



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 09.04.01 ИУ6-32Б

О Т Ч Е Т

по лабораторной работе № 6

Название: Основы Back-End разработки на Golang

Дисциплина: Языки интернет-программирования

Студент

ИУ6-32Б

(Группа)

(Подпись, дата)

Кондратов С.Ю.

(И.О. Фамилия)

Преподаватель

(Подпись, дата)

Шульман В.Д.

(И.О. Фамилия)

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

Задание 1

Напишите веб сервер, который по пути /get отдает текст "Hello, web!".

Порт должен быть :8080.

Рисунок 1

```
1 package main
2
3 import (
4     "fmt"
5     "net/http"
6 )
7
8 func helloHandler(w http.ResponseWriter, r *http.Request) {
9     fmt.Fprint(w, "Hello, web!")
10 }
11
12 func main() {
13     http.HandleFunc("/get", helloHandler)
14     fmt.Println("Server is listening on port 8080")
15     http.ListenAndServe(":8080", nil)
16 }
```

Рисунок 2

На рисунке 2 показан мой результат.

Есть функция helloHandler, обрабатывающая запрос на “get” которая возвращает “Hello, web!”/

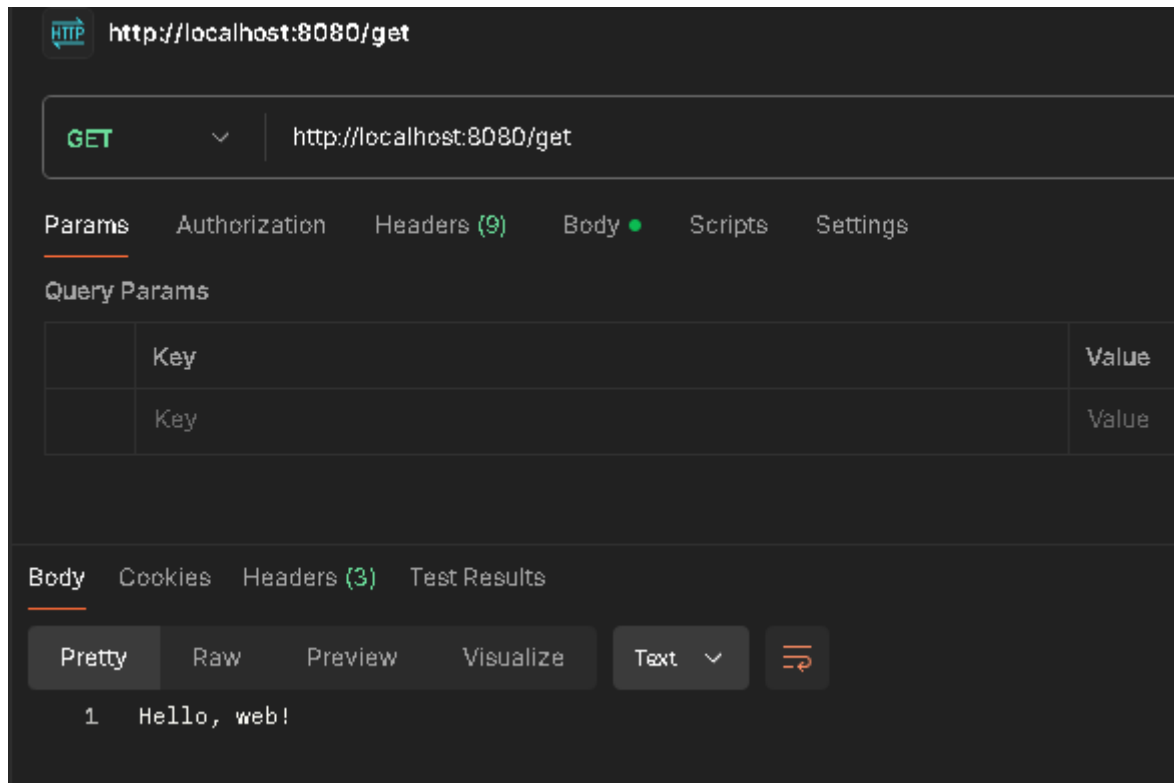


Рисунок 3

На рисунке 3 показан пример вывода.

Задание 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!`

Рисунок 4

Как и в предыдущем задании мы обрабатываем запрос `“api/user”` но теперь в запросе присутствует параметр `name` который задается `api/user?name=Stepan`

```

1  package main
2
3  import (
4      "fmt"
5      "net/http"
6  )
7
8  func helloHandler(w http.ResponseWriter, r *http.Request) {
9      name := r.URL.Query().Get("name")
10     if name == "" {
11         fmt.Fprint(w, "Hello, stranger!")
12         return
13     }
14     fmt.Fprintf(w, "Hello, %s!", name)
15 }
16
17 func main() {
18     http.HandleFunc("/api/user", helloHandler)
19     fmt.Println("Server is listening on port 8080")
20     http.ListenAndServe(":8080", nil)
21 }

```

Рисунок 5

На рис.5 показан мой результат.

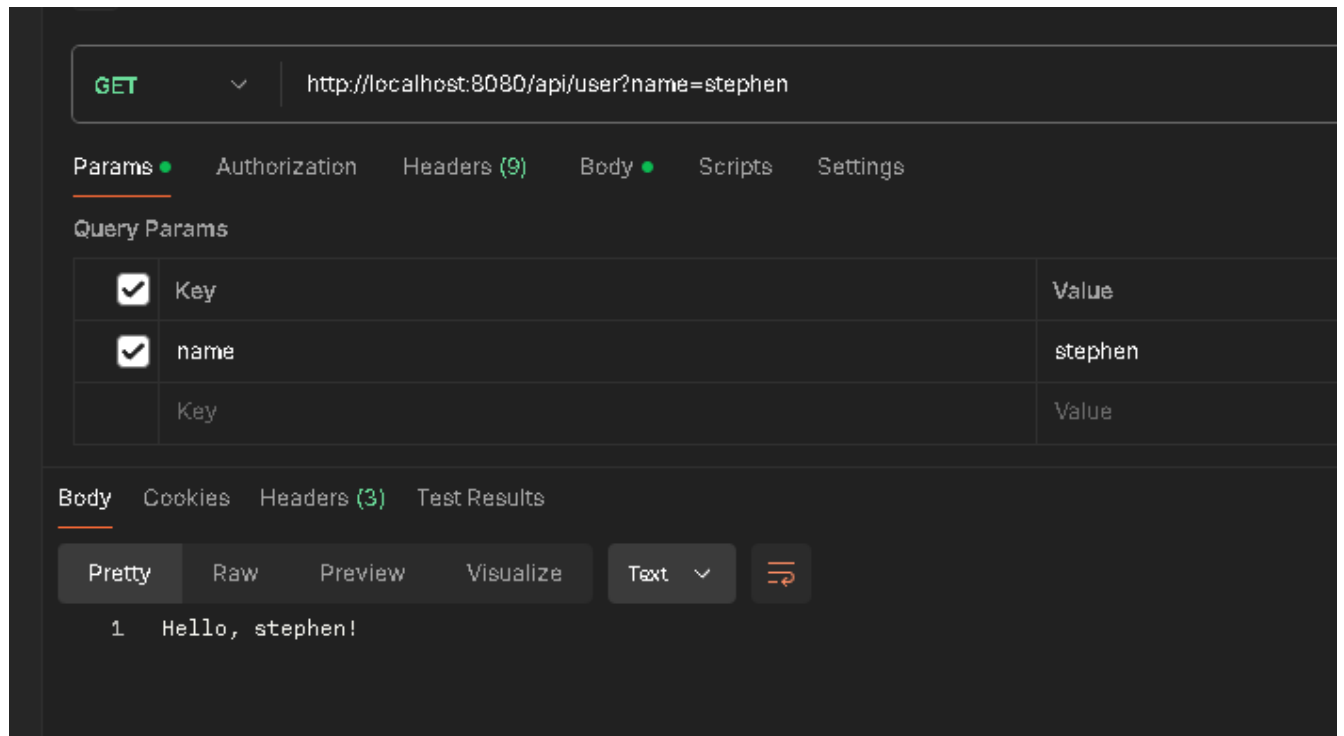


Рисунок 6 (Пример вывода)

Задание 3

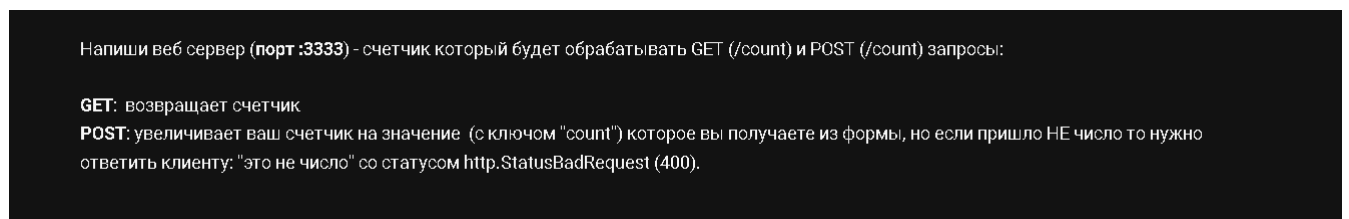


Рисунок 7

Для выполнения задания нам придется создать 2 функции обработчики, для POST и GET метода.

```

func updateCountHandler(w http.ResponseWriter, r *http.Request) {
    if r.Method != http.MethodPost {
        http.Error(w, "Invalid request method", http.StatusBadRequest)
        return
    }

    var update struct {
        Count int `json:"count"`
    }

    err := json.NewDecoder(r.Body).Decode(&update)
    if err != nil {
        http.Error(w, "Invalid request body", http.StatusBadRequest)
        return
    }

    if update.Count == 0 {
        http.Error(w, "Это не число", http.StatusBadRequest)
        return
    }

    counter.Value += update.Count
    w.WriteHeader(http.StatusOK)
}

```

Рисунок 8 (Функция для POST запроса)

```

1  package main
2
3  import (
4      "encoding/json"
5      "fmt"
6      "net/http"
7  )
8
9      Codeium: Refactor | Explain
10 type Counter struct {
11     Value int
12 }
13
14 var counter Counter
15
16      Codeium: Refactor | Explain | Generate GoDoc | X
17 func getCountHandler(w http.ResponseWriter, _ *http.Request) {
18     json.NewEncoder(w).Encode(counter)
19 }

```

Рисунок 9 (Функция для GET запроса)

```

      Codeium: Refactor | Explain | Generate GoDoc | X
4  func main() {
5      http.HandleFunc("/count", func(w http.ResponseWriter, r *http.Request) {
6          if r.Method == http.MethodGet {
7              getCountHandler(w, r)
8          } else if r.Method == http.MethodPost {
9              updateCountHandler(w, r)
10         }
11     })
12
13     fmt.Println("Server is listening on port 3333")
14     http.ListenAndServe(":3333", nil)
15 }

```

Рисунок 11 (Основная функция main)

В ней мы сначала обрабатываем тип запроса и в зависимости от этого выбираем функцию-обработчик.

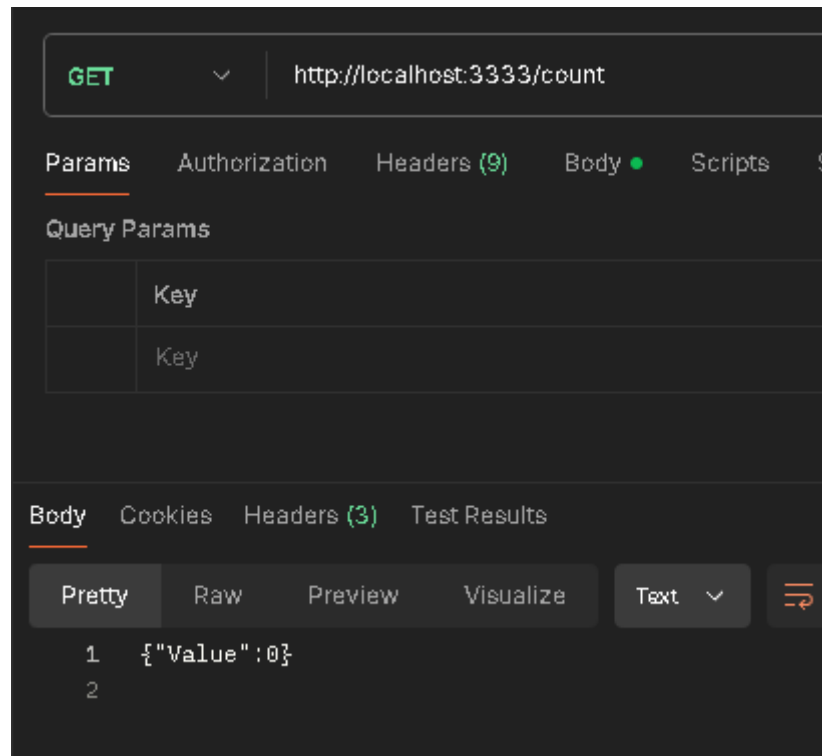


Рисунок 12 (GET запрос)

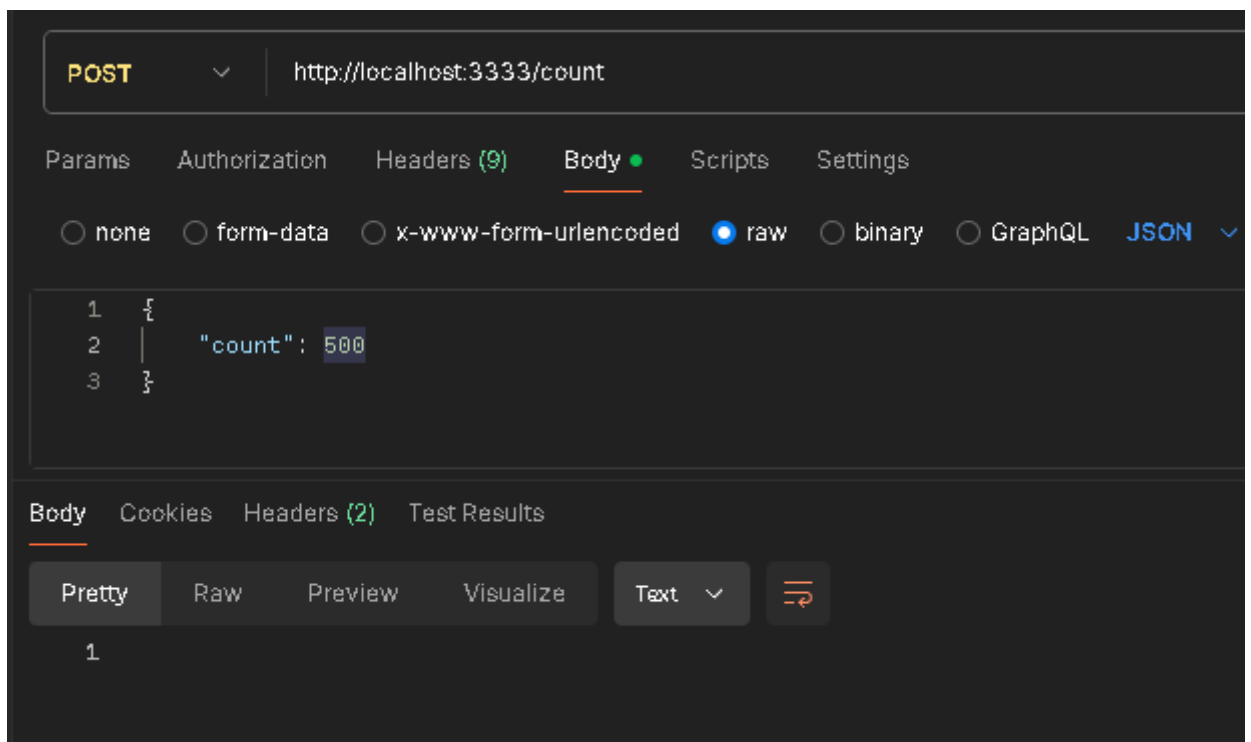


Рисунок 13 (POST запрос)

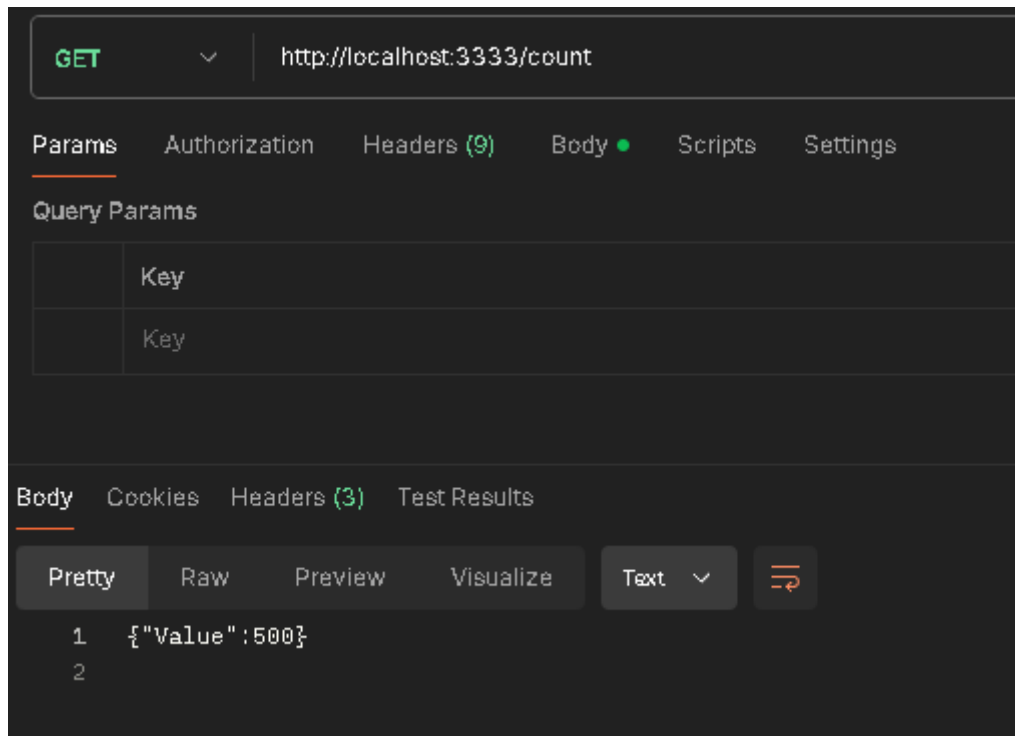


Рисунок 14 (GET запрос)

Контрольные вопросы

1. В чём разница между протоколами TCP и UDP

TCP (Transmission Control Protocol) и UDP (User Datagram Protocol) - это два основных протокола сетевого уровня, используемых в Интернете. Они отличаются по своей надежности, производительности и способу обработки ошибок.

Вот некоторые общие различия между TCP и UDP:

Надежность: TCP обеспечивает надежную передачу данных, а UDP - нет.

Последовательность: TCP обеспечивает последовательную доставку данных, а UDP - нет.

Подтверждения: TCP использует подтверждения, а UDP - нет.

Скорость: TCP медленнее, чем UDP.

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

IP-адрес определяет местоположение веб-сервера в сети, а порт определяет, какое приложение или службу он обслуживает.

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

Набор методов HTTP, реализующий семантику CRUD:

* CREATE - POST

* READ - GET

- * UPDATE - PUT или PATCH
- * DELETE – DELETE

4. Какие группы status code существуют у HTTP-response

Существуют следующие группы статусов HTTP:

- * 1xx - Информационные (100 Continue, 101 Switching Protocols)
- * 2xx - Успешные (200 OK, 201 Created, 204 No Content)
- * 3xx - Перенаправления (301 Moved Permanently, 302 Found, 304 Not Modified)
- * 4xx - Ошибки клиента (400 Bad Request, 401 Unauthorized, 404 Not Found)
- * 5xx - Ошибки сервера (500 Internal Server Error, 502 Bad Gateway, 503 Service Unavailable)

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

HTTP-запрос (HTTP-request) состоит из:

- * Метод (Method, например GET, POST, PUT)
- * URI (Uniform Resource Identifier)
- * Версия HTTP (HTTP-Version)
- * Заголовки (Headers)
- * Тело запроса (Body)

HTTP-ответ (HTTP-response) состоит из:

- * Версия HTTP (HTTP-Version)
- * Статусный код (Status Code)
- * Текстовое описание статуса (Status Text)
- * Заголовки (Headers)
- * Тело ответа (Body)

Заключение: Я научился создавать веб-сервер и обрабатывать GET и POST запросы, при этом получая параметры различными способами.