



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

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)

Н.Е. Мамаев

(И.О. Фамилия)

Преподаватель

(Подпись, дата)

В.Д. Шульман

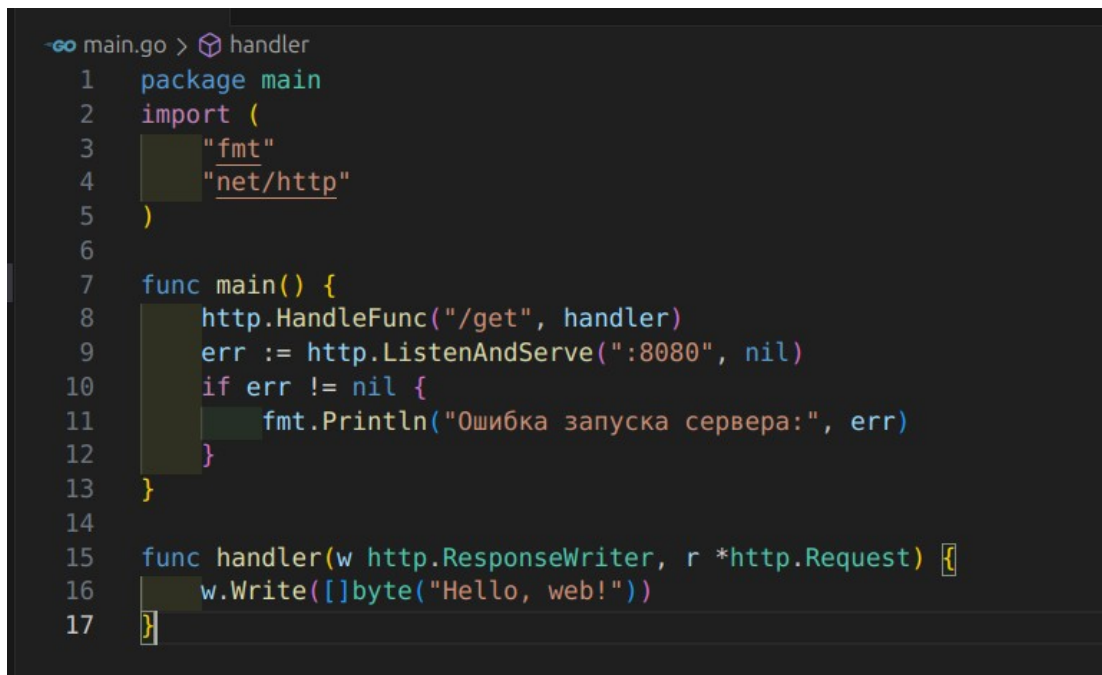
(И.О. Фамилия)

Москва, 2024

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

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

Код программы приведен на рисунке 1:



```
main.go > handler
1 package main
2 import (
3     "fmt"
4     "net/http"
5 )
6
7 func main() {
8     http.HandleFunc("/get", handler)
9     err := http.ListenAndServe(":8080", nil)
10    if err != nil {
11        fmt.Println("Ошибка запуска сервера:", err)
12    }
13 }
14
15 func handler(w http.ResponseWriter, r *http.Request) {
16     w.Write([]byte("Hello, web!"))
17 }
```

Рисунок 1: Код программы 1.

Работа программы продемонстрирована на рисунке 2:

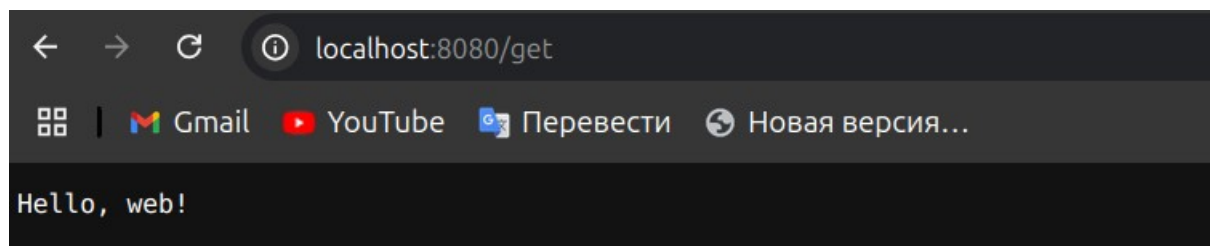


Рисунок 2: Пример работы 1.

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 приведен на рисунке 3:

```

main.go > ...
1  package main
2  import (
3      "fmt"
4      "net/http"
5  )
6
7  func handler(w http.ResponseWriter, r *http.Request) {
8      name := r.URL.Query().Get("name")
9      w.Write([]byte("Hello," + name + "!"))
10 }
11
12 func main() {
13     http.HandleFunc("/api/user", handler)
14
15     err := http.ListenAndServe(":9000", nil)
16     if err != nil {
17         fmt.Println("Ошибка запуска сервера:", err)
18     }
19 }
20
21

```

Работа программы продемонстрирована на рисунке 4:

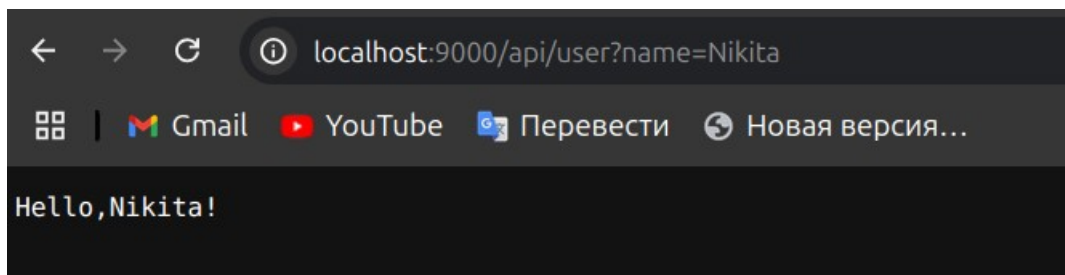


Рисунок 4: Пример работы программы 2.

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 приведен на рисунке 5:

```

9 func handler(w http.ResponseWriter, r *http.Request) {
10     switch r.Method{
11     case http.MethodGet:
12         w.WriteHeader(http.StatusOK)
13         w.Write([]byte(fmt.Sprintf("%d", counter)))
14     case http.MethodPost:
15         err := r.ParseForm()
16         if err == nil{
17             countStr := r.FormValue("count")
18             var count int
19             _, err2 := fmt.Sscanf(countStr, "%d", &count)
20             if err2 != nil{
21                 w.WriteHeader(http.StatusBadRequest)
22                 w.Write([]byte("это не число"))
23             }else{
24                 w.WriteHeader(http.StatusBadRequest)
25                 counter += count
26             }
27         }else{
28             w.WriteHeader(http.StatusBadRequest)
29             return
30         }
31     }
32 }
33
34 func main() {
35     http.HandleFunc("/", handler)
36     err := http.ListenAndServe(":3333", nil)
37     if err != nil {
38         fmt.Println("Ошибка запуска сервера:", err)
39     }
40 }

```

Рисунок 5: Код программы 3.

Демонстрация GET-запроса после двух POST-запросов приведена на рисунке 6:

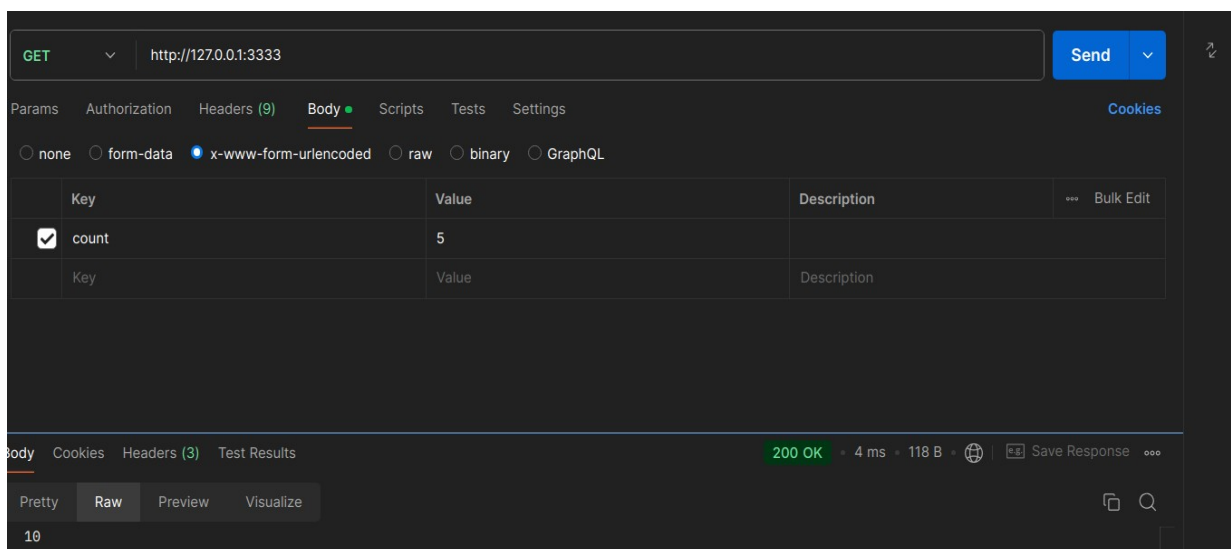


Рисунок 6: Пример работы программы 3.

Контрольные вопросы

1. В чём разница между протоколами TCP и UDP ?

TCP (Transmission Control Protocol) – это надёжный 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) – 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 – это уникальный адрес, который присваивается устройству при подключении компьютерной сети. То есть с помощью IP можно идентифицировать устройство в сети.

Port Number используется для идентификации у устройства программы, которая осуществляет работу с данными в сети.

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

Create – POST

Read – GET

Update – PUT

Delete - DELETE

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

1xx — информационные. Например, 102 – Идёт обработка

2xx — успешные. Например, 202 - Принято

3xx — перенаправление. Например, 305 – Использовать прокси

4xx — клиентские ошибки. Например, 404 – Не найдено

5xx — серверные ошибки. Например, 501 – Не реализовано

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

Структура HTTP запроса и ответа:

- 1) Стартовая строка
- 2) Заголовки
- 3) Тело

Вывод: в ходе лабораторной работы изучены основы сетевого взаимодействия и серверной разработки с использованием языка Golang.