Ministerul Educației a Republicii Moldova

Universitatea Tehnică din Moldova

Facultatea Calculatoare, Informatică și Microelectronică

Raport

La disciplina: Pogramarea în rețea

Lucrarea de laborator nr.5

Tema: “Sockets API”

A efectuat: Vozian T.

studenta grupei TI-161 FR

A verificat: Zgureanu A.

Chișinău 2020

**Link la repozitoriu:** <https://github.com/taniuskav/Programarea-in-retea>

Obiective:

Proiectarea si realizarea unui protocol de transfer date (mesaje), utilizind protocolul de nivel de transport(TCP).

# Executarea lucrării

WebSocket este un protocol care furnizează canale full-duplex de comunicare printr-o singură conexiune TCP. WebSocket este conceput pentru a fi pus în aplicare în browserele web și servere de web, dar poate fi utilizat de către orice client sau server de aplicații. Protocolul WebSocket este un protocol bazat pe TCP.

Server-ul ce utilizează WebSocket la fel ca și în cazul folosirii multicast crează un grup de utilizatori care sunt adăugați într-o listă, mesajele sunt trimise tuturor utilizatorilor înregistrați.

Serverul tratează trei tipuri de evenimente:

* OnOpen – eveniment ce apare la conectare;
* OnClose - eveniment ce apare la conectare;
* OnMessage – evenimentul ce apare la primirea unui mesaj.

Inițial am declarat lista ce va fi folosită pentru adăugarea clienților și adresa serverului (portul). Tratarea evenimentelor va avea loc prin canalul deschis WebSocketServer. În consola serverului la fel vor fi afișate mesajele din log, care cuprind mărimea mesajului primit și transmis și corpul mesajului, pentru a realiza această funcție am folosit Fleck care reprezintă o implimentare WebSocket în limbajul C#.

FleckLog.Level = LogLevel.Debug;

var webSockets = new List<IWebSocketConnection>();

var server = new WebSocketServer("ws://0.0.0.0:30000");

Aplicația folosește propriul protocol și tratează mesajele conform primului caracter primit:

* 1 – conectarea unui utilizator nou;
* 2 – deconectarea unui utilizator;
* / – mesaj privat de la un utilizator către altul.

La conectare are loc adăugarea utilizatorului în lista de utilizatori și formarea mesajului conform protocolului folosit, apoi mesajul este transmis fiecărui participant folosind instrucțiunea ForEach și afișarea mesajului în consolă.

Secvența ce se execută la conectarea unui utilizator poate fi găsită mai jos:

case "1":

users.Add(data[1]);

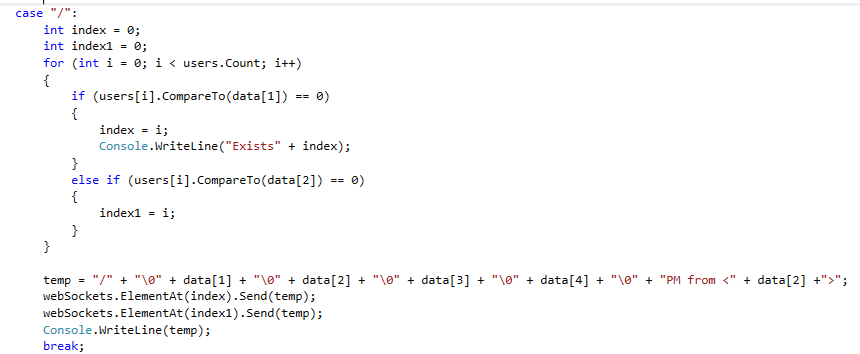
temp = "1" + "\0" + data[1] + "\0" + data[2] + "\0" + "Joins the net";

webSockets.ToList().ForEach(s => s.Send(temp));

Console.WriteLine(temp);

break;

Pentru transmiterea unui mesaj privat indicăm / și numele utilizatorului cărui adresăm acest mesaj în caz de sunt 3 persoane in chat. În Figura de jos este prezentă partea codului ce execută asta:



În codul ce urmează este arătată succesiunea de intrucțiuni executate la conectarea unui utilizator. Conform protocolului utilizat se extrage primul caracter în baza căruia se depistează ce fel de mesaj este primit, restul mesajului se împarte în părți și se afișează în chat.

ws.onmessage = function(e) {

var now = new Date();

var type = e.data.substring(0,1);

var parts = e.data.split('\0');

switch(type) {

// user came online

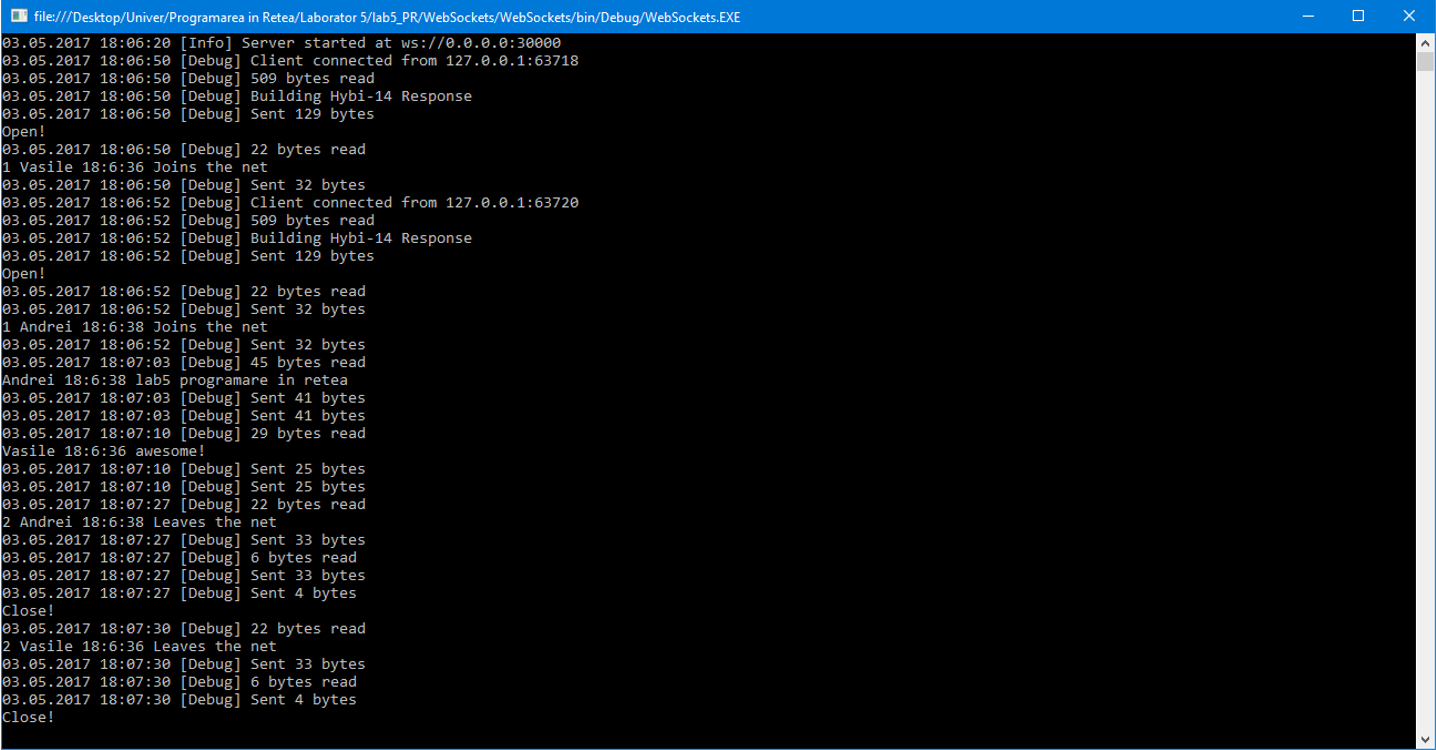
case "1":

list.innerHTML += "<span class='user'>" + parts[1] + " : </span><span class='time'>" +

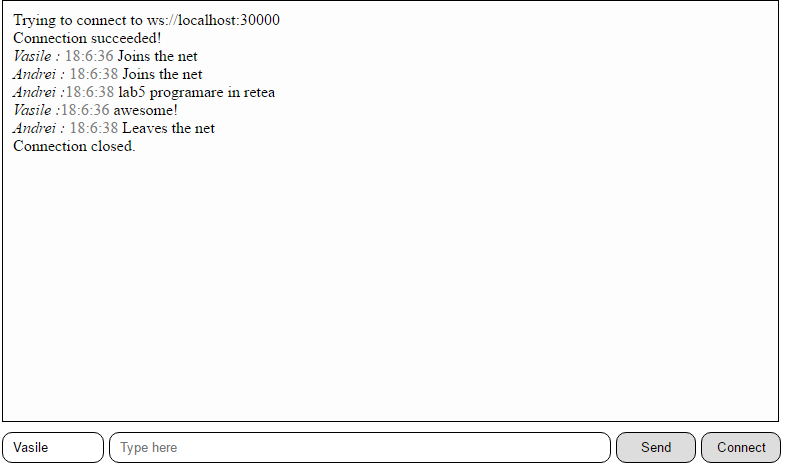
parts[2] + " </span>" + parts[3] + "</br>";

break;

# Rezultate



*Figura 1 - Server*



*Figura 2 – Chat*

# Concluzii

În urma realizării acestei lucrări de laborator am studiat noi abordări pentru comunicare în spațiul Web prin intermediul WebSockets. Pentru a realiza sarcina propusă am creat o aplicație client-server care reprezintă un chat simplu cu posibilitatea de conectare, deconectare și trimiterea mesajelor public și private către toți utilizatorii chatului conectați la server.