



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

ИУ6-33Б

(Группа)

(Подпись, дата)

Б.Ш Истамилов

(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:

```
package main

import (
    "fmt"
    "net/http"
)

func main() {
    http.HandleFunc("/get", func(w http.ResponseWriter, r *http.Request) {
        fmt.Fprintf(w, "Hello, web!")
    })

    http.ListenAndServe(":8080", nil)
}
```

Тест программы:

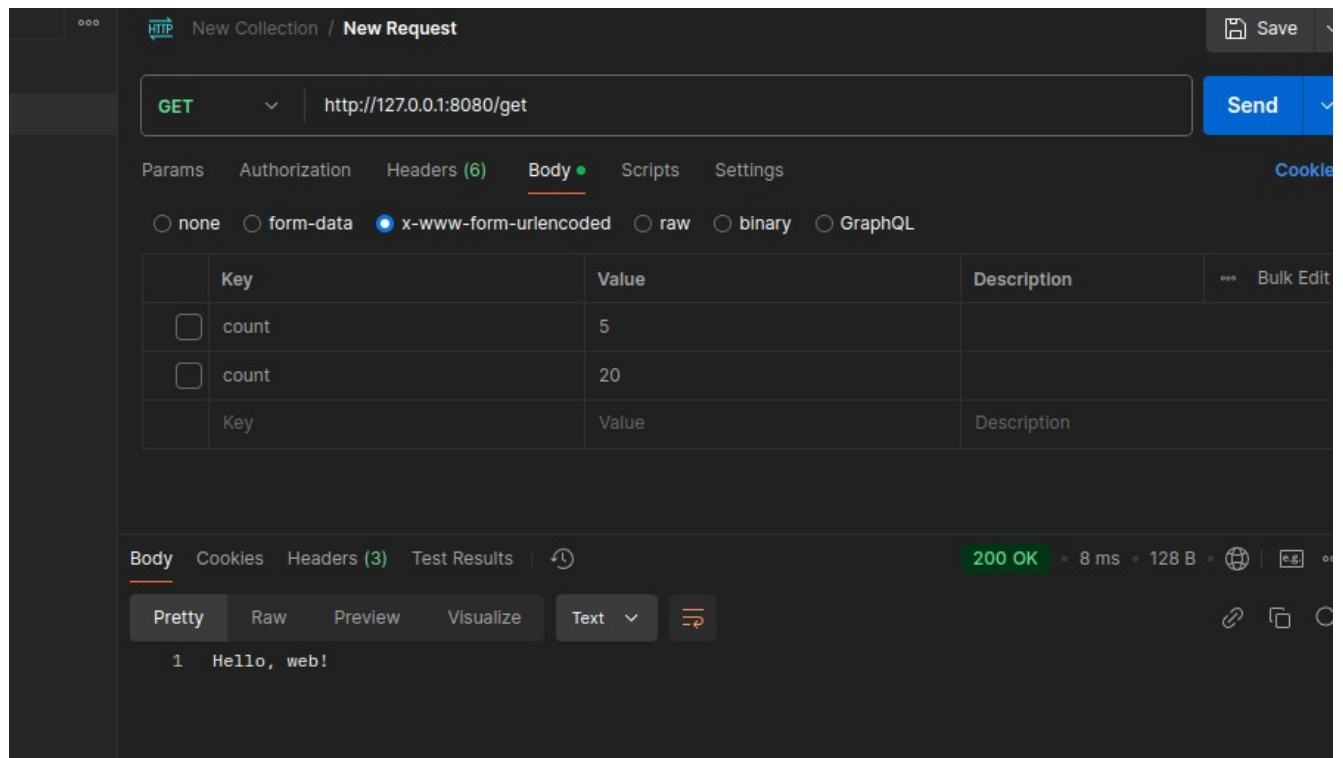


Рисунок 1 — тест программы

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!`

Код программы:

```
package main

import (
    "fmt"
    "net/http"
)

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

Тест программы:

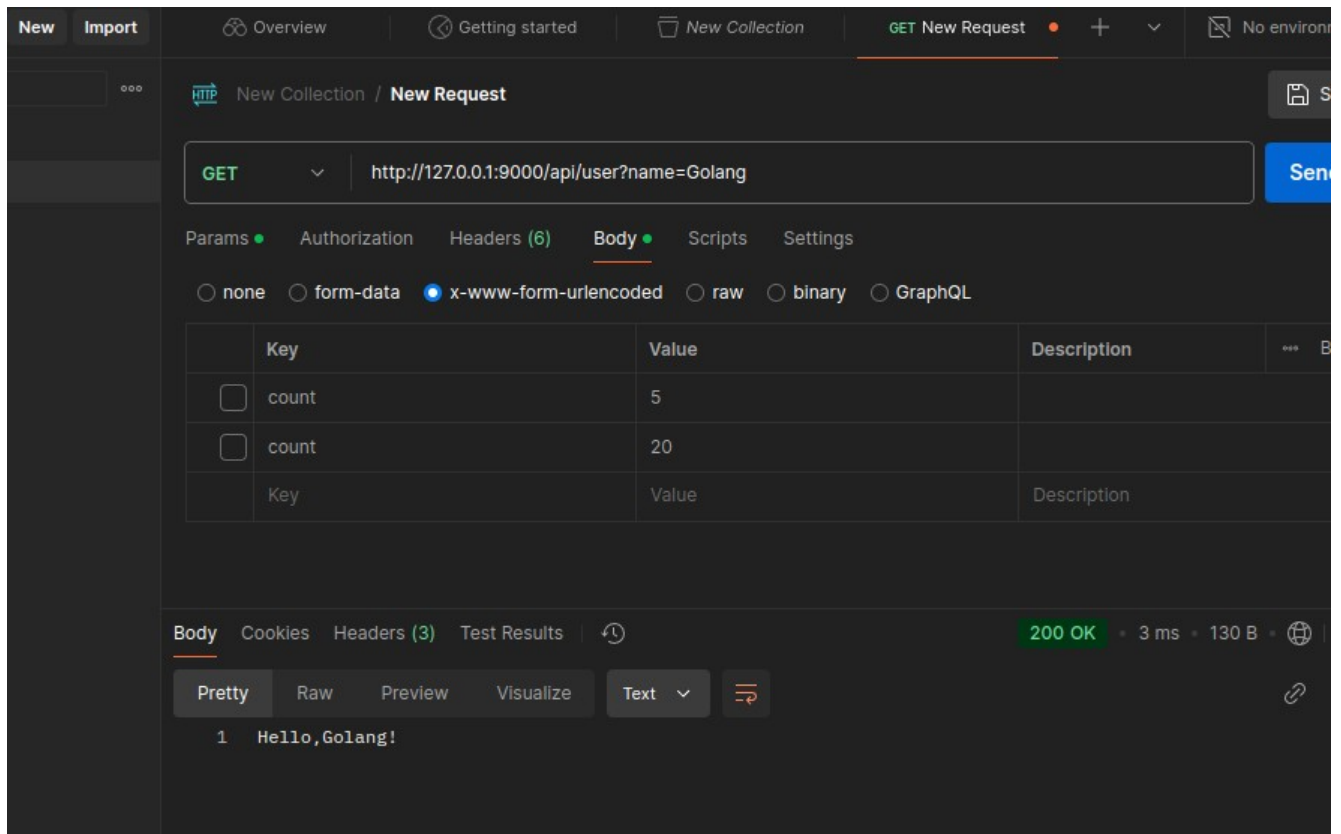


Рисунок 2 — тест программы

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:

`ackage main`

```
import (  
    "fmt"  
    "net/http"  
    "strconv"  
)
```

```
func main() {  
    counter := 0  
    http.HandleFunc("/count", func(w http.ResponseWriter, r *http.Request) {
```

```

if r.Method == http.MethodGet {
w.Write([]byte(strconv.Itoa(counter)))
}
if r.Method == http.MethodPost {
r.ParseForm()
numberString := r.Form.Get("count")
a, err := strconv.Atoi(numberString)
if err != nil {
w.WriteHeader(http.StatusBadRequest)
fmt.Fprintln(w, "это не число")
return
}
counter += a
}
if r.Method != http.MethodPost && r.Method != http.MethodGet {
http.Error(w, "method is not allowed", http.StatusMethodNotAllowed)
}

})
http.ListenAndServe(":3333", nil)
}

```

Тест программы:

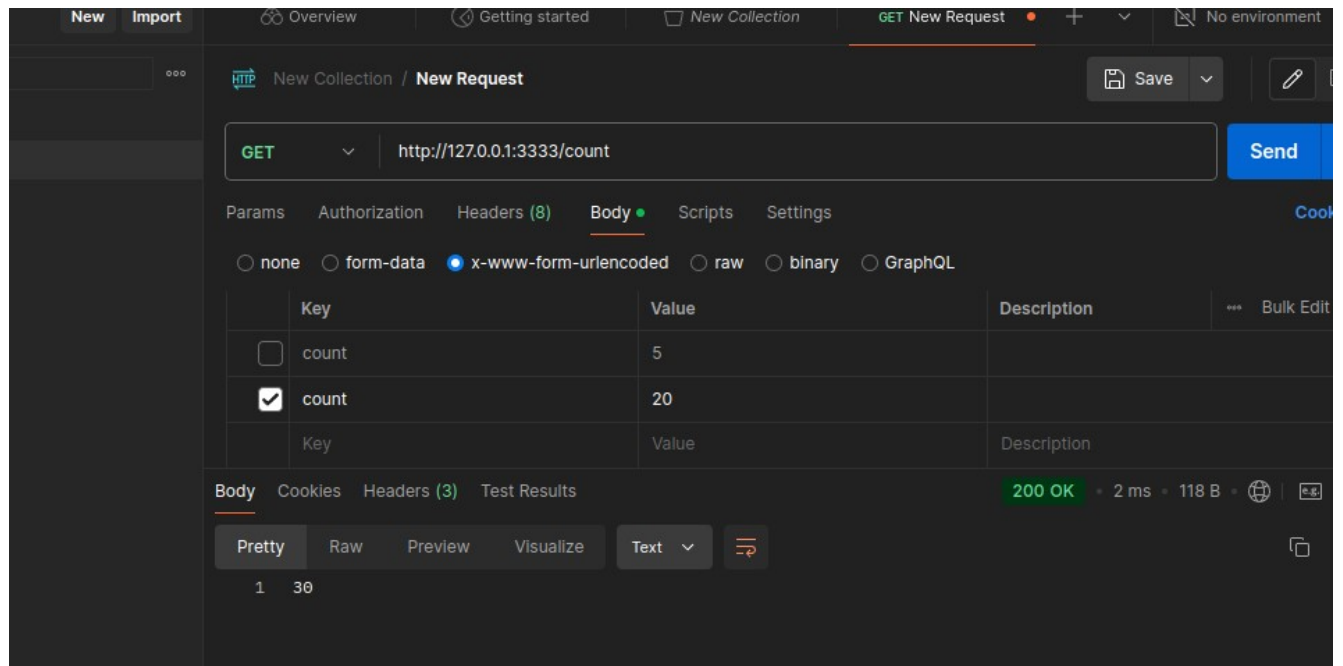


Рисунок 3— тест программы

Вывод: в процессе выполнения лабораторной работы были изучены основы back-end разработки на языке golang