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 лабораторной работы: Основы Back-End разработки на Golang

Дисциплина: Языки интернет программирования

Студент гр. ИУ6-32Б _____ **Суворов Вакао А.**
(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.

В рамках данной лабораторной работы предлагается продолжить изучение Golang и познакомиться с набором стандартных библиотек, используемых для организации сетевого взаимодействия и разработки серверных приложений.

Задание 1_hello

Код:

```
package main

import (
    "fmt"
    "net/http"
)

func handler(w http.ResponseWriter, r *http.Request) {
    if r.URL.Path != "/get" {
       http.NotFound(w, r)
       return
    }
    fmt.Fprint(w, "Hello, web!")
}

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

    fmt.Println("Сервер запущен на порту :8080")
    if err := http.ListenAndServe(":8080", nil); err != nil {
       fmt.Println("Ошибка при запуске сервера:", err)
    }
}
```

Запускаем в Postman (рис. 1)

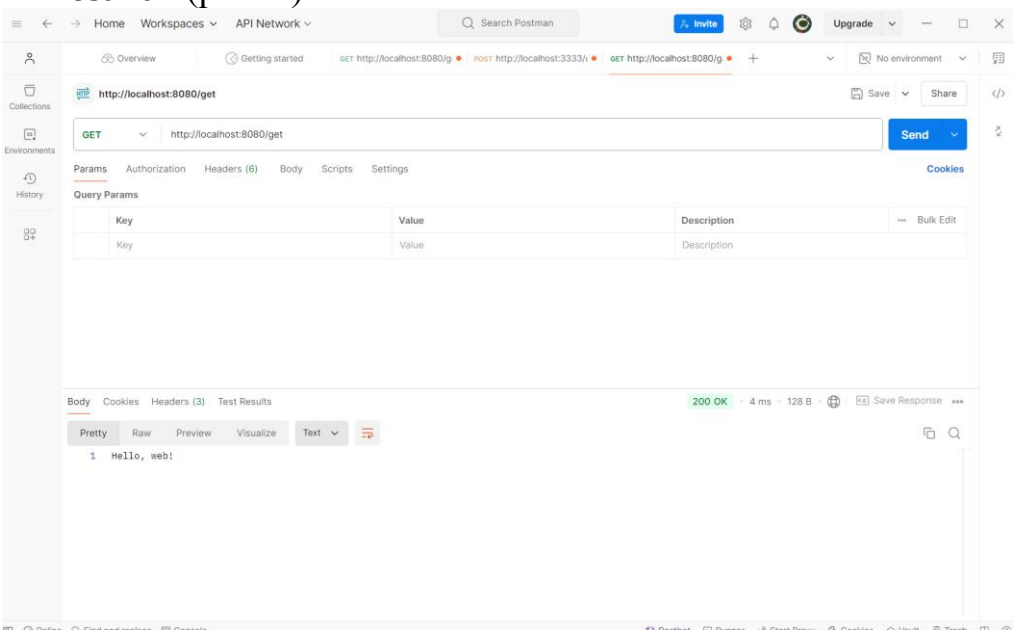

Рис. 1

Задание 2_query

Код:

```
package main

import (
    "fmt"
    "net/http"
)

// Обработчик HTTP-запросов
func handler(w http.ResponseWriter, r *http.Request) {
    w.Write([]byte("Hello " + r.URL.Query().Get("name") + "!"))
}

func main() {
    // Регистрируем обработчик для пути "/"
    http.HandleFunc("/api/user", handler)

    // Запускаем веб-сервер на порту 8080
    fmt.Println("starting server...")
    err := http.ListenAndServe(":9000", nil)
    if err != nil {
        fmt.Println("Ошибка запуска сервера:", err)
    }
}
```

Запускаем в Postman (рис. 2)

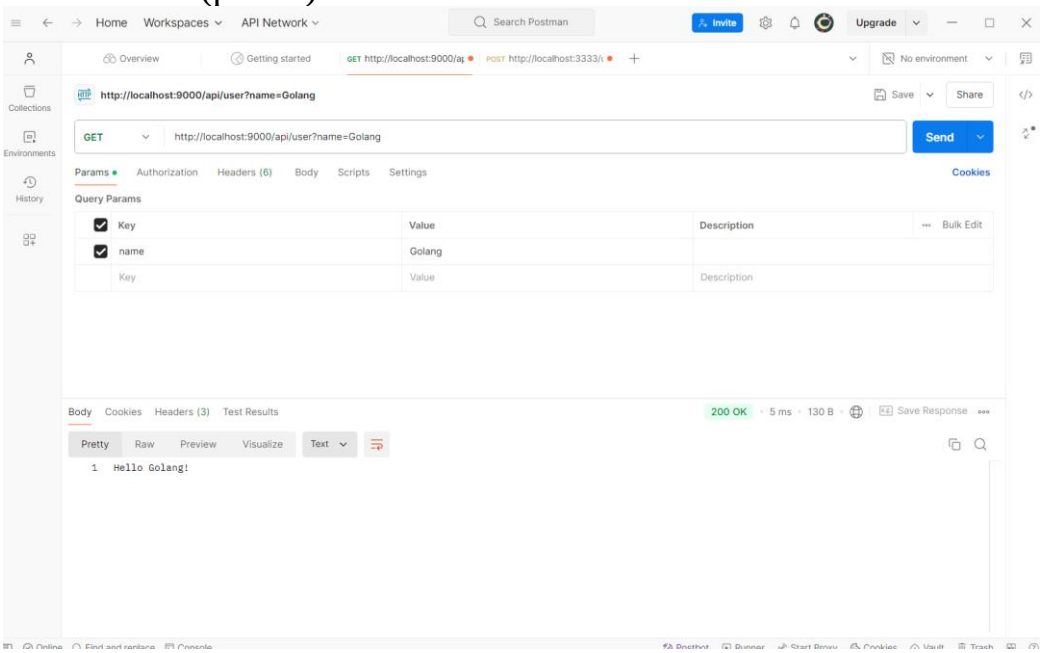

Рис. 2

Задание 3_count
Код:

```
package main

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

// Счетчик и мьютекс для потокобезопасного доступа к счетчику
var (
    counter int
    mu      sync.Mutex
)

func countHandler(w http.ResponseWriter, r *http.Request) {
    switch r.Method {
    case http.MethodGet:
       // Отправляем текущее значение счетчика
       mu.Lock()
       defer mu.Unlock()
       fmt.Fprintf(w, "Счетчик: %d", counter)
    case http.MethodPost:
       // Пытаемся получить значение count из формы
       r.ParseForm()
       countStr := r.FormValue("count")

       // Преобразование строки в число
       count, err := strconv.Atoi(countStr)
       if err != nil {
          http.Error(w, "Это не число", http.StatusBadRequest)
          return
       }

       // Увеличиваем счетчик
       mu.Lock()
       counter += count
       mu.Unlock()

       // Подтверждение успешного добавления
       fmt.Fprintf(w, "Счетчик увеличен на %d. Текущее значение:
%d", count, counter)
    default:
       http.Error(w, "Метод не поддерживается",
http.StatusMethodNotAllowed)
    }
}

func main() {
    // Определяем обработчик для пути /count
    http.HandleFunc("/count", countHandler)
```

```
    fmt.Println("Сервер запущен на порту :3333")
    // Запускаем сервер на порту 3333
    if err := http.ListenAndServe(":3333", nil); err != nil {
        fmt.Println("Ошибка при запуске сервера:", err)
    }
}
```

Сделаем cURL запрос в Постмане используя команду:
«curl -X POST -d "count=5" http://localhost:3333/count» (рис. 3)

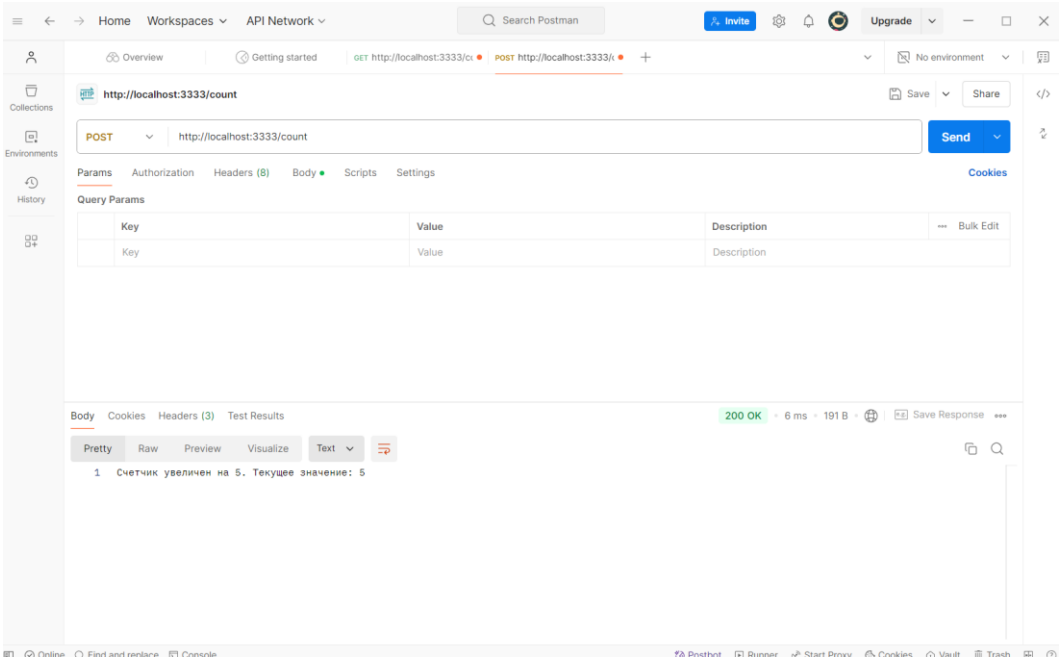

Рис. 3

Сделаем cURL запрос в Постмане используя команду:
«curl -X POST -d "count=abc" http://localhost:3333/count» (рис. 4)

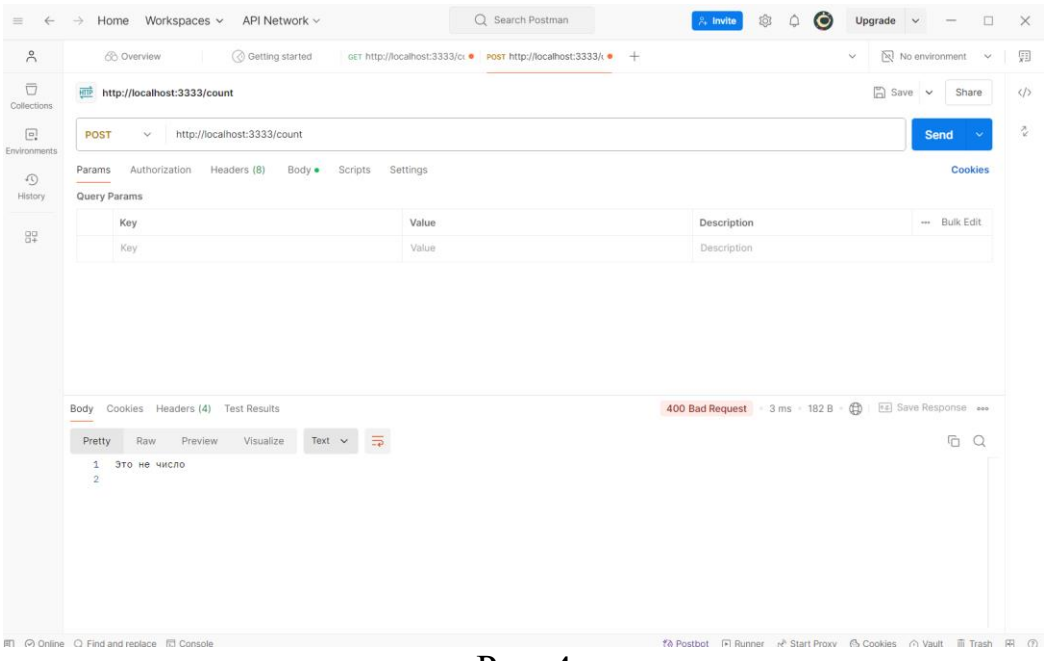

Рис. 4

Вывод: научился работать в postman и с веб серверами на go.