Server

main.cpp

```
#include <QCoreApplication>
#include <QObject>

#include "server.h"

int main(int argc, char *argv[])
{
    QCoreApplication a(argc, argv);

    server *s = new server;

    QObject::connect(s, SIGNAL(sig_quit()), &a, SLOT(quit()));

    return a.exec();
}
```

server.h

```
#ifndef SERVER_H
#define SERVER_H

#include <QObject>
#include <QTcpSocket>
#include <QTcpServer>

#include <iostream>

class server : public QObject
{
    Q_OBJECT
public:
    explicit server(QObject *parent = nullptr);

public slots:
    void slot_acceptConnection();
    void slot_readMessage();

private:
    QTcpSocket *socket;
    QTcpServer *messageServer;

signals:
    void sig_quit();
```

```
};

#endif // SERVER_H
```

server.cpp

```
#include "server.h"

server::server(QObject *parent) : QObject(parent)
{
    socket = new QTcpSocket();
    messageServer = new QTcpServer();

    QObject::connect(messageServer, SIGNAL(newConnection()), this,
SLOT(slot_acceptConnection()));

    messageServer->listen(QHostAddress("127.0.0.1"), 9601);

    std::cout << "Server waiting for messages <127.0.0.1:9601>" << std::endl;

    return;
}

void server::slot_acceptConnection()
{
    socket = messageServer->nextPendingConnection();

    QObject::connect(socket, SIGNAL(readyRead()), this, SLOT(slot_readMessage()));

    return;
}

void server::slot_readMessage()
{
    QByteArray buffer;

    buffer = socket->readAll();

    int bytesRead = buffer.length();

    std::cout << "Message: " << buffer.toStdString() << std::endl;

    std::cout << "Bytes read: " << bytesRead << std::endl;

    messageServer->close();

    emit sig_quit();

    return;
```

```
}
```