

1. Цель и задачи

Цель: Познакомится с основами работы QWebSockets, QSettings, и системой ресурсов в приложениях Qt.

Задачи:

* изучить основы работы с QWebSocket соединениями;
* изучить работу с настройками приложения QSettings;
* изучить работу с системой ресурсов Qt;
* создать более универсальный сервер чата.

1. Задание

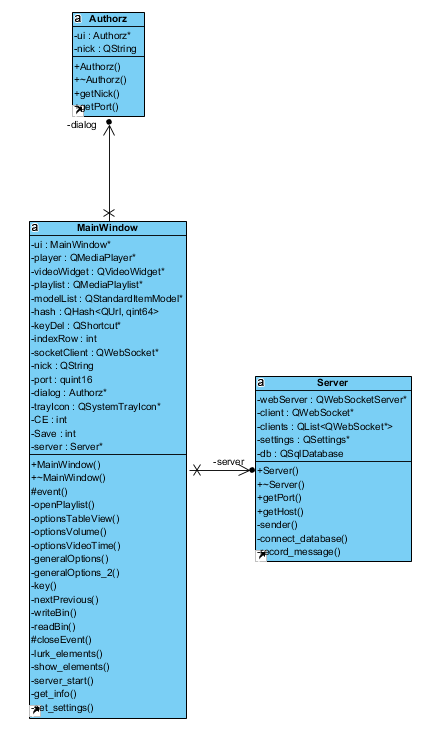
Общее ограничение на лабораторную работу:

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

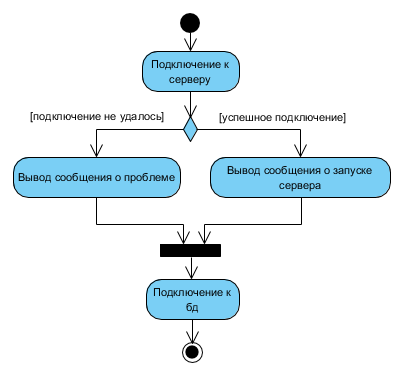
Требуется доработка переработка предыдущей лабораторной работы. Требуется расширить/заменить следующие функции сервера:

* 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.

1. ДИАГРАММЫ



*Рисунок 1 – диаграмма классов*

**

*Рисунок 3 – диаграмма деятельности функции подключения*

1. КОд программы

Основной выполняемый файл клиента main.cpp:

#include "menu.h"

#include <QApplication>

/\*\*

\* @author Oreshnikov Egor

\*/

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

{

QApplication a(argc, argv);

menu w;

w.show();

return a.exec();

}

Файл клиента menu.cpp:

#include "menu.h"

#include "ui\_menu.h"

#include <QMessageBox>

/\*\*

\* @file menu.cpp

\* @author Oreshnikov Egor

\*/

Authorz::Authorz(QWidget \*parent) :

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

{

ui->setupUi(this);

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

this, &Authorz::on\_Start\_clicked);

}

Authorz::~Authorz(){

delete ui;

}

QString Authorz::getNick(){

return nick;

}

quint16 Authorz::getPort(){

quint16 port = 1113;

if (ui->port->displayText() == "") { return port; }

else { return ui->port->displayText().toUInt(); }

}

void Authorz::on\_Start\_clicked()

{

int wordCount = nick.split(" ").count();

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

ui->nick->displayText() == " " || wordCount > 1){

QMessageBox::information(NULL,QObject::tr("Ошибка"), tr("Введите Имя например - Анна!"));

return; }

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

this->close();

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

}

Файл клиента client.cpp:

#include "client.h"

#include "menu.h"

#include "ui\_client.h"

#include <QMessageBox>

#include <QDebug>

/\*\*

\* @file client.cpp

\* @author Oreshnikov Egor

\*/

MainWindow::MainWindow(QWidget \*parent)

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

{

ui->setupUi(this);

dialog = new Authorz();

this->authorz();

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

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

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

if (nick != " "){

socketClient = new QWebSocket();

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

urlServer.setPort(port);

socketClient->open(urlServer);

}

connect(socketClient, &QWebSocket::connected,

this, &MainWindow::connected);

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

this, &MainWindow::execButtonAction);

connect(socketClient, &QWebSocket::textMessageReceived,

this, &MainWindow::read\_Socket);

connect(dialog, &Authorz::Start,

this, &MainWindow::set\_Name);

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::read\_Socket(QString message)

{

ui->textEdit->append(message);

}

void MainWindow::set\_Name(QString nickname){

this->nick = nickname;

ui->Name->setText(nick);

}

void MainWindow::authorz(){

dialog->exec();

this->close();

}

void MainWindow::serverDisconnected()

{

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

tr("Нет подключения к серверу"));

}

Файл клиента menu.h:

/\*\*

\* @file menu.h

\* @author Oreshnikov Egor

\*/

#ifndef AUTHORIZATION\_H

#define AUTHORIZATION\_H

#include <QDialog>

namespace Ui {class Authorz;}

class Authorz : public QDialog

{

Q\_OBJECT

public:

explicit Authorz(QWidget \*parent = nullptr);

~Authorz();

QString getNick();

quint16 getPort();

private:

Ui::Authorz \*ui;

QString nick;

private slots:

void on\_Start\_clicked();

signals:

void Start(QString nick);

};

#endif // AUTHORIZATION\_H

Файл клиента client.h:

#ifndef MAINWINDOW\_H

#define MAINWINDOW\_H

#include <QMainWindow>

#include <QWebSocket>

#include "menu.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;

Authorz\* dialog;

public:

MainWindow(QWidget \*parent = nullptr);

~MainWindow();

private slots:

void execButtonAction();

void read\_Socket(QString message);

void connected();

void set\_Name(QString name);

void authorz();

void serverDisconnected();

};

#endif // MAINWINDOW\_H

Основной выполняемый файл сервера main.cpp:

#include "mainwindow.h"

#include <QApplication>

/\*\*

\* @file main.cpp

\* @author Oreshnikov Egor

\*/

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

{

QApplication a(argc, argv);

MainWindow w;

w.show();

return a.exec();

}

Файл сервера server.cpp:

#include "server.h"

#include <QtWebSockets>

#include <QDebug>

#include <QMessageBox>

#include <QSqlTableModel>

#include <QSqlQuery>

#include <QFile>

/\*\*

\* @file server.cpp

\* @author Oreshnikov Egor

\*/

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

webServer (new QWebSocketServer(QStringLiteral("Сервер"),

QWebSocketServer::NonSecureMode,

this))

{

if (webServer->listen(QHostAddress::Any, read\_settings())){

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

QMessageBox::information(NULL,QObject::tr("Information"), tr("Сервер запущен"));

connect (webServer, &QWebSocketServer::newConnection,

this, &Server::connect\_user);

} else {

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

tr("Сервер не запущен, попытайтесь еще "

"или измените настройки подключения"));

}

clients.clear();

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

connect\_database();

}

Server::~Server(){

clients.clear();

db.close();

emit info("Server close");

webServer->close();

}

void Server::sender(QString message){

foreach (client, clients) {

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

clientSocket->sendTextMessage(message);

}

}

void Server::connect\_database()

{

db.setDatabaseName("db.sqlite");

if (db.open()){

qInfo() << "База данных открыта";

QFile f;

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

if(!f.open(QIODevice::ReadOnly))

{

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

tr("Файл структура базы данных не найден."));

return;

}

QSqlQuery query;

if(query.exec(f.readLine())) {qInfo() << "Таблица создана";}

f.close();

}

else {

qInfo() << "База данных не найдена.";}

}

void Server::record\_message(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("Сообщение записано");}

}

void Server::set\_settings(QString host, quint16 port)

{

QFile f;

f.setFileName("Settings.ini");

if(f.open(QIODevice::ReadWrite)){

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

settings->beginGroup("Settings");

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

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

settings->endGroup();

QMessageBox::information(NULL,QObject::tr("Information"), tr("Настройки изменены, измените настройки."));

}

f.close();

}

quint16 Server::read\_settings()

{

QFile f;

f.setFileName("Settings.ini");

if(f.exists()){

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

settings->beginGroup("Settings");

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

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

settings->endGroup();

return port;

} else{

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

settings->beginGroup("Settings");

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

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

settings->endGroup();

return port;

}

}

void Server::connect\_user()

{

emit info("Подключился новый пользователь!");

client = webServer->nextPendingConnection();

clients.push\_back(client);

Message();

connect(client, &QWebSocket::textMessageReceived,

this, &Server::run\_client\_action);

connect(client, &QWebSocket::disconnected,

this, &Server::disconnectUser);

}

void Server::run\_client\_action(QString message)

{

QString nick, mes;

QStringList mesList;

mesList = message.split(" ");

nick = mesList[0];

mesList[0].clear();

mes = mesList.join(" ");

emit info("Получить сообщение от " + nick + ": " + mes);

sender(nick + ": " + mes);

record\_message(nick, mes);

}

void Server::disconnectUser() {

emit info("Пользователь вышел!");

clients.removeOne(client);

client->disconnect();

}

void Server::Message() {

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++;}

}

QString Server::getPort()

{

settings->beginGroup("Settings");

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

settings->endGroup();

return port;

}

QString Server::getHost()

{

settings->beginGroup("Settings");

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

settings->endGroup();

return host;

}

Файл сервера server.h:

#ifndef SERVER\_H

#define SERVER\_H

#include <QObject>

#include <QWebSocketServer>

#include <QSettings>

#include <QSqlDatabase>

/\*\*

\* @file server.h

\* @author Oreshnikov Egor

\*/

class Server : public QObject

{

Q\_OBJECT

private:

QWebSocketServer\* webServer;

QWebSocket\* client;

QList <QWebSocket\*> clients;

QSettings \*settings;

QSqlDatabase db;

public:

explicit Server(QObject \*parent = nullptr);

~Server();

QString getPort();

QString getHost();

private:

void sender(QString message);

void connect\_database();

void record\_message(QString nick, QString message);

public slots:

void set\_settings(QString host, quint16 port);

private slots:

quint16 read\_settings();

void connect\_user();

void run\_client\_action(QString message);

void disconnectUser();

void Message();

signals:

void info(QString info);

void onoff\_clicked();

};

#endif // SERVER\_H

Файл сервера mainwindow.h:

#ifndef MAINWINDOW\_H

#define MAINWINDOW\_H

#include <QMainWindow>

#include <QMainWindow>

#include <QCloseEvent>

#include <QWebSocketServer>

#include "server.h"

#include <QSystemTrayIcon>

#include <QAction>

/\*\*

\* @file mainwindow.h

\* @author Oreshnikov Egor

\*/

QT\_BEGIN\_NAMESPACE

namespace Ui { class MainWindow; }

QT\_END\_NAMESPACE

class MainWindow : public QMainWindow

{

Q\_OBJECT

public:

explicit MainWindow(QWidget \*parent = 0);

~MainWindow();

protected:

void closeEvent(QCloseEvent \* event);

private slots:

void iconActivated(QSystemTrayIcon::ActivationReason reason);

void on\_callapse\_clicked();

void exit();

void on\_onoff\_clicked();

private:

Ui::MainWindow \* ui;

QSystemTrayIcon \* trayIcon;

int CE = 0;

int Save = 0;

Server\* server;

void lurk\_elements();

void show\_elements();

void server\_start();

void get\_info(QString info);

void set\_settings();

signals:

void callapse();

void onoff\_clicked();

};

#endif // MAINWINDOW\_H

Файл сервера mainwindow.cpp:

#include "mainwindow.h"

#include "ui\_mainwindow.h"

#include "QMenu"

#include "QStyle"

#include <QWidget>

#include <QtWebSockets>

#include <QDebug>

#include <QMessageBox>

/\*\*

\* @file mainwindow.cpp

\* @author Oreshnikov Egor

\*/

MainWindow::MainWindow(QWidget \*parent)

: QMainWindow(parent)

, ui(new Ui::MainWindow)

{

ui->setupUi(this);

if(Save == 0)

{

ui->onoff->setStyleSheet(QString::fromUtf8("background-color: rgb(250, 50, 50);"));

}

ui->onoff->setText(" ");

this->setWindowTitle("Server");

trayIcon = new QSystemTrayIcon(this);

trayIcon->setIcon(this->style()->standardIcon(QStyle::SP\_ComputerIcon));

trayIcon->setToolTip("Server" "\n"

"Работа со сворачиванием программы трей");

QMenu \* menu = new QMenu(this);

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

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

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

connect(quitAction, SIGNAL(triggered()), this, SLOT(close()));

connect(this, SIGNAL(onoff\_clicked()),this,SLOT(on\_onoff\_clicked()));

menu->addAction(viewWindow);

menu->addAction(quitAction);

trayIcon->setContextMenu(menu);

trayIcon->show();

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

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

connect(this, SIGNAL(callapse()),this,SLOT(on\_callapse\_clicked()));

}

MainWindow::~MainWindow()

{

delete ui;

}

void MainWindow::closeEvent(QCloseEvent \* event)

{

if(this->isVisible() && CE == 1){

event->ignore();

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);

exit();

}

}

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

{

switch (reason){

case QSystemTrayIcon::Trigger:

if(CE == 1){

if(!this->isVisible()){

this->show();

CE=0;

} else {

this->hide();

}

}

break;

default:

break;

}

}

void MainWindow::exit()

{

close();

}

void MainWindow::on\_callapse\_clicked()

{

CE = 1;

exit();

}

void MainWindow::lurk\_elements()

{

ui->host->hide();

ui->port->hide();

ui->SaveSetting->hide();

ui->host\_label->hide();

ui->port\_label->hide();

}

void MainWindow::show\_elements()

{

ui->host->show();

ui->port->show();

ui->SaveSetting->show();

ui->host\_label->show();

ui->port\_label->show();

}

void MainWindow::server\_start(){

if (Save == 1){

server = new Server();

connect(server, &Server::info,this, &MainWindow::get\_info);

this->show\_elements();

ui->host->setText(server->getHost());

ui->port->setText(server->getPort());

} else {

delete server;

this->lurk\_elements();

}

}

void MainWindow::get\_info(QString info){

ui->textEdit->append(info);

}

void MainWindow::set\_settings()

{

QString host;

quint16 port;

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

host = "127.0.0.1";

} else {host = ui->host->displayText();}

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

port = 1113;

} else {port = ui->port->displayText().toUInt();}

server->set\_settings(host, port);

}

void MainWindow::on\_onoff\_clicked()

{

if (Save == 0){

ui->onoff->setStyleSheet(QString::fromUtf8("background-color: rgb(102, 250, 0);"));

Save = 1;

}

else{

Save = 0;

ui->onoff->setStyleSheet(QString::fromUtf8("background-color: rgb(250, 50, 50);"));

}

server\_start();

}

void MainWindow::on\_SaveSetting\_clicked()

{

QString host;

quint16 port;

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

host = "127.0.0.1";

} else {host = ui->host->displayText();}

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

port = 1113;

} else {port = ui->port->displayText().toUInt();}

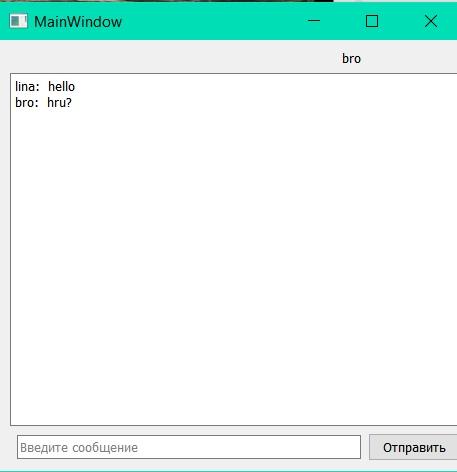
server->set\_settings(host, port);

}

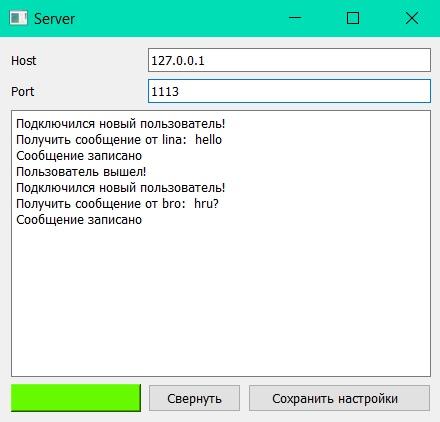
В качестве результата работы программы представлен скриншот выходного файла на рисунках 3-5.



*Рисунок 3 – результат работы программы*

**

*Рисунок 4 – результат работы программы*

**

*Рисунок 5 – результат работы программы*

# ВЫВОД

В процессе выполнения лабораторной работы были изучены основы работы с REST запросами.