ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«РОССИЙСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ НЕФТИ И ГАЗА
(НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ)
ИМЕНИ И.М. ГУБКИНА»

ФАКУЛЬТЕТ КОМПЛЕКСНОЙ БЕЗОПАСНОСТИ ТЭК
КАФЕДРА БЕЗОПАСНОСТИ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ

Лабораторная работа №10

по дисциплине «Специализированные языки и технологии программирования»

на тему «Основы работы с сетью с использованием QtNetwork»

Выполнил студент:
группы КА-22-06
Воронин Алексей Дмитриевич

Преподаватель:
Греков Владимир Сергеевич

Москва, 2025

Оглавление

Цель работы

Цель данной лабораторной работы заключается в овладении основами работы с сетью в рамках Qt, используя модуль QtNetwork. Студенты научатся создавать сетевые приложения, которые могут отправлять запросы на серверы, принимать ответы и обрабатывать их. Особое внимание будет уделено пониманию асинхронного программирования и обработки событий сети, что является ключевым для разработки отзывчивых приложений, способных работать в сетевой среде.

Задание

Разработать простое клиент-серверное приложение для обмена текстовыми сообщениями в реальном времени. Клиентская часть должна позволять пользователю отправлять сообщения на сервер, который, в свою очередь, рассылает их всем подключенным клиентам.

ЧАСТЬ 1 - Детальные инструкции к выполнению

Создайте проект Server с системой сборки qmake.

Для корректной работы в Server.pro нужно добавить определенную конфигурацию:

```
QT = core
QT += core network

CONFIG += c++17 cmdline

# You can make your code fail to compile if it uses deprecated APIs.
# In order to do so, uncomment the following line.
#DEFINES += QT_DISABLE_DEPRECATED_BEFORE=0x060000    # disables all the APIs deprecated before Qt 6.0.0

SOURCES += \
        main.cpp \
        server.cpp

# Default rules for deployment.
qnx: target.path = /tmp/$${TARGET}/bin
else: unix:!android: target.path = /opt/$${TARGET}/bin
!isEmpty(target.path): INSTALLS += target

HEADERS += \
   server.h
```

Добавьте в файл server.cpp следующее:

```
#include <server.h>

Server::Server()
{
   nextBlockSize = 0;

   if (this->listen(QHostAddress::Any, 2323))
   {
      qDebug() << "Start server...";
   }
```

```
        else
        {
            qDebug() << "Error start server...";
        }
    }

    void Server::incomingConnection(qintptr socketDescriptor)
    {
        socket = new QTcpSocket;
        socket->setSocketDescriptor(socketDescriptor);
        connect(socket, &QTcpSocket::readyRead, this, &Server::slotReadyRead);
        connect(socket, &QTcpSocket::disconnected, socket,
&Server::deleteLater);
        Sockets.push_back(socket);
        qDebug() << "Client is connected..." << socketDescriptor;
    }

    void Server::slotReadyRead()
    {
        socket = (QTcpSocket*)sender();
        QDataStream in(socket);
        in.setVersion(QDataStream::Qt_6_2);

        if (in.status() == QDataStream::Ok)
        {
            for (;;)
            {
                if (nextBlockSize == 0)
                {
                    qDebug() << "nextBlockSize = 0";

                    if (socket->bytesAvailable() < 2)
                    {
                        qDebug() << "Data < 2 bytes, break";
                        break;
                    }
```

```
            in >> nextBlockSize;
            qDebug() << "nextBlockSize = " << nextBlockSize;
        }

        if (socket->bytesAvailable() < nextBlockSize)
        {
            qDebug() << "Data is not full, break";
            break;
        }

        QString mssng;
        in >> mssng;
        nextBlockSize = 0;
        qDebug() << mssng;
        SendToClient(mssng);
        break;
      }
   }

   else
   {
      qDebug() << "DataStream eror...";
   }
}

void Server::SendToClient(QString mssng)
{
   Data.clear();
   QDataStream out(&Data, QIODevice::WriteOnly);
   out.setVersion(QDataStream::Qt_6_2);
   out << quint16(0) << mssng;
   out.device()->seek(0);
   out << quint16(Data.size() - sizeof(quint16));
   for (int number = 0; number < Sockets.size(); number++)
   {
      Sockets[number]->write(Data);
   }
```

```
}
```
Добавьте в заголовочный файл server.h следующее:
```
#ifndef SERVER_H
#define SERVER_H

#include <QTcpServer>
#include <QTcpSocket>
#include <QVector>

class Server : public QTcpServer
{
   Q_OBJECT

public:
   Server();
   QTcpSocket *socket;

private:
   QVector <QTcpSocket*> Sockets;
   QByteArray Data;
   quint16 nextBlockSize;

   void SendToClient(QString mssng);

public slots:
   void incomingConnection(qintptr socketDescriptor);
   void slotReadyRead();

};

#endif // SERVER_H
```
Добавьте в main.cpp следующее:
```
#include <QCoreApplication>
#include <server.h>

int main(int argc, char *argv[])
{
```

```
    QCoreApplication a(argc, argv);
    Server server;

    return a.exec();
}
```

Шаг 2. Разработка клиентского приложения

Создайте проект Client с системой сборки qmake.

Для корректной работы в Client.pro нужно добавить определенную конфигурацию:

```
QT += core gui network

greaterThan(QT_MAJOR_VERSION, 4): QT += widgets

CONFIG += c++17

# You can make your code fail to compile if it uses deprecated APIs.
# In order to do so, uncomment the following line.
#DEFINES += QT_DISABLE_DEPRECATED_BEFORE=0x060000    # disables all the APIs deprecated before Qt 6.0.0

SOURCES += \
    main.cpp \
    mainwindow.cpp

HEADERS += \
    mainwindow.h

FORMS += \
    mainwindow.ui

# Default rules for deployment.
qnx: target.path = /tmp/$${TARGET}/bin
else: unix:!android: target.path = /opt/$${TARGET}/bin
!isEmpty(target.path): INSTALLS += target
```

В файл формы mainwindow.ui добавим следующее:

```
<?xml version="1.0" encoding="UTF-8"?>
<ui version="4.0">
 <class>MainWindow</class>
 <widget class="QMainWindow" name="MainWindow">
  <property name="geometry">
   <rect>
    <x>0</x>
    <y>0</y>
    <width>492</width>
    <height>421</height>
   </rect>
  </property>
  <property name="windowTitle">
   <string>Чат</string>
  </property>
  <widget class="QWidget" name="centralwidget">
   <widget class="QPushButton" name="pushButton">
    <property name="geometry">
     <rect>
      <x>310</x>
      <y>40</y>
      <width>161</width>
      <height>32</height>
     </rect>
    </property>
    <property name="text">
     <string>Подключиться к серверу</string>
    </property>
   </widget>
   <widget class="QPushButton" name="pushButton_2">
    <property name="geometry">
     <rect>
      <x>280</x>
      <y>290</y>
      <width>111</width>
      <height>31</height>
     </rect>
```

```
 </property>
 <property name="text">
  <string>Отправить</string>
 </property>
</widget>
<widget class="QTextBrowser" name="textBrowser">
 <property name="geometry">
  <rect>
   <x>10</x>
   <y>90</y>
   <width>381</width>
   <height>181</height>
  </rect>
 </property>
</widget>
<widget class="QLineEdit" name="lineEdit">
 <property name="geometry">
  <rect>
   <x>10</x>
   <y>290</y>
   <width>261</width>
   <height>31</height>
  </rect>
 </property>
 <property name="placeholderText">
  <string>Введите сообщение...</string>
 </property>
</widget>
<widget class="QLineEdit" name="lineEdit_2">
 <property name="geometry">
  <rect>
   <x>150</x>
   <y>30</y>
   <width>151</width>
   <height>21</height>
  </rect>
 </property>
```

```
 <property name="placeholderText">
  <string>127.0.0.1</string>
 </property>
</widget>
<widget class="QLineEdit" name="lineEdit_3">
 <property name="geometry">
  <rect>
   <x>150</x>
   <y>60</y>
   <width>151</width>
   <height>21</height>
  </rect>
 </property>
 <property name="placeholderText">
  <string>Ваше имя</string>
 </property>
</widget>
<widget class="QLabel" name="label">
 <property name="geometry">
  <rect>
   <x>50</x>
   <y>30</y>
   <width>58</width>
   <height>16</height>
  </rect>
 </property>
 <property name="text">
  <string>IP адрес</string>
 </property>
</widget>
<widget class="QLabel" name="label_2">
 <property name="geometry">
  <rect>
   <x>30</x>
   <y>60</y>
   <width>151</width>
   <height>20</height>
```

```
   </rect>
   </property>
   <property name="text">
    <string>Имя пользователя</string>
   </property>
  </widget>
  <widget class="QPushButton" name="pushButton_3">
   <property name="geometry">
    <rect>
     <x>280</x>
     <y>330</y>
     <width>111</width>
     <height>31</height>
    </rect>
   </property>
   <property name="text">
    <string>Очистить чат</string>
   </property>
  </widget>
 </widget>
 <widget class="QMenuBar" name="menubar">
  <property name="geometry">
   <rect>
    <x>0</x>
    <y>0</y>
    <width>492</width>
    <height>22</height>
   </rect>
  </property>
 </widget>
 <widget class="QStatusBar" name="statusbar"/>
</widget>
<resources/>
<connections/>
</ui>
```

В заголовочный файл mainwindow.h добавим следующее:

```
#ifndef MAINWINDOW_H
#define MAINWINDOW_H

#include <QMainWindow>
#include <QTcpSocket>

QT_BEGIN_NAMESPACE
namespace Ui { class MainWindow; }
QT_END_NAMESPACE

class MainWindow : public QMainWindow
{
    Q_OBJECT

public:
    MainWindow(QWidget *parent = nullptr);
    ~MainWindow();

private slots:
    void on_pushButton_clicked();

    void on_pushButton_2_clicked();

    void on_lineEdit_returnPressed();

    void on_lineEdit_2_returnPressed();

    void on_pushButton_3_clicked();

private:
    Ui::MainWindow *ui;
    QTcpSocket *socket;
    QString line_ip;
    QString mssg_buf;
    QString username;
    QByteArray Data;
    quint16 nextBlockSize;
```

```
    void SendToServer(QString mssng);
    void SendToServerConnect(QString mssng);

public slots:
    void slotReadyRead();

};
#endif // MAINWINDOW_H
```

В исходный файл mainwindow.cpp добавить следующее:

```
#include "mainwindow.h"
#include "ui_mainwindow.h"
#include <QMessageBox>

MainWindow::MainWindow(QWidget *parent)
    : QMainWindow(parent)
    , ui(new Ui::MainWindow)
{
    ui->setupUi(this);
    socket = new QTcpSocket(this);
    connect(socket, &QTcpSocket::readyRead, this,
&MainWindow::slotReadyRead);
    connect(socket, &QTcpSocket::disconnected, this,
&MainWindow::deleteLater);
    nextBlockSize = 0;
}

MainWindow::~MainWindow()
{
    delete ui;
}

void MainWindow::on_pushButton_clicked()
{
    if (ui->lineEdit_3->text() != "")
    {
```

```
        username = ui->lineEdit_3->text();
        socket->connectToHost(ui->lineEdit_2->text(), 2323);
        if (socket->waitForConnected(3000))
        {
            SendToServerConnect(username);

            QMessageBox box;
            box.setText("ПОДКЛЮЧЕНО");
            box.exec();
        }
        else
        {
            QMessageBox::warning(0, "ОШИБКА", "Неверный IP адрес");
        }
    }
    else
    {
        QMessageBox::warning(0, "ОШИБКА", "Введите имя пользователя");
    }
}

void MainWindow::SendToServer(QString mssng)
{
    Data.clear();
    QDataStream out(&Data, QIODevice::WriteOnly);
    out.setVersion(QDataStream::Qt_6_2);
    out << quint16(0) << username + ":\n-" + mssng;
    out.device()->seek(0);
    out << quint16(Data.size() - sizeof(quint16));
    socket->write(Data);
    mssg_buf = ui->lineEdit->text();
    ui->lineEdit->clear();
}

void MainWindow::SendToServerConnect(QString mssng)
{
    Data.clear();
```

```
    QDataStream out(&Data, QIODevice::WriteOnly);
    out.setVersion(QDataStream::Qt_6_2);
    out << quint16(0) << mssng;
    out.device()->seek(0);
    out << quint16(Data.size() - sizeof(quint16));
    socket->write(Data);
}

void MainWindow::slotReadyRead()
{
    QDataStream in(socket);
    in.setVersion(QDataStream::Qt_6_2);

    if (in.status() == QDataStream::Ok)
    {
        for (;;)
        {
            if (nextBlockSize == 0)
            {
                if (socket->bytesAvailable() < 2)
                {
                    break;
                }
                in >> nextBlockSize;
            }

            if (socket->bytesAvailable() < nextBlockSize)
            {
                break;
            }

            QString mssng;
            in >> mssng;
            nextBlockSize = 0;

            if (mssng == username + ":\n-" + mssg_buf)
            {
```

```
            ui->textBrowser->setAlignment(Qt::AlignRight);
            ui->textBrowser->append("Вы:");
            ui->textBrowser->append("-" + mssg_buf + "\n");
        }
        else if (mssng == username)
        {
            ui->textBrowser->setAlignment(Qt::AlignCenter);
            ui->textBrowser->append("Вы подключились\n");
        }
        else if (mssng != username && mssng.indexOf(":\n") >= 0)
        {
            ui->textBrowser->setAlignment(Qt::AlignLeft);
            ui->textBrowser->append(mssng + "\n");
        }
        else
        {
            ui->textBrowser->setAlignment(Qt::AlignCenter);
            ui->textBrowser->append(mssng + " подключился\n");
        }
      }
   }
   else
   {
      ui->textBrowser->append("Ошибка чтения...");
   }
}

void MainWindow::on_pushButton_2_clicked()
{
   SendToServer(ui->lineEdit->text());
}

void MainWindow::on_lineEdit_returnPressed()
{
   SendToServer(ui->lineEdit->text());
}
```

```
void MainWindow::on_lineEdit_2_returnPressed()
{
   if (ui->lineEdit_3->text() != "")
   {
      username = ui->lineEdit_3->text();
      socket->connectToHost(ui->lineEdit_2->text(), 2323);
      if (socket->waitForConnected(3000))
      {
         SendToServerConnect(username);

         QMessageBox box;
         box.setText("ПОДКЛЮЧЕНО");
         box.exec();
      }
      else
      {
         QMessageBox::warning(0, "ОШИБКА", "Неверный IP адрес");
      }
   }
   else
   {
      QMessageBox::warning(0, "ОШИБКА", "Введите имя пользователя");
   }
}

void MainWindow::on_pushButton_3_clicked()
{
   ui->textBrowser->clear();
}
```

Тестирование приложения

Для начала запустим серверную часть.

Затем запустим два экземпляра клиентского приложения. Подключимся к серверу и отправим сообщения в чат.

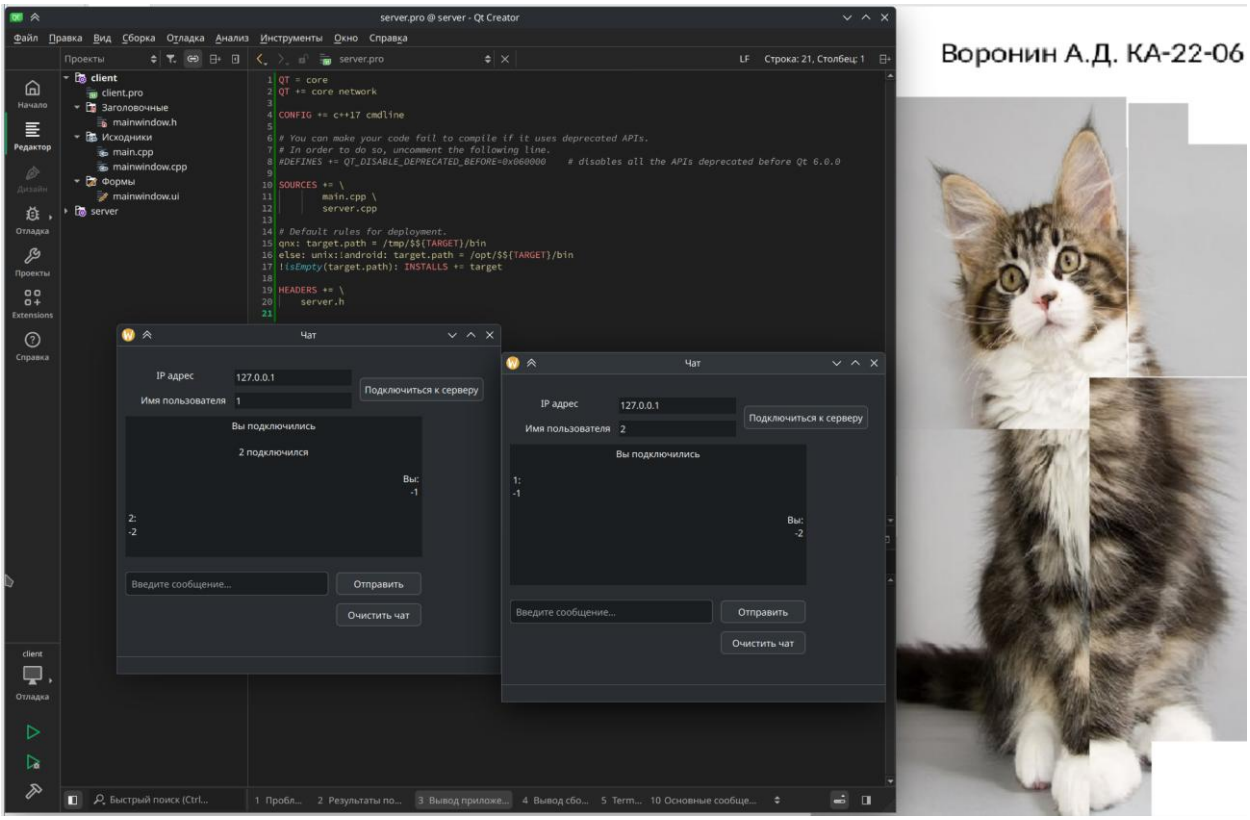

Рисунок 1 – Тестирование приложения

Самостоятельная работа

Задание для самостоятельной работы

- Создайте или доработайте GUI-приложение для администрирования сервера.
- Модифицируйте чат так, чтобы можно было отправлять личные сообщения конкретному пользователю.
- Доработайте клиентское приложения по своему усмотрению (например, добавьте кнопкку для дисконнекта и т.д).

Результат выполнения можно увидеть на скриншоте ниже.

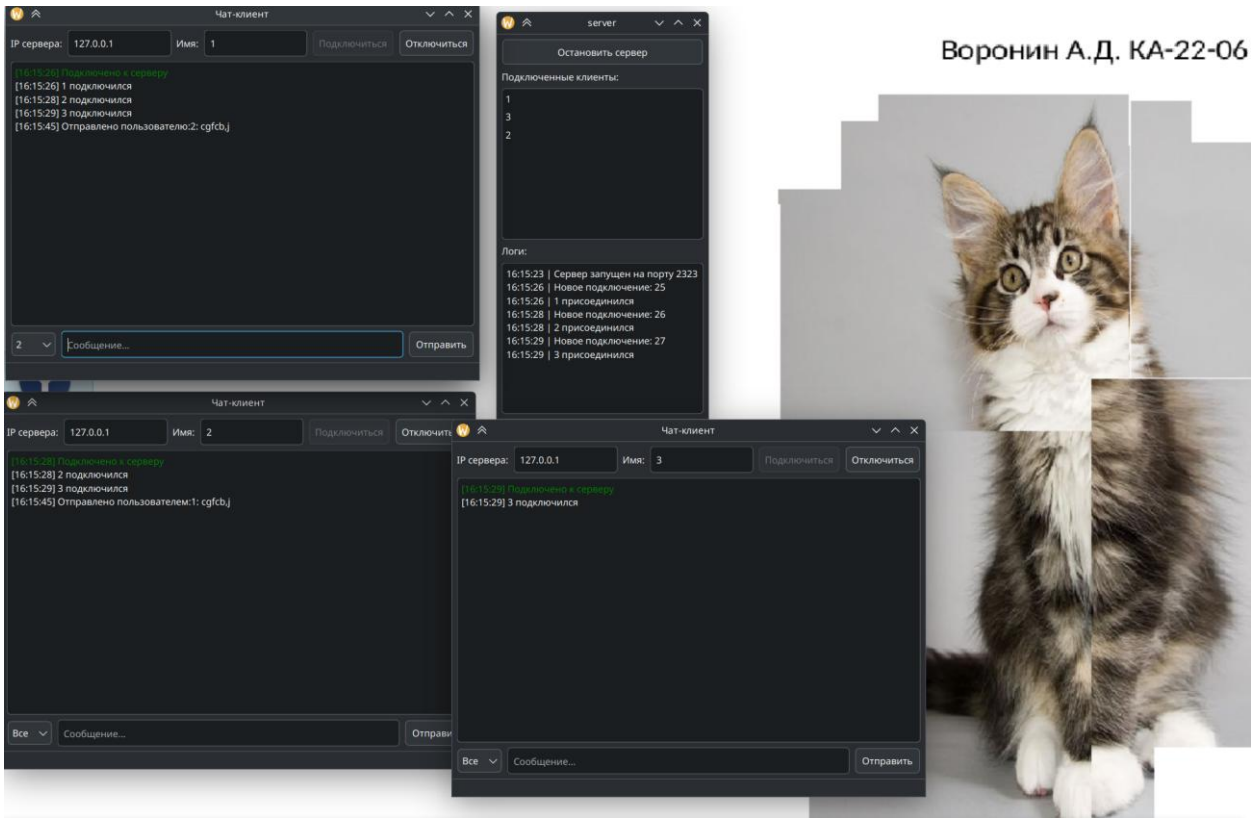

Рисунок 2 – Отправление сообщения пользователем 2 пользователю 1

ЗАКЛЮЧЕНИЕ

Таким образом, задания к данной лабораторной работы были выполнены в полном объеме.

Контрольные вопросы

1) Почему в коде используется quint16 для размера блока данных?

Использование quint16 для размера блока данных обусловлено необходимостью передачи сообщений в сетевом взаимодействии с чётко определённой структурой. quint16 (16-битное беззнаковое целое число) позволяет хранить размер блока данных до 65 535 байт, чего достаточно для текстовых сообщений в чате. Этот тип данных добавляется в начало каждого сообщения, чтобы получатель (клиент или сервер) мог точно определить, сколько байт нужно прочитать для получения полного сообщения. Это предотвращает ошибки, связанные с фрагментацией данных в потоке TCP, обеспечивая корректное разделение сообщений.

2) Как работает механизм сигналов и слотов в Qt при работе с сетью?

Механизм сигналов и слотов в Qt используется для асинхронной обработки событий, включая сетевые. Например, в данном приложении класс QTcpSocket испускает сигнал readyRead(), когда в сокете появляются новые данные для чтения, и сигнал disconnected(), когда соединение разрывается. Эти сигналы подключаются к слотам (slotReadyRead, deleteLater и др.), которые выполняют соответствующую логику обработки, например, чтение данных или удаление сокета. Это позволяет обрабатывать сетевые события без блокировки основного потока приложения, обеспечивая отзывчивость интерфейса и гибкость в управлении соединениями.

3) Почему в данном задании выбран асинхронный подход? Какие у него преимущества и недостатки?

Асинхронный подход выбран для обеспечения отзывчивости приложения и эффективной обработки множественных соединений.

Преимущества:

- Отзывчивость интерфейса: Асинхронная обработка позволяет GUI оставаться активным, пока приложение ожидает сетевые события (например, получение данных).

- Масштабируемость: Сервер может обрабатывать несколько 20 клиентов одновременно без создания отдельных потоков для каждого соединения, что упрощает архитектуру.

- Экономия ресурсов: Не требуется блокировать потоки, что снижает нагрузку на систему.

Недостатки:

- Сложность отладки: Асинхронные события сложнее отслеживать и синхронизировать, особенно при обработке больших объёмов данных.

- Ошибки последовательности: Неправильная обработка порядка событий может привести к некорректной работе приложения.

- Ограниченная производительность для высоконагруженных систем: Для очень большого числа соединений может потребоваться многопоточность или другие подходы.

4) Какие ошибки могут возникнуть при работе с сокетами?

При работе с сокетами могут возникнуть следующие ошибки:

- Ошибки соединения: Неверный IP-адрес или порт, недоступность сервера, тайм-аут подключения (как в коде, где проверяется waitForConnected(3000)).

- Ошибки чтения/записи: Неполное получение данных, повреждение пакетов или разрыв соединения во время передачи.

- Ошибки формата данных: Некорректная работа с QDataStream, например, несоответствие версий Qt или неправильная обработка nextBlockSize.

- Ошибки управления ресурсами: Утечки памяти при неправильном удалении объектов сокетов (например, если не вызывается deleteLater()).

- Сетевые ограничения: Блокировка соединений брандмауэром, перегрузка сети или потеря пакетов.
- Синхронизация данных: Проблемы с порядком обработки сообщений или их дублированием при асинхронной работе.