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

дисциплина «Программирование сетевых приложений»

на тему «Программирование приложений с использованием WebSocket»

Выполнил: студент гр. ИТП-41

Болотникова Е.В.

Принял: преподаватель

Гуменников Е.Д.

Гомель 2022

**Задание:** разработать интерактивное веб-приложение с использованием технологии WebSocket – общий чат.

.

Для выполнения лабораторной работы было разработано клиент-серверное приложение. Сервер принимает сообщение и рассылает его всем остальным участникам.

Результат работы первого клиента приведен на рисунке 1.



Рисунок 1 – Результат работы чата

Листинг приложения приведён в приложении А.

**Вывод**: в ходе выполнения лабораторной работы было разработано интерактивное веб-приложение, которое организует общение в реальном времени между пользователями чата с использованием протокола WebSocket.

Приложение А

server.js:

'use strict';

// Use HTTP server

const http = require('http');

// Use WebSocket

const Websocket = require('websocket').server;

// Use logger module

const logger = require('./tools/logger');

// Use router module

const router = require('./tools/router');

// Use server module

const serve = require('./tools/serve');

// Use clustering

const cluster = require('cluster');

const os = require('os');

// Process id

const pid = process.pid;

// Selecting port

const PORT = 8000;

// Clustering

if (cluster.isMaster) {

    const count = os.cpus().length;

    console.log(`Master pid: ${pid}`);

    console.log(`Starting ${count} forks`);

    for (let i = 0; i < count; i++) cluster.fork();

}

else {

    const id = cluster.worker.id;

    console.log(`Worker: ${id}, pid: ${pid}, port: ${PORT}`);

    // Server creating

    const server = http.createServer((req, res) => {

        logger.log(req);

        res.setHeader('Process-Id', pid);

        const data = router.routing.get(req.url);

        serve.serve(data, req, res);

    });

    // Server starting

    server.listen(PORT, () => {

        console.log('Listen port ' + PORT);

    });

    // WebSocket creating

    const ws = new Websocket({

        httpServer: server,

        autoAcceptConnections: false

    });

    // Clients list

    const clients = [];

    // Using WebSocket to get requests

    ws.on('request', req => {

        // Adding client

        const connection = req.accept('', req.origin);

        clients.push(connection);

        console.log('Connected ' + connection.remoteAddress);

        // Getting client message

        connection.on('message', message => {

            const dataName = message.type + 'Data';

            const data = message[dataName];

            console.dir(message);

            console.log('Received: ' + data);

            clients.forEach(client => {

                if (connection !== client) {

                    client.send(data);

                }

            });

      });

      // On client disconnection

      connection.on('close', (reasonCode, description) => {

            console.log('Disconnected ' + connection.remoteAddress);

            console.dir({ reasonCode, description });

        });

    });

}

chat.html:

<!DOCTYPE html>

<html>

    <head>

        <title>Chat</title>

        <link rel="stylesheet" href="style.css">

    </head>

    <body>

        <div >

            <h1>chat</h1>

            <ul id="chat"></ul>

            <div id="message\_container">

                <input id="name" type="text" placeholder="Your name">

                <input id="msg" type="text" placeholder="Your message">

                <button id="send">Send</button>

            </div>

        </div>

        <script>

            const socket = new WebSocket('ws://127.0.0.1:8000/chat');

            const chat = document.getElementById('chat');

            const msg = document.getElementById('msg');

            const name = document.getElementById('name');

            const conteiner = document.getElementById('message\_container')

            const botton = document.getElementById('send')

            const writeLine = text => {

                const line = document.createElement('li');

                line.innerHTML = `<p>${text}</p>`;

                chat.appendChild(line);

            };

            const writeLineConect = text => {

                const line = document.createElement('div');

                line.innerHTML = `<h2>${text}</h2>`;

                chat.appendChild(line);

            };

            socket.onopen = () => {

                writeLineConect('connected');

            };

            socket.onclose = () => {

                writeLineConect('closed');

            };

            socket.onmessage = event => {

                writeLine(event.data);

            };

            botton.addEventListener('click', function(e) {

              e.preventDefault();

              if(msg.value && name.value){

                    const s = name.value + ": "+ msg.value;

                    msg.value = '';

                    writeLine(s);

                    socket.send(s);

                }

            });

        </script>

      </body>

</html>