Федеральное государственное бюджетное образовательное учреждение
высшего профессионального образования
«Московский государственный технический университет
имени Н.Э. Баумана»
(МГТУ им. Н.Э. Баумана)

Факультет: Информатика и системы управления
Кафедра: Теоретическая информатика и компьютерные технологии

Лабораторная работа №5
«Реализация WebSocket клиента и сервера на языке Golang»
по курсу: «Компьютерные сети»

Выполнил:
Студент группы ИУ9-32Б
Гнатенко Т. А.

Проверил:
Посевин Д. П.

Москва, 2022

Цели

Изучить принципы работы WebSocket

Задачи

Реализовать сетевую службу на языке программирования Golang взаимодействующую по протоколу связи WebSocket по вариантам. Клиентское приложение получает через стандартный поток ввода данные и в формате JSON передает их на сервер, сервер выполняет вычисления и возвращает результат обратно клиенту, который в свою очередь выводит полученный результат в стандартный поток вывода.

Решение

Исходный код

client.go

```
package main

import (
    "bufio"
    "flag"
    "log"
    "net/url"
    "os"
    "os/signal"

    log2 "github.com/mgutz/logxi/v1"

    "github.com/gorilla/websocket"
)

var addr = flag.String("addr", "localhost:8080", "http
 ↪  service address")
```

```
func main() {
    flag.Parse()
    log.SetFlags(0)

    interrupt := make(chan os.Signal, 1)
    signal.Notify(interrupt, os.Interrupt)

    u := url.URL{Scheme: "ws", Host: *addr, Path: "/echo"}
    log2.Info("connecting to ", "server", u.String())

    c, _, err := websocket.DefaultDialer.Dial(u.String(),
↪   nil)
    if err != nil {
        log2.Error(err.Error())
        return
    }
    defer c.Close()

    done := make(chan struct{})

    go func() {
        defer close(done)
        for {
            log2.Info("Enter the task")
            _, message, err := c.ReadMessage()
            if err != nil {
                log2.Error(err.Error())
                return
            }

            log2.Info("Answer got successful", "answer: ",
↪   string(message))
        }
    }()

    outgoing := make(chan string)
    go func() {
        reader := bufio.NewReader(os.Stdin)
        for {
            line, _, err := reader.ReadLine()
```

```
            if err != nil {
                break
            }
            outgoing <- string(line)
        }
    }()

    for {

        select {
        case <-done:
            return
        case line := <-outgoing:
            err := c.WriteMessage(websocket.TextMessage,
    []byte(line))
            if err != nil {
                log2.Error(err.Error())
                return
            }
        case <-interrupt:
            log2.Info("Interrupt")
            err := c.WriteMessage(websocket.CloseMessage,
    websocket.FormatCloseMessage(websocket.CloseNormalClosure,
    ""))
            if err != nil {
                log2.Error(err.Error())
                return
            }
            select {
            case <-done:
            case <-outgoing:
            }
            return
        }
    }
}
```

server.go

```
package main
```

```
import (
    "flag"
    "fmt"
    "log"
    "math"
    "net/http"
    "strconv"
    "strings"

    log2 "github.com/mgutz/logxi/v1"

    "github.com/gorilla/websocket"
)

var e = 0.001
var addr = flag.String("addr", "localhost:8080", "http
 ↪  service address")

var upgrader = websocket.Upgrader{} // use default options

func echo(w http.ResponseWriter, r *http.Request) {
    c, err := upgrader.Upgrade(w, r, nil)
    if err != nil {
        log2.Error(err.Error())
        return
    }
    defer c.Close()
    for {
        mt, message, err := c.ReadMessage()
        task := string(message)
        if err != nil {
            log2.Error(err.Error())
            break
        }
        log2.Info("Task got successful", "task: ", task)
        coefficients := strings.Split(task, " ")

        if len(coefficients) == 3 {
            a, err := strconv.ParseFloat(coefficients[0],
 ↪  64)
```

```
            if err != nil {
                log2.Error(err.Error())
                return
            }
            b, err := strconv.ParseFloat(coefficients[1],
↪   64)
            if err != nil {
                log2.Error(err.Error())
                return
            }
            c, err := strconv.ParseFloat(coefficients[2],
↪   64)
            if err != nil {
                log2.Error(err.Error())
                return
            }

            D := b*b - 4*a*c
            if D > -e && D < e {
                res := -b / (2 * a)
                task = fmt.Sprintf("Solution is %f", res)
            } else if D < 0 {
                task = "Not solution"
            } else {
                res1, res2 := (-b+math.Sqrt(D))/(2*a),
↪   (-b-math.Sqrt(D))/(2*a)
                task = fmt.Sprintf("Solution is %f and %f",
↪   res1, res2)
            }
        } else {
            task = "Entered data's flawed"
        }
        err = c.WriteMessage(mt, []byte(task))
        if err != nil {
            log2.Error(err.Error())
            break
        }
    }
}
```

```
func main() {
    flag.Parse()
    log.SetFlags(0)
    http.HandleFunc("/echo", echo)
    log2.Info(http.ListenAndServe(*addr, nil).Error())
}
```

Вывод