**МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ**

**УЧРЕЖДЕНИЕ ОБРАЗОВАНИЯ**

**ГОМЕЛЬСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ ИМЕНИ П. О. СУХОГО**

Факультет автоматизированных и информационных систем

Кафедра «Информационные технологии»

ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ № 7

по дисциплине «Программирование сетевых приложений»

на тему:

**«Программирование приложений с использованием**

**WebSocket»**

Выполнил: студент гр. ИТИ-41

Гончар А.С.

Принял: преподаватель

Карабчикова Е.А.

Гомель 2020

**Цель работы:** научиться разрабатывать приложения с использованием web-сокетов.

**Задание**: Разработать интерактивное веб­приложение с использованием технологии WebSocket. (рекомендуется использовать .Net, Java или Python, можно использовать другие языки программирования по согласованию с преподавателем).

Разработать конечный автомат для анализа надежности и корректности пароля.

Длина пароля не менее 6 символов. Разрешены только латинские буквы, цифры и знаки: пробел \_­. В пароле обязательно должны присутствовать знаки в верхнем и нижнем регистре и цифры.

Пользователю возвращается информация о том, можно ли использовать указанный пароль.

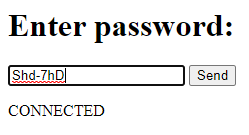


Рисунок 1 – html

**Вывод:** в ходе выполнения лабораторной работы был разработан https­сервер с использованием web-сокетов.

**Приложение А**

**Листинг программы**

ClientObject:

using System;

using System.Net.Sockets;

using System.Text;

using System.Text.RegularExpressions;

namespace Lab7PSP

{

public class ClientObject

{

private TcpClient client;

private NetworkStream stream = null;

private string clientName = string.Empty;

public ClientObject(TcpClient client)

{

this.client = client;

}

public void Process()

{

try

{

NetworkStream stream = client.GetStream();

// enter to an infinite cycle to be able to handle every change in stream

while (true)

{

while (!stream.DataAvailable) ;

while (client.Available < 3) ; // match against "get"

byte[] bytes = new byte[client.Available];

stream.Read(bytes, 0, client.Available);

string s = Encoding.UTF8.GetString(bytes);

if (Regex.IsMatch(s, "^GET", RegexOptions.IgnoreCase))

{

Console.WriteLine("=====Handshaking from client=====\n{0}", s);

// 1. Obtain the value of the "Sec-WebSocket-Key" request header without any leading or trailing whitespace

// 2. Concatenate it with "258EAFA5-E914-47DA-95CA-C5AB0DC85B11" (a special GUID specified by RFC 6455)

// 3. Compute SHA-1 and Base64 hash of the new value

// 4. Write the hash back as the value of "Sec-WebSocket-Accept" response header in an HTTP response

string swk = Regex.Match(s, "Sec-WebSocket-Key: (.\*)").Groups[1].Value.Trim();

string swka = swk + "258EAFA5-E914-47DA-95CA-C5AB0DC85B11";

byte[] swkaSha1 = System.Security.Cryptography.SHA1.Create().ComputeHash(Encoding.UTF8.GetBytes(swka));

string swkaSha1Base64 = Convert.ToBase64String(swkaSha1);

// HTTP/1.1 defines the sequence CR LF as the end-of-line marker

byte[] response = Encoding.UTF8.GetBytes(

"HTTP/1.1 101 Switching Protocols\r\n" +

"Connection: Upgrade\r\n" +

"Upgrade: websocket\r\n" +

"Sec-WebSocket-Accept: " + swkaSha1Base64 + "\r\n\r\n");

stream.Write(response, 0, response.Length);

}

else

{

bool fin = (bytes[0] & 0b10000000) != 0,

mask = (bytes[1] & 0b10000000) != 0; // must be true, "All messages from the client to the server have this bit set"

int opcode = bytes[0] & 0b00001111, // expecting 1 - text message

msglen = bytes[1] - 128, // & 0111 1111

offset = 2;

if (msglen == 126)

{

// was ToUInt16(bytes, offset) but the result is incorrect

msglen = BitConverter.ToUInt16(new byte[] { bytes[3], bytes[2] }, 0);

offset = 4;

}

else if (msglen == 127)

{

Console.WriteLine("TODO: msglen == 127, needs qword to store msglen");

// i don't really know the byte order, please edit this

// msglen = BitConverter.ToUInt64(new byte[] { bytes[5], bytes[4], bytes[3], bytes[2], bytes[9], bytes[8], bytes[7], bytes[6] }, 0);

// offset = 10;

}

if (msglen == 0)

Console.WriteLine("msglen == 0");

else if (mask)

{

byte[] decoded = new byte[msglen];

byte[] masks = new byte[4] { bytes[offset], bytes[offset + 1], bytes[offset + 2], bytes[offset + 3] };

offset += 4;

for (int i = 0; i < msglen; ++i)

decoded[i] = (byte)(bytes[offset + i] ^ masks[i % 4]);

string text = Encoding.UTF8.GetString(decoded);

Console.WriteLine("{0}", text);

var correctStr = CheckPass(text) ? "Correct" : "Incorrect";

Byte[] response = Encoding.UTF8.GetBytes(" " + correctStr);

response[0] = 0x81; // denotes this is the final message and it is in text

response[1] = (byte)(response.Length - 2); // payload size = message - header size

stream.Write(response, 0, response.Length);

}

else

Console.WriteLine("mask bit not set");

Console.WriteLine();

}

}

}

catch (Exception ex)

{

Console.WriteLine(ex.Message);

}

finally

{

if (stream != null)

stream.Close();

if (client != null)

client.Close();

Console.ReadKey();

}

}

private bool CheckPass(string password)

{

Console.WriteLine(password);

bool correct = false;

var regexstr = @"^(?=.\*?[A-Z])(?=.\*?[a-z])(?=.\*?[0-9])(?=.\*?[-\_]).{6,}$";

Regex regex = new Regex(regexstr);

if (regex.IsMatch(password))

{

if (!Regex.Match(password, @"[а-яА-Я]").Success && !Regex.Match(password, @"[/\*+=!@#$%^,.<>;:()&|\`~{}]").Success)

{

correct = true;

}

}

return correct;

}

}

}

Program:

using Lab7PSP;

using System;

using System.Net;

using System.Net.Sockets;

using System.Threading;

class Server

{

public static void Main()

{

string ip = "127.0.0.1";

int port = 8080;

var server = new TcpListener(IPAddress.Parse(ip), port);

server.Start();

Console.WriteLine("Server has started on {0}:{1}, Waiting for a connection...", ip, port);

while (true)

{

TcpClient client = server.AcceptTcpClient();

Console.WriteLine("A client connected.");

ClientObject clientobject = new ClientObject(client);

Thread clientThread = new Thread(new ThreadStart(clientobject.Process));

clientThread.Start();

}

}

}