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 ИУ6-33Б _____ С.В. Сонин
(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.

Задание 1:

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

Порт должен быть :8080.

Код программы:

```go
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("localhost:8080", nil)
    if err != nil {
        fmt.Println("Ошибка запуска сервера:", err)
    }
}
```

```
+ curl -iL -w '\n' -X GET localhost:8080/get/
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100    11  100    11    0     0     52      0 --:--:-- --:--:-- --:--:--    53HTTP/1.1 200
OK
Date: Mon, 30 Sep 2024 05:29:53 GMT
Content-Length: 11
Content-Type: text/plain; charset=utf-8

Hello, web!
```

Рисунок 1 – Результат работы программы.

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!

порт :9000

Код программы:

```
    package main

import (
    "net/http"
)

func handler(w http.ResponseWriter, r *http.Request) {
    w.Write([]byte("Hello," + r.URL.Query().Get("name") +
"!"))
}
func main() {
    http.HandleFunc("/api/user/", handler)
    http.ListenAndServe(":9000", nil)
}
```

```
+ curl -iL -w '\n' -X GET 'localhost:9000/api/user?name=Golang'
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100    57  100    57    0     0  24432      0 --:--:-- --:--:-- --:--:-- 28500
100    13  100    13    0     0   4868      0 --:--:-- --:--:-- --:--:--  4868HTTP/1.1 301
Moved Permanently
Content-Type: text/html; charset=utf-8
Location: /api/user/?name=Golang
Date: Mon, 30 Sep 2024 05:39:54 GMT
Content-Length: 57

HTTP/1.1 200 OK
Date: Mon, 30 Sep 2024 05:39:54 GMT
Content-Length: 13
Content-Type: text/plain; charset=utf-8

Hello,Golang!
```

Рисунок 2 – Результат работы программы.

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).

Код программы:

```
package main

import (
```

```
    "encoding/json"
    "io"
    "net/http"
    "strconv"
)

var count int

type req struct {
    Count string `json:"count"`
}

func handler(w http.ResponseWriter, r *http.Request) {
    if r.Method == "POST" {
        data, _ := io.ReadAll(r.Body)
        var rq req
        json.Unmarshal(data, &rq)
        i, err := strconv.Atoi(rq.Count)
        if err != nil {
            w.WriteHeader(http.StatusBadRequest)
            w.Write([]byte("это не число"))
            return
        }
        count += i

    }
    if r.Method == "GET" {
        w.Write([]byte(strconv.Itoa(count)))
    }
}
func main() {
    http.HandleFunc("/", handler)
    http.ListenAndServe(":3333", nil)
}
```

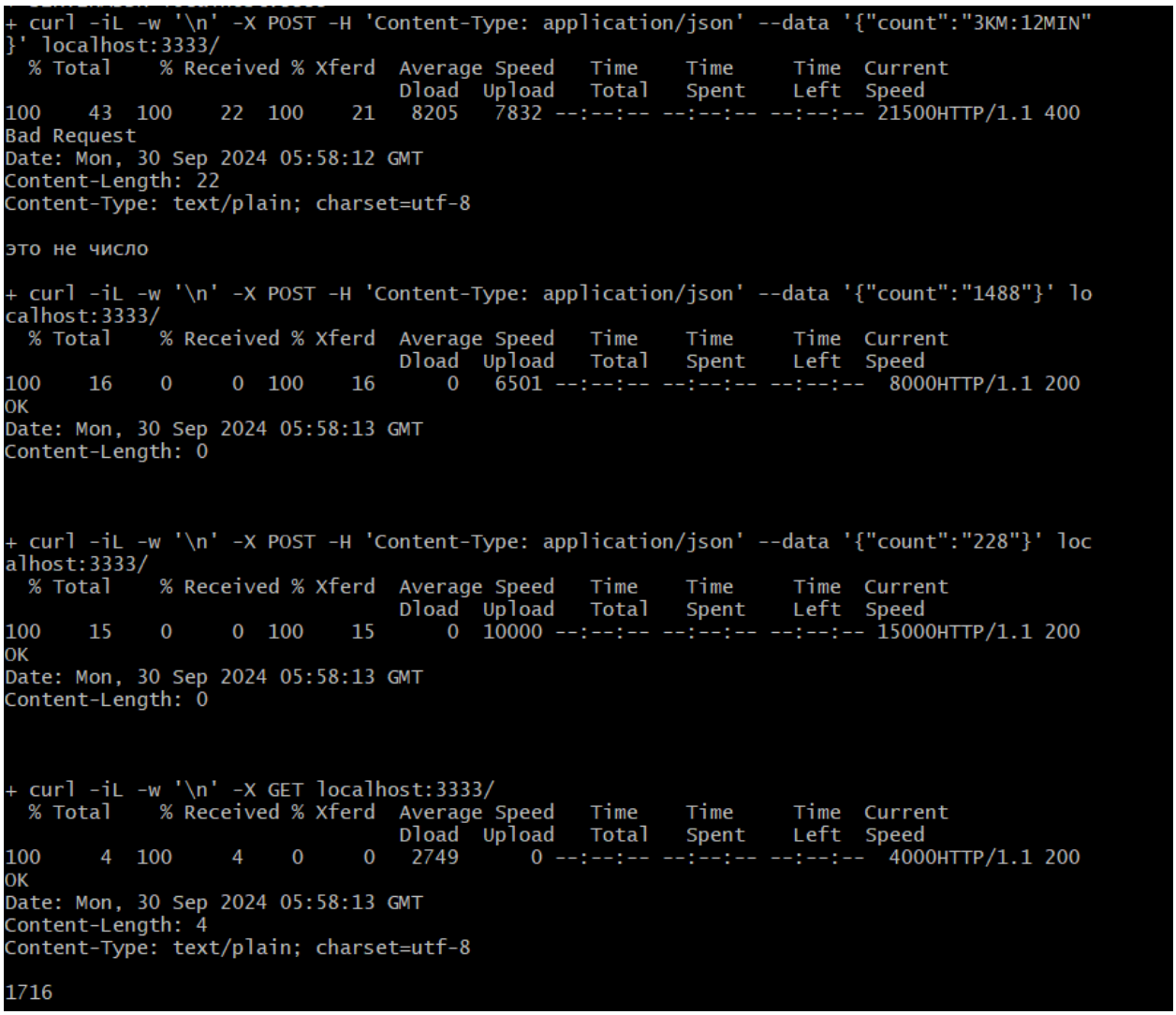

Рисунок 3 – Результат работы программы.

Вывод:

Изучил основы сетевого взаимодействия и серверной разработки с использованием языка Golang.

Контрольные вопросы:

В чём разница между протоколами TCP и UDP?

Протокол TCP позволяет установить устойчивое соединение, вследствие чего гарантирует доставку пакетов, в то время как UDP – нет.

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

IP-адрес – это идентификатор устройства в сети, в то время как порт – это специфический канал на этом устройстве для передачи данных.

Какой набор методов в HTTP-request в полной мере релизует семантику CRUD?

GET, POST, DELETE, PATCH

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

- 1xx: Informational (информационные):
  - 100 Continue («продолжайте»)[3];
  - 101 Switching Protocols («переключение протоколов»)[3];
  - 102 Processing («идёт обработка»);
  - 103 Early Hints («предварительный ответ»);
- 2xx: Success (успешно):
  - 200 OK («хорошо»)[3];
  - 201 Created («создано»)[3][4];
  - 202 Accepted («принято»)[3];
  - 203 Non-Authoritative Information («информация не авторитетна»)[3];
  - 204 No Content («нет содержимого»)[3];
  - 205 Reset Content («сбросить содержимое»)[3];
  - 206 Partial Content («частичное содержимое»)[5];
  - 207 Multi-Status («многостатусный»)[6];
  - 208 Already Reported («уже сообщалось»)[7];
  - 226 IM Used («использовано IM»).
- 3xx: Redirection (перенаправление):
  - 300 Multiple Choices («множество выборов»)[3];
  - 301 Moved Permanently («перемещено навсегда»)[3];
  - 302 Found («найдено»)[3];
  - 303 See Other («смотреть другое»)[3];
  - 304 Not Modified («не изменялось»)[8];
  - 305 Use Proxy («использовать прокси»)[3];
  - 306 — *зарезервировано* (код использовался только в ранних спецификациях)[3];
  - 307 Temporary Redirect («временное перенаправление»)[3];
  - 308 Permanent Redirect («постоянное перенаправление»)[9].
- 4xx: Client Error (ошибка клиента):
  - 400 Bad Request («неправильный, некорректный запрос»)[3][4];
  - 401 Unauthorized («не авторизован»)[10];
  - 402 Payment Required («необходима оплата») — *зарезервировано* для использования в будущем[3];
  - 403 Forbidden («запрещено (не уполномочен)»)[3];
  - 404 Not Found («не найдено»)[3];
  - 405 Method Not Allowed («метод не поддерживается»)[3];
  - 406 Not Acceptable («неприемлемо»)[3];
  - 407 Proxy Authentication Required («необходима аутентификация прокси»)[10];
  - 408 Request Timeout («истекло время ожидания»)[3];
  - 409 Conflict («конфликт»)[3][4];
  - 410 Gone («удалён»)[3];
  - 411 Length Required («необходима длина»)[3];
  - 412 Precondition Failed («условие ложно»)[8][11];
  - 413 Payload Too Large («полезная нагрузка слишком велика»)[3];
  - 414 URI Too Long («URI слишком длинный»)[3];
  - 415 Unsupported Media Type («неподдерживаемый тип данных»)[3];
  - 416 Range Not Satisfiable («диапазон не достижим»)[12];
  - 417 Expectation Failed («ожидание не оправдалось»)[3];
  - 418 I'm a teapot («я — чайник»);
  - 419 Authentication Timeout (not in RFC 2616) («обычно ошибка проверки CSRF»);

- 421 Misdirected Request[13];
- 422 Unprocessable Entity («необрабатываемый экземпляр»);
- 423 Locked («заблокировано»);
- 424 Failed Dependency («невыполненная зависимость»);
- 425 Too Early («слишком рано»);
- 426 Upgrade Required («необходимо обновление»)[3];
- 428 Precondition Required («необходимо предусловие»)[14];
- 429 Too Many Requests («слишком много запросов»)[14];
- 431 Request Header Fields Too Large («поля заголовка запроса слишком большие»)[14];
- 449 Retry With («повторить с»)[1];
- 451 Unavailable For Legal Reasons («недоступно по юридическим причинам»)[15].
- 499 Client Closed Request (клиент закрыл соединение);

- 5xx: Server Error (ошибка сервера):
  - 500 Internal Server Error («внутренняя ошибка сервера»)[3];
  - 501 Not Implemented («не реализовано»)[3];
  - 502 Bad Gateway («плохой, ошибочный шлюз»)[3];
  - 503 Service Unavailable («сервис недоступен»)[3];
  - 504 Gateway Timeout («шлюз не отвечает»)[3];
  - 505 HTTP Version Not Supported («версия HTTP не поддерживается»)[3];
  - 506 Variant Also Negotiates («вариант тоже проводит согласование»)[16];
  - 507 Insufficient Storage («переполнение хранилища»);
  - 508 Loop Detected («обнаружено бесконечное перенаправление»)[17];
  - 509 Bandwidth Limit Exceeded («исчерпана пропускная ширина канала»);
  - 510 Not Extended («не расширено»);
  - 511 Network Authentication Required («требуется сетевая аутентификация»)[14];
  - 520 Unknown Error («неизвестная ошибка»)[18];
  - 521 Web Server Is Down («веб-сервер не работает»)[18];
  - 522 Connection Timed Out («соединение не отвечает»)[18];
  - 523 Origin Is Unreachable («источник недоступен»)[18];
  - 524 A Timeout Occurred («время ожидания истекло»)[18];
  - 525 SSL Handshake Failed («квитирование SSL не удалось»)[18];
  - 526 Invalid SSL Certificate («недействительный сертификат SSL»)[18].
  -

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

HTTP-request:

Стартовая строка, заголовки, [тело].

GET /index.html HTTP/1.1
Host: www.ru
Connection: keep-alive
Cache-Control: max-age=0
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/119.0.0.0 Safari/537.36 Edg/119.0.0.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/

webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.7
Accept-Encoding: gzip, deflate
Accept-Language: ru,en;q=0.9,en-GB;q=0.8,en-US;q=0.7

HTTP-response:

Строка состояния, заголовки, [тело].

HTTP/1.1 200 OK
Date: Mon, 27 Jul 2009 12:28:53 GMT
Server: Apache/2.2.14 (Win32)
Last-Modified: Wed, 22 Jul 2009 19:15:56 GMT
Content-Length: 88
Content-Type: text/html
Connection: Closed

Список использованных источников:

https://github.com/Sergey-pixel-dev/web-6

https://stepik.org/course/54403/info

https://ru.wikipedia.org/wiki/%D0%A1%D0%BF%D0%B8%D1%81%D0%BE%D0%BA_%D0%BA%D0%BE%D0%B4%D0%BE%D0%B2_%D1%81%D0%BE%D1%81%D1%82%D0%BE%D1%8F%D0%BD%D0%B8%D1%8F_HTTP