Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение высшего образования
«Московский государственный технический университет имени Н.Э. Баумана (национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ **ИНФОРМАТИКА И СИСТЕМЫ УПРАВЛЕНИЯ**

КАФЕДРА **КОМПЬЮТЕРНЫЕ СИСТЕМЫ И СЕТИ (ИУ6)**

НАПРАВЛЕНИЕ ПОДГОТОВКИ **09.03.01 Информатика и вычислительная техника**

О Т Ч Е Т

по рубежному контролю № 1

Название: Разработка WEB-сервера на Golang

Дисциплина: Языки интернет-программирования

Студент ИУ6-31Б _____ Л.В. Зимин
(Группа) (Подпись, дата) (И.О. Фамилия)

Преподаватель _____ В.Д. Шульман
(Подпись, дата) (И.О. Фамилия)

Москва, 2024

Вариант 20

Условие задания:

Существование треугольника:

Необходимо написать веб-севрер на GO, определяющий существует ли треугольник. Сервер должен запускаться по адресу `127.0.0.1:8081`.

У севрера должна быть ручка (handler) `POST /triangle`. Эта ручка ожидает, что через JSON будет передано 3 параметра типа int: `side1`, `side2` и `side3`.

При обработке http-запроса должно вычисляться, существует ли треугольник со сторонами, равными `side1`, `side2` и `side3`. Треугольник существует только тогда, когда сумма двух его сторон больше третьей.

В качестве ответа сервер должен возвращать JSON с единственным полем `result`.

Ход работы

Решение:

```
package main //20 variant

import (
	"encoding/json"
	"fmt"
	"net/http"
	"strings"
)

type Triangle struct {
	Side1 *int `json:"side1"`
	Side2 *int `json:"side2"`
	Side3 *int `json:"side3"`
}

type Result struct {
	Result string `json:"result"`
}

func triangleHandle(w http.ResponseWriter, r *http.Request) {
	if r.Method != "POST" {
		w.WriteHeader(http.StatusMethodNotAllowed)
		w.Write([]byte("Данный метод не поддерживается"))
		return
	}
	var triangle Triangle
	decoder := json.NewDecoder(r.Body)
```

```
	err := decoder.Decode(&triangle)
	if err != nil {
		w.WriteHeader(http.StatusBadRequest)
		if strings.HasPrefix(err.Error(), "json: cannot unmarshal number")
{
			w.Write([]byte("Число вышло за пределы разрядной
сетки или не является целым числом!"))
		} else if strings.HasPrefix(err.Error(), "json: cannot unmarshal
string") {
			w.Write([]byte("Введите число!"))
		} else {
			w.Write([]byte("Ошибка запроса! " + err.Error()))
		}
		return
	}
	if triangle.Side1 == nil {
		w.WriteHeader(http.StatusBadRequest)
		w.Write([]byte("1 сторона не указана!"))
		return
	}
	if triangle.Side2 == nil {
		w.WriteHeader(http.StatusBadRequest)
		w.Write([]byte("2 сторона не указана!"))
		return
	}
	if triangle.Side3 == nil {
		w.WriteHeader(http.StatusBadRequest)
		w.Write([]byte("3 сторона не указана!"))
		return
	}
```

```
	if int(*triangle.Side1) <= 0 || int(*triangle.Side2) <= 0 ||
int(*triangle.Side3) <= 0 {
		w.WriteHeader(400)
		w.Write([]byte("Все стороны должны быть положительны!"))
		return
	}

	var res Result
	if *triangle.Side1+*triangle.Side2 > *triangle.Side3 &&
*triangle.Side1+*triangle.Side3 > *triangle.Side2 &&
*triangle.Side3+*triangle.Side2 > *triangle.Side1 {
		res.Result = "Треугольник существует"
	} else {
		res.Result = "Треугольник не существует"
	}
	w.Header().Set("Content-Type", "application/json") //в формате json
	w.WriteHeader(http.StatusOK)
	respByte, _ := json.Marshal(res)
	w.Write(respByte)

}
func main() {
	http.HandleFunc("/triangle", triangleHandle)

	fmt.Println("Запуск сервера:")
	err := http.ListenAndServe("127.0.0.1:8081", nil)
	if err != nil {
		fmt.Println("Ошибка запуска сервера!: ", err)
	}
}
```

Тестирование

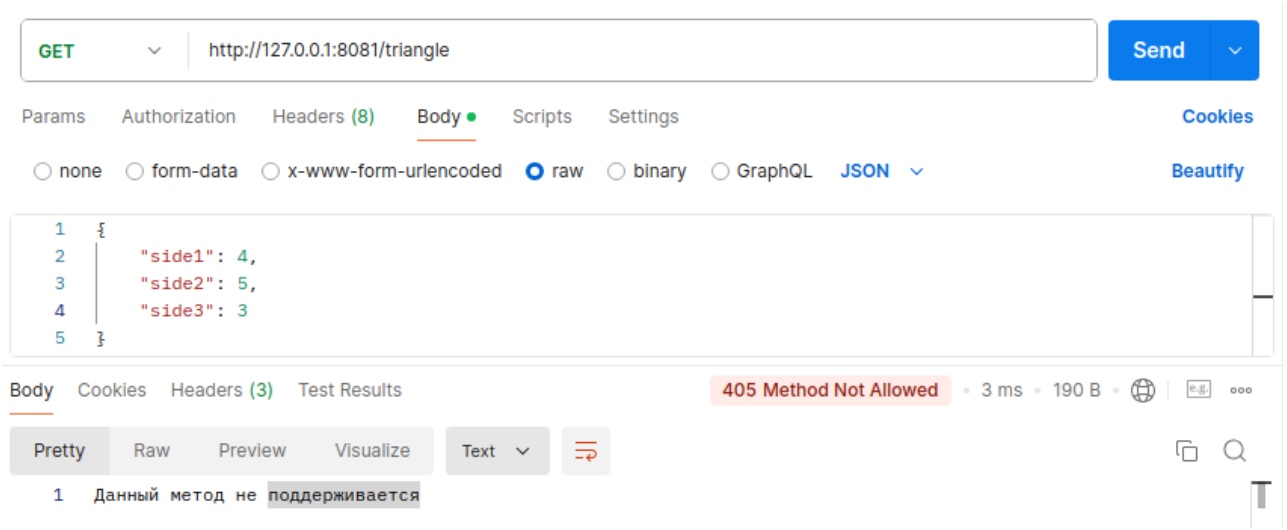

Рисунок 1 — Тестирование 1

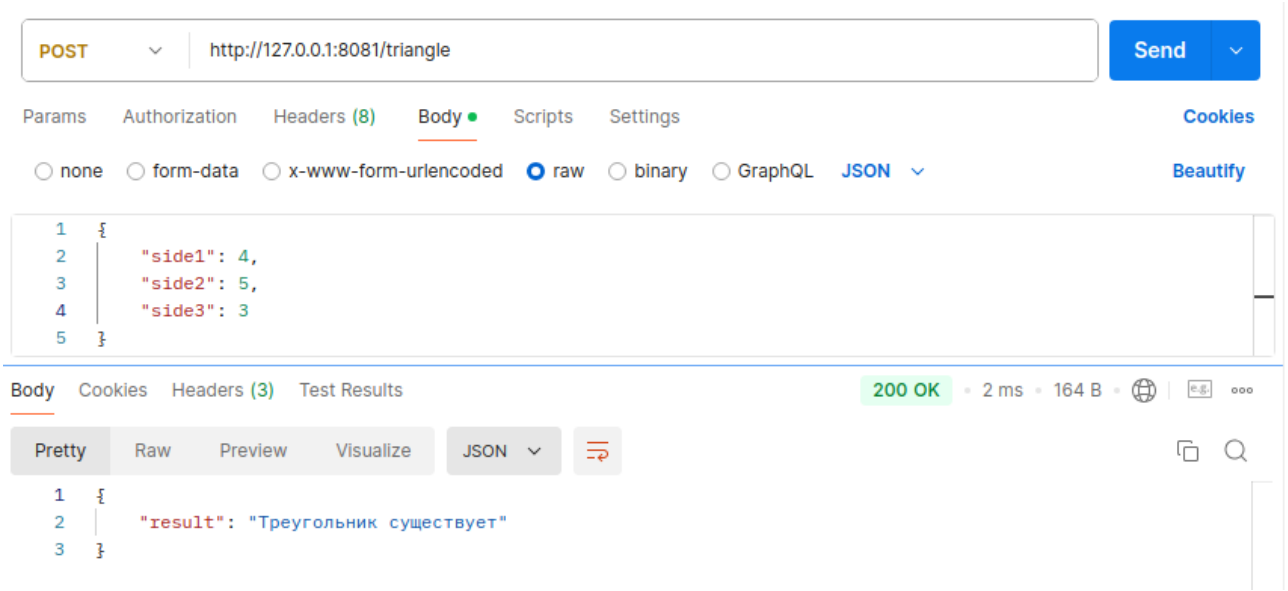

Рисунок 2 — Тестирование 2

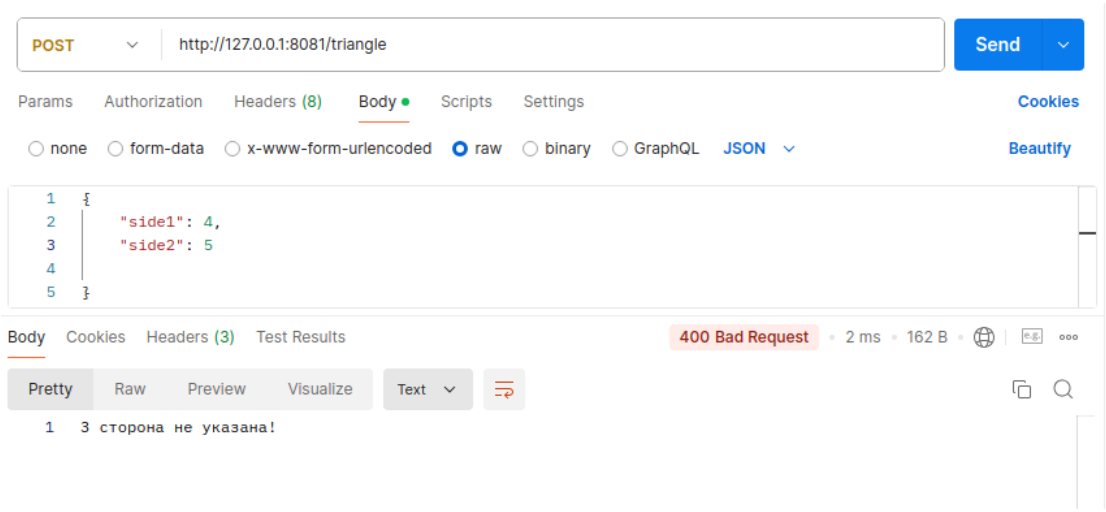

Рисунок 3 — Тестирование 3

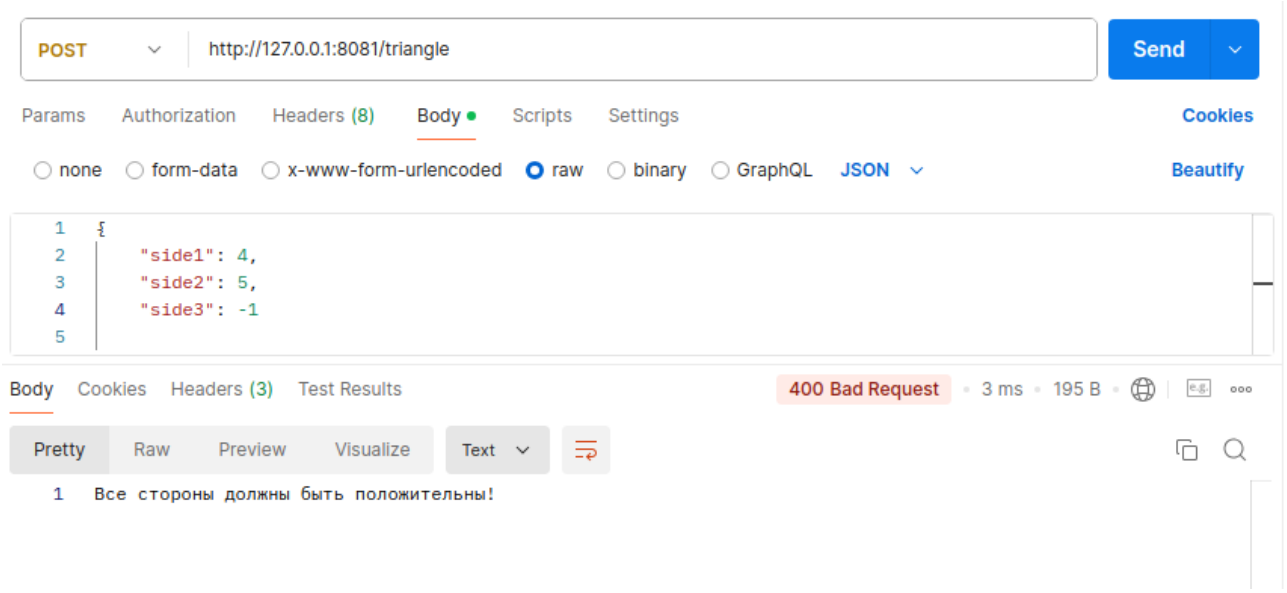

Рисунок 4 — Тестирование 4

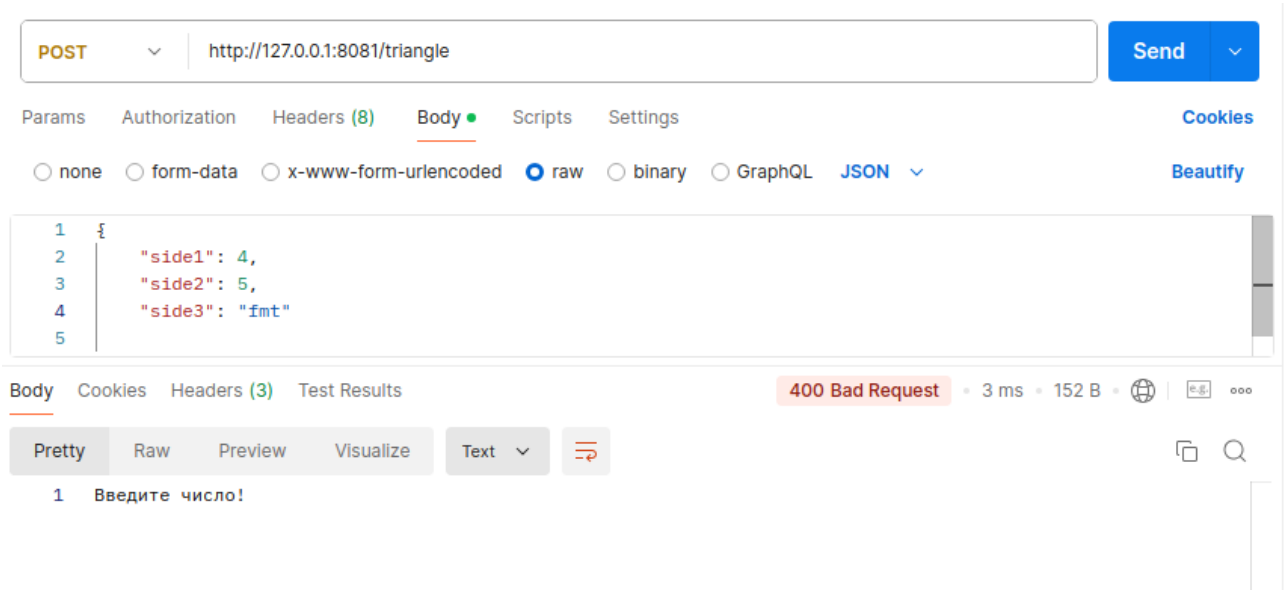

Рисунок 5 — Тестирование 5

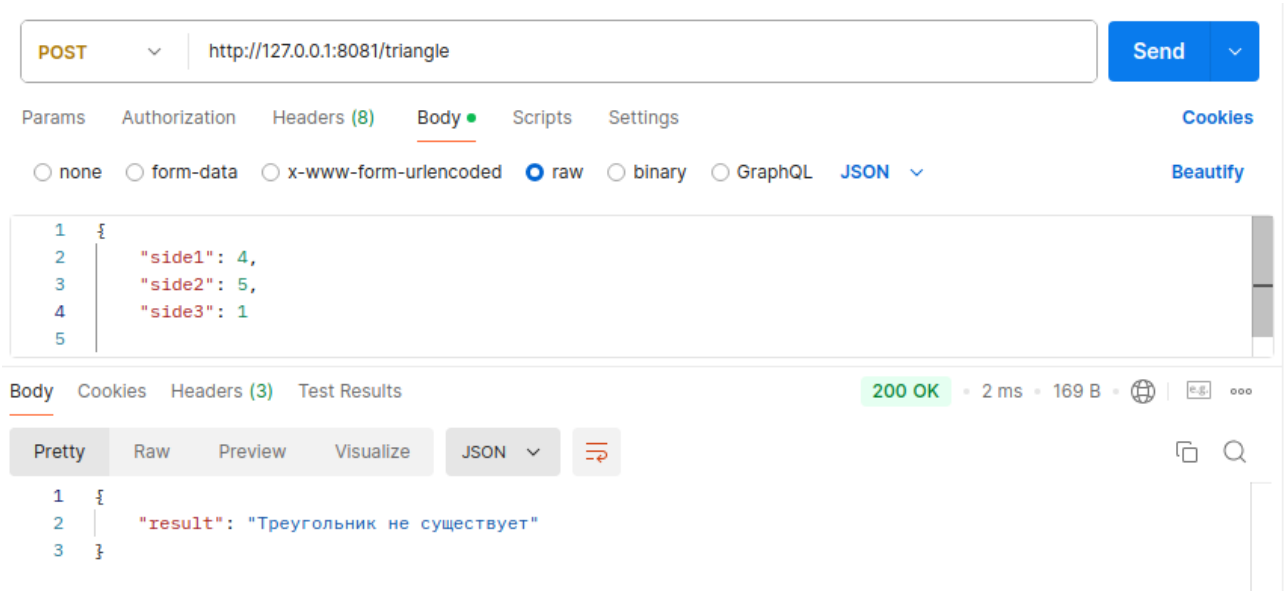

Рисунок 6 — Тестирование 6

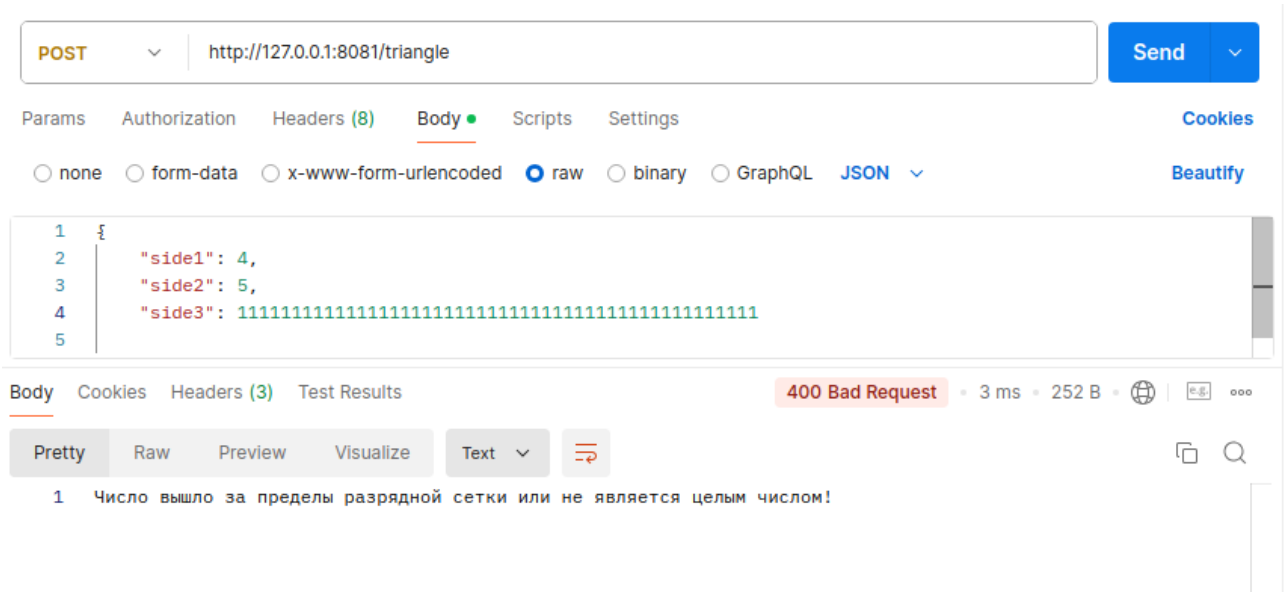

Рисунок 7 — Тестирование 7

Загрузим решения на GitHub и сделаем Pull request из dev в мастер с помощью интерфейса GitHub.

Заключение:

Язык программирования Golang позволяет полноценно работать с сетью. Например, есть возможность создать веб-сервер без подключения дополнительных сторонних библиотек с возможностью обработки HTTP запросов.