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 ИУ6-32Б (Группа) — 28.10.24 (Подпись, дата) — В.А. Баринова (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.

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

Ход работы

Задание 1. Напишите веб сервер, который по пути /get отдает текст "Hello, web!". Порт должен быть :8080.

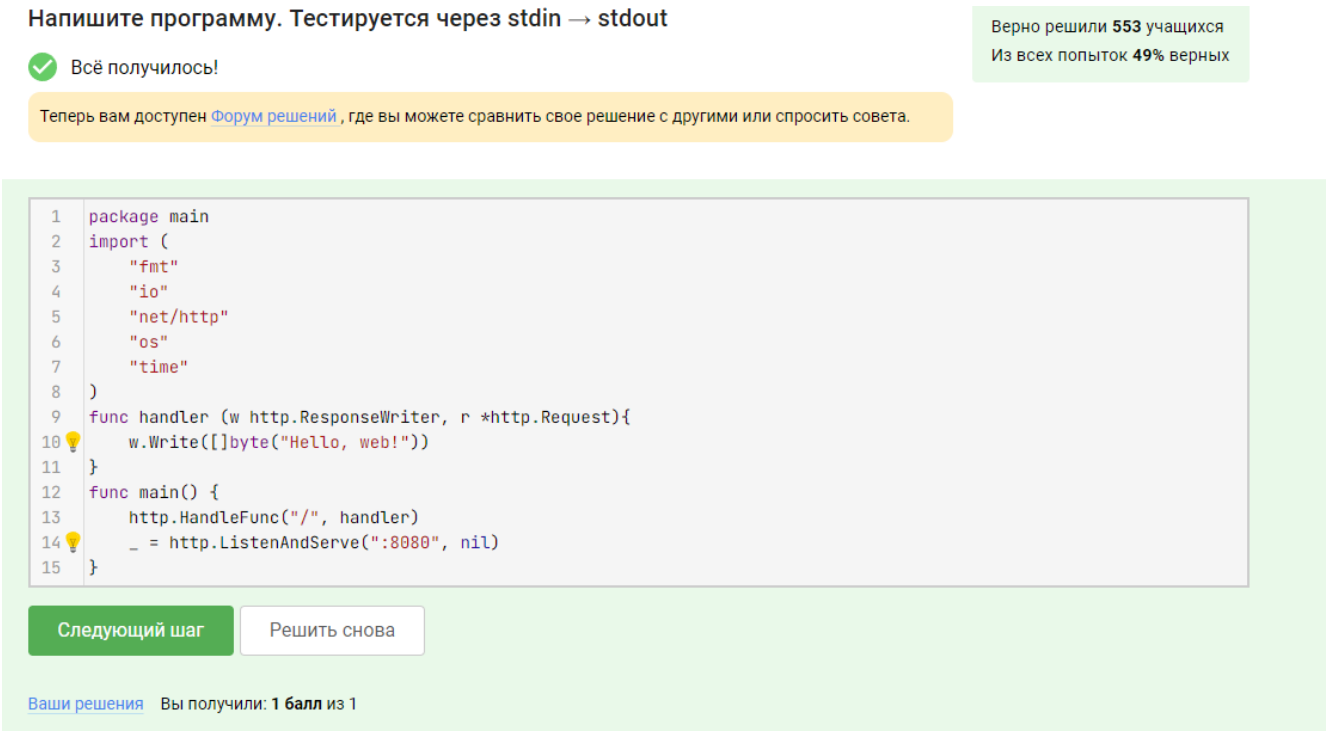

Рисунок 1—Прохождение тестов по заданию 1 на Stepik

Проведем тестирование программы, приведенной далее, с помощью ПО Postman (рис. 2):

```
package main

import (
    "net/http"
)

func handler(w http.ResponseWriter, r *http.Request) {
    w.Write([]byte("Hello, web!"))
}
func main() {
    http.HandleFunc("/", handler)
    _ = http.ListenAndServe(":8080", nil)
}
```

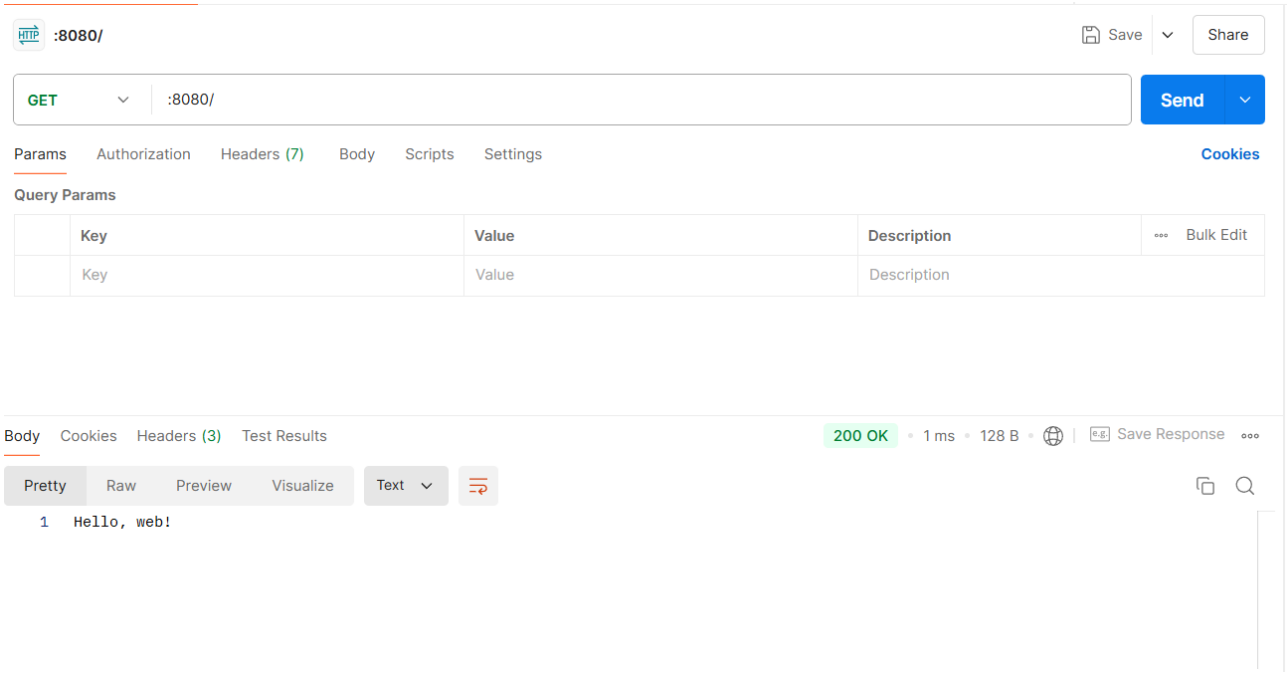

Рисунок 2—Тестирование (задание 1)

Задание 2. Напишите 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. Порт :9000.

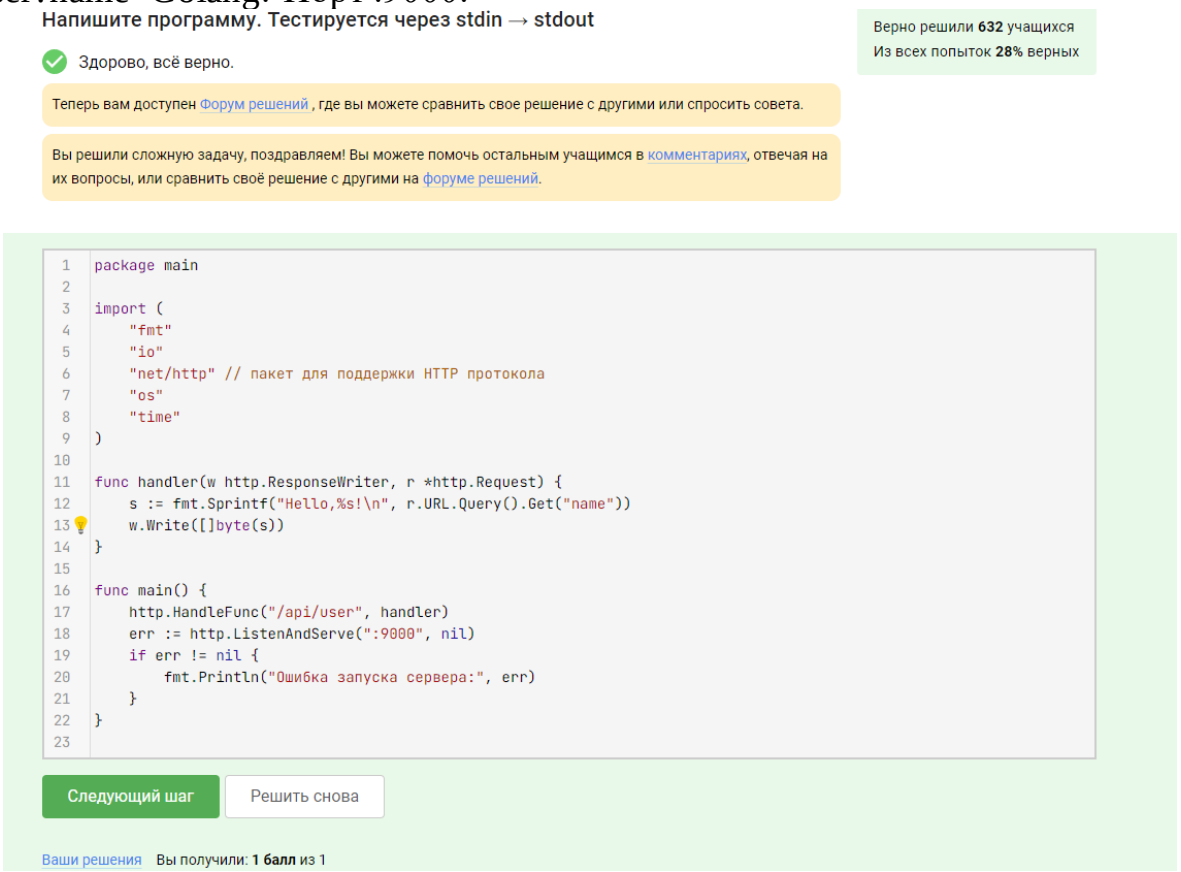

Рисунок 3—Прохождение тестов по заданию 2 на Stepik

Проведем тестирование программы, приведенной далее, с помощью ПО Postman (рис. 4):

```
package main
```

```go
import (
    "fmt"
    "net/http" // пакет для поддержки HTTP протокола
)

func handler(w http.ResponseWriter, r *http.Request) {
    s := fmt.Sprintf("Hello,%s!\n", r.URL.Query().Get("name"))
    w.Write([]byte(s))
}

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

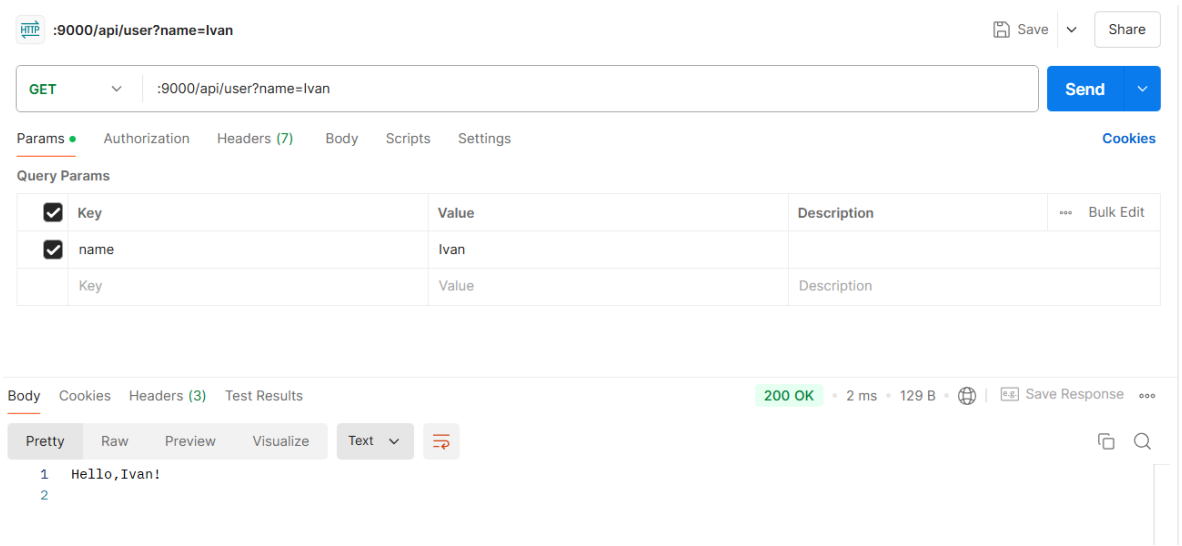

Рисунок 4—Тестирование (задание 2)

Задание 3. Напишите 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).

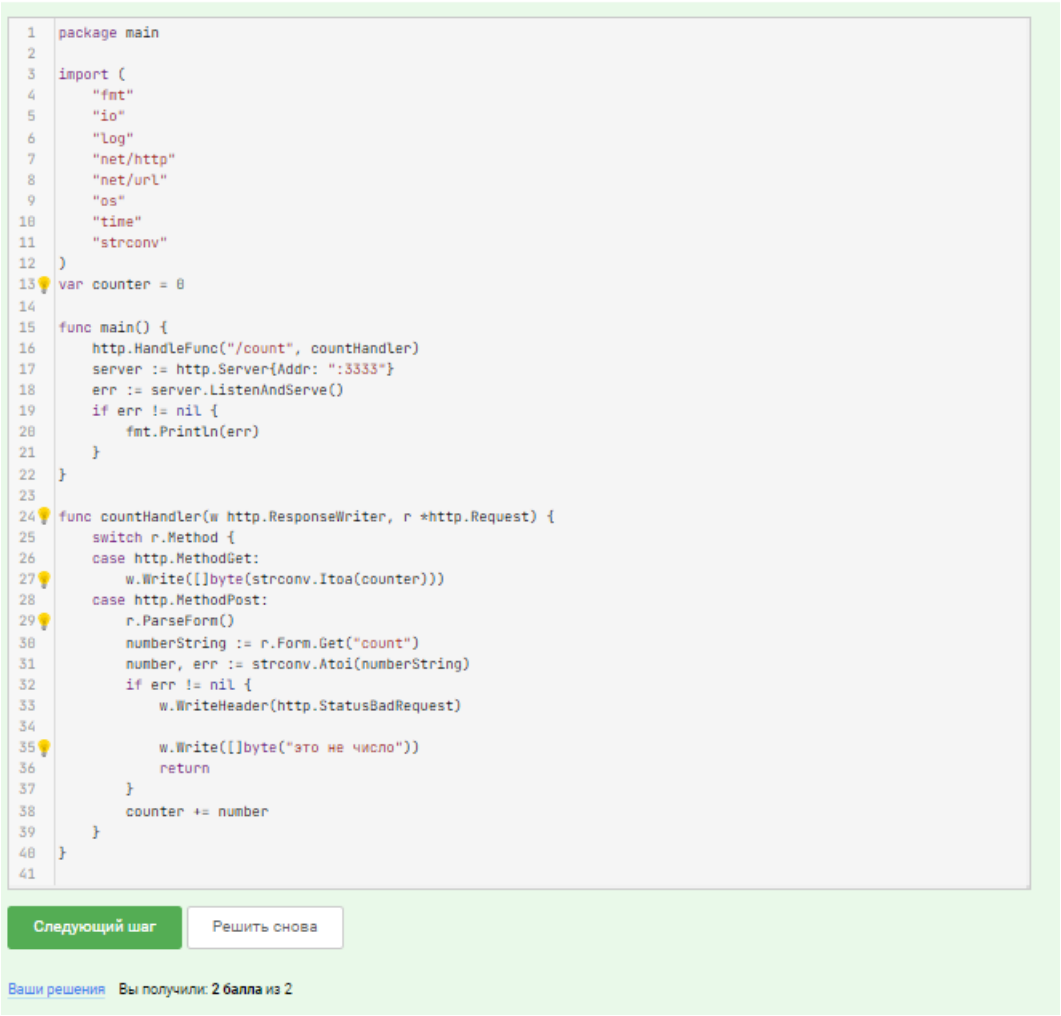

Рисунок 5—Прохождение тестов по заданию 3 на Stepik

Проведем тестирование программы, приведенной далее, с помощью ПО Postman (рис. 6, 7, 8):

```
package main

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

var counter = 0

func main() {
    http.HandleFunc("/count", countHandler)
    server := http.Server{Addr: ":3333"}
    err := server.ListenAndServe()
    if err != nil {
        fmt.Println(err)
    }
}

func countHandler(w http.ResponseWriter, r *http.Request) {
    switch r.Method {
```

```
    case http.MethodGet:
        w.Write([]byte(strconv.Itoa(counter)))
    case http.MethodPost:
        r.ParseForm()
        numberString := r.Form.Get("count")
        number, err := strconv.Atoi(numberString)
        if err != nil {
            w.WriteHeader(http.StatusBadRequest)

            w.Write([]byte("это не число"))
            return
        }
        counter += number
    }
}
```

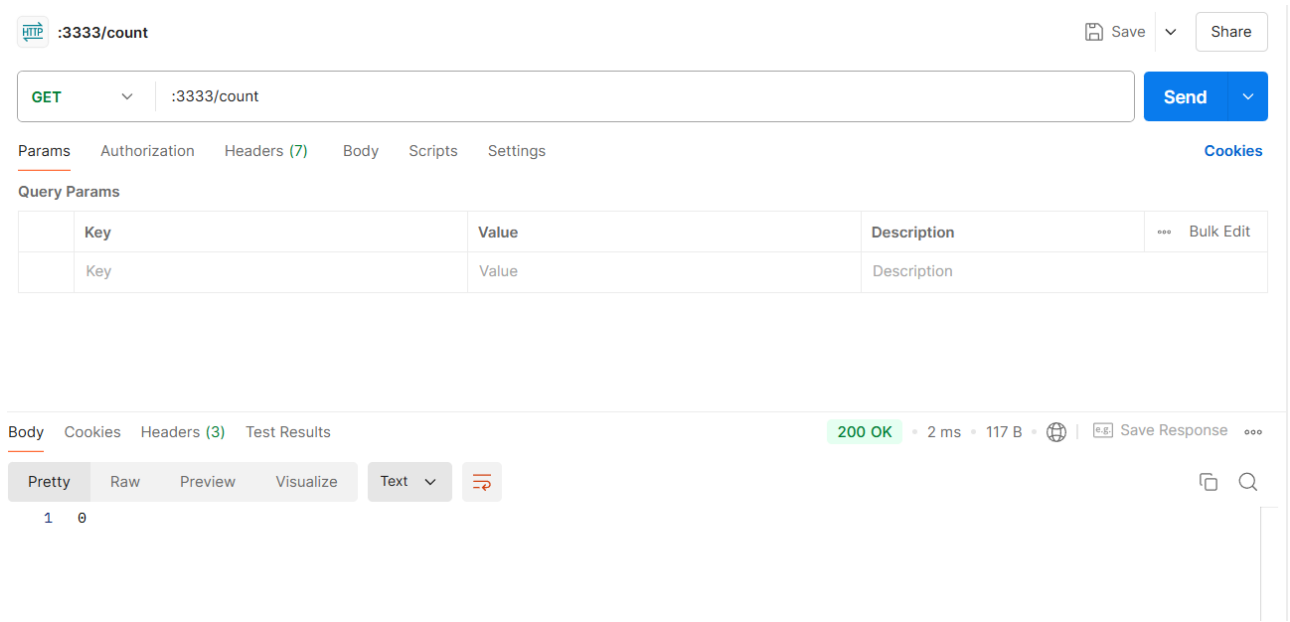

Рисунок 6—Тестирование (задание 3)

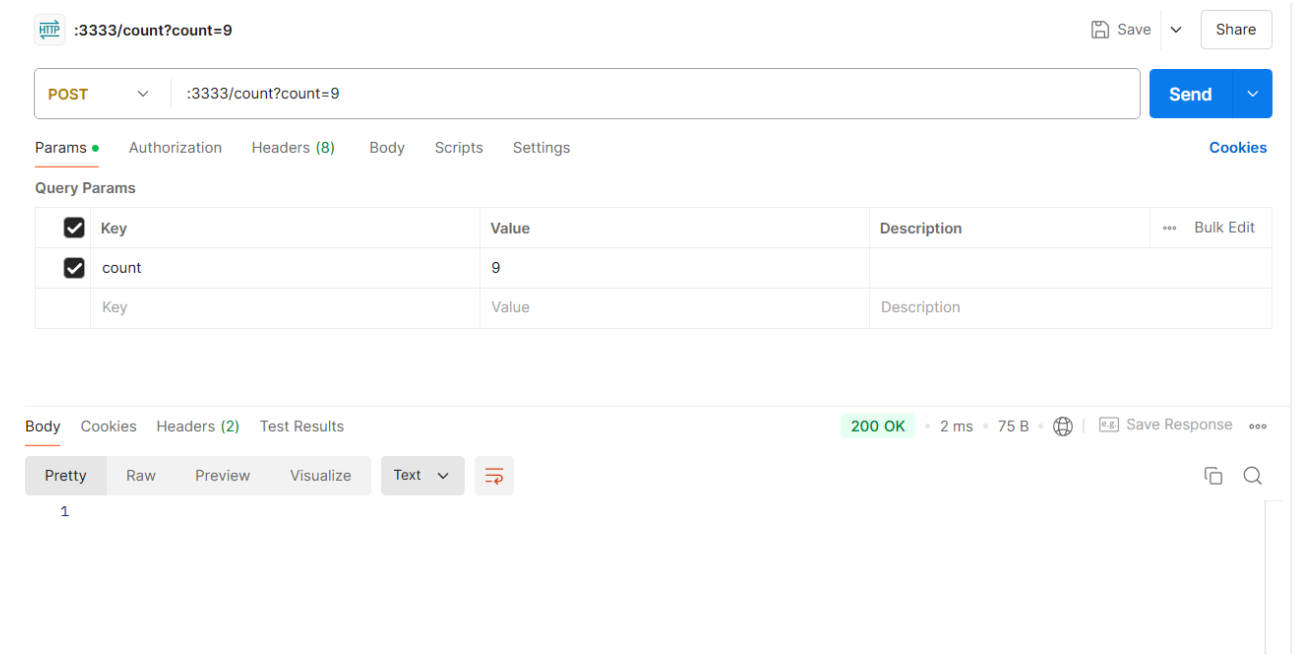

Рисунок 7—Тестирование (задание 3)

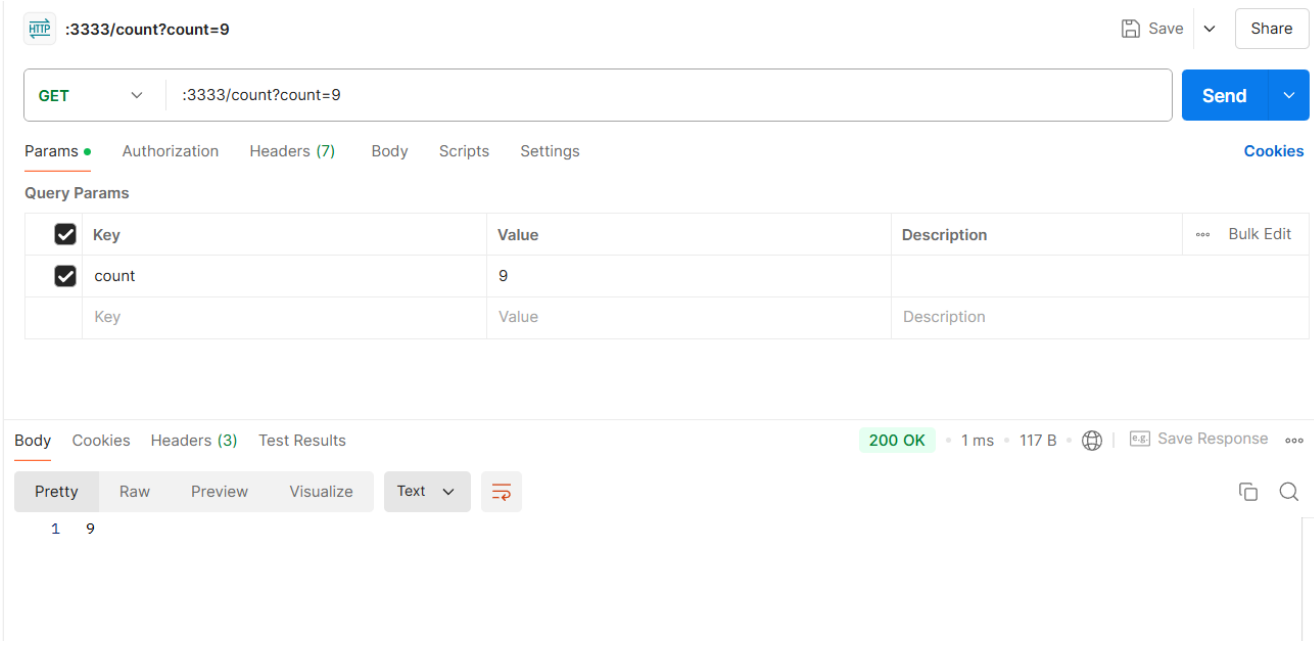

Рисунок 8—Тестирование (задание 3)

Заключение: в рамках данной лабораторной работы продолжили изучение Golang и познакомились с набором стандартных библиотек, используемых для организации сетевого взаимодействия и разработки серверных приложений.

Контрольные вопросы

1.В чём разница между протоколами TCP и UDP?

TCP (Transmission Control Protocol):

- TCP — это *надежный и устойчивый* протокол передачи данных в сетях. Он обеспечивает установление соединения между отправителем и получателем, а также обеспечивает гарантию доставки данных в правильном порядке и контроль ошибок.

- TCP используется для приложений, которым важна надежная передача данных, таких как веб-серверы, электронная почта и файловые передачи.

UDP (User Datagram Protocol):

- UDP — это *простой и быстрый* протокол передачи данных в сетях. Он *не* гарантирует надежную доставку данных, *не* устанавливает соединение и *не* контролирует порядок доставки.
- UDP используется в приложениях, где небольшая потеря данных не критична, например, в видеозвонках и стриминге.

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

IP-адрес (Internet Protocol address) — это уникальный адрес, присваиваемый устройству в сети. Он используется для идентификации устройства и его местоположения в сети. IP-адреса могут быть как статическими (постоянными), так и динамическими (изменяющимися).

Назначение:

- Определяет конкретное устройство или сервер в сети.
- Позволяет отправлять и получать данные между устройствами.
- Отвечает за маршрутизацию пакетов данных через сеть.

- Пример: 192.168.1.1

Номер порта — это числовой идентификатор, который используется для различения различных приложений или служб, работающих на одном и том же устройстве. Он определяет конкретный процесс или сервис, с которым необходимо установить соединение.

-Назначение:

- Позволяет различным приложениям на одном сервере принимать и обрабатывать потоки данных.
- Указывает, к какому сервису или приложению следует направить входящий трафик.

Пример:

- Порт 80 используется для HTTP-трафика.
- Порт 21 используется для FTP.

Разница между IP-адресом и номером порта

1. Функция:
   - IP-адрес идентифицирует устройство в сети, позволяя другим устройствам находить его.
   - Номер порта идентифицирует конкретное приложение или сервис на этом устройстве.

2. Уровень работы:
   - IP-адрес работает на сетевом уровне (3-й уровень модели OSI).
   - Номера портов работают на транспортном уровне (4-й уровень модели OSI), управляя тем, какие данные должны быть направлены конкретному приложению или службе.

3. Формат:
- IP-адрес представляет собой набор чисел (например, 192.168.1.1).
- Номер порта — это целое число в диапазоне от 0 до 65535.

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

Семантика CRUD (Create, Read, Update, Delete) представляет собой набор операций, которые используются для создания, чтения, обновления и удаления ресурсов в приложениях.

1. Create (Создание):
- Метод: POST

Описание: используется для создания нового ресурса.

2. Read (Чтение):
- Метод: GET
- Описание: используется для запроса и получения представления ресурса.

Метод GET не должен изменять состояние сервера и только извлекает данные.

3. Update (Обновление):
- Методы:
  - PUT — для обновления всего ресурса.
  - PATCH — для частичного обновления ресурса.

4. Delete (Удаление):
- Метод: DELETE
- Описание: используется для удаления существующего ресурса с сервера.

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

Код ответа (состояния) HTTP показывает, был ли успешно выполнен определённый HTTP запрос. Коды сгруппированы в 5 классов:

Информационные (100 – 199)

Примеры:

100 Continue («продолжайте»);
101 Switching Protocols («переключение протоколов»);
102 Processing («идёт обработка»);
103 Early Hints («предварительный ответ»).

Успешные (200 – 299)

Примеры:

200 OK («хорошо»);
201 Created («создано»);
202 Accepted («принято»);
204 No Content («нет содержимого»).

Перенаправления (300 – 399)

Примеры:

300 Multiple Choices («множество выборов»);

301 Moved Permanently («перемещено навсегда»);
302 Found («найдено»);
303 See Other («смотреть другое»).

Клиентские ошибки (400 – 499)

Примеры:

400 Bad Request («неправильный, некорректный запрос»);
401 Unauthorized («не авторизован»);
403 Forbidden («запрещено (не уполномочен)»);
404 Not Found («не найдено»).

Серверные ошибки (500 – 599)

Примеры:

500 Internal Server Error («внутренняя ошибка сервера»);
502 Bad Gateway («плохой, ошибочный шлюз»);
503 Service Unavailable («сервис недоступен»);
504 Gateway Timeout («шлюз не отвечает»).

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

HTTP-протокол описывает взаимодействие между двумя компьютерами (клиентом и сервером), построенное на базе сообщений, называемых запрос (Request) и ответ (Response). Каждое сообщение состоит из трех частей: стартовая строка, заголовки и тело. При этом обязательной является только стартовая строка.

Список использованных источников

1. https://stepik.org/course/54403/info
2. https://go.dev/doc