



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

(Группа)

\_\_\_\_\_  
(Подпись, дата)

А.В. Палий

(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.

## Ход работы

Для начала ознакомились с данными в курсе <https://stepik.org/course/54403/info> в разделе "4. Списки, сеть и сервера". Здесь основное внимание стоит уделить урокам "4.2 Работа с сетью" и "4.3 Веб-сервера".

### Задание 1

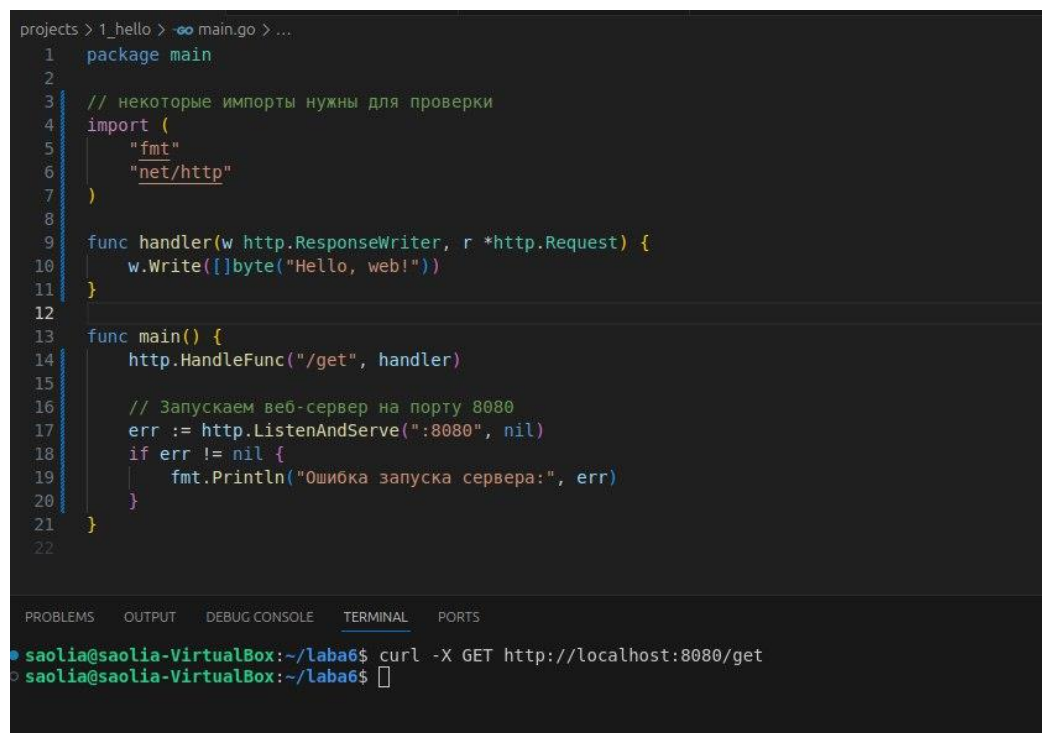
Необходимо написать веб-сервер, который по пути /get отдает текст "Hello, web!". Порт должен быть :8080.

После решения задания полученный код main.go необходимо перенести в данный репозиторий.

Код должен компилироваться, а сервер запускаться и корректно обрабатывать запросы.

Для локальной отладки можно использовать Postman или Insomnia.

Результат работы представлен на рисунке 1.



```
projects > 1_hello > -o main.go > ...
1 package main
2
3 // некоторые импорты нужны для проверки
4 import (
5     "fmt"
6     "net/http"
7 )
8
9 func handler(w http.ResponseWriter, r *http.Request) {
10     w.Write([]byte("Hello, web!"))
11 }
12
13 func main() {
14     http.HandleFunc("/get", handler)
15
16     // Запускаем веб-сервер на порту 8080
17     err := http.ListenAndServe(":8080", nil)
18     if err != nil {
19         fmt.Println("Ошибка запуска сервера:", err)
20     }
21 }
22
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
saolia@saolia-VirtualBox:~/laba6$ curl -X GET http://localhost:8080/get
saolia@saolia-VirtualBox:~/laba6$
```

Рисунок 1 - Результат

### Задание 2

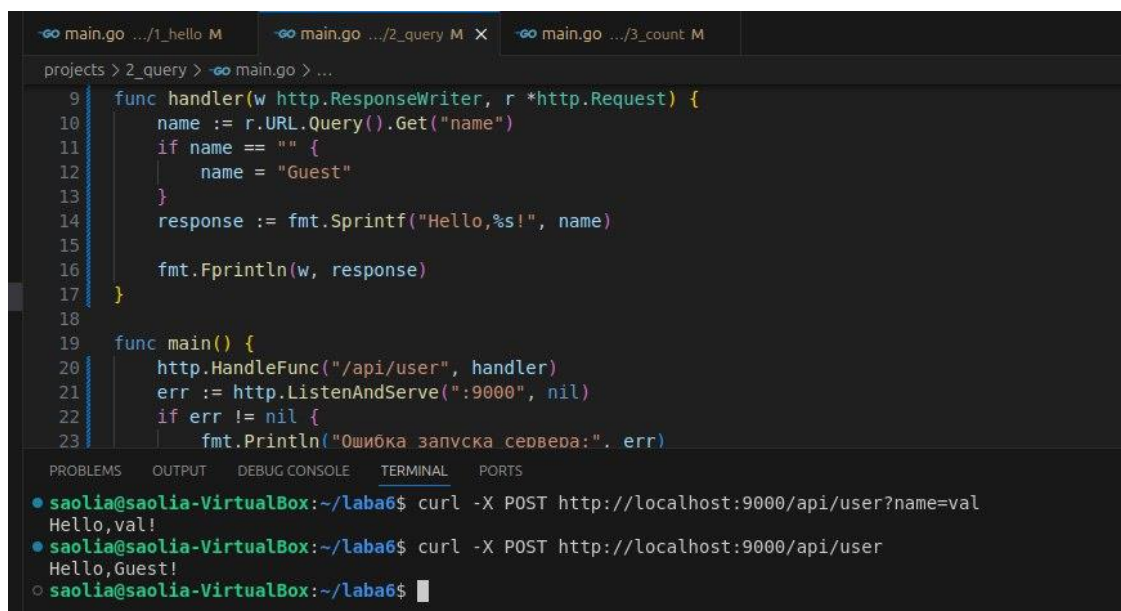
Необходимо написать веб-сервер, который по пути `/api/user` приветствует пользователя. Сервер по этому пути должен принимать и парсить параметр `name`, после этого отвечая в формате: `"Hello,<name>!"`. Пример url: `/api/user?name=Golang`

После решения задания полученный код `main.go` необходимо перенести в данный репозиторий в директорию с данным файлом `README.md`

Код должен компилироваться, а сервер запускаться и корректно обрабатывать запросы.

Для локальной отладки можно использовать Postman или Insomnia.

Результат работы представлен на рисунке 2.



```
9 func handler(w http.ResponseWriter, r *http.Request) {
10     name := r.URL.Query().Get("name")
11     if name == "" {
12         name = "Guest"
13     }
14     response := fmt.Sprintf("Hello,%s!", name)
15
16     fmt.Fprintln(w, response)
17 }
18
19 func main() {
20     http.HandleFunc("/api/user", handler)
21     err := http.ListenAndServe(":9000", nil)
22     if err != nil {
23         fmt.Println("Ошибка запуска сервера:", err)
24     }
25 }
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
● saolia@saolia-VirtualBox:~/laba6$ curl -X POST http://localhost:9000/api/user?name=val
Hello,val!
● saolia@saolia-VirtualBox:~/laba6$ curl -X POST http://localhost:9000/api/user
Hello,Guest!
○ saolia@saolia-VirtualBox:~/laba6$
```

Рисунок 2 – Результат

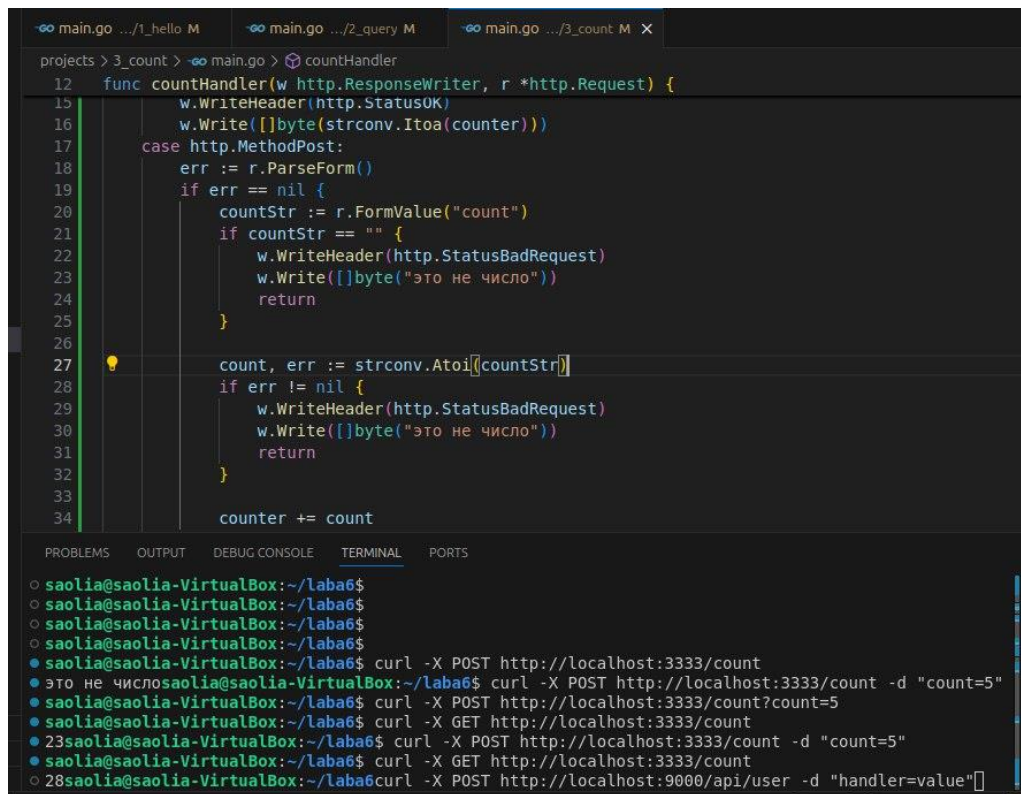
### Задание 3

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).

Результат работы представлен на рисунке 3.



```
main.go .../1_hello M  main.go .../2_query M  main.go .../3_count M X
projects > 3_count > main.go > countHandler
12 func countHandler(w http.ResponseWriter, r *http.Request) {
15     w.WriteHeader(http.StatusOK)
16     w.Write([]byte(strconv.Itoa(counter)))
17     case http.MethodPost:
18         err := r.ParseForm()
19         if err == nil {
20             countStr := r.FormValue("count")
21             if countStr == "" {
22                 w.WriteHeader(http.StatusBadRequest)
23                 w.Write([]byte("это не число"))
24                 return
25             }
26
27             count, err := strconv.Atoi(countStr)
28             if err != nil {
29                 w.WriteHeader(http.StatusBadRequest)
30                 w.Write([]byte("это не число"))
31                 return
32             }
33
34             counter += count
35         }
36     }
37 }
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
saolia@saolia-VirtualBox:~/laba6$
saolia@saolia-VirtualBox:~/laba6$
saolia@saolia-VirtualBox:~/laba6$
saolia@saolia-VirtualBox:~/laba6$
saolia@saolia-VirtualBox:~/laba6$ curl -X POST http://localhost:3333/count
saolia@saolia-VirtualBox:~/laba6$ curl -X POST http://localhost:3333/count -d "count=5"
saolia@saolia-VirtualBox:~/laba6$ curl -X POST http://localhost:3333/count?count=5
saolia@saolia-VirtualBox:~/laba6$ curl -X GET http://localhost:3333/count
23saolia@saolia-VirtualBox:~/laba6$ curl -X POST http://localhost:3333/count -d "count=5"
saolia@saolia-VirtualBox:~/laba6$ curl -X GET http://localhost:3333/count
28saolia@saolia-VirtualBox:~/laba6$ curl -X POST http://localhost:9000/api/user -d "handler=value"
```

Рисунок 3 – Результат

## Заключение

В ходе выполнения лабораторной работы изучили основы сетевого взаимодействия и серверной разработки с использованием языка Golang.

## Контрольные вопросы

### 1. Разница между протоколами TCP и UDP:

- TCP (Transmission Control Protocol) - надежный, ориентированный на соединение протокол. Он обеспечивает гарантированную доставку данных, контроль потока и порядка пакетов.

- UDP (User Datagram Protocol) - ненадежный, без установления соединения протокол. Он не гарантирует доставку пакетов и не контролирует их порядок. Применяется для передачи данных, где важна скорость, а не надежность (например, потоковое видео).

### 2. IP-адрес и номер порта веб-сервера:

- IP-адрес (Internet Protocol address) - уникальный идентификатор устройства в сети Интернет. Он позволяет маршрутизировать трафик до нужного устройства.

- Номер порта (Port number) - номер логического "канала" на хосте, используемый для идентификации приложения, принимающего и отправляющего

сетевые пакеты. Это позволяет нескольким приложениям на одном хосте обмениваться данными независимо.

### 3. Методы HTTP, реализующие CRUD:

- Create (POST)
- Read (GET)
- Update (PUT/PATCH)
- Delete (DELETE)

### 4. Группы кодов состояния HTTP-ответов:

- 1xx (Informational) - запрос принят, продолжается обработка
- 2xx (Success) - запрос успешно обработан (например, 200 OK)
- 3xx (Redirection) - клиенту требуется выполнить дополнительные действия (например, 301 Moved Permanently)
- 4xx (Client Error) - ошибка на стороне клиента (например, 404 Not Found)
- 5xx (Server Error) - ошибка на стороне сервера (например, 500 Internal Server Error)

### 5. Элементы HTTP-запроса и HTTP-ответа:

#### HTTP-запрос:

- Метод (GET, POST, PUT, DELETE, etc.)
- URL
- Заголовки (Headers)
- Тело (Body)

#### HTTP-ответ:

- Версия протокола
- Код состояния (Status Code)
- Заголовки (Headers)
- Тело (Body)