



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 лабораторной работе № 6

Название: Основы Back-End разработки на Golang

Дисциплина: Основы web-программирования

Студент

ИУ6-31Б

(Группа)

Минбулатов
А.А.

(Подпись, дата)

(И.О. Фамилия)

Преподаватель

Шульман В.Д.

(Подпись, дата)

(И.О. Фамилия)

Москва, 2024

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

Задание

1. Ознакомиться с разделом "4. Списки, сеть и сервера" курса
2. Сделать форк данного репозитория в GitHub, клонировать получившуюся копию локально, создать от мастера ветку dev и переключиться на неё
3. Выполнить задания. Ссылки на задания можно найти в README-файлах в директории projects
4. (опционально) Проверить свой коды линтерами с помощью команды `make lint`
5. Сделать отчёт и поместить его в директорию docs
6. Зафиксировать изменения, сделать коммит и отправить получившееся состояние ветки dev в личный форк данного репозитория в GitHub
7. Через интерфейс GitHub создать Pull Request dev --> master
8. На защите лабораторной работы продемонстрировать открытый Pull Request. PR должен быть направлен в master ветку форка, а не исходного репозитория

Задачи:

1. Необходимо написать веб-сервер, который по пути `«/get»` отдает текст `«Hello, web!»`. Порт должен быть `:8080`. Код должен компилироваться, а сервер запускаться и корректно обрабатывать запросы. Для локальной отладки можно

использовать Postman или Insomnia.

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

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)`.

Код для задания 1:

```
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!")
        }
    }
}

```

Тест задания представлен на рисунке 1.

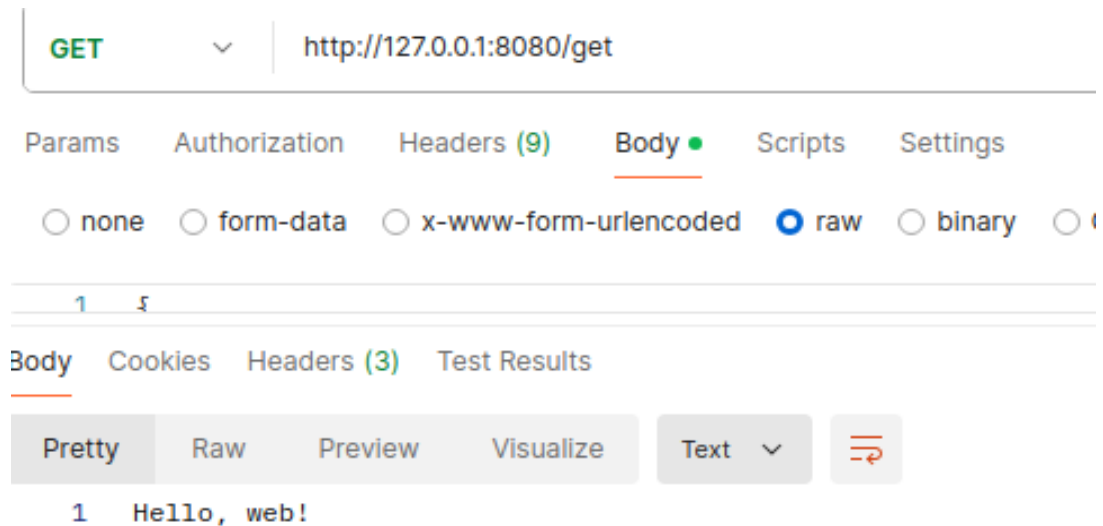


рис. 1 - тест

Код задания 2:

```

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")
    w.Write([]byte("Hello," + name + "!"))
}

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

```

Тест задания 2 представлен на рисунке 2.

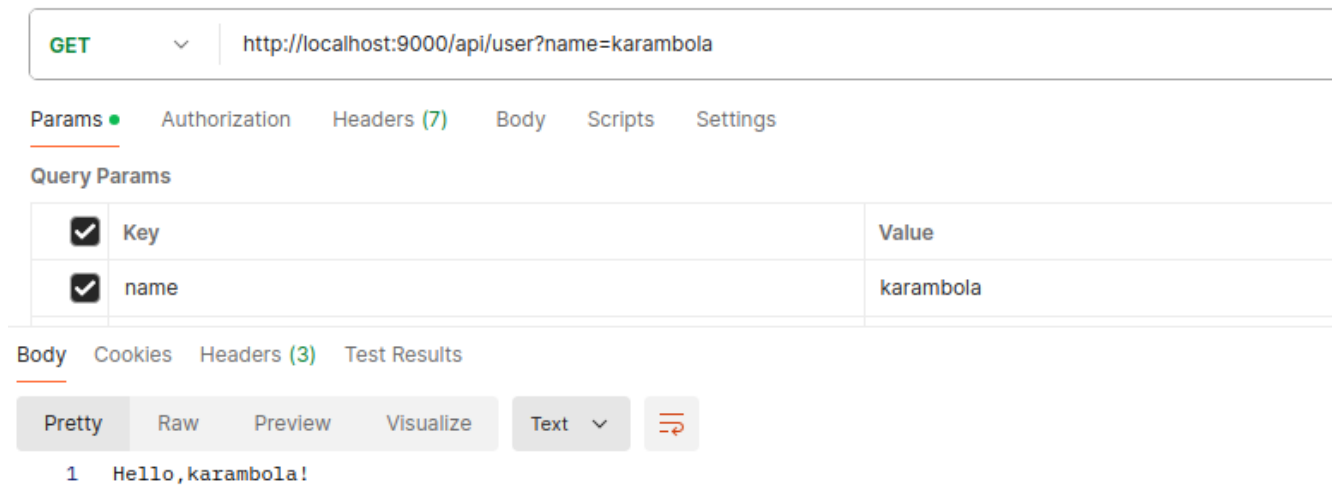


рис. 2 - тест

Код задания 3:

```
package main

// некоторые импорты нужны для проверки
import (
    "fmt"
    "net/http"
    "strconv" // вдруг понадобится вам ;)
)

var count1 int = 0

func handler(w http.ResponseWriter, r *http.Request) {
    if r.Method == "GET" {
        w.WriteHeader(http.StatusOK)
        w.Write([]byte(strconv.Itoa(count1)))
        return
    } else if r.Method == "POST" {
        r.ParseForm()
        s := r.FormValue("count")
        if s == "" {
            w.WriteHeader(http.StatusBadRequest)
            w.Write([]byte("это не число"))
            return
        }
        number, err := strconv.Atoi(s)
        if err != nil {
            w.WriteHeader(http.StatusBadRequest)
            w.Write([]byte("это не число"))
            return
        }
        count1 += number
        return
    } else {
        w.WriteHeader(http.StatusMethodNotAllowed)
        w.Write([]byte("Метод не поддерживается"))
        return
    }
}

func main() {
    http.HandleFunc("/count", handler)
    err := http.ListenAndServe(":3333", nil)
```

```

    if err != nil {
        fmt.Println("Ошибка запуска сервера!")
    }
}

```

Тест задания представлен на рисунках 3-5.

POST
⌵
http://localhost:3333/count?count=15

Params
Authorization
Headers (8)
Body
Scripts
Settings

Query Params

<input checked="" type="checkbox"/>	Key	Value
<input checked="" type="checkbox"/>	count	15
	Key	Value

рис. 3 - тест

POST
⌵
http://localhost:3333/count?count=1234

Params
Authorization
Headers (8)
Body
Scripts
Settings

Query Params

<input checked="" type="checkbox"/>	Key	Value
<input checked="" type="checkbox"/>	count	1234
	Key	Value

рис. 4 - тест

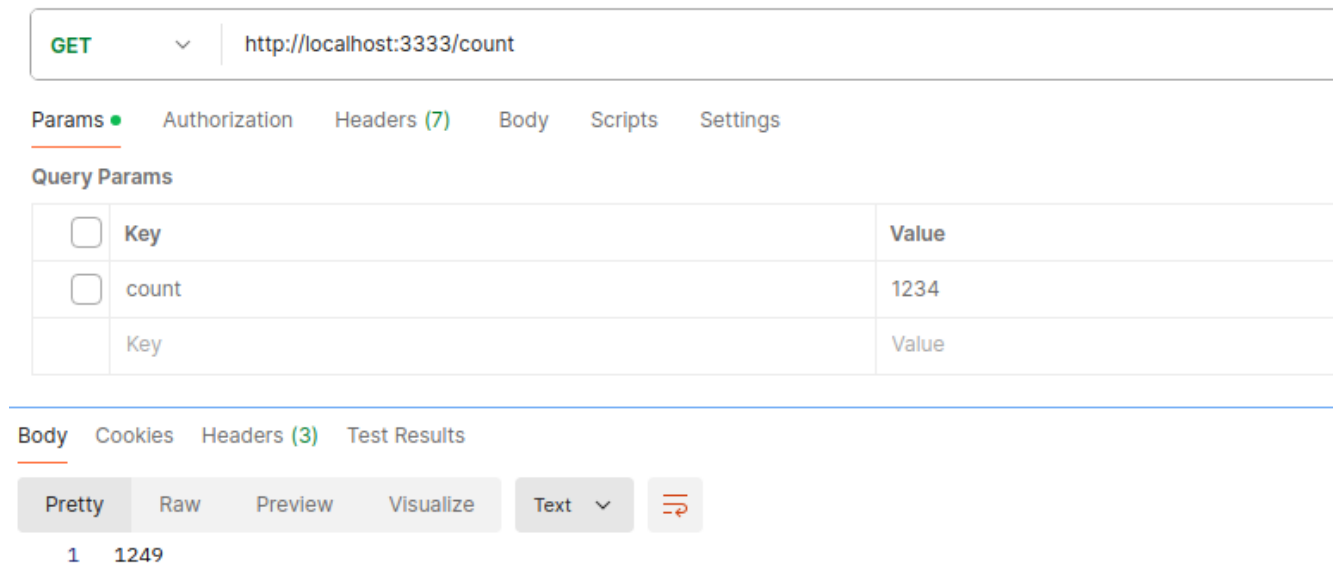


рис. 5 — тест

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