотеки

* **Перечисление MessageType**

Представляет собой все возможные флаги сообщений, которые используются при взаимодействии сервера и клиента. Значения флагов описаны в разделе 4.

* **Класс ConnectionException**

Наследуется от класса System.Exception. Используется для прекращения прослушивания входящих сообщений от сервера/клиента при получении сообщения, после которого необходимо разорвать соединение.

* **Класс CColorsLoader**

Данный класс реализует работу с цветами: хранит таблицу соответствия идентификаторов пользователей и цветов, выполняет преобразование идентификатора в соответствующий ему цвет.

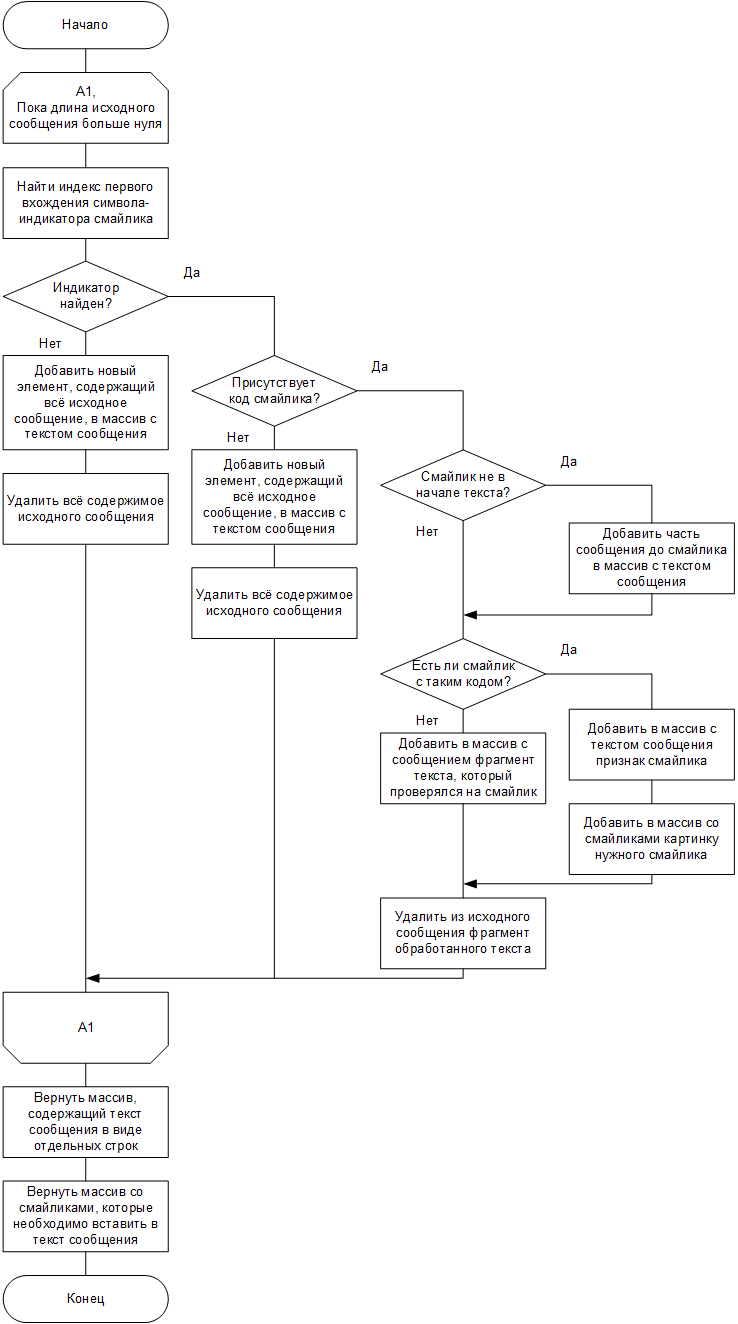


Рисунок 5.2 – Схема алгоритма парсинга сообщения

# Тестирование программного средства

Таблица 3 – Тест 1

|  |  |
| --- | --- |
| Тестовая  ситуация | Проверка возможности запуска сервера с заданными параметрами |
| Исходный набор данных | IP-адрес: 127.0.0.1  Номер порта: 8888  Максимальное число клиентов: 25 |
| Ожидаемый результат | Появление на экране сообщения, свидетельствующего об успешном запуске сервера |
| Фактический результат | Вывод в окне приложения сервера сообщения «Сервер запущен. Ожидание подключений…» |

Таблица 4 – Тест 2

|  |  |
| --- | --- |
| Тестовая  ситуация | Проверка возможности подключения нового пользователя к серверу |
| Исходный набор данных | Запущенный сервер с уже поключённым пользователем Bob |
| Ожидаемый результат | Подлючение к серверу пользователя Alice. Вывод на сервере сообщения о новом пользователе. Вывод у клиента Bob информационного сообщения и добавление пользователя Alice в список контактов. Вывод у пользователя Alice приветственного сообщения |
| Фактический результат | Вывод на сервере и у клиента Bob сообщения «Пользователь <Alice> подключился к серверу». Добавление контакта Alice в список контактов пользователя Bob. Отображения у клиента Alice сообщения «Добро пожаловать, Alice» и отображение в списке контактов пользователя Bob |

Таблица 5 – Тест 3

|  |  |
| --- | --- |
| Тестовая  ситуация | Проверка отказа в подключении новому клиенту |
| Исходный набор данных | Запущенный сервер, который временно закрыт для подключения новых пользователей |
| Ожидаемый результат | Отказ в подключении новому клиенту и вывод у него сообщения с описанеим причины отказа |
| Фактический результат | Вывод у клиента сообщения «Не удалось подключиться к серверу. Сервер закрыт для входящих подключений.» |

Таблица 6 – Тест 4

|  |  |
| --- | --- |
| Тестовая  ситуация | Проверка отказа в подключении новому клиенту |
| Исходный набор данных | Запущенный сервер, на котором достигнуто максимальное количество подключённых пользователей |

Окончание таблицы 6

|  |  |
| --- | --- |
| Ожидаемый результат | Отказ в подключении новому клиенту и вывод у него сообщения с описанеим причины отказа |
| Фактический результат | Вывод у клиента сообщения «Не удалось подключиться к серверу. Сервер перегружен.» |

Таблица 7 – Тест 5

|  |  |
| --- | --- |
| Тестовая  ситуация | Проверка отказа в подключении новому клиенту |
| Исходный набор данных | Запущенный сервер. Среди подключённых клиентов имеется пользователь с именем Bob |
| Ожидаемый результат | Отказ в подключении новому клиенту c именем Bob и вывод у него сообщения с описанеим причины отказа |
| Фактический результат | Вывод у клиента сообщения «Не удалось подключиться к серверу. Пользователь с таким именем уже существует.» |

Таблица 8 – Тест 6

|  |  |
| --- | --- |
| Тестовая  ситуация | Проверка отправки группового сообщения |
| Исходный набор данных | Запущенный сервер с клиентами Alice, Bob, Mark. Пользователь Alice отправляет сообщение «Hello, world!» в групповой чат |
| Ожидаемый результат | Отображение у пользователей Bob и Mark в групповом чате сообщения от Alice |
| Фактический результат | Отображение у пользователей Bob и Mark в групповом чате сообщения «<Alice>: Hello, world!» |

Таблица 9 – Тест 7

|  |  |
| --- | --- |
| Тестовая  ситуация | Проверка отправки личного сообщения |
| Исходный набор данных | Запущенный сервер с клиентами Alice, Bob, Mark. Пользователь Bob отправляет сообщение «Hi, Alice» пользователю Alice |
| Ожидаемый результат | Отображение личного сообщения от Bob у пользователя  Alice |
| Фактический результат | У пользователя Alice появление уведомления «+1» рядом с контактом Bob. После перехода к чату с пользователем Bob отображение сообщения «<Bob>: Hi, Alice» |

Таблица 10 – Тест 8

|  |  |
| --- | --- |
| Тестовая  ситуация | Проверка блокировки клиента сервером |
| Исходный набор данных | Запущенный сервер с клиентами Alice, Bob, Mark. Сервер блокирует пользователя Bob |

Окончание таблицы 10

|  |  |
| --- | --- |
| Ожидаемый результат | На сервере пользователь Bob помечается как заблокированный. Уведомление пользователя Bob о блокировке и отсутствие возможности отправлять сообщения в общий чат |
| Фактический результат | Индикатор пользователя Bob на сервере меняет цвет с зелёного на красный. У пользователя Bob в общем чате появляется сообщение «Функция отправки сообщений заблокирована.». Появление у клиента Bob сообщения «Невозможно отправить сообщение по причине блокировки» при попытке отправить сообщение в общий чат, но отсутствие каких-либо ошибок при отправке личных сообщений |

Таблица 11 – Тест 9

|  |  |
| --- | --- |
| Тестовая  ситуация | Проверка блокировки одного клиента другим клиентом |
| Исходный набор данных | Запущенный сервер с клиентами Alice, Bob, Mark. Пользователь Alice блокирует пользователя Bob |
| Ожидаемый результат | У клиента Alice пользователь Bob помечается как заблокированный. Уведомление пользователя Bob о блокировке пользователем Alice и отсутствие возможности отправлять сообщения данному собеседнику |
| Фактический результат | Индикатор пользователя Bob у клиента Alice меняет цвет с зелёного на красный. У пользователя Bob в личном чате с клиентом Alice появляется сообщение «Пользователь  <Alice> добавил Вас в чёрный список!». Появление у клиента Bob сообщения «Невозможно отправить сообщение по причине блокировки» при попытке отправить личное сообщение клиенту Alice, но отсутствие каких-либо ошибок при отправке групповых и личных сообщений другим клиентам |

Таблица 12 – Тест 10

|  |  |
| --- | --- |
| Тестовая  ситуация | Проверка отключения клиента сервером |
| Исходный набор данных | Запущенный сервер с клиентами Alice, Bob, Mark. Сервер отключает пользователя Bob |
| Ожидаемый результат | Появление соответствующих сообщений на сервере, у отключённого и оставшихся клиентов. Обновление списка контактов у клиентов, а также списка клиентов на сервере |
| Фактический результат | Появление у клиента Bob сообщения «Вы отключены от сервера» и отсутствие соединения с сервером. Появление на сервере и у клиентов Alice и Mark сообщения «Пользователь Bob отключён от сервера.». Удаление пользователя Bob из списка контактов у клиентов Alice и Mark, а также из списка подключённых клиентов на сервере |

Таблица 13 – Тест 11

|  |  |
| --- | --- |
| Тестовая  ситуация | Проверка самостоятельного отключения клиента от сервера |
| Исходный набор данных | Запущенный сервер с клиентами Alice, Bob, Mark. Пользователь Bob самостояельно отключается от сервера |
| Ожидаемый результат | Появление соответствующих сообщение на сервере, у отключённого и оставшихся клиентов. Обновление списка контактов у клиентов, а также списка подключённых пользователей на сервере |
| Фактический результат | Появление у клиента Bob сообщения «Вы покинули сервер» и отсутствие соединения с сервером. Появление на сервере и у клиентов Alice и Mark сообщения «Пользователь Bob отключён от сервера.». Удаление пользователя Bob из списка контактов у клиентов Alice и Mark, а также из списка подключённых клиентов на сервере |

# Руководство пользователя

## Работа с приложением «Cервер»

Для запуска приложения «Сервер» сделайте двойной щелчок левой кнопкой мыши по файлу ChatBoxServer.exe. Перед Вами откроется главное окно приложения (рисунок 7.1). Здесь необходимо задать параметры, с которыми будет работать сервер: IP-адрес и номер порта, максимальное количество пользователей, а также список цветов, которые будут выдаваться клиентам для отображения их сообщений. Если IP-адрес или порт введён некорректно, то соответствующее поле ввода изменит свой цвет с зелёного на красный. Для изменения цвета из таблицы цветов просто щелкните по нему левой кнопкой мыши и перед Вами откроется окно выбора нового цвета.

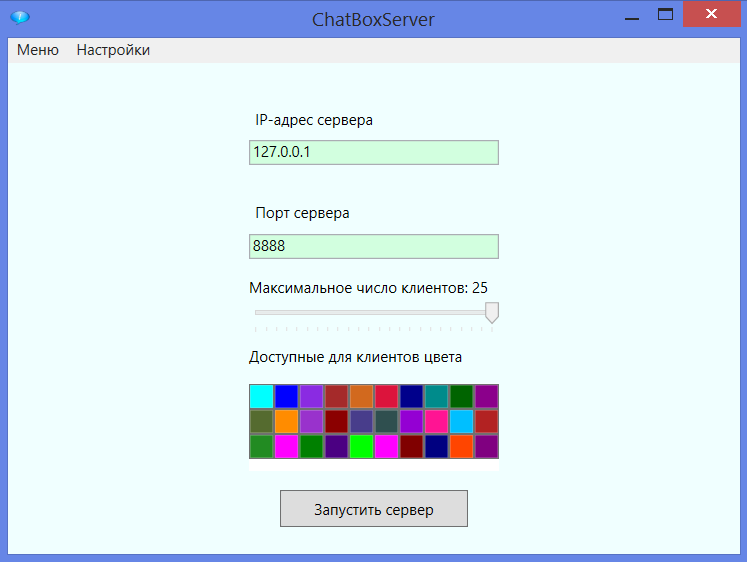


Рисунок 7.1 – Главное окно приложения

После того как все параметры верно указаны, кнопка «Запустить сервер» станет активной. Нажатие на неё запустит сервер и откроется окно управления работой сервера, описание которого приведено на рисунке 7.2.

Каждый подключённый клиент отображается в соответствующей области. Имя клиента написано тем цветом, котрое ему выделил сервер. Для отключения клиента нажмите кнопку с красным крестиком возле его имени. Если клиент активен, то его индикатор имеет зелёный цвет, иначе – красный. Для блокировки/разблокирования клиента нажмите на соответствующий индикатор, после чего он изменит цвет и статус клиента будет изменён на противоположный.

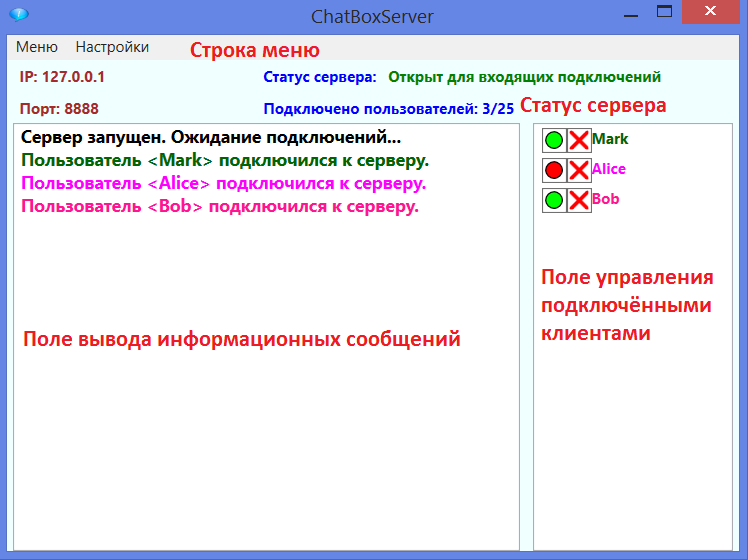


Рисунок 7.2 – Описание окна управления работой сервера

Дополнительное управление осуществляется при помощи пунктов главного меню (рисунки 7.3, 7.4).

Пункт главного меню «Меню» имеет следующие подпункты:

* «Главное меню»: выполняет переход на начальный экран приложения (доступно при отключённом сервере);
* «Выход» закрывает окно приложения при отключённом сервере, отключает всех клиентов и переходит в главное меню при запущенном сервере.

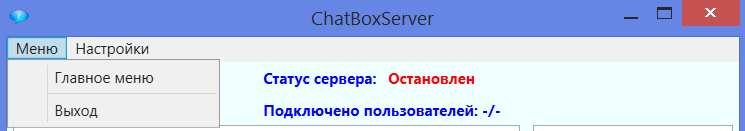


Рисунок 7.3 – Пункт главного меню «Меню»

Пункт главного меню «Настройки» имеет следующие подпункты:

* «Открыть сервер для входящих подключений»: разрешает подключение новых пользователей (доступно при закрытом сервере);
* «Закрыть сервер для входящих подключений»: запрещает подключение новых пользователей (доступно при открытом сервере);
* «Отключить всех клиентов»: отключает всех клиентов от сервера (доступно при запущенном сервере);
* «Остановить сервер» отключает всех клиентов и завершает работу сервера (доступно при запущенном сервере).

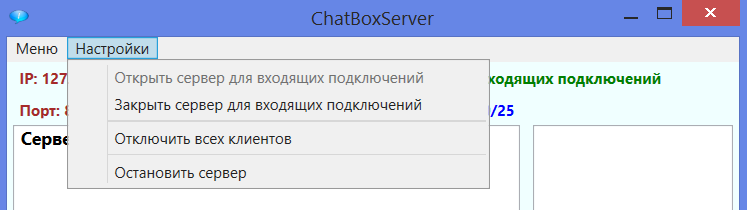


Рисунок 7.4 – Пункт главного меню «Настройки»

## Работа с приложением «Клиент»

Для запуска приложения «Клиент» сделайте двойной щелчок левой кнопкой мыши по файлу ChatBoxClient.exe. Перед Вами откроется главное окно приложения (рисунок 7.5). Здесь необходимо задать параметры сервера, к которому Вы хотите подключиться: IP-адрес и номер порта, а также имя, под которым Вас будут видеть остальные пользователи. Если IP-адрес или порт введён некорректно, то соответствующее поле ввода изменит свой цвет с зелёного на красный.

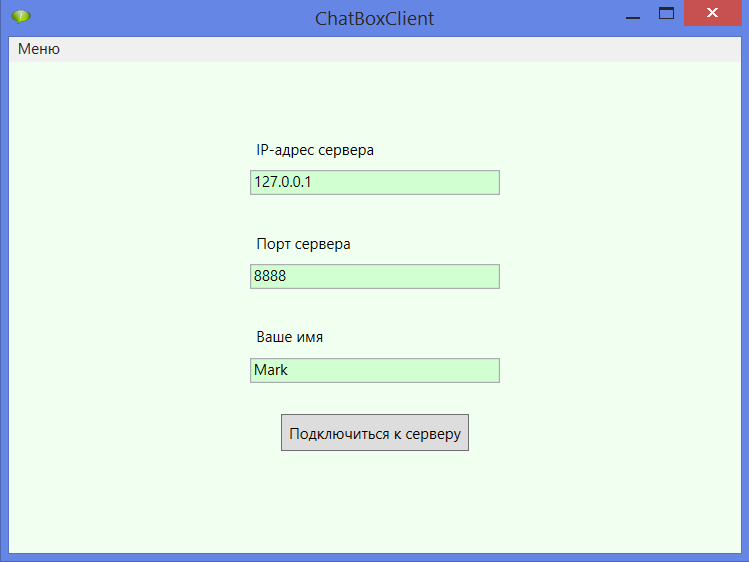


Рисунок 7.5 – Главное окно приложения

После того как все параметры верно указаны, кнопка «Подключиться к серверу» станет активной. Нажатие на неё произведёт попытку подключения к серверу и перед Вами откроется окно с диалогами, описание которого приведено на рисунке 7.6.

Справа отображён список всех пользователей, которые вместе с Вами находятся на сервере. Чтобы перейти к диалогу с конкретным пользователем просто выберите его из списка контактов. После этого откроется диалог с данным клиентом и Вы сможете перейти к общению. Групповой чат в списке контактов обозначен именем «Общий чат». Возле каждого контакта есть счётчик сообщений, отображающий количество непрочитанных сообщений от данного собеседника. Чтобы заблокировать или разблокировать какого-либо клиента необходимо нажать на индикатор рядом с его именем. После блокировки данный клиент не сможет отправлять Вам сообщения, а индикатор с его именем изменит свой цвет с зелёного на красный. Нажатие на индикатор возле группового чата приведёт к отключению Вас от сервера.

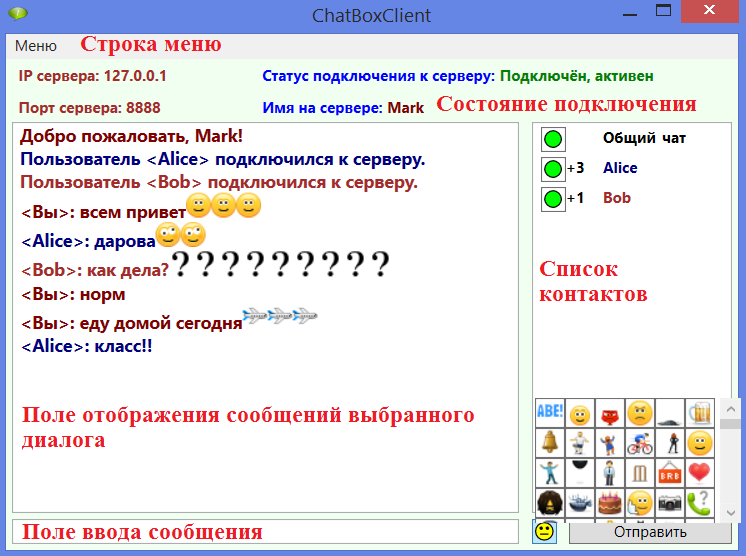


Рисунок 7.6 — Описание окна с диалогами

Для добавления смайлика в сообщение нажмите на изображение смайлика рядом с кнопкой «Отправить». После этого откроется список со смайликами. Выбрав понравившийся смайлик, просто нажмите на него и он появится в Вашем сообщении. Отправка сообщения осуществляется нажатием клавиши «Enter» на клавиатуре либо кнопки «Отправить» в окне приложения.

Дополнительное управление приложением осуществляется через строку главного меню (рисунок 7.7).

Пункт главного меню «Меню» содержит следующие подпункты:

* «Отключиться от сервера»: разрывает соединение с сервером (доступно при наличии подключения к серверу);
* «Главное меню»: выполняет переход на начальный экран приложения (доступно при отсутствии подключения к серверу);
* «Выход» закрывает окно приложения (доступно при отсутствии подключения к серверу).

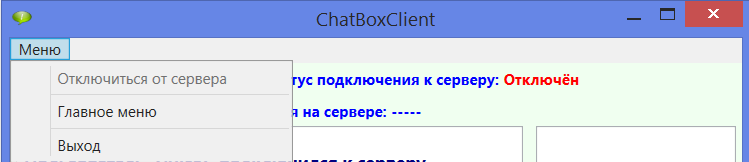


Рисунок 7.7 — Пункт главного меню «Меню»

Заключение

В ходе разработки программного средства был создан продукт, предоставляющий возможность общения между пользователями в локальной сети. Богатый функционал приложения позволяет его использовать, например, в сети какой-либо организации, школы, и т.д. Приложение имеет высокую скорость работы, интуитивно понятный интерфейс, подходящий практически для любого пользователя.

При тестировании приложения не было выявлено случаев некорректной работы программы, появления ошибок и т.д.

При разработке программного средства были усовершенствованы знания по языку программирования C#, получен опыт проектирования объектно-ориентированных приложений, изучена библиотека WPF для создания приложений с пользовательским интерфейсом. Также были изучены возможности работы с сетевыми протоколами в языке C#.

В дальнейшем разработанное приложение может быть усовершенствовано. Например, может быть добавлена регистрация пользователей, сохранение для каждого пользователя его «чёрного списка», поиск и добавление пользователей в контакты.

Список использованных источников

[1] Интернет во всех сферах жизни [Электронный ресурс] – Режим доступа: http://www.chopper-bike.ru/view\_invited\_paper.php – Дата доступа: 08.12.2016

[1] Шилд, Герберт. C# 4.0: полное руководство.: Пер. с англ. – М.: ООО «И. Д. Вильямс», 2016 – 1056 с.

[2] Что такое чат и как им пользоваться [Электронный ресурс]. – Электронные данные. – Режим доступа: <http://useroff.com/chto-takoe-chat.html> – Дата доступа: 17.05.2017

[3] IRC – Википедия [Электронный ресурс]. – Электронные данные. – Режим доступа: <https://ru.wikipedia.org/wiki/IRC> – Дата доступа: 17.05.2017

[4] Tox – Википедия [Электронный ресурс]. – Электронные данные. – Режим доступа: https://ru.wikipedia.org/wiki/Tox – Дата доступа: 17.05.2017

[5] интегрированная среда разработки Visual Studio [Электронный ресурс]. – Электронные данные. – Режим доступа: https://msdn.microsoft.com/ru-ru/library/dn762121.aspx – Дата доступа: 17.05.2017

[6] Сокеты в C# и .NET [Электронный ресурс]. – Электронные данные. – Режим доступа: https://metanit.com/sharp/net/3.1.php – Дата доступа: 17.05.2017

[7] WPF | Введение [Электронный ресурс]. – Электронные данные. – Режим доступа: https://metanit.com/sharp/wpf/1.php – Дата доступа: 17.05.2017

Приложение А

(обязательное)

Текст программного модуля CServer

class CServer

{

public int AmoOfConnectedClients { get; private set; }

public bool IsServerWork { get; private set; }

public bool IsServerClosed { get; private set; }

const bool SERVER\_MESSAGE = true;

const bool NEED\_MESSAGE = true;

const bool NOT\_NEED\_MESSAGE = false;

const string MESSG\_WAITING\_FOR\_CLIENTS = "Сервер запущен. Ожидание

подключений...";

const string MESSG\_SERVER\_STOPPED = "Сервер остановлен.";

const string MESSG\_CLIENT\_CONNECTED\_TO\_SERVER = "Пользователь <{0}>

подключился к серверу.";

const string MESSG\_CLIENT\_DISCONNECTED\_FROM\_SERVER = "Пользователь <{0}>

отключён от сервера.";

int maxAmoOfConnectedClients;

CIDManager idManager;

IPEndPoint listenPoint;

Socket listenSocket;

Thread thrListeningForNewClients;

CView UI;

List<CClient> clients;

/\* Конструктор класса \*/

public CServer(string ip, int port, int clientsAmo, CView currUI)

{

idManager = CIDManager.GetInstance();

clients = new List<CClient>();

listenPoint = new IPEndPoint(IPAddress.Parse(ip), port);

listenSocket = new Socket(AddressFamily.InterNetwork,

SocketType.Stream, ProtocolType.Tcp);

UI = currUI;

maxAmoOfConnectedClients = clientsAmo;

AmoOfConnectedClients = 0;

IsServerWork = false;

IsServerClosed = false;

}

/\* Подпрограмма для запуска сервера \*/

public void Start()

{

thrListeningForNewClients = new Thread(ListenIncomingConnections);

thrListeningForNewClients.IsBackground = true;

thrListeningForNewClients.Start();

}

/\* Подпрограмма прослушивания входящих подключений \*/

private void ListenIncomingConnections()

{

try

{

listenSocket.Bind(listenPoint);

listenSocket.Listen(10);

UI.ShowText(MESSG\_WAITING\_FOR\_CLIENTS,

idManager.ID\_SERVER\_DEFAULT);

IsServerWork = true;

UI.UpdateServerInfo();

}

catch (Exception ex)

{

UI.ShowText(ex.Message, idManager.ID\_SERVER\_ERROR);

}

if (IsServerWork)

{

try

{

while (true)

{

Socket handler = listenSocket.Accept();

try

{

TryToConnectNewClient(handler);

}

catch (Exception) { }

}

}

catch (Exception)

{

}

finally

{

RemoveAllConnections(NEED\_MESSAGE);

UI.ShowText(MESSG\_SERVER\_STOPPED,

idManager.ID\_SERVER\_DEFAULT);

IsServerWork = false;

}

}

}

/\* Подпрограмма для обработки запроса на подключение \*/

private void TryToConnectNewClient(Socket newClientSocket)

{

sbyte newClientID = idManager.GetID();

CClient newClient = new CClient(newClientSocket, this, newClientID,

UI);

newClient.GetClientName();

MessageType errorType;

if (IsPossibleToConnectClient(newClient, out errorType))

{

newClient.ConfirmConnection();

newClient.Start();

UI.UpdateServerInfo();

}

else

{

newClient.RejectConnection(errorType);

idManager.FreeID(newClientID);

}

}

/\* Подпрограмма проверки нового клиента \*/

private bool IsPossibleToConnectClient(CClient newClient, out MessageType

errorType)

{

string messageForOthers;

int numOfNewClient;

if (IsServerClosed)

{

errorType = MessageType.ServerClosed;

return false;

}

if (AmoOfConnectedClients >= maxAmoOfConnectedClients)

{

errorType = MessageType.ServerOverloaded;

return false;

}

if (IsNameConflict(newClient.ClientName))

{

errorType = MessageType.NameConflict;

return false;

}

AmoOfConnectedClients++;

numOfNewClient = AmoOfConnectedClients - 1;

clients.Add(newClient);

messageForOthers = String.Format(MESSG\_CLIENT\_CONNECTED\_TO\_SERVER,

clients[numOfNewClient].ClientName);

UI.ShowText(messageForOthers, clients[numOfNewClient].ID);

UI.AddClientToList(clients[numOfNewClient].ClientName,

clients[numOfNewClient].ID);

errorType = MessageType.ConnectionConfirmed;

return true;

}

/\* Подпрограмм проверки, есть ли пользователь

\* с заданным именем на сервере \*/

private bool IsNameConflict(string name)

{

bool res;

int i;

res = false;

i = 0;

while ((i < AmoOfConnectedClients) && (!res))

{

if (string.Compare(name, clients[i].ClientName, true) == 0)

res = true;

i++;

}

return res;

}

/\* Подпрограмма отправки новому пользователю

\* списка всех клиентов сервера \*/

public void SendClientsList(CClient receiver)

{

receiver.Writer.Write((byte)(clients.Count - 1));

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

{

if (clients[i].ID != receiver.ID)

{

receiver.Writer.Write(clients[i].ID);

receiver.Writer.Write(clients[i].ClientName);

}

}

}

/\* Подпрограмма отправки новому пользователю

\* таблицы цветов \*/

public void SendPalette(CClient receiver)

{

Color[] curPalette = CColorsLoader.GetInstance().GetCurrentPalette();

receiver.Writer.Write(curPalette.Length);

for (int i = 0; i < curPalette.Length; i++)

{

receiver.Writer.Write(curPalette[i].R);

receiver.Writer.Write(curPalette[i].G);

receiver.Writer.Write(curPalette[i].B);

}

}

/\* Подпрограмма отключения заданного клиента \*/

public void DisconnectClient(sbyte clientID)

{

int discClientInd;

string discClientName;

discClientInd = GetClientIndByID(clientID);

if (discClientInd < 0)

{

return;

}

discClientName = clients[discClientInd].ClientName;

clients[discClientInd].Stop();

RemoveConnection(clients[discClientInd].ID);

UI.UpdateServerInfo();

NotifyAllClientsAboutUserLeft(discClientName, clientID);

}

/\* Подпрограмма отключения всех клиентов \*/

public void RemoveAllConnections(bool isNeedMessage)

{

for (int i = 0; i < AmoOfConnectedClients; i++)

{

if (clients[i] != null)

{

clients[i].Stop();

idManager.FreeID(clients[i].ID);

if (isNeedMessage)

UI.ShowText(String.Format(

MESSG\_CLIENT\_DISCONNECTED\_FROM\_SERVER,

clients[i].ClientName), clients[i].ID);

}

}

clients.Clear();

AmoOfConnectedClients = 0;

UI.RemoveAllClientsFromList();

UI.UpdateServerInfo();

}

/\* Подпрограмма проверки, заблокирован

\* ли сервером клиент с заданным ID \*/

public bool IsClientBanned(sbyte ID)

{

int clientInd = GetClientIndByID(ID);

if (clientInd < 0)

{

return false;

}

return clients[clientInd].IsClientBanned;

}

/\* Подпрограмма блокировки клиента с ID \*/

public void BanClient(sbyte clientID)

{

int clientInd = GetClientIndByID(clientID);

if (clientInd < 0)

{

return;

}

clients[clientInd].IsClientBanned = true;

try

{

clients[clientInd].Writer.Write((byte)MessageType.ClientBanned);

}

catch (Exception)

{

UI.RemoveClientFromList(clients[clientInd].ID);

DisconnectClient(clients[clientInd].ID);

}

}

/\* Подпрограмма снятия блокировки с клиента с заданным ID \*/

public void ActivateClient(sbyte clientID)

{

int clientInd = GetClientIndByID(clientID);

if (clientInd < 0)

{

return;

}

clients[clientInd].IsClientBanned = false;

try

{

clients[clientInd].Writer.Write

((byte)MessageType.ClientActivated);

}

catch (Exception)

{

UI.RemoveClientFromList(clients[clientInd].ID);

DisconnectClient(clients[clientInd].ID);

}

}

/\* Подпрограмма отключения клиента с заданным ID \*/

public void RemoveConnection(sbyte ID)

{

int clientInd = GetClientIndByID(ID);

if (clientInd < 0)

{

return;

}

AmoOfConnectedClients--;

idManager.FreeID(clients[clientInd].ID);

clients.RemoveAt(clientInd);

}

/\* Подпрограмма получения индекса клиента по его ID \*/

public int GetClientIndByID(int ID)

{

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

{

if (clients[i].ID == ID)

{

return i;

}

}

return -1;

}

/\* Подпрограмма остановки сервера \*/

public void Stop()

{

if (IsServerWork && listenSocket.IsBound)

{

try

{

listenSocket.Close();

}

catch (Exception ex)

{

UI.ShowText(ex.Message, idManager.ID\_SERVER\_ERROR);

}

}

}

/\* Подпрограмма отправки группового сообщения \*/

public void SendPublicMessage(string message, sbyte senderID)

{

for (int i = 0; i < AmoOfConnectedClients; i++)

{

if (clients[i].ID != senderID)

{

try

{

clients[i].Writer.Write((byte)MessageType.PublicMessage);

clients[i].Writer.Write(senderID);

clients[i].Writer.Write(message);

}

catch (Exception)

{

UI.RemoveClientFromList(clients[i].ID);

DisconnectClient(clients[i].ID);

i--;

}

}

}

}

/\* Подпрограмма отправки личного сообщения \*/

public void SendPrivateMessage(string message, sbyte senderID, sbyte

receiverID)

{

int receiverInd = GetClientIndByID(receiverID);

if (receiverInd < 0)

{

return;

}

try

{

clients[receiverInd].Writer.Write

((byte)MessageType.PrivateMessage);

clients[receiverInd].Writer.Write(senderID);

clients[receiverInd].Writer.Write(message);

}

catch (Exception)

{

UI.RemoveClientFromList(clients[receiverInd].ID);

DisconnectClient(clients[receiverInd].ID);

}

}

/\* Подпрограмма блокировки одного пользователя другим \*/

public void BanUserByAnotherUser(sbyte senderID, sbyte receiverID)

{

int receiverInd = GetClientIndByID(receiverID);

if (receiverInd < 0)

{

return;

}

try

{

clients[receiverInd].Writer.Write((byte)MessageType.BannedByUser);

clients[receiverInd].Writer.Write(senderID);

}

catch (Exception)

{

UI.RemoveClientFromList(clients[receiverInd].ID);

DisconnectClient(clients[receiverInd].ID);

}

}

/\* Подпрограмма разблокировки одного пользователя другим \*/

public void ActivateUserByAnotherUser(sbyte senderID, sbyte receiverID)

{

int receiverInd = GetClientIndByID(receiverID);

if (receiverInd < 0)

{

return;

}

try

{

clients[receiverInd].Writer.Write

((byte)MessageType.ActivatedByUser);

clients[receiverInd].Writer.Write(senderID);

}

catch (Exception)

{

UI.RemoveClientFromList(clients[receiverInd].ID);

DisconnectClient(clients[receiverInd].ID);

}

}

/\* Подпрограмма уведомления всех клиентов о

\* подключении нового пользователя \*/

public void NotifyAllClientsAboutNewUser(string username, sbyte userID)

{

for (int i = 0; i < AmoOfConnectedClients; i++)

{

if (clients[i].ID != userID)

{

try

{

clients[i].Writer.Write((byte)MessageType.NewUser);

clients[i].Writer.Write(username);

clients[i].Writer.Write(userID);

}

catch (Exception)

{

UI.RemoveClientFromList(clients[i].ID);

DisconnectClient(clients[i].ID);

i--;

}

}

}

}

/\* Подпрограмма уведомления всех клиентов об отключении

\* одного из пользователей \*/

public void NotifyAllClientsAboutUserLeft(string username, sbyte userID)

{

for (int i = 0; i < AmoOfConnectedClients; i++)

{

if (clients[i].ID != userID)

{

try

{

clients[i].Writer.Write((byte)MessageType.UserLeave);

clients[i].Writer.Write(userID);

}

catch (Exception)

{

UI.RemoveClientFromList(clients[i].ID);

DisconnectClient(clients[i].ID);

i--;

}

}

}

string message = String.Format(MESSG\_CLIENT\_DISCONNECTED\_FROM\_SERVER,

username);

UI.ShowText(message, userID);

}

/\* Подпрограмма закрытия сервера для новых клиентов \*/

public void CloseForIncomingConnections()

{

IsServerClosed = true;

}

/\* Подпрограмма открытия сервера для новых клиентов \*/

public void OpenForIncomingConnections()

{

IsServerClosed = false;

}

}

ВЕДОМОСТЬ ДОКУМЕНТОВ

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Обозначение | | | | | Наименование | | | | Дополнитель-ные сведения | | |
|  | | | | | Текстовые документы | | | |  | | |
|  | | | | |  | | | |  | | |
| БГУИР КП 1–40 01 01 23 ПЗ | | | | | Пояснительная записка | | | | 45 с. | | |
|  | | | | |  | | | |  | | |
|  | | | | | Графические документы | | | |  | | |
|  | | | | |  | | | |  | | |
| ГУИР.551004-01 СА | | | | | Обработка сервером запросов на подключение | | | | Формат А1 | | |
|  | | | | | Схема алгоритма | | | |  | | |
|  | | | | |  | | | |  | | |
|  | | | | |  | | | |  | | |
|  | | | | |  | | | |  | | |
|  | | | | |  | | | |  | | |
|  | | | | |  | | | |  | | |
|  | | | | |  | | | |  | | |
|  | | | | |  | | | |  | | |
|  | | | | |  | | | |  | | |
|  | | | | |  | | | |  | | |
|  | | | | |  | | | |  | | |
|  | | | | |  | | | |  | | |
|  | | | | |  | | | |  | | |
|  | | | | |  | | | |  | | |
|  | | | | |  | | | |  | | |
|  | | | | |  | | | |  | | |
|  | | | | |  | | | |  | | |
|  | | | | |  | | | |  | | |
|  | | | | |  | | | |  | | |
|  | | | | |  | | | |  | | |
|  | | | | |  | | | |  | | |
|  | | | | |  | | | |  | | |
|  |  |  |  |  | БГУИР КП 1–40 01 01 23 ПЗ | | | | | | |
|  |  |  |  |  |
|  |  |  |  |  |
| Измммм | Лист | № докум. | Подп | Дата | Программное средство  мгновенного обмена сообщениями в локальной сети | Литера | | | | Лист | Листов |
| Разраб. | | Филипук Д.С. |  | 01.06 | Т |  |  | | 45 | 45 |
| Пров. | | Красковский П.Н. |  | 01.06 | Кафедра ПОИТ  гр. 551004 | | | | | |
|  | |  |  |  |
|  | |  |  |  |
|  | |  |  |  |