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 Программно-аппаратное обеспечение систем искусственного интеллекта**

О Т Ч Е Т

по лабораторной работе № 6

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

Дисциплина: Языки интернет программирования

Студент ИУ6-33Б _____ _____ С.И.Козярская
(Группа) (Подпись, дата) (И.О. Фамилия)

Преподаватель _____ _____
(Подпись, дата) (И.О. Фамилия)

Москва, 2024

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

Ход работы:

1. 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
   Далее представлены программный код и результат выполнения (рисунок 1)

```go
package main

// некоторые импорты нужны для проверки
import (
    "fmt"
    "net/http" // пакет для поддержки HTTP протокола
)

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

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

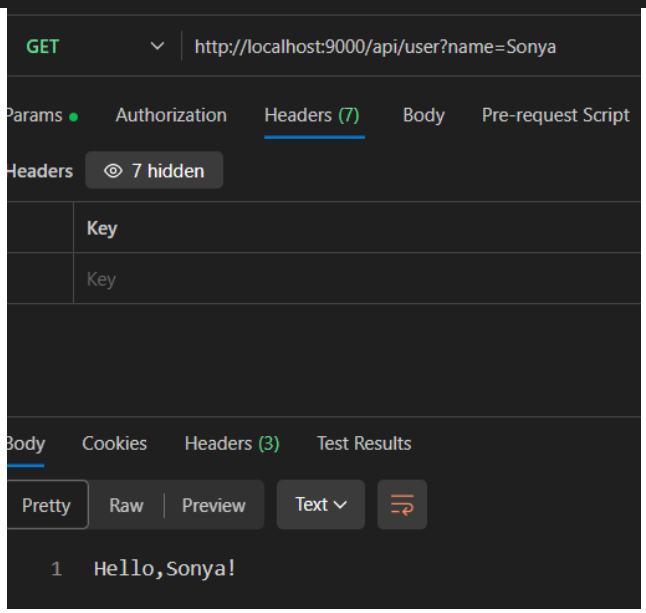

Рисунок 1- задача 1 код и вывод

2. Напишите веб сервер, который по пути /get отдает текст "Hello, web!". Порт должен быть :8080.
   Далее представлены программный код и результат выполнения (рисунок 2)

```go
package main

// некоторые импорты нужны для проверки
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(":8080", nil)
    if err != nil {
        fmt.Println("Ошибка запуска сервера:", err)
    }
}
```

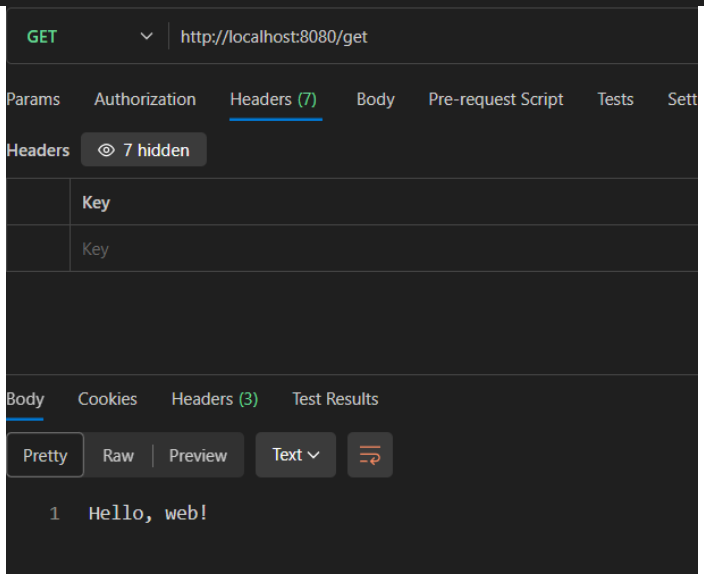

Рисунок 2- задача 2 код и вывод

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).

Далее представлены программный код и результат выполнения в postman (рисунки 3-5)

```go
package main

import (
    "fmt"
    "net/http"
    "strconv"
)

var count int = 0

func handler(w http.ResponseWriter, r *http.Request) {
    switch r.Method {
    case http.MethodGet:
        w.Write([]byte(strconv.Itoa(count)))
    case http.MethodPost:
        r.ParseForm()
        num_str := r.Form.Get("count")
        num, err := strconv.Atoi(num_str)
        if err != nil {
            w.WriteHeader(http.StatusBadRequest)
            w.Write([]byte("это не число"))
            return
        }
        count += num
    }
}

func main() {
    http.HandleFunc("/count", handler)
    err := http.ListenAndServe(":3333", nil)
    if err != nil {
        fmt.Println("Ошибка запуска сервера:", err)
    }
}
```

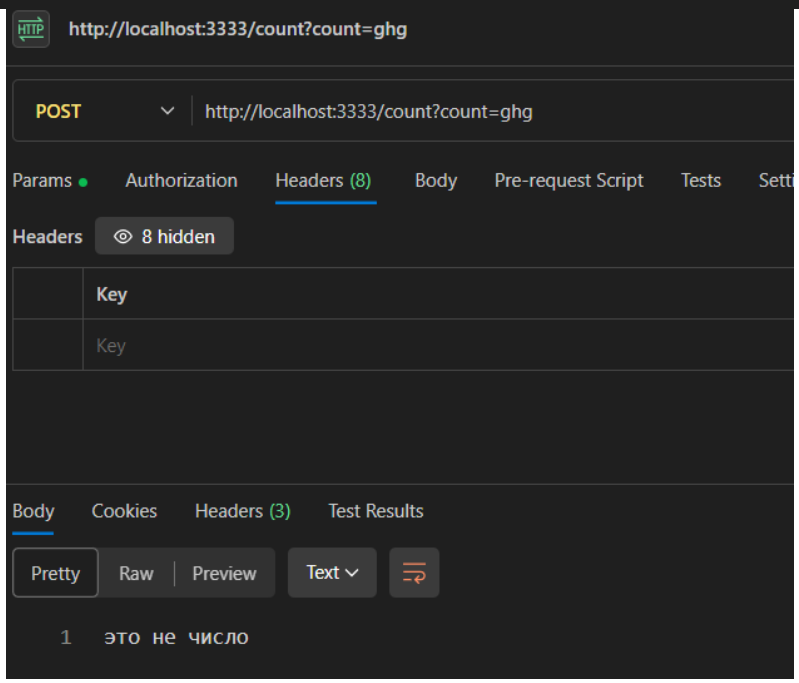

Рисунок 3-POST вывод если не число

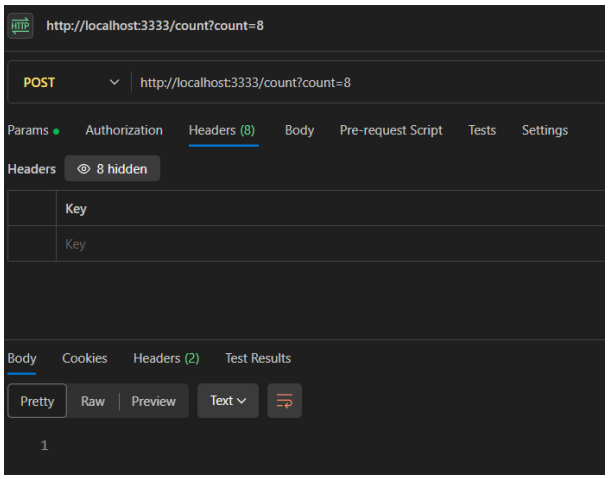

Рисунок 4- POST правильное выполнение

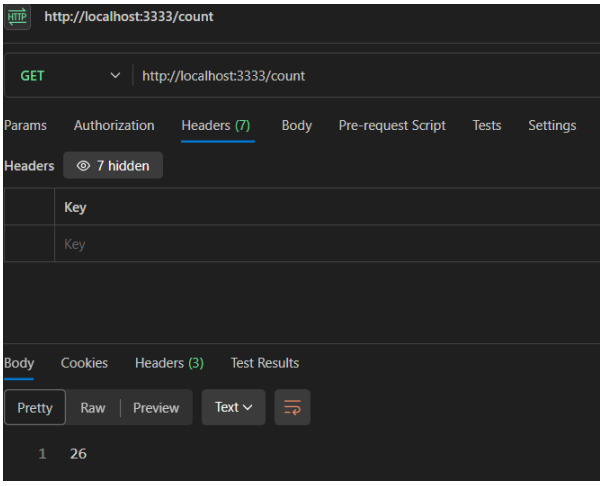

Рисунок 5 - GET вывод результата

Заключение : в результате проделанной работы были получены теоретические знания и практические навыки основ сетевого взаимодействия и серверной разработки с использованием языка Golang. Были решены 3 задачи, с проверкой через POSTMAN

Результаты выполнения работы были отправлены на GitHub