Санкт-Петербургский государственный политехнический университет

Институт Прикладной математики и механики
Кафедра Прикладной математики

C.A.Бондарев

КОРПОРАТИВНЫЙ ЧАТ С ВОЗМОЖНОСТЬЮ ПОДКЛЮЧЕНИЯ ПО IP

Курсовой проект

Направление подготовки бакалавров: 010800 Механика и математическое моделирование

Профиль ООП: 010800.62.01 Механика деформируемого твердого тела

Группа 23604/1

Руководитель проекта: Панченко А.Ю.

Допущен к защите:

«\_\_» 20\_\_ г.

 Санкт-Петербург

2014

#

# **Оглавление**

[***Оглавление*** 2](file:///C%3A%5CUsers%5C%D0%A1%D0%B5%5CDownloads%5CProgramming_project1.docx#_Toc388226015)

[***Введение*** 3](file:///C%3A%5CUsers%5C%D0%A1%D0%B5%5CDownloads%5CProgramming_project1.docx#_Toc388226016)

[***Глава 1. История возникновения чата. Его виды*** 4](file:///C%3A%5CUsers%5C%D0%A1%D0%B5%5CDownloads%5CProgramming_project1.docx#_Toc388226017)

[1.1 История возникновения чата 4](file:///C%3A%5CUsers%5C%D0%A1%D0%B5%5CDownloads%5CProgramming_project1.docx#_Toc388226018)

[1.2 Виды чатов 6](file:///C%3A%5CUsers%5C%D0%A1%D0%B5%5CDownloads%5CProgramming_project1.docx#_Toc388226019)

[***Глава 2.***  7](file:///C%3A%5CUsers%5C%D0%A1%D0%B5%5CDownloads%5CProgramming_project1.docx#_Toc388226021)

[2.1 Начало работы с проектом 7](file:///C%3A%5CUsers%5C%D0%A1%D0%B5%5CDownloads%5CProgramming_project1.docx#_Toc388226022)

[2.2 Написание кода 8](file:///C%3A%5CUsers%5C%D0%A1%D0%B5%5CDownloads%5CProgramming_project1.docx#_Toc388226023)

[2.3 Итог работы](file:///C%3A%5CUsers%5C%D0%A1%D0%B5%5CDownloads%5CProgramming_project1.docx#_Toc388226025) 11

[***Заключение***](file:///C%3A%5CUsers%5C%D0%A1%D0%B5%5CDownloads%5CProgramming_project1.docx#_Toc388226026) 12

[***Литература***](file:///C%3A%5CUsers%5C%D0%A1%D0%B5%5CDownloads%5CProgramming_project1.docx#_Toc388226027) 14

***Введение***

Целью данной курсовой работы является создание функционального окна, через которое пользователи смогут общаться на расстоянии, каждый сидя за своим компьютером или другим электронно-информационным устройством. В процессе выполнения курсовой работы необходимо решить следующие задачи:
 ***-*** Создать два функциональных окна – одно для сервера, другое для клиента.
  ***-*** Пристроить к функциональным окнам кнопки, через которое будет осуществляться общение.

 В первой главе приведена история возникновения чата как такового, а также виды чатов.
 Во второй главе описывается то, как создается это приложение с примерами кодов страниц.

***Глава 1. История возникновения чата. Его виды.***

* 1. **История возникновения чата**

Вопросы общения интересовали людей всегда. Для того, чтобы можно было обмениваться информацией не только при личной встрече, но и на огромных расстояниях, люди изобретали всё новые и новые технические средства, организовывали почтовые системы, протягивали кабели через континенты и океаны, запускали спутники связи. С развитием информационных технологий стали возможным еще более глобальные коммуникации. Историческим «докомпьютерным» предшественником чатов, несомненно, был телефон. Ни почта, ни телеграф не позволяли общаться в реальном времени и не были доступны в домашней обстановке. Изобретение и распространение телефона по планете вызвало настоящую революцию в средствах и способах общения. Возможность поговорить с собеседником на другой стороне Земли казалась настоящим чудом.

Во второй половине XX века начали бурно развиваться компьютеры. Однако долгое время они были большими и слишком дорогими, что препятствовало тому, чтобы расходовать драгоценное машинное время на забавы с обменом сообщениями вместо расчетов атомных бомб. К тому же, до конца 60-х годов они не были связаны друг с другом. Предок Интернета, сеть ARPANET, в 1969 году насчитывала только четыре связанных друг с другом научных компьютера. Чуть позже, в 1971 году, была придумана электронная почта, которая стала необычайно популярна ввиду своего удобства. Постепенно появились новые службы сообщений, такие, как списки почтовой рассылки, новостные группы и доски объявлений. Однако в то время сеть ARPANET ещё не могла легко взаимодействовать с другими сетями, построенными на других технических стандартах, что затрудняло её распространение. Но тем не менее, эта проблема вскоре была решена после перехода сетей на протокол обмена данными TCP/IP, который успешно применяется до сих пор. Именно в 1983 году термин «Интернет» закрепился за сетью ARPANET.

Программы для обмена текстовыми строками, несмотря на простоту самой идеи, появились не сразу. Примерно в 1974 году для мэйнфрейма PLATO был разработана программа Talkomatic, потенциально позволявшая общаться между тысячей терминалов системы. В 1980-x появилась система Freelancing' Round table. Однако по-настоящему популярным стал разработанный в 1988 году протокол, названный Internet Relay Chat (IRC), что примерно можно перевести как ретранслируемый интернет-разговор. Где-то в это же время появилось и распространилось само понятие «чат». Общение в IRC быстро стало популярным из-за простоты процесса и дружественности среды. В 1991 году во время операции «Буря в пустыне» была организована IRC-трансляция новостей — сообщения со всего мира собирались в одном месте и в режиме реального времени передавались в IRC. Есть сведения, что подобным образом IRC использовался и во время путча в СССР, когда пользователи из Москвы моментально сообщали всему миру о происходящем на улицах. Для клиентов IRC, написано множество ботов, например, Eggdrop, автоматизирующие многие рутинные операции. Самым известным из клиентов IRC стал mIRC; благодаря простой и эффективной системе команд для него было написано множество скриптов, которые также позволяют выполнять широкий спектр действий. Боты и mIRC-боты используются для различных игр в каналах — «Мафия», «Викторина», и других. Разработчики IRC настолько хорошо продумали его архитектуру, что её с тех пор практически не требовалось изменять. Конечно, у него есть недостатки: короткие сообщения, проблема с кодировками, невозможность посмотреть историю сообщений при подключении. Однако он был и остаётся популярным средством для чата, хотя и в значительной мере потеснен со своих позиций. В частности, в 1998 году был придуман похожего назначения протокол Jabber - даже его название (англ. jabber болтовня, трёп; тарабарщина) отсылало к слову chat. Jabber содержал в себе многие технические новшества и постепенно получил широкое распространение, а также стал основой многих сервисов. Были и другие протоколы, менее известные, например, SIP.

* 1. **Виды чатов.**

Существует несколько разновидностей программной реализации чатов:

1. HTTP или веб-чаты. Такой чат выглядит как обычная веб-страница, где можно прочесть последние несколько десятков фраз, написанные участниками чата и модераторами. Чаты, использующие технологию Adobe Flash. Вместо периодической перезагрузки страницы, между клиентом и сервером открывается сокет, что позволяет моментально отправлять или получать сообщения, расходуя меньше трафика.
2. IRC, специализированный протокол для чатов.
3. Программы-чаты для общения в локальных сетях (например, Vypress Chat, Intranet Chat). Часто есть возможность передачи файлов.
4. Чаты, реализованные поверх сторонних протоколов (например, чат, использующий ICQ).

По применению чаты делятся на:

1. all2all - групповая коммуникация (например, IRC, Jabber, Yahoo! Chat, AVACS Live Chat);
2. p2p - персональные коммуникации (например, ICQ, Jabber, Skype, Yahoo! Messenger, AOL Instant Messenger, Hamachi) — личное общение.

***Глава 2. Создание приложения***

**2.1 Начало работы с проектом**

Перед нами встала задача – как написать код? Программа должна включать в себя следующие опции:
1) Корректное отображение окон сервера и клиента.
2) Возможность подключиться клиенту через ip адрес.
3) Возможность запустить сервер, остановить его, отправить сообщение всем или только выборочным участникам чата.
4) Удачное разъединение соединения, смена порта и/или ip адреса.

**2.2 Написание кода**

Разберем написание кода на примере одного cpp. Файла, отвечающего за работу сервера.

#include "myserver.h"

MyServer::MyServer(QWidget \*widget, QObject \*parent) :QTcpServer(parent)

{

 \_widget = widget;

}

bool MyServer::doStartServer(QHostAddress addr, qint16 port) // проверка на запуск сервера.

{

 if (!listen(addr, port))

 {

 qDebug() << "Server not started at" << addr << ":" << port;

 return false;

 }

 qDebug() << "Server started at" << addr << ":" << port;

 return true;

}

void MyServer::doSendToAllUserJoin(QString name) // вывести на экран всем в чат, что пользователь добавился в чат.

{

 QByteArray block;

 QDataStream out(&block, QIODevice::WriteOnly);

 //резервируем место под размер блока

 out << (quint16)0 << MyClient::comUserJoin << name;

 //пишем на зарезервированное место размер блока

 out.device()->*seek*(0);

 out << (quint16)(block.size() - sizeof(quint16));

 //отправляем всем авторизованным, кроме того, кто вошел

 for (int i = 0; i < \_clients.length(); ++i)

 if (\_clients.at(i)->getName() != name && \_clients.at(i)->getAutched())

 \_clients.at(i)->\_sok->write(block);

}

void MyServer::doSendToAllUserLeft(QString name)// вывести на экран всем в чат, что пользователь ушел из чата.

{

 QByteArray block;

 QDataStream out(&block, QIODevice::WriteOnly);

 out << (quint16)0 << MyClient::comUserLeft << name;

 out.device()->*seek*(0);

 out << (quint16)(block.size() - sizeof(quint16));

 for (int i = 0; i < \_clients.length(); ++i)

 if (\_clients.at(i)->getName() != name && \_clients.at(i)->getAutched())

 \_clients.at(i)->\_sok->write(block);

}

void MyServer::doSendToAllMessage(QString message, QString fromUsername) // отправить сообщение всем в чат(клиентом).

{

 QByteArray block;

 QDataStream out(&block, QIODevice::WriteOnly);

 out << (quint16)0 << MyClient::comMessageToAll << fromUsername << message;

 out.device()->*seek*(0);

 out << (quint16)(block.size() - sizeof(quint16));

 for (int i = 0; i < \_clients.length(); ++i)

 if (\_clients.at(i)->getAutched())

 \_clients.at(i)->\_sok->write(block);

}

void MyServer::doSendToAllServerMessage(QString message)// отправить сообщение сервера всем в чат.

{

 QByteArray block;

 QDataStream out(&block, QIODevice::WriteOnly);

 out << (quint16)0 << MyClient::comPublicServerMessage << message;

 out.device()->*seek*(0);

 out << (quint16)(block.size() - sizeof(quint16));

 for (int i = 0; i < \_clients.length(); ++i)

 if (\_clients.at(i)->getAutched())

 \_clients.at(i)->\_sok->write(block);

}

void MyServer::doSendServerMessageToUsers(QString message, const QStringList &users)// отправить сообщения сервера в пользователю.

{

 QByteArray block;

 QDataStream out(&block, QIODevice::WriteOnly);

 out << (quint16)0 << MyClient::comPrivateServerMessage << message;

 out.device()->*seek*(0);

 out << (quint16)(block.size() - sizeof(quint16));

 for (int j = 0; j < \_clients.length(); ++j)

 if (users.contains(\_clients.at(j)->getName()))

 \_clients.at(j)->\_sok->write(block);

}

void MyServer::doSendMessageToUsers(QString message, const QStringList &users, QString fromUsername) // отправить сообщение в чат.

{

 QByteArray block, blockToSender;

 QDataStream out(&block, QIODevice::WriteOnly);

 out << (quint16)0 << MyClient::comMessageToUsers << fromUsername << message;

 out.device()->*seek*(0);

 out << (quint16)(block.size() - sizeof(quint16));

 //тому, кто отправил нужно отправить не его имя, а список тех, кому он отправил

 QDataStream outToSender(&blockToSender, QIODevice::WriteOnly);

 outToSender << (quint16)0 << MyClient::comMessageToUsers << users.join(",") << message;

 outToSender.device()->*seek*(0);

 outToSender << (quint16)(blockToSender.size() - sizeof(quint16));

 for (int j = 0; j < \_clients.length(); ++j)

 if (users.contains(\_clients.at(j)->getName()))

 \_clients.at(j)->\_sok->write(block);

 else if (\_clients.at(j)->getName() == fromUsername)

 \_clients.at(j)->\_sok->write(blockToSender);

}

QStringList MyServer::getUsersOnline() const // выводим на экран пользователей, которые в чате.

{

 QStringList l;

 foreach (MyClient \* c, \_clients)

 if (c->getAutched())

 l << c->getName();

 return l;

}

bool MyServer::isNameValid(QString name) const // проверяем - годно ли данное имя или нет.

{

 if (name.length() > 20 || name.length() < 5)

 return false;

 QRegExp r("[A-Za-z0-9\_]+");

 return r.exactMatch(name);

}

bool MyServer::isNameUsed(QString name) const // проверяем - занято ли имя или нет.

{

 for (int i = 0; i < \_clients.length(); ++i)

 if (\_clients.at(i)->getName() == name)

 return true;

 return false;

}

void MyServer::*incomingConnection*(int handle)

{

 //создаем клиента

 MyClient \*client = new MyClient(handle, this, this);

 if (\_widget != 0)

 {

 connect(client, SIGNAL(addUserToGui(QString)), \_widget, SLOT(onAddUserToGui(QString)));

 connect(client, SIGNAL(removeUserFromGui(QString)), \_widget, SLOT(onRemoveUserFromGui(QString)));

 connect(client, SIGNAL(messageToGui(QString,QString,QStringList)), \_widget, SLOT(onMessageToGui(QString,QString,QStringList)));

 }

 connect(client, SIGNAL(removeUser(MyClient\*)), this, SLOT(onRemoveUser(MyClient\*)));

 \_clients.append(client);

}

void MyServer::onRemoveUser(MyClient \*client)

{

 \_clients.removeAt(\_clients.indexOf(client));

}

void MyServer::onMessageFromGui(QString message, const QStringList &users)

{

 if (users.isEmpty())

 doSendToAllServerMessage(message);

 else

 doSendServerMessageToUsers(message, users);

}

**2.3 Итог работы.**

Мы написали чат, мы можем общаться. Ура!