Учреждение образования

«Белорусский государственный университет

информатики и радиоэлектроники»

Факультет компьютерных систем и сетей

Кафедра программное обеспечение информационных технологий

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

к курсовому проекту

«Сетевой чат»

по дисциплине

«Компьютерные системы и сети»

Выполнил: Руководитель:

студент гр. 351003 Третьяков Ф. И.

Наумец Юлия Станиславовна

Минск 2014

Задание на курсовую работу

Разработать клиент-серверный чат.

Использовать операционную систему Microsoft® Windows, файловую систему – любую, поддерживаемую операционной системой, интерфейс – графический, язык разработки – C#.

Разработать отдельно приложение сервера и приложение клиента. Серверная и клиентская часть должны выполнять свои стандартные функции.

Реализовать интуитивно понятный пользовательский интерфейс.

Содержание

[Введение](#_Toc406555029)

[1. Обзор источников](#_Toc406555030)

[1.1 Основные термины](#_Toc406555031)

[1.2 Обзор аналогов](#_Toc406555032)

[1.3 Выбор технологий для создания проекта](#_Toc406555033)

[1.4 Основные функции программного обеспечения](#_Toc406555034)

[2. Структурное проектирование](#_Toc406555035)

[2.1 Структурная схема приложения Server](#_Toc406555036)

[2.2 Структурная схема приложения Chatter](#_Toc406555037)

[3. Функциональное проектирование](#_Toc406555041)

3.1 Контроллер

3.2 Модель

[4. Разработка программных модулей](#_Toc406555048)

[5. Тестирование](#_Toc406555054)

[Заключение](#_Toc406555055)

[Список литературы](#_Toc406555056)

# Введение

Во время внедрения новых технологий в повседневную жизнь, всё больше и больше людей вовлечены в процесс виртуального общения. Распространение сети Интернет в каждый дом изначально было запланировано исключительно для обмена информацией. Только позже стало понятно, что Интернет предоставляет более широкие возможности, и приложения для общения отошли на второй план. Поэтому теперь ценится не внешний вид приложений, и иногда даже не функционал, а их простота интерфейса и быстродействие.

В связи с этим, было принято решение разработать для пользователей удобный программный продукт, используя который, пользователь может обмениваться мгновенными личными сообщениями с другими пользователями.

Кроме того, любой пользователь может создать свой сервер, к которому смогут подключиться только те, кто знает IP-адрес сервера, что поможет защититься от нежелательных подключений. Поэтому, данный программный продукт может быть использован в офисах, для обеспечения связи между коллегами.

При дальнейшем развитии данное приложение может стать востребованным, благодаря тому, что оно не требует большого количества ресурсов компьютера. Также, при надобности есть возможность сделать приложение кроссплатформенным.

# 1. Обзор источников

## 1.1 Основные термины

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 ответов сразу 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. Под словом чат обычно понимается групповое общение, хотя к ним можно отнести и обмен текстом «один на один» посредством программ [мгновенного обмена сообщениями](https://ru.wikipedia.org/wiki/%D0%9F%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%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), например, [XMPP](https://ru.wikipedia.org/wiki/XMPP), [ICQ](https://ru.wikipedia.org/wiki/ICQ) или даже [SMS](https://ru.wikipedia.org/wiki/SMS) (необычно это слышать, но это факт, SMS — частный пример чата)[1].

Клиент — это аппаратный или программный компонент вычислительной системы, посылающий запросы [серверу](https://ru.wikipedia.org/wiki/%D0%A1%D0%B5%D1%80%D0%B2%D0%B5%D1%80_(%D0%BF%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)). [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), являющаяся клиентом, взаимодействует с сервером, используя определённый [протокол](https://ru.wikipedia.org/wiki/%D0%9F%D1%80%D0%BE%D1%82%D0%BE%D0%BA%D0%BE%D0%BB%D1%8B_%D0%BF%D0%B5%D1%80%D0%B5%D0%B4%D0%B0%D1%87%D0%B8_%D0%B4%D0%B0%D0%BD%D0%BD%D1%8B%D1%85). Она может запрашивать с сервера какие-либо [данные](https://ru.wikipedia.org/wiki/%D0%94%D0%B0%D0%BD%D0%BD%D1%8B%D0%B5), манипулировать данными непосредственно на сервере, запускать на сервере новые [процессы](https://ru.wikipedia.org/wiki/%D0%9F%D1%80%D0%BE%D1%86%D0%B5%D1%81%D1%81_(%D0%B8%D0%BD%D1%84%D0%BE%D1%80%D0%BC%D0%B0%D1%82%D0%B8%D0%BA%D0%B0)) и т. п. Полученные от сервера данные клиентская программа может предоставлять [пользователю](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) или использовать как-либо иначе, в зависимости от назначения программы. Программа-клиент и программа-сервер могут работать как на одном и том же компьютере, так и на разных. Во втором случае для обмена информацией между ними используется [сетевое соединение](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_%D1%81%D0%B5%D1%82%D1%8C)[2].

Серверное программное обеспечение — программный компонент вычислительной системы, выполняющий сервисные (обслуживающие) функции по запросу [клиента](https://ru.wikipedia.org/wiki/%D0%9A%D0%BB%D0%B8%D0%B5%D0%BD%D1%82_(%D0%B8%D0%BD%D1%84%D0%BE%D1%80%D0%BC%D0%B0%D1%82%D0%B8%D0%BA%D0%B0)), предоставляя ему доступ к определённым ресурсам или услугам. Для взаимодействия с клиентом (или клиентами, если поддерживается одновременная работа с несколькими клиентами) сервер выделяет необходимые ресурсы [межпроцессного взаимодействия](https://ru.wikipedia.org/wiki/%D0%9C%D0%B5%D0%B6%D0%BF%D1%80%D0%BE%D1%86%D0%B5%D1%81%D1%81%D0%BD%D0%BE%D0%B5_%D0%B2%D0%B7%D0%B0%D0%B8%D0%BC%D0%BE%D0%B4%D0%B5%D0%B9%D1%81%D1%82%D0%B2%D0%B8%D0%B5) ([разделяемая память](https://ru.wikipedia.org/wiki/%D0%A0%D0%B0%D0%B7%D0%B4%D0%B5%D0%BB%D1%8F%D0%B5%D0%BC%D0%B0%D1%8F_%D0%BF%D0%B0%D0%BC%D1%8F%D1%82%D1%8C), [пайп](https://ru.wikipedia.org/wiki/%D0%98%D0%BC%D0%B5%D0%BD%D0%BE%D0%B2%D0%B0%D0%BD%D0%BD%D1%8B%D0%B9_%D0%BA%D0%B0%D0%BD%D0%B0%D0%BB" \o "Именованный канал), [сокет](https://ru.wikipedia.org/wiki/%D0%A1%D0%BE%D0%BA%D0%B5%D1%82_(%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%BD%D1%8B%D0%B9_%D0%B8%D0%BD%D1%82%D0%B5%D1%80%D1%84%D0%B5%D0%B9%D1%81)) и т. п.) и ожидает запросы на открытие соединения (или, собственно, запросы на предоставляемый сервис)[3].

Клиент-сервер — вычислительная или сетевая архитектура, в которой задания или сетевая нагрузка распределены между поставщиками услуг, называемыми серверами, и заказчиками услуг, называемыми клиентами. Физически клиент и сервер — это [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). Обычно они взаимодействуют через компьютерную сеть посредством [сетевых протоколов](https://ru.wikipedia.org/wiki/%D0%A1%D0%B5%D1%82%D0%B5%D0%B2%D0%BE%D0%B9_%D0%BF%D1%80%D0%BE%D1%82%D0%BE%D0%BA%D0%BE%D0%BB) и находятся на разных вычислительных машинах, но могут выполняться также и на одной машине.

Преимущества:

* Отсутствие дублирования кода программы-сервера программами-клиентами.
* Так как все вычисления выполняются на сервере, то требования к компьютерам, на которых установлен клиент, снижаются.
* Все данные хранятся на сервере, который, как правило, защищён гораздо лучше большинства клиентов. На сервере проще организовать контроль полномочий, чтобы разрешать доступ к данным только клиентам с соответствующими правами доступа.

Недостатки:

* Неработоспособность сервера может сделать неработоспособной всю вычислительную сеть. Неработоспособным сервером следует считать сервер, производительности которого не хватает на обслуживание всех клиентов, а также сервер, находящийся на ремонте, профилактике и т. п.
* Поддержка работы данной системы требует отдельного специалиста — системного администратора[4].

## 1.2 Обзор аналогов

MyChat — клиент-серверный чат, он идеально подходит для передачи сообщений и обмена файлами в сети офиса или компании. MyChat может легко работать внутри корпоративной сети или городских сетей, а также через Интернет. Продукт будет полезен компаниям, которым нужно средство корпоративного общения, но недопустимо использование таких IMсистем, как ICQ или Skype, в связи с ограниченным доступом к интернету или корпоративными требованиями к безопасности, а также возможности утечки информации. С MyChat таких проблем не будет: вы сами устанавливаете сервер и сможете контролировать абсолютно всё.

* менеджер учётных записей
* система массового оповещения пользователей
* гибкие настройки
* персональный профиль
* передача файлов
* проверка правописания[5]

Simple Chat – это простой и мощный [кроссплатформенный](https://schat.me/ru/cross-platform) клиент-серверный чат для локальной сети и Интернета, с открытым исходным кодом.

* передача файлов
* анимированные смайлы
* комнаты
* звуковые оповещения[6]

## 1.3 Выбор технологий для создания проекта

Для создания сетевого чата будем использовать фреймворк ASP.NET MVC – фреймворк,  для создания веб-приложений, который реализует шаблон Model-view-controller. Данный фреймворк добавлен Microsoft в ASP.NET.

Концепция паттерна (шаблона) MVC (model - view - controller) предполагает разделение приложения на три компонента:

* Контроллер (controller) представляет класс, обеспечивающий связь между пользователем и системой, представлением и хранилищем данных. Он получает вводимые пользователем данные и обрабатывает их. И в зависимости от результатов обработки отправляет пользователю определенный вывод, например, в виде представления.
* Представление (view) - это собственно визуальная часть или пользовательский интерфейс приложения. Как правило, html-страница, которую пользователь видит, зайдя на сайт.
* Модель (model) представляет класс, описывающий логику используемых данных.

Общую схему взаимодействия этих компонентов можно представить следующим образом:



Windows Forms — интерфейс программирования приложений (API), отвечающий за графический интерфейс пользователя и являющийся частью Microsoft .NET Framework. Данный интерфейс упрощает доступ к элементам интерфейса Microsoft Windows за счет создания обѐртки для существующего Win32 API в управляемом коде. Причем управляемый код — классы, реализующие API для Windows Forms, не зависят от языка разработки. То есть программист одинаково может использовать Windows Forms как при написании ПО на C#, С++, так и на VB.Net, J# и др.

Платформа .NET Framework — это интегрированный компонент Windows, который поддерживает создание и выполнение нового поколения приложений и веб-служб XML. При разработке платформы .NET Framework учитывались следующие цели.

* Обеспечение согласованной объектно-ориентированной среды программирования для локального сохранения и выполнения объектного кода, для локального выполнения кода, распределенного в Интернете, либо для удаленного выполнения.
* Обеспечение среды выполнения кода, минимизирующей конфликты при развертывании программного обеспечения и управлении версиями.
* Обеспечение среды выполнения кода, гарантирующей безопасное выполнение кода, включая код, созданный неизвестным или не полностью доверенным сторонним изготовителем.
* Обеспечение среды выполнения кода, исключающей проблемы с производительностью сред выполнения сценариев или интерпретируемого кода.
* Обеспечение единых принципов работы разработчиков для разных типов приложений, таких как приложения Windows и веб-приложения.
* Разработка взаимодействия на основе промышленных стандартов, которое обеспечит интеграцию кода платформы .NET Framework с любым другим кодом. [7]

## 1.4 Основные функции программного обеспечения

Основными функциями клиента являются:

* отправка запроса на авторизацию
* вход в чат
* выход из сетевого чата
* отправка сообщений друзьям через сервер
* отправка запроса о состоянии друзей
* отображение истории сообщений текущего сеанса

Основными функциями сервера являются:

* хранение учётных записей пользователей
* хранение списка пользователей онлайн
* ответ на запросы авторизации/регистрации/добавления друга/отправки сообщения/состоянии пользователей
* вывод журнала событий на сервере в окно приложения

# 2. Структурное проектирование

Далее будет рассмотрен каркас приложений и их основная логика работы.

# 3. Функциональное проектирование

3.1 Модель

Рассмотрим модель, которую мы будем использовать. Основная модель – ChatModel. Она использует две модели для хранения сообщений и пользователей соответственно: ChatMessage и ChatUser:

public class ChatModel

{

// Все пользователи чата

public List<ChatUser> Users;

// все сообщения

public List<ChatMessage> Messages;

public ChatModel()

{

Users = new List<ChatUser>();

Messages = new List<ChatMessage>();

Messages.Add(new ChatMessage()

{

Text = "Чат запущен " + DateTime.Now

});

}

}

public class ChatUser

{

public string Name;

public DateTime LoginTime;

public DateTime LastPing;

}

public class ChatMessage

{

// автор сообщения, если null - автор сервер

public ChatUser User;

// время сообщения

public DateTime Date = DateTime.Now;

// текст

public string Text = "";

}

Все данные - пользователи и их сообщения будут храниться в переменной chatModel. И так как она будет общим для всех пользователей, она объявлена статической.

3.2. Контроллер

Теперь рассмотрим структуру контроллера. В нём реализован метод Index: public ActionResult Index(string user, bool? logOn, bool? logOff, string chatMessage).

Метод Index принимает четыре параметра.

* Первый параметр идентифицирует пользователя.
* Параметр logOn при значении true указывает, что пользователь осуществил вход.
* Параметр logOff при значении true, наоборот, указывает, что пользователь вышел из чата.
* И последний параметр содержит сообщение пользователя. И в зависимости от значений параметров возвращаем либо обычное представление Index.cshtml, либо одно из частичных представлений.

Если запрос не является AJAX-запросом, передаеся обычное представление. Если установлен параметр logOn или logOff, то возвращаем частичное представление ChatRoom. И если установлен последний параметр с сообщением пользователя, то возвращается частичное представление History.

# 4. Разработка программных модулей

# 5. Тестирование

# Заключение

# Список литературы

[1]. [https://ru.wikipedia.org/wiki/](https://ru.wikipedia.org/wiki/%D0%A7%D0%B0%D1%82)Чат

[2]. https://ru.wikipedia.org/wiki/Клиент\_(информатика)

[3]. <https://ru.wikipedia.org/wiki/Сервер_(программное_обеспечение)>

[4]. https://ru.wikipedia.org/wiki/Клиент-сервер

[5]. <http://www.nsoft-s.com/aboutmychat.html#_>

[6]. <https://schat.me/ru/features>

[7]. https://msdn.microsoft.com/ru-ru/library/vstudio/zw4w595w(v=vs.100).aspx