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 ИУ9-31Б
Старовойтов А. И.

Проверил:
Посевин Д. П.

Москва, 2022

Цели

Изучение протокола 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.

Вариант 3. Интеграл.

Решение

Рабочее окружение в этой лабе полностью реализовано в докере. Написаны функциональные тесты на pytest и юнит-тесты для модуля вычисления интеграла.

`cmd/client/client.go`

```
package main

import (
    "context"
    "fmt"
    "time"

    "github.com/stewkk/iu9-networks/lab5/internal/api"
    "nhooyr.io/websocket"
    "nhooyr.io/websocket/wsjson"
)

func main() {
    ctx, cancel := context.WithTimeout(context.Background(),
    ↪ time.Minute)
    defer cancel()

    c, _, err := websocket.Dial(ctx, "ws://localhost:5332", nil)
    if err != nil {
        panic(err)
    }
    defer c.Close(websocket.StatusInternalError, "internal error")

    var integral api.Input
    fmt.Scan(&integral.A, &integral.B, &integral.C, &integral.Start,
    ↪ &integral.End)
```

```
    err = wsjson.Write(ctx, c, integral)
    if err != nil {
        panic(err)
    }

    var result api.Result
    err = wsjson.Read(ctx, c, &result)
    if err != nil {
        panic(err)
    }

    fmt.Println(result.Sum)

    c.Close(websocket.StatusNormalClosure, "")
}
```

cmd/lab5/lab5.go

```
package main

import "github.com/stewkk/iu9-networks/lab5/internal/api"

func main() {
    api.RunServer()
}
```

Модуль для вычисления определенного интеграла

internal/integral/integral.go

```
package integral

type Integral struct {
    Polynom `json:"polynom"`
    Range   `json:"range"`
}

func (i *Integral) Calc() float64 {
    steps := 10000
    result := i.calcSum(steps)
    steps *= 2
    for newResult := i.calcSum(steps); ne(newResult, result); {
        result = newResult
        steps *= 2
    }
    return result
}

func (integral *Integral) calcSum(steps int) float64 {
    var sum float64
    step := integral.size() / float64(steps)
    if integral.Start > integral.End {
```

```
        step = -step
    }
    x := integral.Start + step/2
    for i := 0; i < steps; i++ {
        sum += integral.Polynom.calc(x) * step
        x += step
    }
    return sum
}
```

internal/integral/integral_test.go

```
package integral

import (
    "testing"

    "github.com/stretchr/testify/require"
    "github.com/stretchr/testify/suite"
)

type IntegralTestSuite struct {
    suite.Suite
}

func (suite *IntegralTestSuite) TestCalculatesConstant() {
    integral := Integral{
        Polynom: Polynom{
            A: 0,
            B: 0,
            C: 1,
        },
        Range: Range{
            Start: 0,
            End:   1,
        },
    }

    require.InDelta(suite.T(), 1.0, integral.Calc(), eps)
}

func (suite *IntegralTestSuite) TestCalculatesLinear() {
    integral := Integral{
        Polynom: Polynom{
            A: 0,
            B: 1,
            C: 0,
        },
        Range: Range{
            Start: 0,
            End:   1,
```

```
        },
    }

    require.InDelta(suite.T(), 0.5, integral.Calc(), eps)
}

func (suite *IntegralTestSuite) TestCalculatesQuadraticFunction() {
    integral := Integral{
        Polynom: Polynom{
            A: 2,
            B: 1,
            C: -5,
        },
        Range: Range{
            Start: -1,
            End:   10,
        },
    }

    require.InDelta(suite.T(), 661.833333, integral.Calc(), eps*3)
}

func (suite *IntegralTestSuite) TestZeroRangeYieldsZero() {
    integral := Integral{
        Polynom: Polynom{
            A: 2,
            B: 1,
            C: -5,
        },
        Range: Range{
            Start: 0,
            End:   0,
        },
    }

    require.InDelta(suite.T(), 0.0, integral.Calc(), eps)
}

func (suite *IntegralTestSuite) TestNegativeRange() {
    integral := Integral{
        Polynom: Polynom{
            A: 0,
            B: 1,
            C: 0,
        },
        Range: Range{
            Start: 1,
            End:   0,
        },
    }
```

```
    require.InDelta(suite.T(), -0.5, integral.Calc(), eps)
}

func TestExampleTestSuite(t *testing.T) {
    suite.Run(t, new(IntegralTestSuite))
}
```

internal/integral/range.go

```
package integral

type Range struct {
    Start float64 `json:"start"`
    End   float64 `json:"end"`
}

func (rng Range) size() float64 {
    return abs(rng.End - rng.Start)
}
```

internal/integral/utils.go

```
package integral

func lt(a, b float64) bool {
    return a+eps < b
}

func ne(a, b float64) bool {
    return abs(a-b) >= eps
}

var eps = 1e-7

func abs(n float64) float64 {
    if n < 0 {
        return -n
    }
    return n
}
```

internal/integral/polynom.go

```
package integral

type Polynom struct {
    A float64 `json:"a"`
    B float64 `json:"b"`
    C float64 `json:"c"`
}

func (p *Polynom) calc(x float64) float64 {
```

```
    return p.A*x*x + p.B*x + p.C
}
```

Моудль API

internal/api/proto.go

```
package api

import "github.com/stewkk/iu9-networks/lab5/internal/integral"

type Result struct {
    Sum float64 `json:"sum"`
}

type Input integral.Integral
```

internal/api/websocket.go

```
package api

import (
    "context"
    "fmt"
    "log"
    "net/http"
    "time"

    "github.com/stewkk/iu9-networks/lab5/internal/integral"
    "nhooyr.io/websocket"
    "nhooyr.io/websocket/wsjson"
)

func RunServer() {
    server := http.Server{
        Addr: "0.0.0.0:5332",
        Handler: http.HandlerFunc(handler),
    }

    log.Println("Listenig...")
    server.ListenAndServe()
}

func handler(w http.ResponseWriter, r *http.Request) {
    c, err := websocket.Accept(w, r, nil)
    if err != nil {
        fmt.Println(err)
        return
    }
    defer c.Close(websocket.StatusInternalError, "internal error")

    for {
```

```
        ctx, cancel := context.WithTimeout(r.Context(), time.Second*10)
        defer cancel()

        var v integral.Integral
        err = wsjson.Read(ctx, c, &v)
        if err != nil {
            fmt.Println(err)
            return
        }

        sum := v.Calc()
        log.Printf("received: %v\tsum: %v", v, sum)

        wsjson.Write(ctx, c, Result{
            Sum: sum,
        })
    }
}
```

Функциональные тесты

tests/conftest.py

```
#!/usr/bin/env python3

import pytest

from websocket import create_connection
import json


pytest_plugins = ["docker_compose"]

@pytest.fixture(scope="function")
def wait_for_api(function_scoped_container_getter):
    service =
 ↪  function_scoped_container_getter.get("app").network_info[0]

    return service

@pytest.fixture
def service(wait_for_api):
    return Service(wait_for_api)


class Service:
    def __init__(self, service):
        self.ws = create_connection("ws://localhost:%s" %
         ↪  service.host_port)
    def make_request(self, payload):
        self.ws.send(json.dumps(payload))
```

```
        return self.ws.recv_data()
```

tests/test_api.py

```
#!/usr/bin/env python3

import pytest
import json
import struct

from conftest import Service


def test_calculates_integral(service):
    _, response = service.make_request({
        "polynom": {
            "a": 0.0,
            "b": 0.0,
            "c": 1.0,
        },
        "range": {
            "start": 0.0,
            "end": 1.0,
        }
    })

    assert json.loads(response)["sum"] == pytest.approx(1.0, 1e-6)

def test_returns_error_on_wrong_format(service):
    code, response = service.make_request([])

    assert code == 8
    assert struct.unpack("!H", response[0:2])[0] == 1007

def test_calculates_two_integrals(service):
    payload = {
        "polynom": {
            "a": 0.0,
            "b": 0.0,
            "c": 1.0,
        },
        "range": {
            "start": 0.0,
            "end": 1.0,
        }
    }
    _, response = service.make_request(payload)

    payload["range"]["end"] = 2.0
    _, response = service.make_request(payload)
```

```
assert json.loads(response)["sum"] == pytest.approx(2.0, 1e-6)
```