Министерство цифрового развития, связи и массовых коммуникаций РФ Сибирский Государственный Университет Телекоммуникаций и Информатики СибГУТИ

Кафедра САПР

Курсовая работа

По дисциплине: «Программирование и обработка графического интерфейса»

Выполнил: Бондаренко А.Л.

Группа: ТТМ-23

Проверил: Голованчиков С.А.

Новосибирск 2024гЦель работы

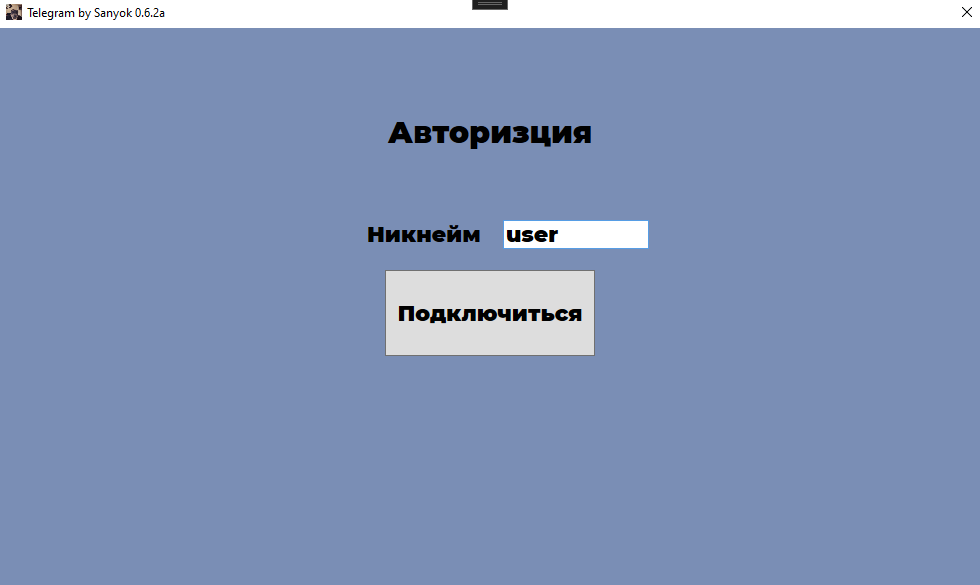
Индивидуальная тема. Клиент-серверное приложение (Мессенджер).

**Целью работы является создание** клиент-серверного приложения «Мессенджер», используя элементы управления WPF. При помощи него должна осуществляется связь между множеством клиентов, используя интернет.

Описание работы

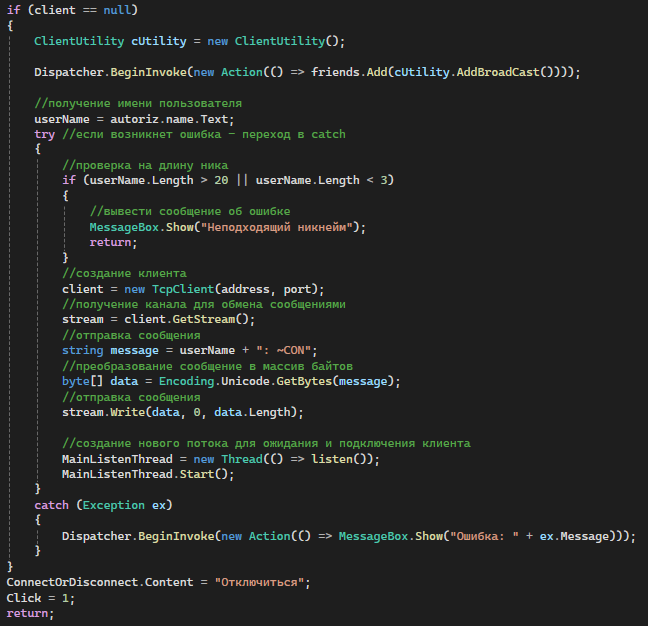
1. Клиент пользователя

Пользователя встречает окно авторизации, где он может подключиться к серверу.

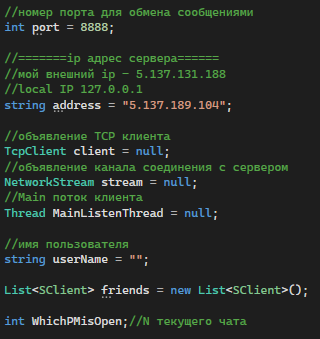


Окно авторизации представляет собой собственный элемент UserControl.

По нажатию кнопки «Подключиться» выполняется код:



Создается новый клиент и канал связи. При инициализации программы были созданы глобальные переменные с параметрами ниже:



Подключившись к серверу, можно увидеть функционал мессенджера:



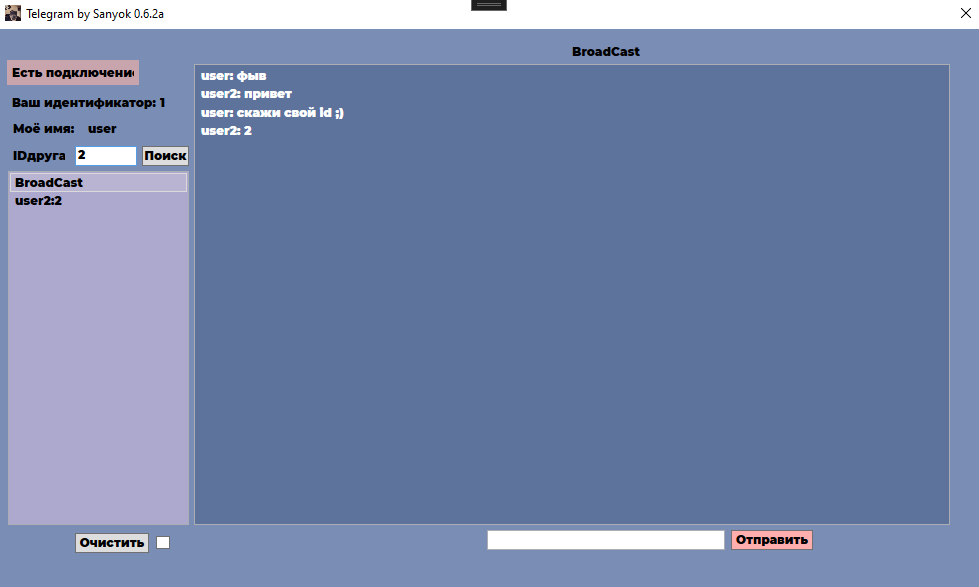
Программа поделена на сегменты:

1. Основное поле переписки;

2. Список чатов (и поиск пользователей);

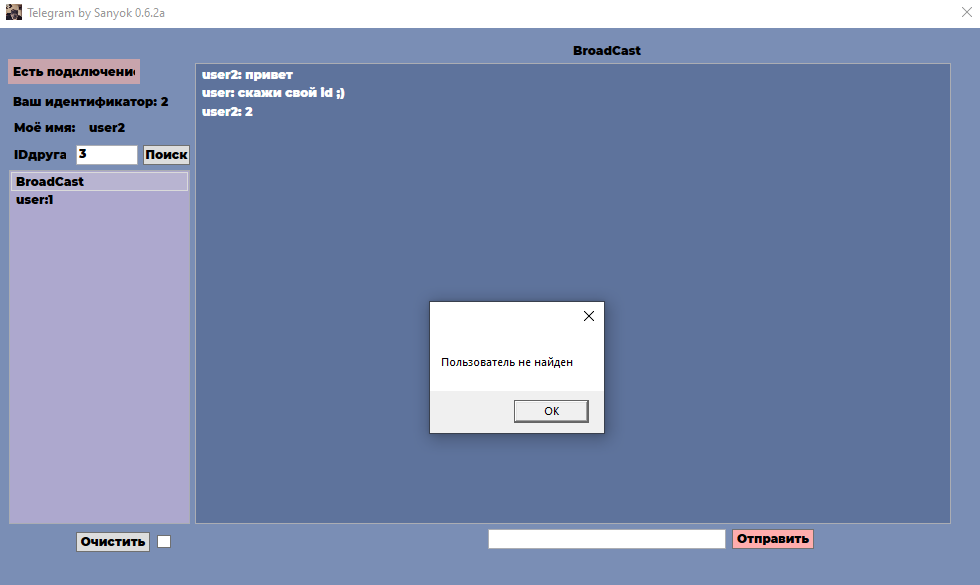
3. Дополнительная информация о пользователе.

В 3-ем сегменте можно увидеть свой идентификатор, его можно сообщить другим пользователям, чтобы общаться в личном чате.



Введя, верный ID в поисковую строку, добавиться личный чат с другим пользователем.

Также введя, неверный ID будет вызвано оповещение:



Кликнув по выпавшему элементу в ListBox, вверху меняется название чата:



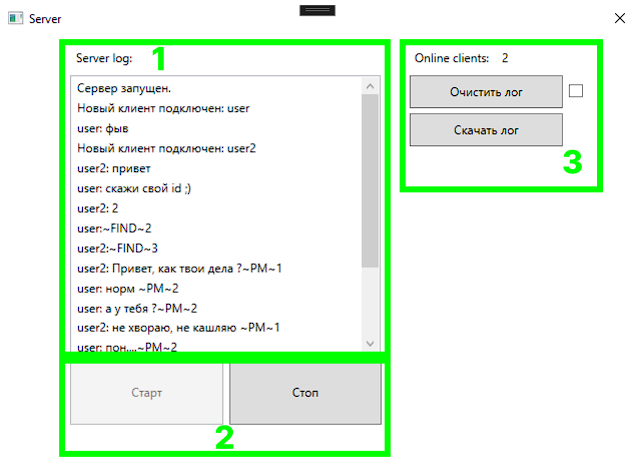
Существует возможность удаления чата. Удаляя чат, он удаляется у обоих пользователей:



Нажав на CheckBox всплывающая подсказка не будет появляться, а чаты сразу удалятся. Всплывающая подсказка также сделана как отдельное окно Window.

2.Клиент сервера

Клиент сервера выглядит следующем образом:



Сегменты:

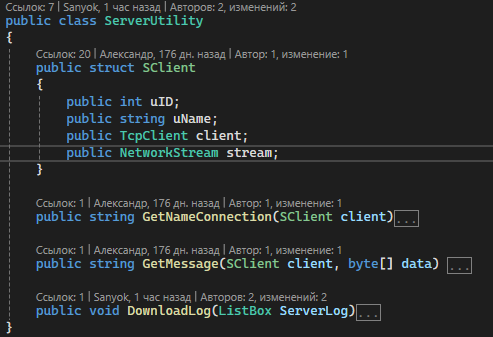
1. Лог сервера, где отображаются все ошибки и запросы.

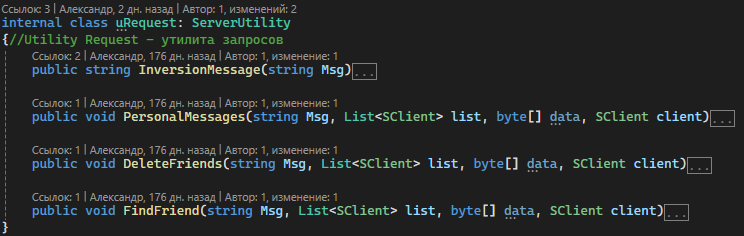
2. Кнопки запуска и остановки сервера

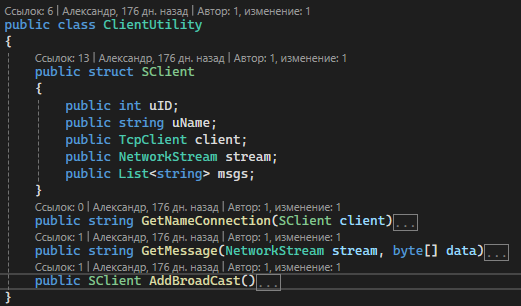
3. Дополнительный функционал (Кнопка очищения лога и кнопка его скачивания в .txt, также отображение количества активных пользователей)

3.Код

Чтобы не дублировать код и уменьшить были написаны 3 класса: ServerUtility, URequest, ClientUtility.





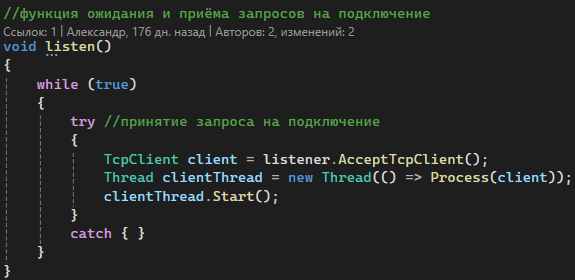


В основном теле программы есть 2 главных метода который прослушивают канал связи в ожидании ответа друг от друга. У клиента и сервера - listen(), а у сервера доп. - Process().

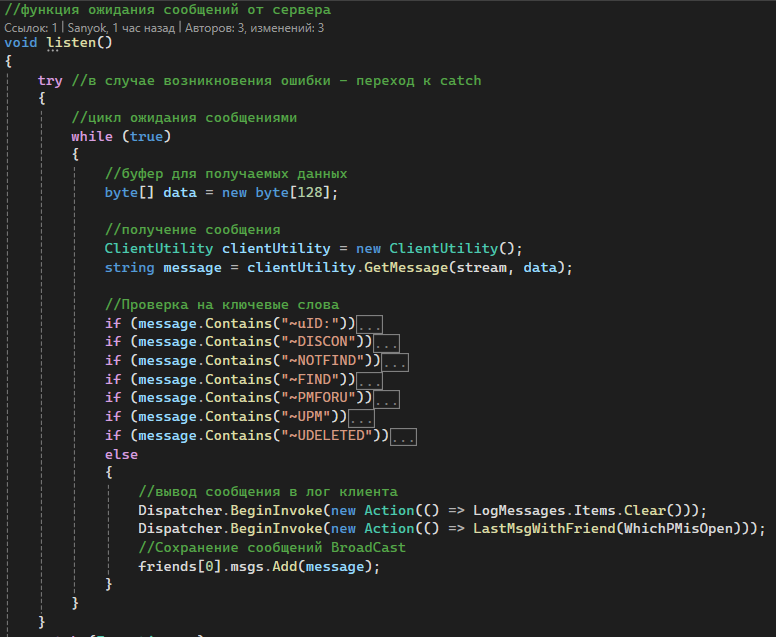
На сервере программа принимает подключение клиента и ждет от него сообщений, в методе Process():



Метод listen():



У клиента метод listen() выглядит по другому:



Методы listen() состоят из блоков try-catch-finally, дабы программа могла корректно отрабатывать различные сценарии взаимодействия подключения множества пользователей одновременно.

Также был разработан собственный язык запросов между сервером и клиентом:

отправка с Client:

~uID

~DISCON

~FIND

~NOTFIND

~PMFORU

~UPM

~UDELETED

~DELFRNDs

~CON

-------------------------------------------

Пользователь

/INV

-------------------------------------------

отправка с Server:

~PM

~DELFRNDs

~FIND

~DISCON

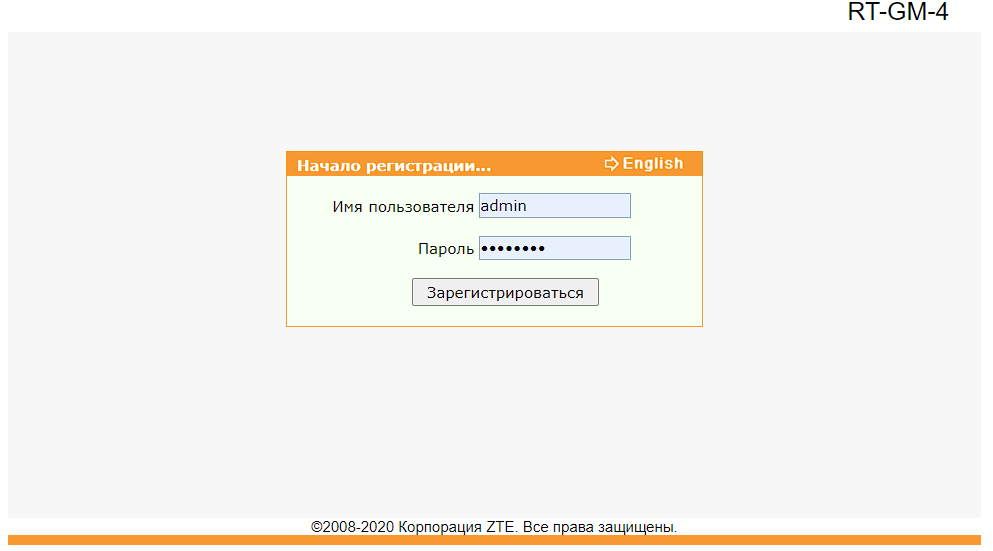
~CON

~DECLINE

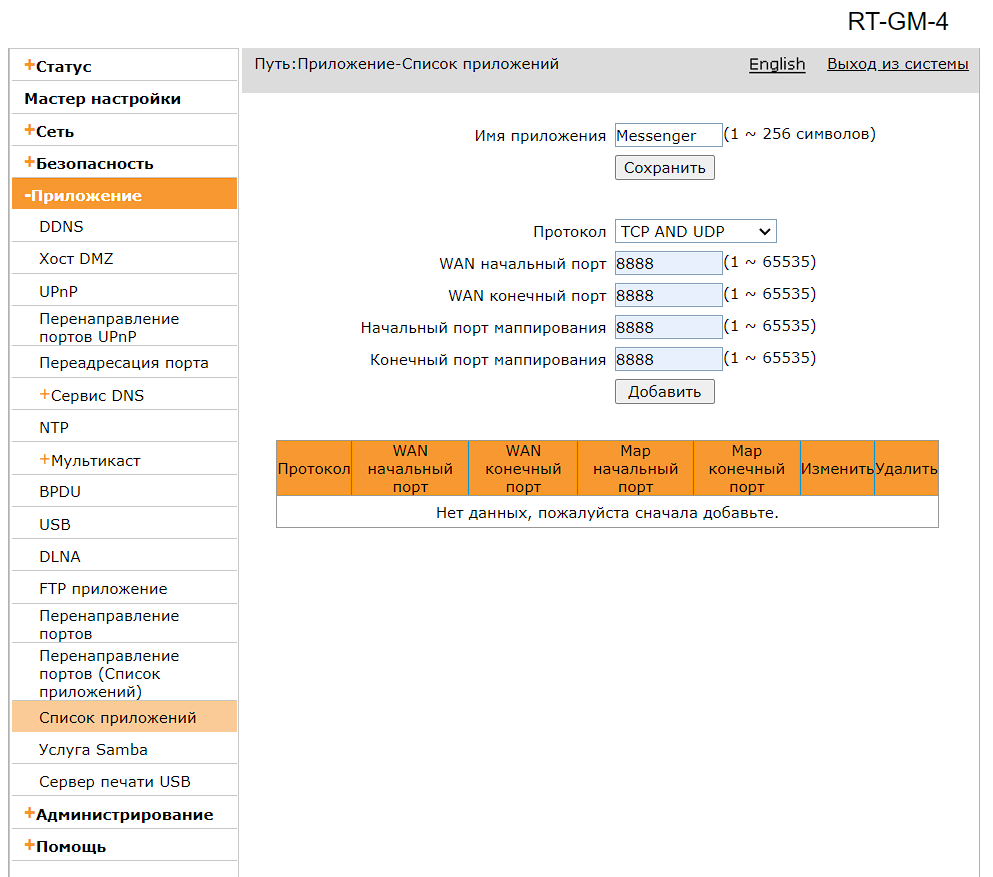
4. Интернет

Для того чтобы к серверу могли подключаться пользователи с других устройств через интернет – установка внутреннего ip на сервере, открытие порта на модеме, установка внешнего ip в клиенте.

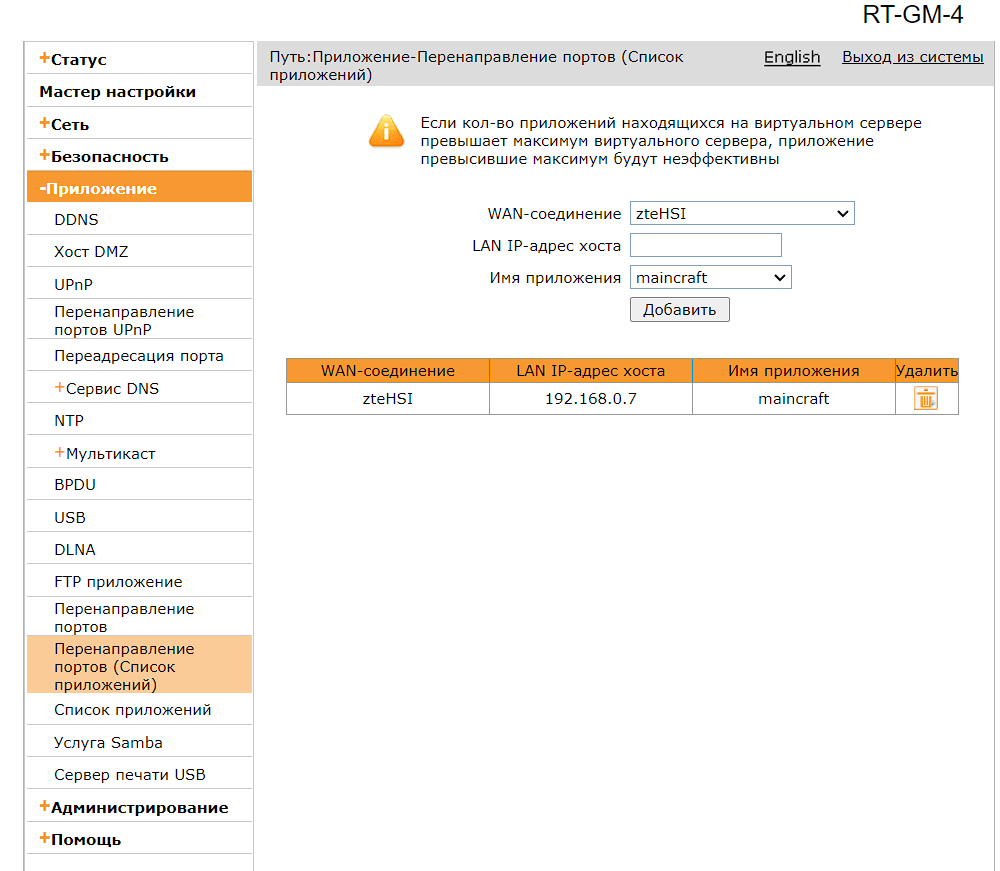
Вход в настройки роутера:



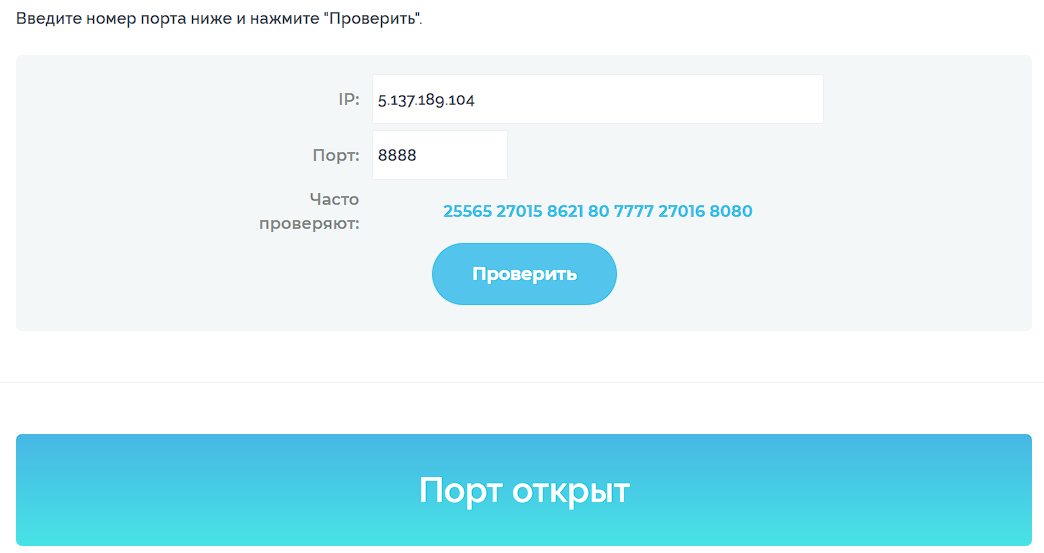
Создание приложения и открытие порта:



Установка соответствия на какое устройство идет подключение во внутренней сети:

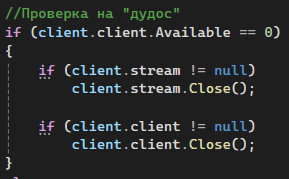


Проверка порта на доступность:

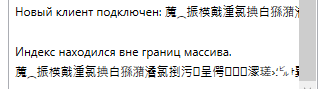


4.1. Проблемы

Проблемой, с которой столкнулся при развертывании сервера и подключения его к интернету, стали скан-боты. Они обваливали сервер кидая пустые запросы. Для обхода данной проблемы было написано простенькое условие:



Подобным образом выглядит запрос скан-бота:



Вывод

**В итоге было разработано клиент-серверное приложение, с функциональностью мессенджера. Внутри которого есть личные чаты. Самым интересным было подключение к интернету, создание своего языка запросов и обход «дудоса» от скан-ботов.**

**На курсовую ушло много времени в районе 2 месяцев, на текущий момент это стало моим самым крупным проектом написанным на C#.**

**Проект выложен на гитхаб:**

<https://github.com/hocokich/TelegramBySanyok>