



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: Языки интернет-программирования

Студент

ИУ6-31Б  
(Группа)

(Подпись, дата)

К.С. Доронина  
(И.О. Фамилия)

Преподаватель

(Подпись, дата)

В.Д. Шульман  
(И.О. Фамилия)

Москва, 2024

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

**Порядок выполнения:**

1. Ознакомиться с 4 разделом курса на Stepik
2. Сделать форк репозитория
3. Выполнить задания projects
4. Проверить свой код линтерами
5. Сделать отчет
6. Зафиксировать изменения
7. Создать Pull Request

**Выполнение:**

Задание 1 – hello (рис. 1)

Необходимо написать веб-сервер, который по пути `/get` отдает текст `"Hello, web!"`. Порт должен быть `:8080`.

После решения задания полученный код `main.go` необходимо перенести в данный репозиторий.

Код должен компилироваться, а сервер запускаться и корректно обрабатывать запросы.

Для локальной отладки можно использовать Postman или Insomnia.

Рисунок 1 – текст задания 1

Код программы:

```
package main

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)
}
}

```

Пример выполнения в Postman (рис. 2)

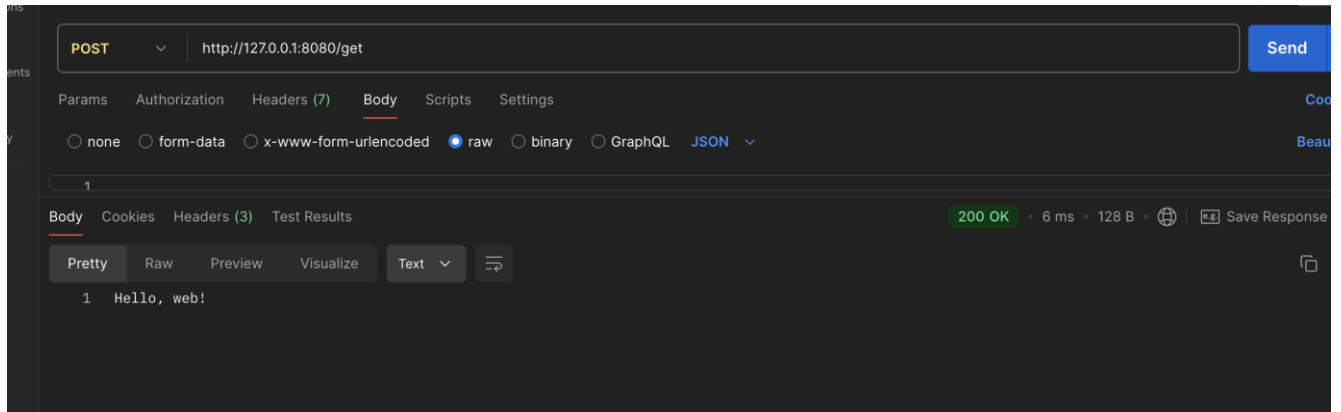


Рисунок 2 – результат в Postman

Задание 2 – query (рис. 3)

Необходимо написать веб-сервер, который по пути `/api/user` приветствует пользователя. Сервер по этому пути должен принимать и парсить параметр `name`, после этого отвечая в формате: `"Hello, <name>!"`. Пример url: `/api/user?name=Golang`

После решения задания полученный код `main.go` необходимо перенести в данный репозиторий в директорию с данным файлом README.md

Код должен компилироваться, а сервер запускаться и корректно обрабатывать запросы.

Для локальной отладки можно использовать Postman или Insomnia.

Рисунок 3 – текст задания 2

Код программы:

```

package main
import (
    "fmt"
    "net/http"
)

```

```

func handler(w http.ResponseWriter, r *http.Request) {
    name := r.URL.Query().Get("name")
    if name == "" {
        http.Error(w, "Нет параметра", http.StatusBadRequest)
        return
    }
    fmt.Fprintf(w, "Hello,%s!", name)
}

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

```

Пример выполнения в Postman (рис.4)

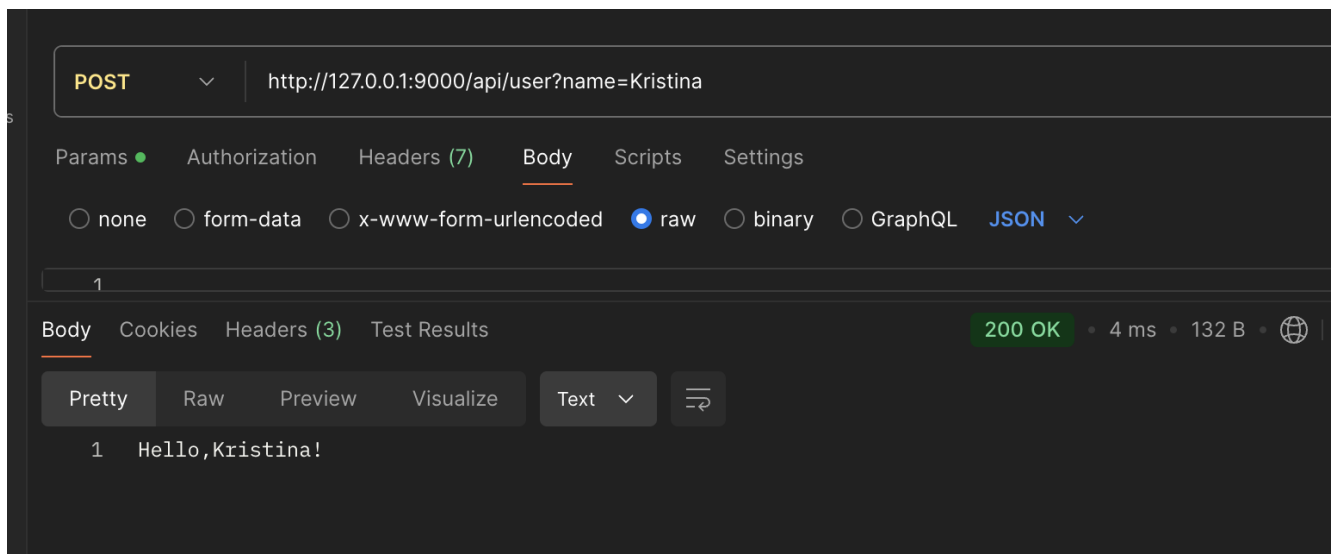


Рисунок 4 – пример выполнения

Задание 3 – count (рис. 5)

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).

После решения задания полученный код `main.go` необходимо перенести в данный репозиторий в директорию с данным файлом README.md

Код должен компилироваться, а сервер запускаться и корректно обрабатывать запросы.

Для локальной отладки можно использовать Postman или Insomnia.

Рисунок 5 – текст задания 3

Код программы:

```
package main
import (
    "fmt"
    "log"
    "net/http"
    "strconv"
)
var globCount int
func main() {
    http.HandleFunc("/", handleRequest)
    err := http.ListenAndServe(":3333", nil)
    if err != nil {
        log.Println(err)
    }
}
func handleRequest(w http.ResponseWriter, r *http.Request) {
    switch r.Method {
    case http.MethodGet:
        handleGet(w, r)
    case http.MethodPost:
        handlePost(w, r)
    }
}
func handleGet(w http.ResponseWriter, r *http.Request) {
```

```

fmt.Println(globCount)
}

func handlePost(w http.ResponseWriter, r *http.Request) {
intCount, err := strconv.Atoi(r.FormValue("count"))
if err != nil {
w.WriteHeader(http.StatusBadRequest)
w.Write([]byte("это не число"))
} else {
globCount += intCount
}
}
}

```

Пример выполнения в Postman (рис. 6)

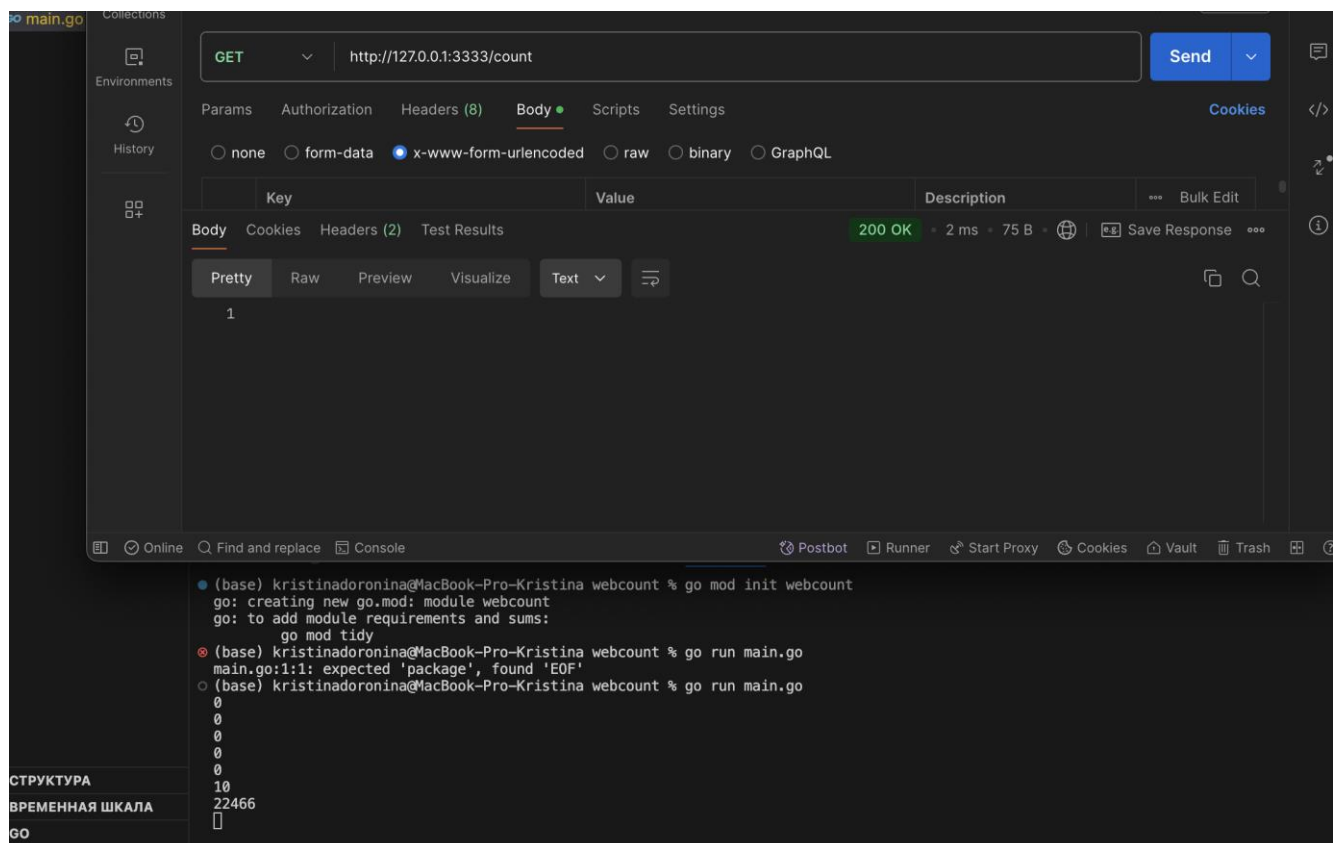


Рисунок 6 – пример выполнения

### Контрольные вопросы:

1. В чём разница между протоколами TCP и UDP ?

TCP в отличие от UDP гарантирует доставку данных в правильном порядке и без потерь.

2. Для чего нужны IP Address и Port Number у веб-сервера и в чём разница?

IP Address - это уникальный адрес, который идентифицирует устройство в сети. Он необходим для маршрутизации данных между устройствами.

Port Number - это число, которое указывает на конкретное программное приложение или службу на устройстве, использующем IP-адрес.

IP-адрес указывает на устройство в сети, а номер порта указывает на конкретное приложение или службу на этом устройстве.

3. Какой набор методов в HTTP-request в полной мере реализует семантику CRUD ?

POST, GET, PUT, DELETE

4. Какие группы status code существуют у HTTP-response (желательно, с примерами) ?

1xx (Информационные):

- 100 Continue - сервер получил начальные данные от клиента и ждет дальнейшие.

2xx (Успех):

- 200 OK - запрос успешно выполнен.
- 201 Created - ресурс был успешно создан.

3xx (Перенаправление):

- 301 Moved Permanently - ресурс был перемещен на новый URL.
- 302 Found - ресурс временно доступен по другому URL.

4xx (Ошибки клиента):

- 400 Bad Request - сервер не понял запрос из-за неверного синтаксиса.
- 404 Not Found - запрашиваемый ресурс не найден.

5xx (Ошибки сервера):

- 500 Internal Server Error - произошла ошибка на стороне сервера.
- 503 Service Unavailable - сервер недоступен для обработки запроса.

5. Из каких составных элементов состоит HTTP-request и HTTP-response ?

HTTP-request состоит из:

1. Стартовая строка
2. Заголовки (Headers) - метайнформация о запросе (Host, User-Agent, Accept).
3. Тело запроса (Body) - содержимое запроса (обычно для методов POST, PUT).

HTTP-response состоит из:

1. Стартовая строка
2. Заголовки (Headers) - метайнформация о ответе (Content-Type, Content-Length, Server).
3. Тело ответа (Body) - фактические данные, возвращаемые в ответе (HTML-страница, JSON-данные).