**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 О.В. Белякова
(Группа) (Подпись, дата) (И.О. Фамилия)

Преподаватель В.Д. Шульман
(Подпись, дата) (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)

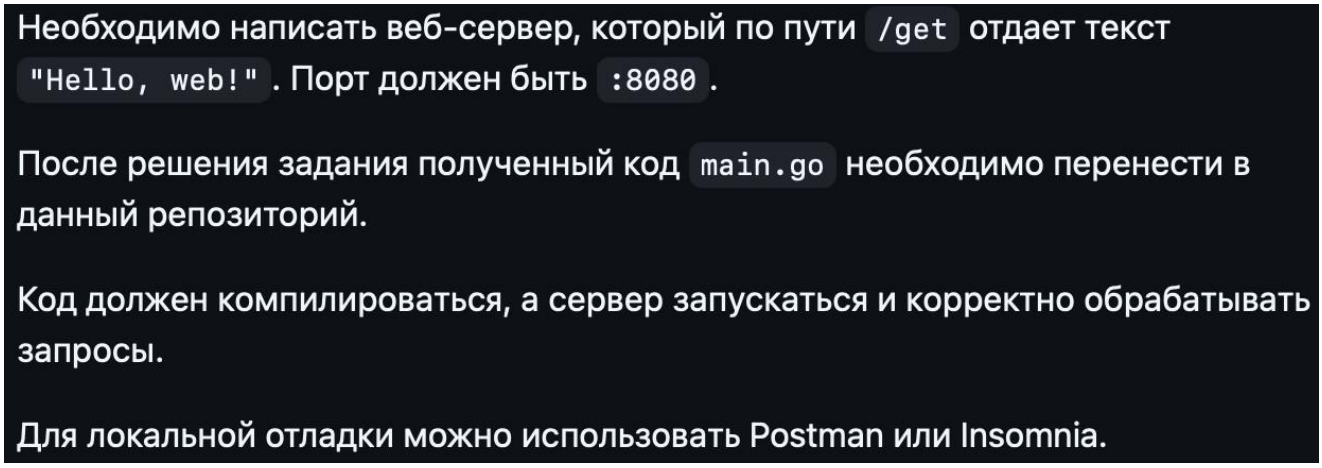


Рисунок 1 – текст задания 1

Код программы:

```
package main

import (
 "fmt"
 "net/http"
)

func main() {
 http.HandleFunc("/get", func(w http.ResponseWriter, r *http.Request) {
  fmt.Fprintln(w, "Hello, web!")
 })
```

```
 fmt.Println("Starting server on :8080...")
 http.ListenAndServe(":8080", nil)
}
```

Пример выполнения в Postman (рис. 2)

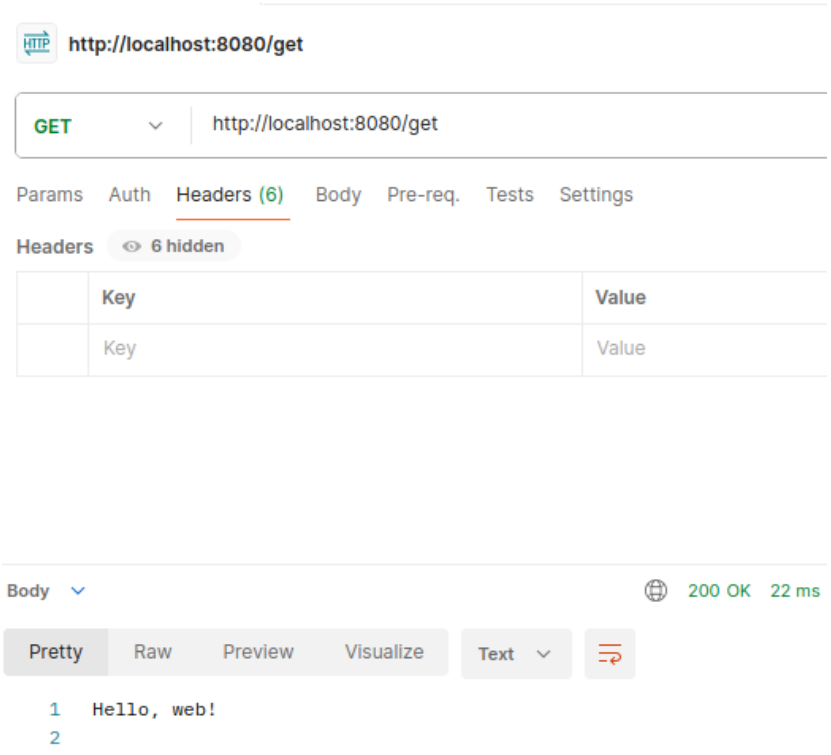


Рисунок 2 – результат в Postman

Задание 2 – query (рис. 3)

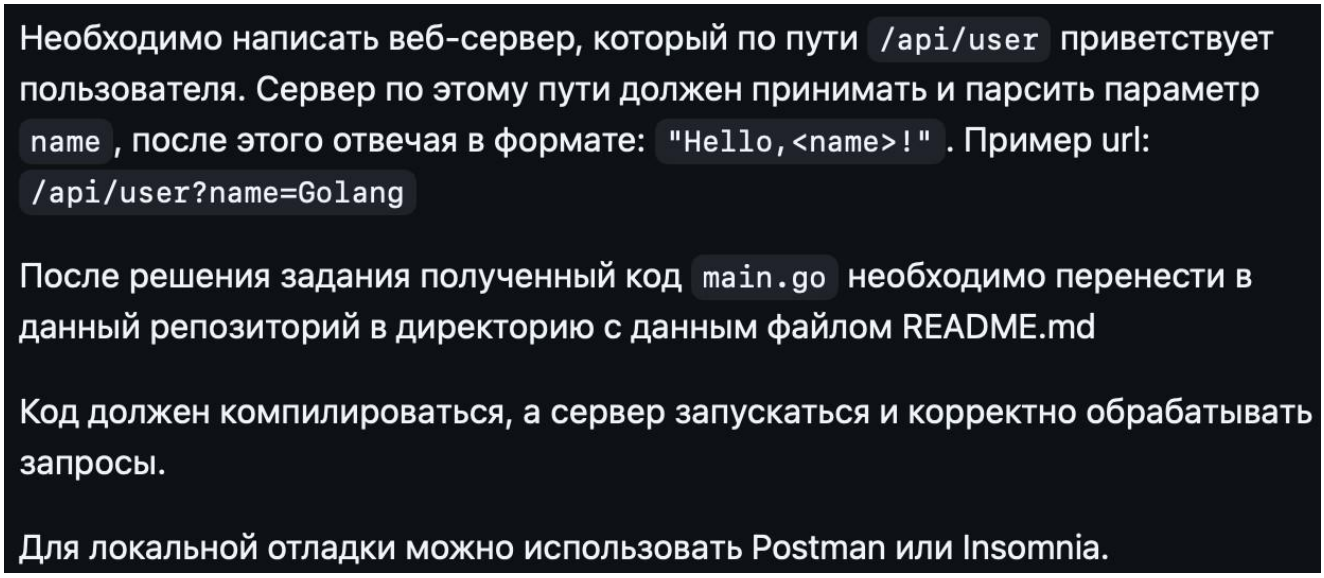


Рисунок 3 – текст задания 2

Код программы:

```
package main

import (
 "fmt"
 "net/http"
)

func main() {
 http.HandleFunc("/api/user", func(w http.ResponseWriter, r *http.Request) {
  name := r.URL.Query().Get("name")
  if name == "" {
   http.Error(w, "Name parameter is missing", http.StatusBadRequest)
   return
  }
  fmt.Fprintf(w, "Hello, %s!", name)
 })

 fmt.Println("Starting server on :8080...")
 http.ListenAndServe(":8080", nil)
}
```

Пример выполнения в Postman (рис.4)

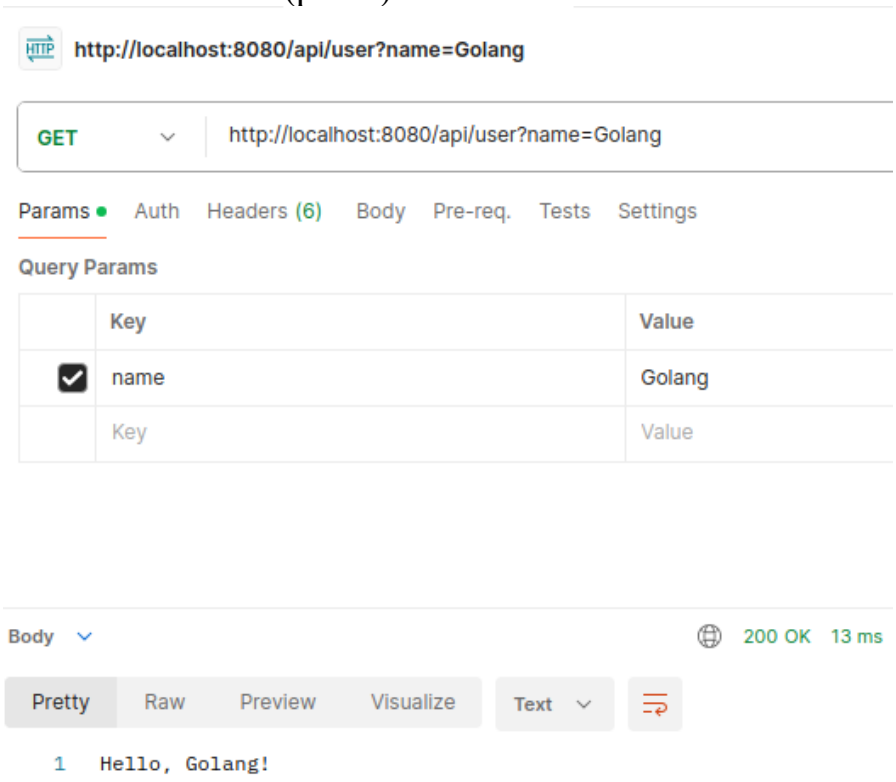


Рисунок 4 – пример выполнения

Задание 3 – count (рис. 5)

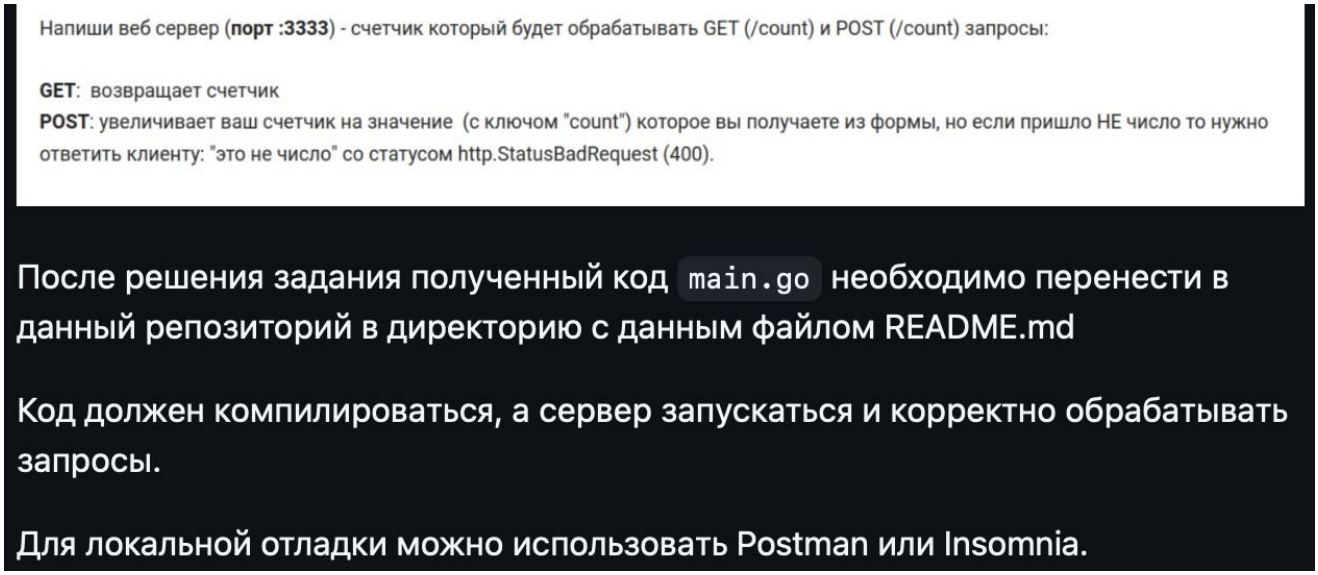


Рисунок 5 – текст задания 3

Код программы:

```
package main

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

var (
 counter int
 mu      sync.Mutex
)

func main() {
 http.HandleFunc("/count", func(w http.ResponseWriter, r *http.Request) {
  switch r.Method {
  case http.MethodGet:
   mu.Lock()
   fmt.Fprintf(w, "Counter: %d", counter)
   mu.Unlock()
  case http.MethodPost:
   if err := r.ParseForm(); err != nil {
    http.Error(w, "Invalid form data", http.StatusBadRequest)
    return
   }

   value := r.FormValue("count")
   num, err := strconv.Atoi(value)
   if err != nil {
    http.Error(w, "это не число", http.StatusBadRequest)
    return
   }

   mu.Lock()
   counter += num
   mu.Unlock()
```

```
  fmt.Fprintf(w, "Counter increased by %d", num)
 default:
  http.Error(w, "Method not allowed", http.StatusMethodNotAllowed)
 }
})

fmt.Println("Starting server on :3333...")
http.ListenAndServe(":3333", nil)
}
```

Пример выполнения в Postman (рис. 6)

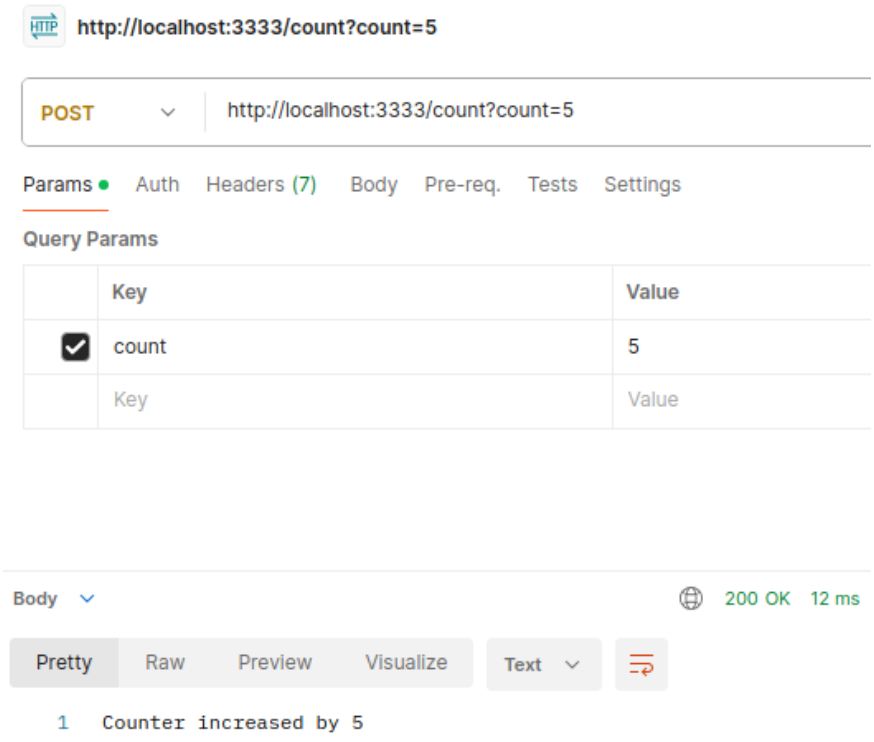


Рисунок 6 – пример выполнения

**Контрольные вопросы:**

1. В чём разница между протоколами TCP и UDP ?

TCP в отличии 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 релизует 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) - метаинформация 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) - метаинформация о ответе (Content-Type, Content-Length, Server).

3. Тело ответа (Body) - фактические данные, возвращаемые в ответе (HTML-страница, JSON-данные).