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)**

О Т Ч Е Т

по лабораторной работе № 6

Название: **Основы Back-End разработки на Golang**

Дисциплина: **Языки интернет программирования**

Студент	ИУ6-32Б		Кирикович М.А
	(Группа)	(Подпись, дата)	(И.О. Фамилия)
Преподаватель			И.О. Фамилия
		(Подпись, дата)	(И.О. Фамилия)

Москва, 2024

Цель работы — изучение основ сетевого взаимодействия и серверной разработки с использованием языка Golang.

Часть 1

Напишите веб сервер, который по пути /get отдает текст "Hello, web!".

Порт должен быть :8080.

Рисунок 1: Задание

```go
package main

// здесь надо написать код
import (
	"fmt"
	"net/http"
)

func handler(w http.ResponseWriter, r *http.Request) {
	w.Write([]byte("hello,web!"))
}

func main() {
	http.HandleFunc("/get", handler)

	err := http.ListenAndServe(":8083", nil)
	if err != nil {
		fmt.Println("ошибка запуска сервера")
	}
}

```

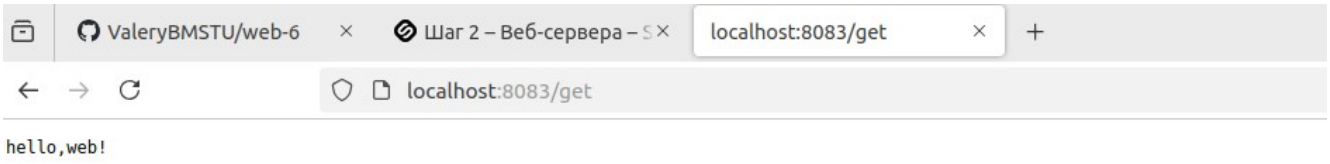

Рисунок 2: Результат

Часть 2

Напишите веб-сервер который по пути /api/user приветствует пользователя:
Принимает и парсит параметр `name` и делает ответ "Hello,<name>!"
Пример: /api/user?name=Golang
Ответ: Hello,Golang!

порт :9000

Рисунок 3: Задание 2

```
package main

import (
	"fmt"
	"net/http"
)

func handler(w http.ResponseWriter, r *http.Request) {
	inputname := r.URL.Query().Get("name")
	w.Write([]byte("Hello," + inputname))

}

func main() {
	http.HandleFunc("/api/user", handler)
	err := http.ListenAndServe(":9000", nil)
	if err != nil {
		fmt.Println("ошибка запуска сервера")
	}
}
```

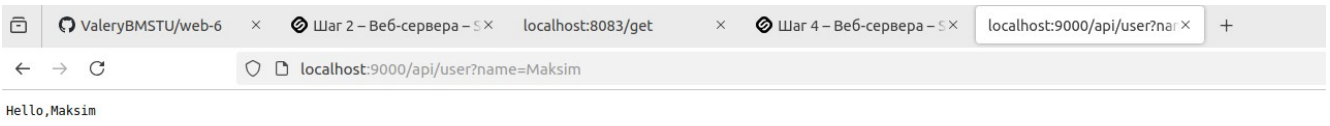

Рисунок 4: Результат тестирования

Часть 3

Напиши веб сервер (**порт :3333**) - счетчик который будет обрабатывать GET (/count) и POST (/count) запросы:

GET: возвращает счетчик
POST: увеличивает ваш счетчик на значение (с ключом "count") которое вы получаете из формы, но если пришло НЕ число то нужно ответить клиенту: "это не число" со статусом http.StatusBadRequest (400).

Рисунок 5: задание 3

```go
package main

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

var counter int = 0

type kol struct {
	Count int `json:"count"`
}

func handler(w http.ResponseWriter, r *http.Request) {
	switch r.Method {
	case "GET":
		w.Header().Set("Content-Type", "application/json")
		json.NewEncoder(w).Encode(map[string]int{"count": counter})

	case "POST":
		var c kol
		err := json.NewDecoder(r.Body).Decode(&c)
		if err != nil {
			http.Error(w, err.Error(), http.StatusBadRequest)
		}
		fmt.Println(c.Count)
		counter += c.Count
	default:
		http.Error(w, "неправильный запрос", 405)
	}
}
func main() {
	http.HandleFunc("/count", handler)

	err := http.ListenAndServe(":3333", nil)
	if err != nil {
		fmt.Println("Ошибка запуска сервера")
	}
}

```

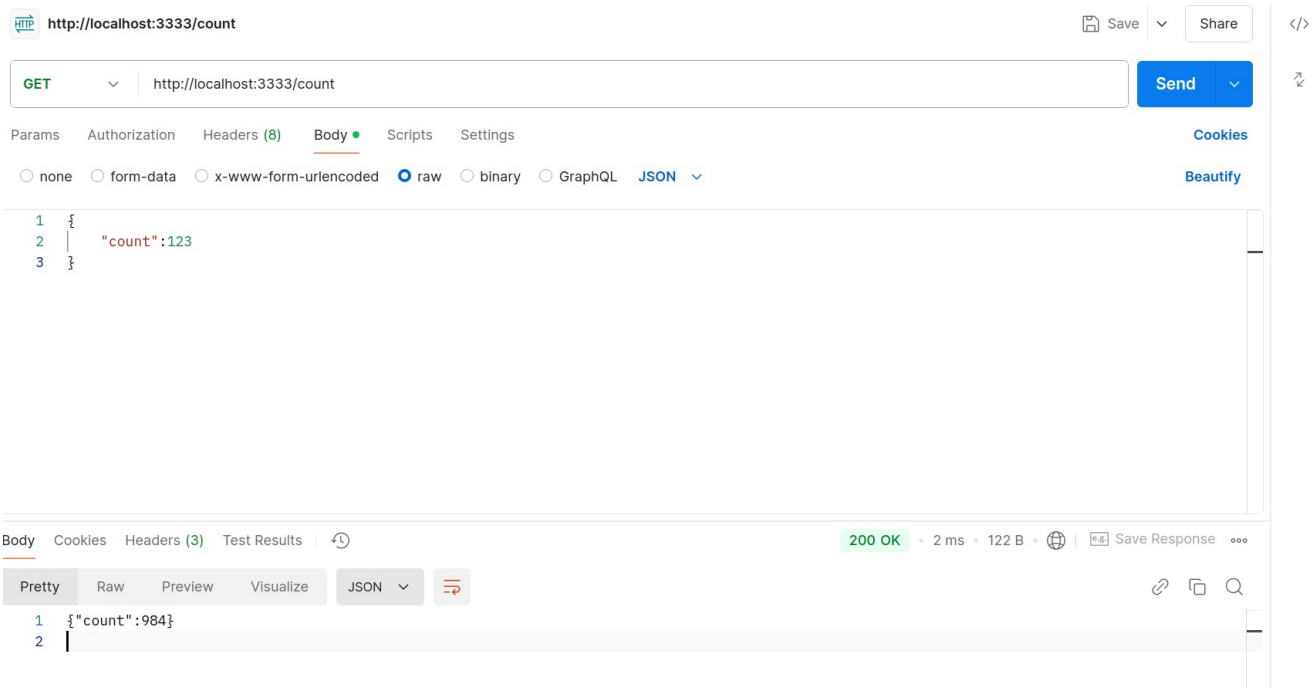

Рисунок 6: пример GET запроса

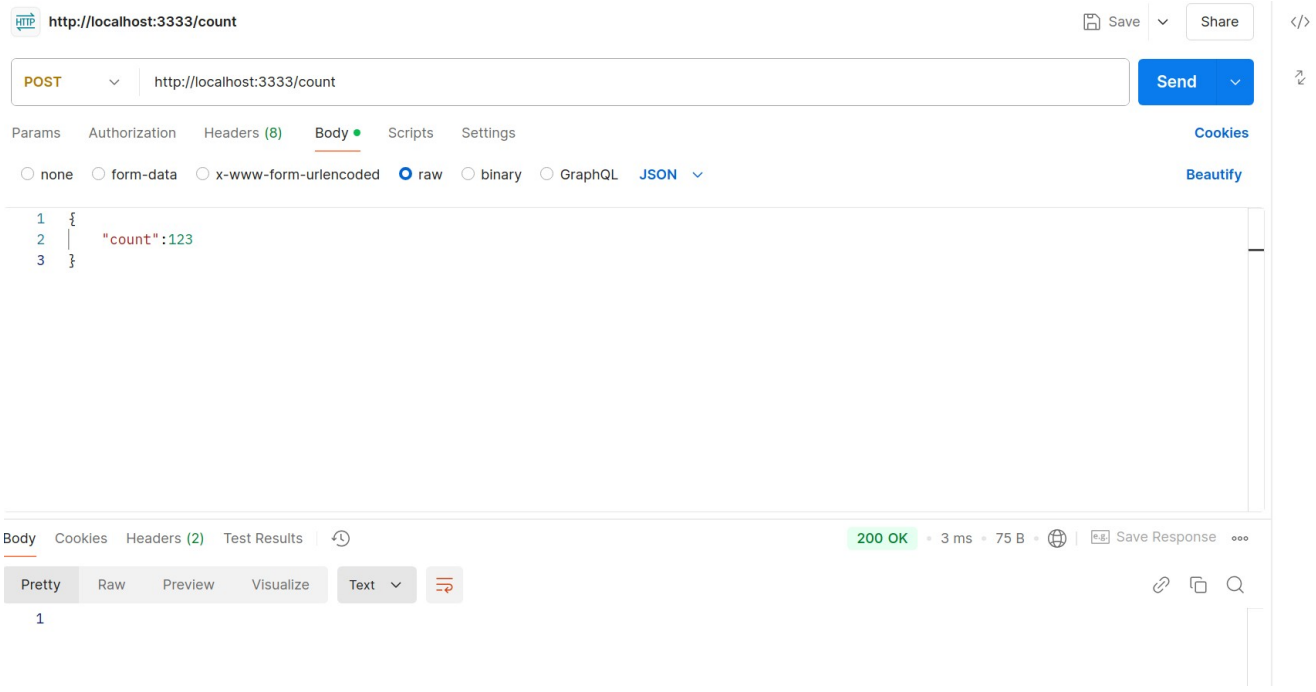

Рисунок 7: Пример POST запроса