Специальность: 09.02.07 Информационные системы и программирование

Проверил:

\_\_\_\_\_\_\_\_\_\_\_ Антонов М.Б.

Дата: «5» апреля 2020 г.

Оценка: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Министерство образования и науки Российской Федерации

Федеральное государственное автономное образовательное учреждение высшего образования

«Санкт-Петербургский национальный исследовательский университет информационных технологий, механики и оптики»

(Университет ИТМО)

факультет среднего профессионального образования

Отчет

о Лабораторной работе №6

по теме: Простейший WebSocket Server чата

по дисциплине: Разработка программных модулей

Выполнил:

студент группы 2333

\_\_\_\_\_\_\_\_\_\_\_ Агапова Е.А.

Санкт-Петербург 2020

Содержание

[1 Постановка задачи 3](#_Toc35698558)

[1.1 Цель 3](#_Toc35698559)

[1.2 Задачи 3](#_Toc35698560)

[1.3 Задание 3](#_Toc35698561)

[2 Диаграммы 5](#_Toc35698562)

[3 Исходный код 6](#_Toc35698563)

[4 ЛИСТИНГ ВЫполнения программы 19](#_Toc35698564)

[5 Вывод 21](#_Toc35698565)

1. Постановка задачи
   1. Цель

Познакомиться с основами работы QWebSocket, QSettings и системой ресурсов в приложениях Qt.

* 1. Задачи

1. изучить основы работы с QWebSocket соединениями;
2. изучить работу с настройками приложения QSettings;
3. изучить работу с системой ресурсов Qt;
4. создать более универсальный сервер чата.
   1. Задание

Требуется доработка/переработка предыдущей лабораторной работы. Требуется расширить/заменить следующие функции сервера:

* QTcpSocket требуется заменить на QWebSocket, соответсвующую доработку требуется произвести и на клиенте. Клиент кроме этой доработки менять более не потребуется;
* настройки порта на котором должен быть запущен сервер требуется хранить в .ini файле и читать с использованием QSettings;
* менять настройки можно непосредственно в приложении и пересохранять .ini файл;
* сервер должен сворачиваться в системный трей и разворачиваться из него для изменения настроек сервера;
* включение/отключение сервера должно быть реализовано как в главном окне приложения, так и через контекстное меню системного трея;
* иконка системного трея и иконка основного окна должна быть сохранена в системе ресурсов .qrc Qt;
* все сообщения, приходящие через сокеты должны сохраняться в базу данных .sqllite;
* пустая структура БД должна быть сохранена в системе ресурсов Qt;
* последние 10 сообщений для новых пользователей должны читаться из базы данных .sqllite.

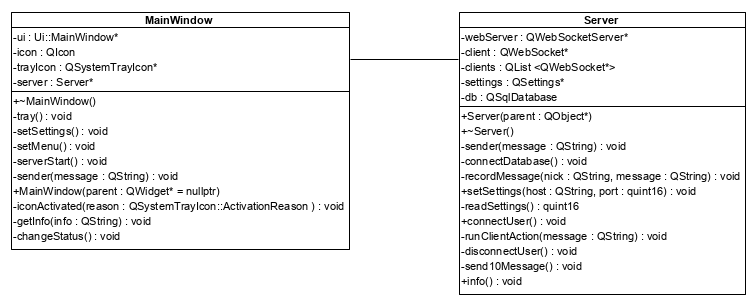
Общее ограничение на лабораторную работу:

* использовать разбитие на файлы основной программы (в main.cpp только функция main.cpp);
* документировать весь код используя аннотации Doxygen.

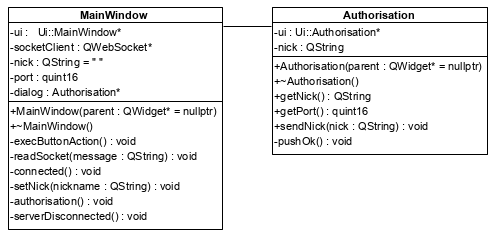
Требования к отчёту:

* цели лабораторной работы;
* задачи лабораторной работы;
* выполняемое задание;
* диаграмма(ы) классов;
* диаграммы деятельности (для методов и/или функций больше 10 строк кода (исключая форматирования и пустые строки));
* исходный код программы с комментариями (2 пробела вместо 4, 10pt, моноширинный шрифт);
* листинг выполнения программы;
* исходный код в виде файлов для запуска на произвольном ПК.

1. Диаграммы



*Рисунок 1 – Диаграмма классов серверной части*



*Рисунок 2 – Диаграмма классов клиентской части*

1. Исходный код

Листинги 2.1-2.6 – исходный код серверной части.

Листинг 2.1 – Код программы

/\*\*

\* @file main.cpp

\* @author Agapova Ekaterina

\*/

#include "mainwindow.h"

#include <QApplication>

int main(int argc, char \*argv[])

{

QApplication a(argc, argv);

MainWindow w;

w.show();

return a.exec();

}

Листинг 2.2 – Заголовочный файл mainwindow.h

/\*\*

\* @file mainwindow.h

\* @brief server

\* @author Agapova Ekaterina

\*/

#ifndef MAINWINDOW\_H

#define MAINWINDOW\_H

#include <QMainWindow>

#include <QWebSocketServer>

#include <QSystemTrayIcon>

#include <QAction>

#include "server.h"

QT\_BEGIN\_NAMESPACE

namespace Ui { class MainWindow; }

QT\_END\_NAMESPACE

class MainWindow : public QMainWindow

{

Q\_OBJECT

private:

Ui::MainWindow \*ui;

QIcon icon;

QSystemTrayIcon\* trayIcon;

Server\* server;

public:

/\*\*

\* @brief Constructor

\* @param parent

\*/

MainWindow(QWidget \*parent = nullptr);

/\*\*

\* @brief Destructor

\*/

~MainWindow();

private:

/\*\*

\* @brief set menu of tray

\*/

void setMenu();

/\*\*

\* @brief Send message for all users

\* @param message

\*/

void sender(QString message);

/\*\*

\* @brief hide elements

\*/

void hideEl();

/\*\*

\* @brief show elements

\*/

void showEl();

private slots:

/\*\*

\* @brief collapse to tray

\*/

void tray();

/\*\*

\* @brief Start or close server

\*/

void serverStart();

/\*\*

\* @brief change status of server

\*/

void changeStatus();

/\*\*

\* @brief iconActivated

\* @param reason

\*/

void iconActivated(QSystemTrayIcon::ActivationReason reason);

/\*\*

\* @brief getInfo

\* @param info

\*/

void getInfo(QString info);

/\*\*

\* @brief set server settings

\*/

void setSettings();

};

#endif // MAINWINDOW\_H

Листинг 2.3 – Заголовочный файл mainwindow.cpp

/\*\*

\* @file mainwindow.cpp

\* @brief server

\* @author Agapova Ekaterina

\*/

#include "mainwindow.h"

#include "ui\_mainwindow.h"

#include <QWidget>

#include <QtWebSockets>

#include <QDebug>

#include <QMessageBox>

MainWindow::MainWindow(QWidget \*parent)

: QMainWindow(parent), ui(new Ui::MainWindow)

{

ui->setupUi(this);

this->hideEl();

this->setWindowTitle("Server");

icon.addFile(":/images/icon.png");

trayIcon = new QSystemTrayIcon(this);

trayIcon->setIcon(icon);

trayIcon->setToolTip("Tray Program" "\n"

"Работа со сворачиванием программы трей");

setMenu();

trayIcon->show();

connect(trayIcon, SIGNAL(activated(QSystemTrayIcon::ActivationReason)),

this, SLOT(iconActivated(QSystemTrayIcon::ActivationReason)));

connect(ui->status, &QComboBox::currentTextChanged,

this, &MainWindow::serverStart);

connect(ui->tray, &QPushButton::clicked, this, &MainWindow::tray);

connect(ui->settings, &QPushButton::clicked,

this, &MainWindow::setSettings);

}

MainWindow::~MainWindow()

{

delete ui;

}

void MainWindow::tray()

{

if(this->isVisible()){

this->hide();

QSystemTrayIcon::MessageIcon icon = QSystemTrayIcon::MessageIcon(QSystemTrayIcon::Information);

trayIcon->showMessage("Tray Program",

"Приложение свернуто в трей. Для того чтобы, развернуть окно приложения, щелкните по иконке приложения в трее",

icon, 2000);

}

}

void MainWindow::iconActivated(QSystemTrayIcon::ActivationReason reason)

{

switch (reason){

case QSystemTrayIcon::Trigger:

if(!this->isVisible()){

this->show();

} else {

this->hide();

}

break;

default:

break;

}

}

void MainWindow::setMenu(){

QMenu\* menu = new QMenu(this);

QAction\* viewWin = new QAction("Развернуть окно", this);

connect(viewWin, SIGNAL(triggered()), this, SLOT(show()));

QAction\* start = new QAction("Запустить сервер", this);

connect(start, &QAction::triggered, this, &MainWindow::changeStatus);

QAction\* closeServ = new QAction("Закрыть сервер", this);

connect(closeServ, &QAction::triggered, this, &MainWindow::changeStatus);

QAction\* quitWin = new QAction("Выход", this);

connect(quitWin, &QAction::triggered, this, &MainWindow::close);

menu->addAction(viewWin);

menu->addAction(start);

menu->addAction(closeServ);

menu->addAction(quitWin);

trayIcon->setContextMenu(menu);

}

void MainWindow::serverStart(){

if (ui->status->currentText() == "ON"){

server = new Server();

connect(server, &Server::info, this, &MainWindow::getInfo);

this->showEl();

} else {

delete server;

this->hideEl();

}

}

void MainWindow::changeStatus()

{

if(ui->status->currentText() == "ON"){

ui->status->setCurrentIndex(0);

} else{

ui->status->setCurrentIndex(1);

}

}

void MainWindow::getInfo(QString info)

{

ui->textEdit->append(info);

}

void MainWindow::setSettings()

{

QString host;

quint16 port;

if(ui->host->displayText() == "")

host = "127.0.0.1";

else

host = ui->host->displayText();

if(ui->port->displayText() == "")

port = 1337;

else

port = ui->port->displayText().toUInt();

server->setSettings(host, port);

}

Листинг 2.4 – Заголовочный файл server.h

/\*\*

\* @file server.h

\* @brief server

\* @author Agapova Ekaterina

\*/

#ifndef SERVER\_H

#define SERVER\_H

#include <QObject>

#include <QWebSocketServer>

#include <QSettings>

#include <QSqlDatabase>

class Server : public QObject

{

Q\_OBJECT

private:

/\*\*

\* @brief connecting

\*/

QWebSocketServer\* webServer;

/\*\*

\* @brief connecting to the client

\*/

QWebSocket\* client;

/\*\*

\* @brief list of clients

\*/

QList <QWebSocket\*> clients;

/\*\*

\* @brief settings

\*/

QSettings \*settings;

/\*\*

\* @brief database

\*/

QSqlDatabase db;

public:

/\*\*

\* @brief Server Constructor

\* @param parent

\*/

explicit Server(QObject \*parent = nullptr);

/\*\*

\* @brief Server destructor

\*/

~Server();

private:

/\*\*

\* @brief Send message for all users

\* @param message

\*/

void sender(QString message);

/\*\*

\* @brief connect database

\*/

void connectDatabase();

/\*\*

\* @brief record message

\* @param nick

\* @param message

\*/

void recordMessage(QString nick, QString message);

public slots:

/\*\*

\* @brief set settings for server

\* @param host

\* @param port

\*/

void setSettings(QString host, quint16 port);

private slots:

/\*\*

\* @brief read settings

\* @return port

\*/

quint16 readSettings();

/\*\*

\* @brief connect User

\*/

void connectUser();

/\*\*

\* @brief message from client

\*/

void runClientAction(QString message);

/\*\*

\* @brief disconnect User

\*/

void disconnectUser();

/\*\*

\* @brief send 10 messages for new user

\*/

void send10Message();

signals:

/\*\*

\* @brief information

\* @param info - message with information

\*/

void info(QString info);

};

#endif // SERVER\_H

Листинг 2.5 – Заголовочный файл server.cpp

/\*\*

\* @file server.cpp

\* @brief server

\* @author Agapova Ekaterina

\*/

#include "server.h"

#include <QtWebSockets>

#include <QDebug>

#include <QMessageBox>

#include <QSqlTableModel>

#include <QSqlQuery>

#include <QFile>

Server::Server(QObject \*parent) : QObject(parent),

webServer (new QWebSocketServer(QStringLiteral("Chat Server"),

QWebSocketServer::NonSecureMode,

this))

{

if (webServer->listen(QHostAddress::Any, readSettings())){

qDebug() << webServer->serverUrl();

QMessageBox::information(NULL,QObject::tr("Информация"), tr("Server started"));

connect (webServer, &QWebSocketServer::newConnection,

this, &Server::connectUser);

} else {

QMessageBox::information(NULL,QObject::tr("Информация"),

tr("Unable to start server"));

}

clients.clear();

db = QSqlDatabase::addDatabase("QSQLITE");

connectDatabase();

}

Server::~Server()

{

clients.clear();

db.close();

emit info("Server close");

webServer->close();

}

void Server::sender(QString message){

foreach (client, clients) {

QWebSocket\* clientSock = (QWebSocket\*) client;

clientSock->sendTextMessage(message);

}

}

void Server::connectDatabase()

{

db.setDatabaseName("db.sqlite");

if (db.open()){

qInfo() << "Database open";

QFile f;

f.setFileName(":/dbStructure.txt");

f.open(QIODevice::ReadOnly);

QSqlQuery query;

if(query.exec(f.readLine()))

qInfo() << "Таблица создана";

f.close();

}

else

qInfo() << "Невозможно открыть БД";

}

void Server::recordMessage(QString nick, QString message)

{

QSqlQuery query;

query.prepare("INSERT INTO messages (id, nickname, message) VALUES (NULL, :nick, :mes)");

query.bindValue(":nick", nick);

query.bindValue(":mes", message);

if(query.exec())

emit info(" Message recorded");

}

void Server::setSettings(QString host, quint16 port)

{

settings = new QSettings(":/setting.ini", QSettings::IniFormat);

settings->beginGroup("Settings");

settings->setValue( "host", host);

settings->setValue("port", port);

settings->endGroup();

QMessageBox::information(NULL,QObject::tr("Информация"),tr("Настройки сервера изменены. Перезапустите сервер"));

}

quint16 Server::readSettings()

{

settings = new QSettings(":/setting.ini", QSettings::IniFormat);

settings->beginGroup("Settings");

QString host = settings->value("host").toString();

quint16 port = settings->value("port").toUInt();

settings->endGroup();

return port;

}

void Server::connectUser()

{

emit info("We have new user!");

client = webServer->nextPendingConnection();

clients.push\_back(client);

sender("1 new user");

send10Message();

connect(client, &QWebSocket::textMessageReceived,

this, &Server::runClientAction);

connect(client, &QWebSocket::disconnected,

this, &Server::disconnectUser);

}

void Server::runClientAction(QString message)

{

QString nick, mes;

QStringList mesList;

mesList = message.split(" ");

nick = mesList[0];

mesList.removeFirst();

mes = mesList.join(" ");

emit info("Get message from " + nick + ": " + mes);

sender(nick + ": " + mes);

recordMessage(nick, mes);

}

void Server::disconnectUser()

{

emit info("We lost user!");

sender("1 user left");

clients.removeOne(client);

client->disconnect();

}

void Server::send10Message()

{

QSqlQuery query;

int k, i=0;

query.exec("SELECT COUNT(\*) FROM messages");

query.next();

k = query.value(0).toInt();

query.exec("SELECT \* FROM messages");

query.next();

if(k > 10){

while (i < (k-10)){

query.next();

i++;

}

}

while(i<k){

client->sendTextMessage(query.value(1).toString() + ": " + query.value(2).toString());

query.next();

i++;

}

}

Листинг 2.6 – Файл res.qrc

<RCC>

<qresource prefix="/images">

<file>icon.png</file>

</qresource>

<qresource prefix="/">

<file>setting.ini</file>

<file>dbStructure.txt</file>

</qresource>

</RCC>

Листинги 2.7-2.11 – исходные коды клиентской части

Листинг 2.7 – Заголовочный файл main.cpp

/\*\*

\* @file main.cpp

\* @author Agapova Ekaterina

\*/

#include "mainwindow.h"

#include <QApplication>

int main(int argc, char \*argv[])

{

QApplication a(argc, argv);

MainWindow w;

w.show();

return a.exec();

}

Листинг 2.8 – Заголовочный файл mainwindow.h

/\*\*

\* @file mainwindow.h

\* @brief client

\* @author Agapova Ekaterina

\*/

#ifndef MAINWINDOW\_H

#define MAINWINDOW\_H

#include <QMainWindow>

#include <QWebSocket>

#include "authorisation.h"

QT\_BEGIN\_NAMESPACE

namespace Ui { class MainWindow; }

QT\_END\_NAMESPACE

class MainWindow : public QMainWindow

{

Q\_OBJECT

private:

Ui::MainWindow \*ui;

QWebSocket \*socketClient;

QString nick = " ";

quint16 port;

Authorisation\* dialog;

public:

/\*\*

\* @brief Client constructor

\* @param parent

\*/

MainWindow(QWidget \*parent = nullptr);

/\*\*

\* @brief Client destructor

\*/

~MainWindow();

private slots:

/\*\*

\* @brief send messages

\*/

void execButtonAction();

/\*\*

\* @brief read Socket

\*/

void readSocket(QString message);

/\*\*

\* @brief connected

\*/

void connected();

/\*\*

\* @brief set nickname of user

\* @param nickname

\*/

void setNick(QString nickname);

/\*\*

\* @brief authorisation

\*/

void authorisation();

/\*\*

\* @brief server disconnected

\*/

void serverDisconnected();

};

#endif // MAINWINDOW\_H

Листинг 2.9 – Заголовочный файл mainwindow.cpp

/\*\*

\* @file mainwindow.h

\* @brief client

\* @author Agapova Ekaterina

\*/

#include "mainwindow.h"

#include "ui\_mainwindow.h"

#include <QMessageBox>

MainWindow::MainWindow(QWidget \*parent)

: QMainWindow(parent), ui(new Ui::MainWindow)

{

ui->setupUi(this);

dialog = new Authorisation();

this->authorisation();

this->nick = dialog->getNick();

this->port = dialog->getPort();

ui->nick->setText(this->nick);

if (nick != " "){

socketClient = new QWebSocket();

QUrl urlServ = QUrl(QString("ws://127.0.0.1:"));

urlServ.setPort(port);

socketClient->open(urlServ);

}

connect(socketClient, &QWebSocket::connected,

this, &MainWindow::connected);

connect(ui->send, &QPushButton::clicked,

this, &MainWindow::execButtonAction);

connect(socketClient, &QWebSocket::textMessageReceived,

this, &MainWindow::readSocket);

connect(dialog, &Authorisation::sendNick,

this, &MainWindow::setNick);

connect(socketClient, &QWebSocket::disconnected,

this, &MainWindow::serverDisconnected);

}

MainWindow::~MainWindow()

{

delete ui;

}

void MainWindow::connected(){

QMessageBox::information(NULL, QObject::tr("Информация"),

tr("Подключение прошло успешно!"));

}

void MainWindow::execButtonAction()

{

if(ui->message->text().toUtf8() == ""){

QMessageBox::information(NULL,QObject::tr("Ошибка"),tr("Введите сообщение"));

return;

} else {

socketClient->sendTextMessage(nick + " " + ui->message->displayText());

ui->message->clear();

}

}

void MainWindow::readSocket(QString message)

{

ui->textEdit->append(message);

}

void MainWindow::setNick(QString nickname){

this->nick = nickname;

ui->nick->setText(nick);

}

void MainWindow::authorisation(){

dialog->exec();

this->close();

}

void MainWindow::serverDisconnected()

{

QMessageBox::information(NULL,QObject::tr("Информация"),tr("Нет подключения к серверу"));

}

Листинг 2.10 – Заголовочный файл authorisation.h

/\*\*

\* @file authorisation.h

\* @brief authorisation

\* @author Agapova Ekaterina

\*/

#ifndef AUTHORISATION\_H

#define AUTHORISATION\_H

#include <QDialog>

namespace Ui {class Authorisation;}

class Authorisation : public QDialog

{

Q\_OBJECT

public:

/\*\*

\* @brief Constructor

\* @param parent

\*/

explicit Authorisation(QWidget \*parent = nullptr);

/\*\*

\* @brief Destructor

\*/

~Authorisation();

/\*\*

\* @brief get Nickname

\* @return nickname

\*/

QString getNick();

/\*\*

\* @brief get Port

\* @return port

\*/

quint16 getPort();

private:

Ui::Authorisation \*ui;

QString nick;

private slots:

/\*\*

\* @brief push button "Ok"

\*/

void pushOk();

signals:

/\*\*

\* @brief send Nickname

\* @param nick

\*/

void sendNick(QString nick);

};

#endif // AUTHORISATION\_H

Листинг 2.11 – Заголовочный файл mainwindow.cpp

/\*\*

\* @file authorisation.cpp

\* @brief authorisation

\* @author Agapova Ekaterina

\*/

#include "authorisation.h"

#include "ui\_authorisation.h"

#include <QMessageBox>

Authorisation::Authorisation(QWidget \*parent) :

QDialog(parent), ui(new Ui::Authorisation)

{

ui->setupUi(this);

connect(ui->ok, &QPushButton::clicked,

this, &Authorisation::pushOk);

}

Authorisation::~Authorisation()

{

delete ui;

}

void Authorisation::pushOk(){

if(ui->nick->displayText() == "" || ui->nick->displayText() == " "){

QMessageBox::information(NULL,QObject::tr("Ошибка"), tr("Введите никнейм!"));

return;

}

nick = ui->nick->displayText();

this->close();

emit sendNick(ui->nick->displayText());

}

QString Authorisation::getNick()

{

return nick;

}

quint16 Authorisation::getPort()

{

quint16 port = 1337;

if (ui->port->displayText() == "")

return port;

else

return ui->port->displayText().toUInt();

}

1. ЛИСТИНГ ВЫполнения программы

На рисунках 3-5 представлены интерфейсы работы программы.

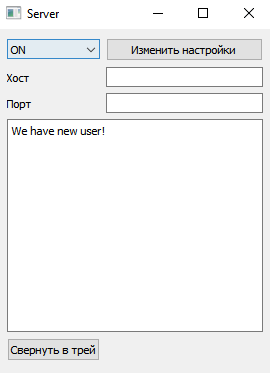


Рисунок 3 – Окно сервера

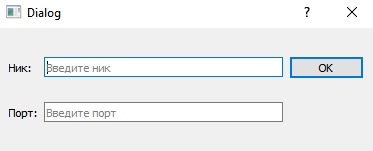


Рисунок 4 – Авторизация пользователя

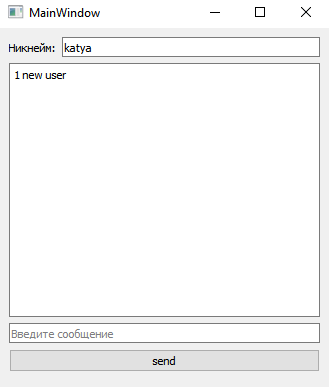


Рисунок 5 – Окно пользователя

1. Вывод

В ходе выполнения лабораторной работы была написана программа, которая реализует работу простейшего чата на вебсокетах. Сервер чата сворачивается в трей, настройки сохранены в файле .ini. Все необходимые файлы сохранены в ресурсах. Для пользователя реализовано отдельное окно авторизации.