|  |  |
| --- | --- |
| Gerb-BMSTU_01 | **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 Название направления «Информатика и вычислительная техника»**

**Отчет**

**по лабораторной работе № 6**

**Название: Основы Back-End разработки на Golang**

**Дисциплина:** Языки интернет программирования

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Студент | ИУ6-32Б |  |  | Е.Б. Гаппова |
|  | (Группа) |  | (Подпись, дата) | (И.О. Фамилия) |
|  |  |  |  |  |
| Преподаватель |  |  |  | В. Д. Шульман |
|  |  |  | (Подпись, дата) | (И.О. Фамилия) |

Москва, 2024

**Задание 1**:

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

Код программы:

package main

import (

    "fmt"

    "net/http"

)

// Обработчик HTTP-запросов

func handler(w http.ResponseWriter, r \*http.Request) {

    w.Write([]byte("Hello, web!"))

}

func main() {

    // Регистрируем обработчик для пути "/"

    http.HandleFunc("/get", handler)

    // Запускаем веб-сервер на порту 8080

    err := http.ListenAndServe(":8081", nil)

    if err != nil {

        fmt.Println("Ошибка запуска сервера:", err)

    }

}

Тестирование программы:

На рисунке 1 видим результат запуска веб сервера по пути /get.

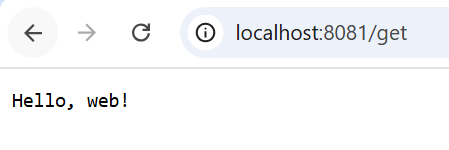


Рисунок 1 - запуск веб сервера

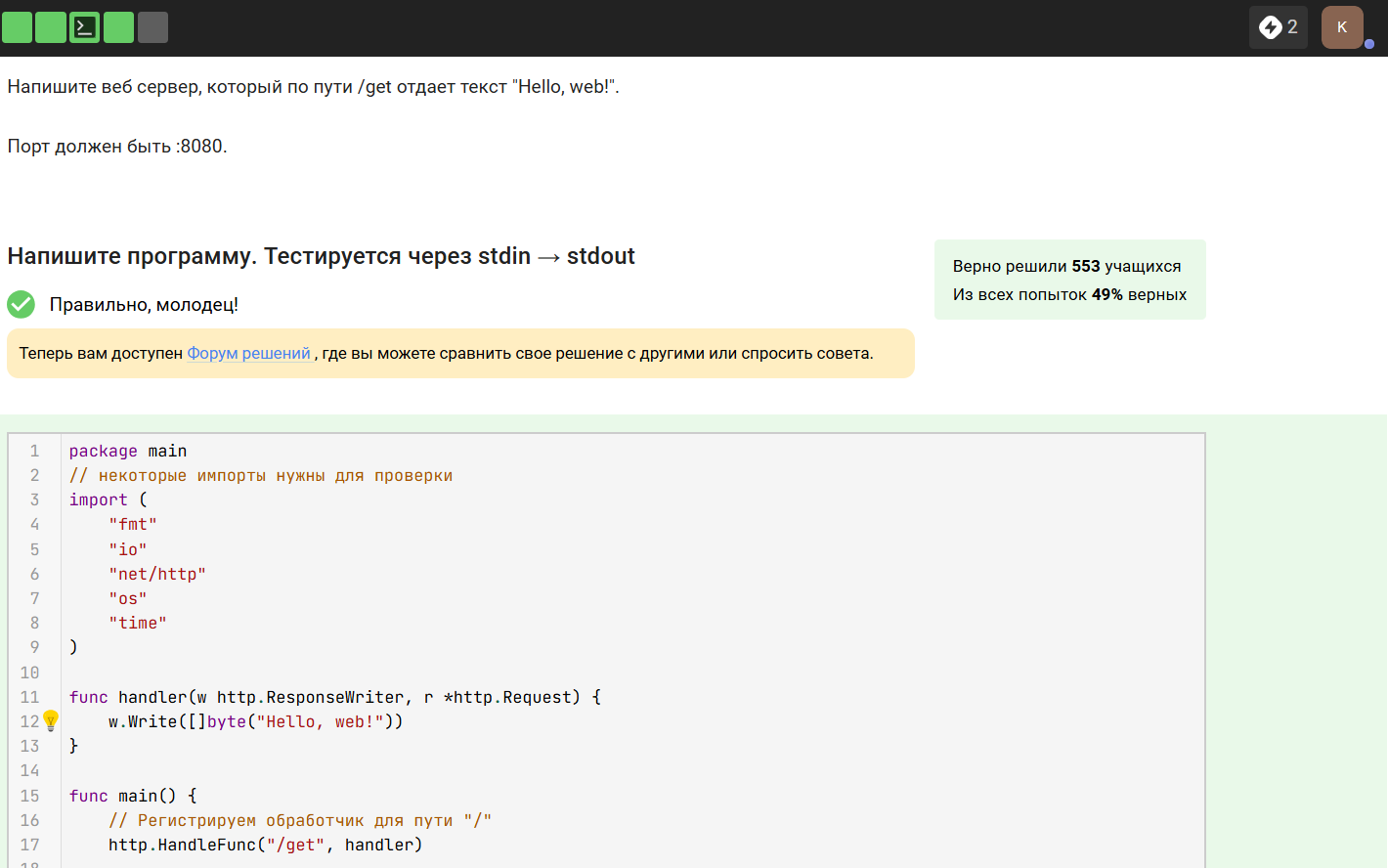


Рисунок 2 – тесты на Stepik прошли

**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!

Код программы:

package main

import (

    "fmt"

    //"io"

    "net/http" // пакет для поддержки HTTP протокола

    //"os"

    //time"

)

// Обработчик HTTP-запросов

func handler(w http.ResponseWriter, r \*http.Request) {

    s := fmt.Sprintf("Hello,%s!", r.URL.Query().Get("name"))

    w.Write([]byte(s))

}

func main() {

    // Регистрируем обработчик для пути "/"

    http.HandleFunc("/api/user", handler)

    // Запускаем веб-сервер на порту

    err := http.ListenAndServe(":9000", nil)

    if err != nil {

        fmt.Println("Ошибка запуска сервера:", err)

    }

}

Тестирование программы:

Протестируем программу, используя клиент postman.

Query параметр name передается через URL с помощью ?name= (Рисунок 3)

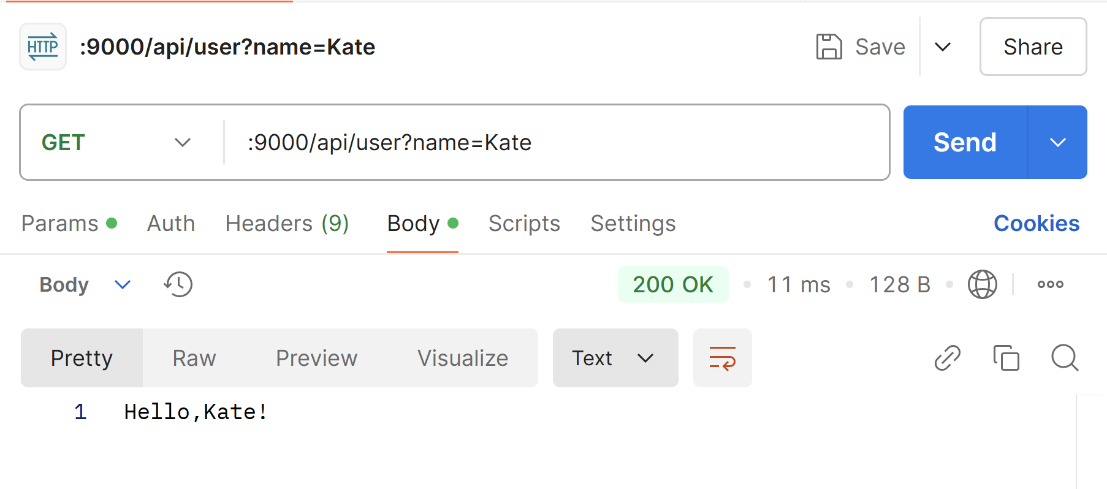


Рисунок 3 – тестирование программы

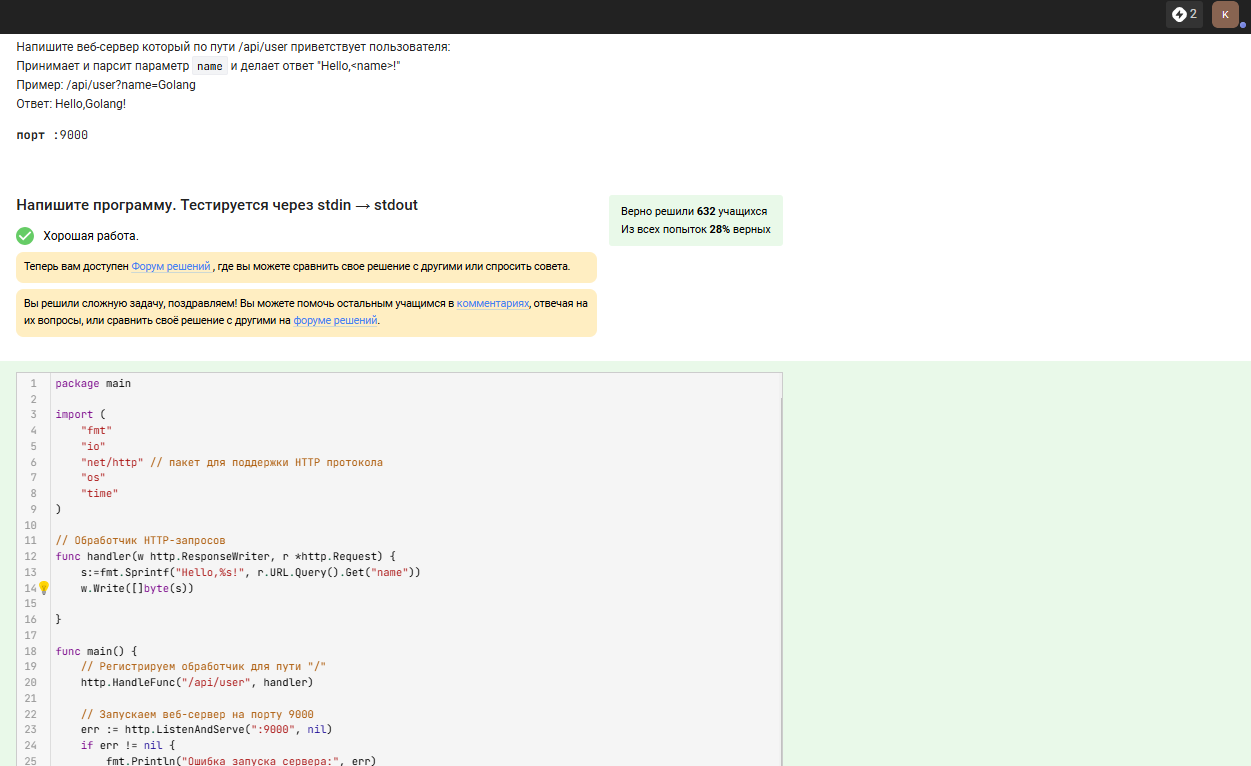


Рисунок 4 - тесты на Stepik прошли

**Задание 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).

Клд программы:

package main

import (

    "fmt"

    "net/http"

    "strconv"

)

var c int

func handler(w http.ResponseWriter, r \*http.Request) {

    if r.Method == "POST" {

        r.ParseForm()

        s := r.Form.Get("count")

        if n, err := strconv.Atoi(s); err != nil {

            w.WriteHeader(http.StatusBadRequest) //400

            w.Write([]byte("это не число"))

            return

        } else {

            c += n

            w.WriteHeader(200)

            return

        }

    }

    if r.Method == "GET" {

        w.Write([]byte(strconv.Itoa(c)))

        w.WriteHeader(http.StatusOK) // 200

        return

    }

}

func main() {

    c = 0

    // Регистрируем обработчик для пути "/"

    http.HandleFunc("/count", handler)

    // Запускаем веб-сервер на порту 3333

    err := http.ListenAndServe(":3333", nil)

    if err != nil {

        fmt.Println("Ошибка запуска сервера:", err)

    }

}

Тестирование программы:

Сначала отправим запрос get и получим значение 0 параметра count (Рисунок 5):

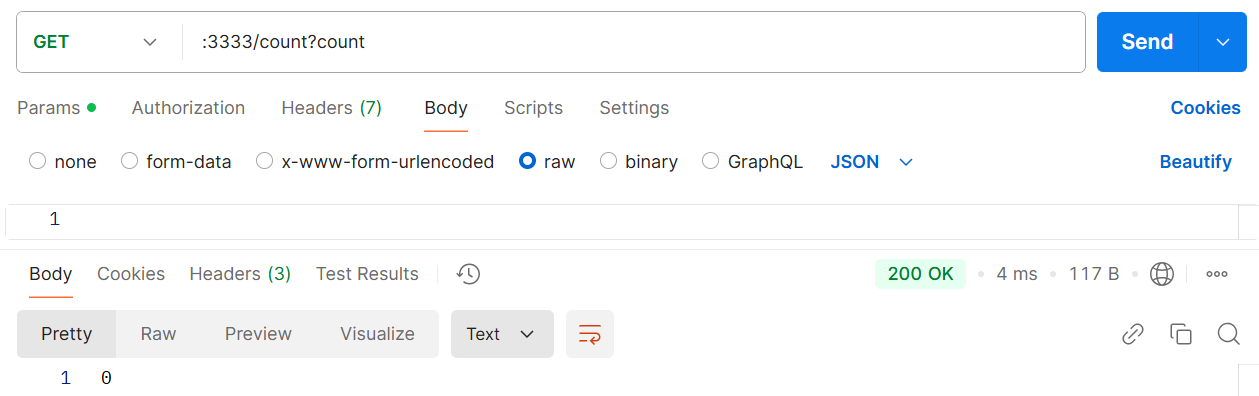


Рисунок 5 – первый запрос get

Далее отправим запрос post с count=5,запрос get и получим,что значение счетчика стало 5 (Рисунок 6)



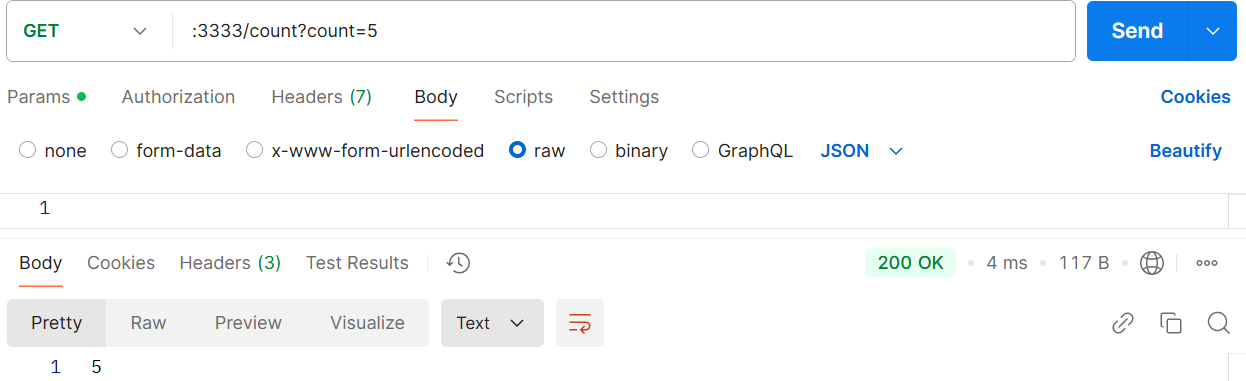


Рисунок 6 – Изменение count на 5

Далее повторим действия с значением 6 и к счетчику прибавится 6

(Рисунок 7):



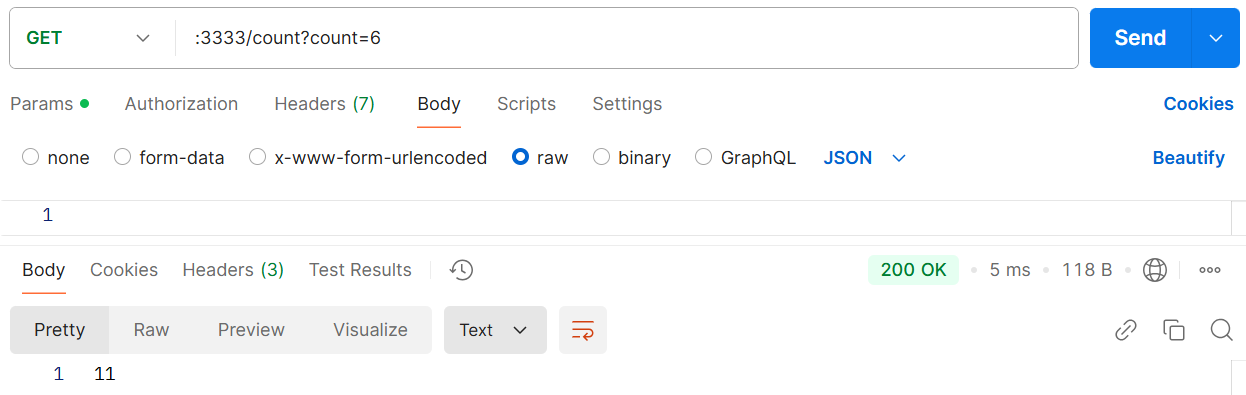


Рисунок 7 – изменение count на 6

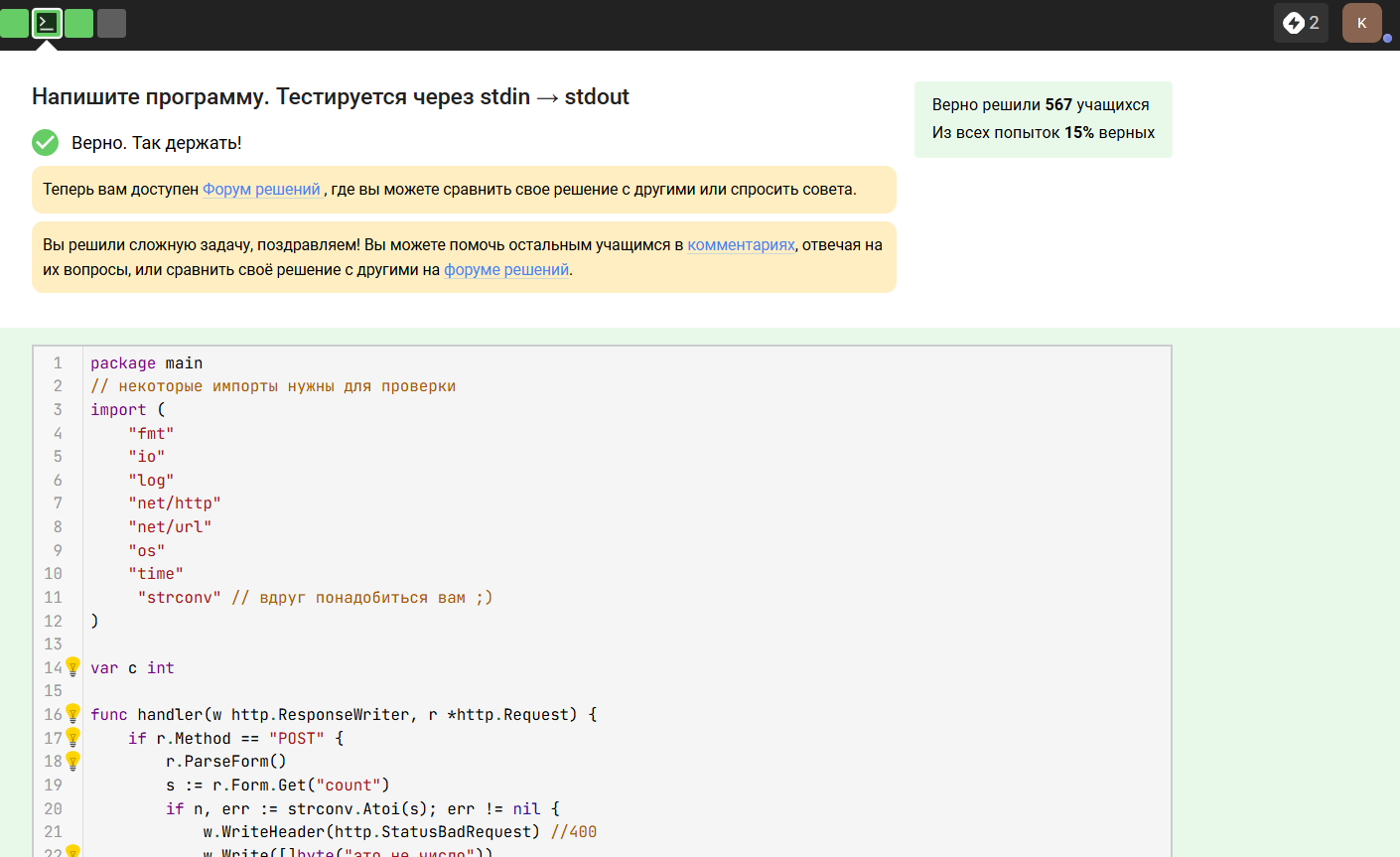


Рисунок 8 - тесты на Stepik прошли

Вывод : произошло ознакомление с Back-End разработкой на Golang, использованы http запросы.

Контрольные вопросы:

1. В чём разница между протоколами TCP и UDP?

Протокол TCP обеспечивает соединение между отправителем и получателем, гарантирует доставку данных в верном порядке, в отличие от UDP, где скорость передачи данных быстрее, но соединение не устанавливается, может быть нарушена целостность данных и их порядок.

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

IP (internet protocol) Adress – уникальный адрес, идентифицирующий устройство в интернете или локальной сети.

Port number, номер порта, - идентификатор конкретной программы (на каком-либо IP адресе)

1. Какой набор методов в HTTP-request в полной мере релализует семантику CRUD?

Create – POST

Read – GET

Update – PUT

Delete – DELETE

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

Коды сгруппированы в 5 классов:

1. Информационные 100 - 199
2. Успешные 200 - 299
3. Перенаправления 300 - 399
4. Клиентские ошибки 400 - 499
5. Серверные ошибки 500 – 599

100 – Continue (чтобы клиент продолжал запрос)

200 – OK

301 Moved Permanently (URL-адрес запрошенного ресурса был изменен навсегда. Новый URL-адрес указан в ответе.)

400 Bad Request

508 -Loop Detected (Обнаружен бесконечный цикл обработки данных)

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

 Каждое сообщение состоит из трех частей: стартовая строка, заголовки и тело. При этом обязательной является только стартовая строка.

Стартовая строка запроса выглядит так:  
  
*METHOD URI* HTTP/*VERSION*,  
  
где METHOD — это как раз метод HTTP-запроса, URI — идентификатор ресурса, VERSION — версия протокола (на данный момент актуальна версия 1.1).

Прим. GET /index.php HTTP/1.1

Источники:

1. Задание 1: <https://stepik.org/lesson/1101590/step/2?unit=1112590>
2. Задание 2: <https://stepik.org/lesson/1101590/step/4?unit=1112590>
3. Задание 3: <https://stepik.org/lesson/1101590/step/7?unit=1112590>