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: Back-end разработка на GoLang

Дисциплина: Языки интернет программирования

Студент	ИУ6-33Б	11.10.24	Пономаренко В.М.
	(Группа)	(Подпись, дата)	(И.О. Фамилия)
Преподаватель		11.10.24	Шульман В.Д.
		(Подпись, дата)	(И.О. Фамилия)

Москва, 2024

Цель работы: Изучение основ сетевого взаимодействия и серверной разработки с использованием языка Golang

Задание: Написать несколько веб-серверов:

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

2) Задание “Query”
   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

3) Задание “Count”
   Написать 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).

Ход работы:

Задание 1. “Hello”

Листинг:

```go
package main

import (
    "fmt"
    "net/http"
)

func handler(w http.ResponseWriter, r *http.Request) {
    w.Write([]byte("Hello, web!"))
    // w - интерфейс для ответа клиенту
    // r - структура с данными о запросе

}

func main() {
    http.HandleFunc("/get", handler)         // регистрация обработчика для пути "/get"
    err := http.ListenAndServe(":8080", nil) // запуск сервера на порту 8080
    if err != nil {
        fmt.Println("error")
    }
}
//http:///localhost:8080/get - url, указывает на веб ресурс
```

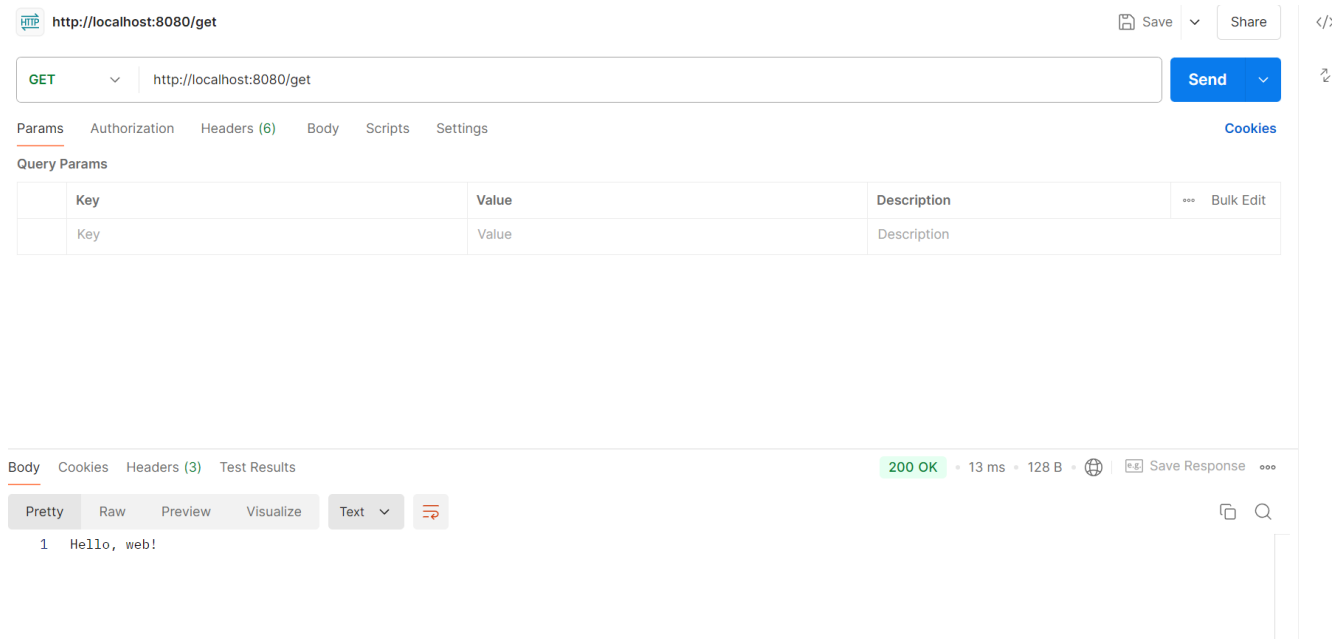

Рисунок 1 - тестирование задачи 1 через Postman

Задание 2. “Query”

Листинг задачи 2:

```go
package main

import (
    "fmt"
    "net/http"
)

func handler(w http.ResponseWriter, r *http.Request){
    // через метод URL получаем запрошенный адрес
    // у адреса вызывваем метод Query(), возвращает строку запроса
    // Get() для получения значения отдельного параметра
    name := r.URL.Query().Get("name")
    ans := "Hello," + name + "!"
    w.Write([]byte(ans))
}

func main(){
    http.HandleFunc("/api/user", handler)
    err := http.ListenAndServe(":9000", nil)
    if err != nil {
        fmt.Println("error")
    }
}
```

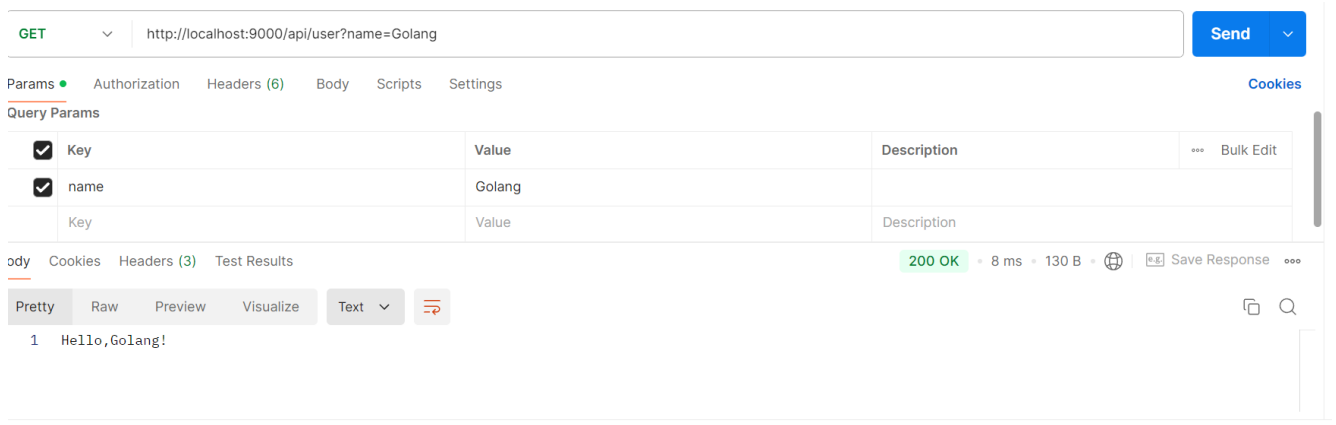

Рисунок 2 - тестирование 1 задачи 2

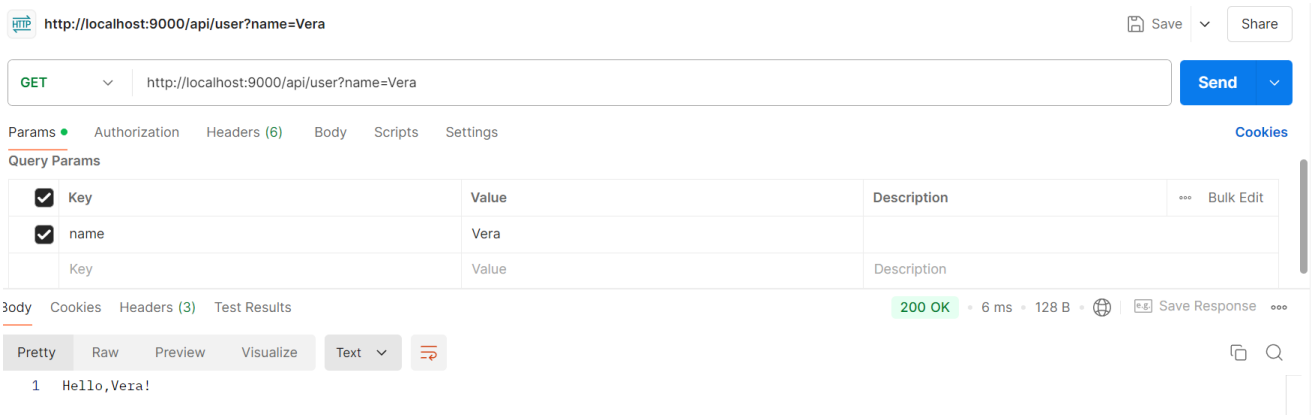

Рисунок 3 - тестирование 2 задачи 2

Задание 3. “Count”

Листинг задачи 3

```go
package main

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

var counter = 0

func handler(w http.ResponseWriter, r *http.Request) {
    if r.Method == "GET" {
        w.Write([]byte(strconv.Itoa(counter)))
        return
    } else if r.Method == "POST" {
        r.ParseForm()               // в POST предаются данные, собираем их в структуру, доступную через r.Form
        count := r.Form.Get("count") //получаем значение по ключу count
        if countInt, err := strconv.Atoi(count); err != nil {
            w.WriteHeader(http.StatusBadRequest)
            w.Write([]byte("Это не число"))
            return
        } else {
            counter += countInt
            w.WriteHeader(http.StatusOK)
            w.Write([]byte(strconv.Itoa(counter)))
            return
        }
    } else {
        w.WriteHeader(http.StatusMethodNotAllowed)
        w.Write([]byte("этот метод не разрешен"))
        return
    }

}

func main() {
    http.HandleFunc("/count", handler)
    err := http.ListenAndServe(":3333", nil)
    if err != nil {
        fmt.Println("error")
    }
}
```

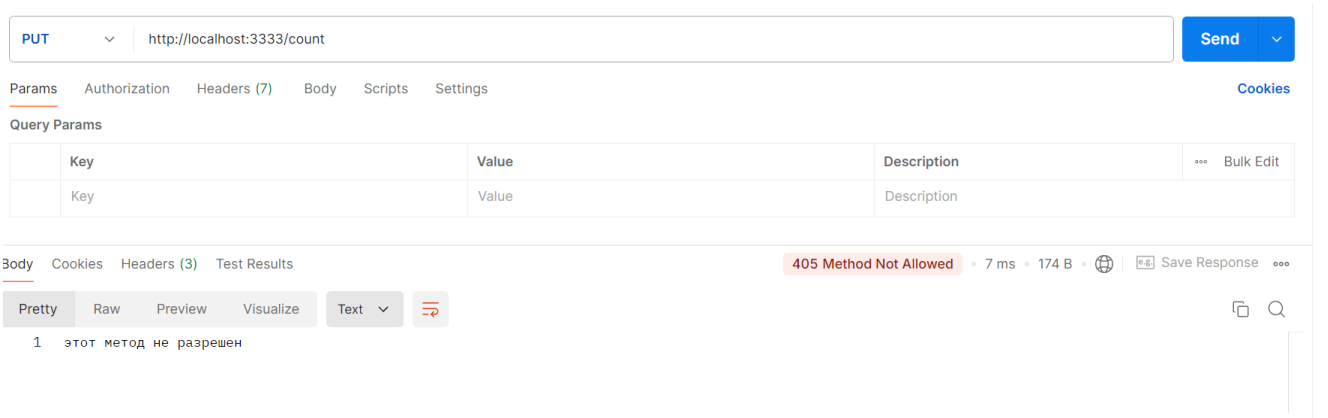

Рисунок 4 - тест 1 задачи 3 (недопустимый метод)

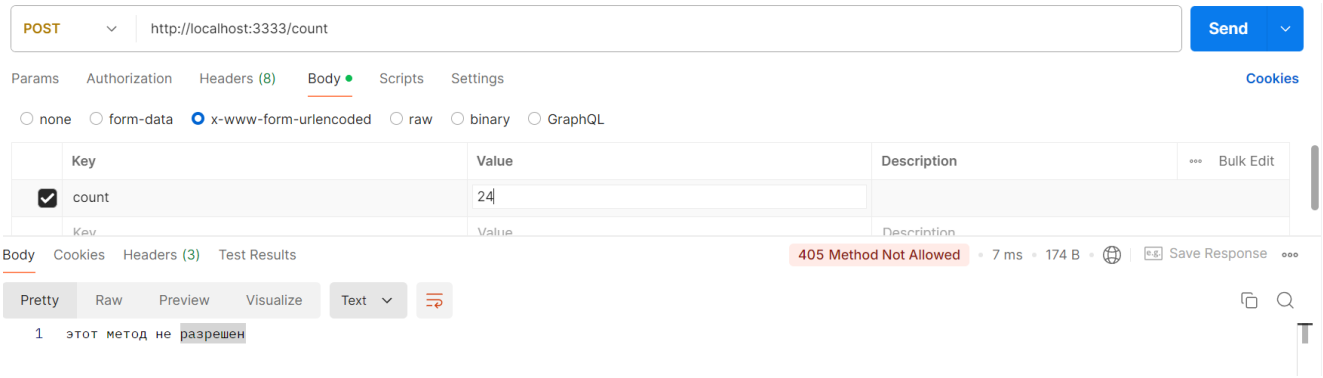

Рисунок 5 - настройка body для тестирования POST

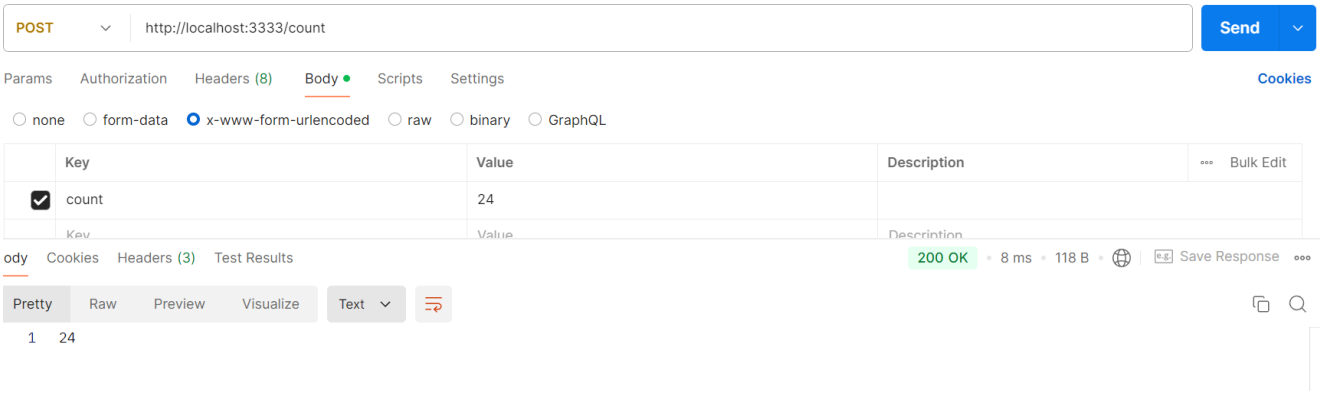

Рисунок 6 - тест 2 задачи 3 (метод POST с корректными данными)

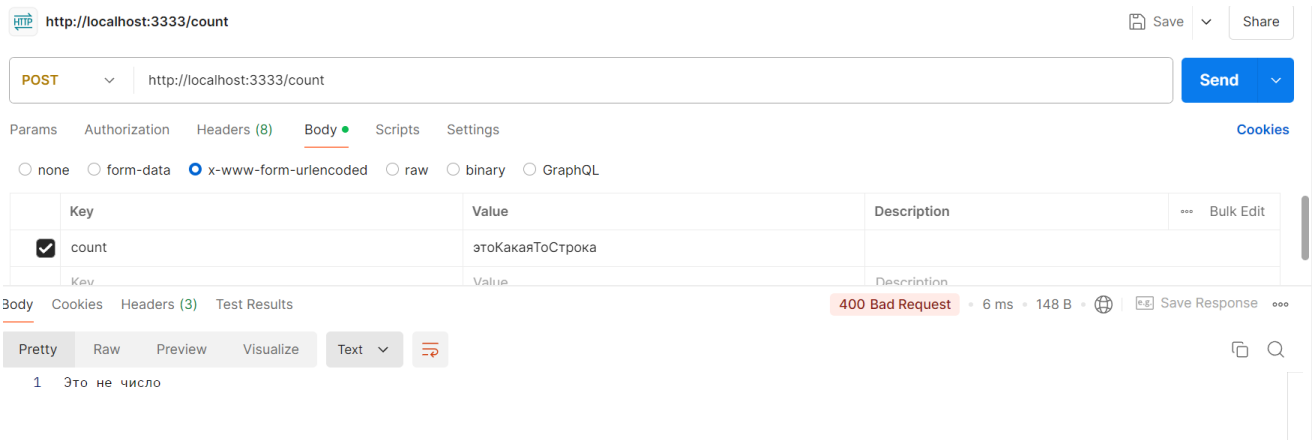

Рисунок 7 – тест 3 задачи 3 (метод POST с НЕкорректными данными)

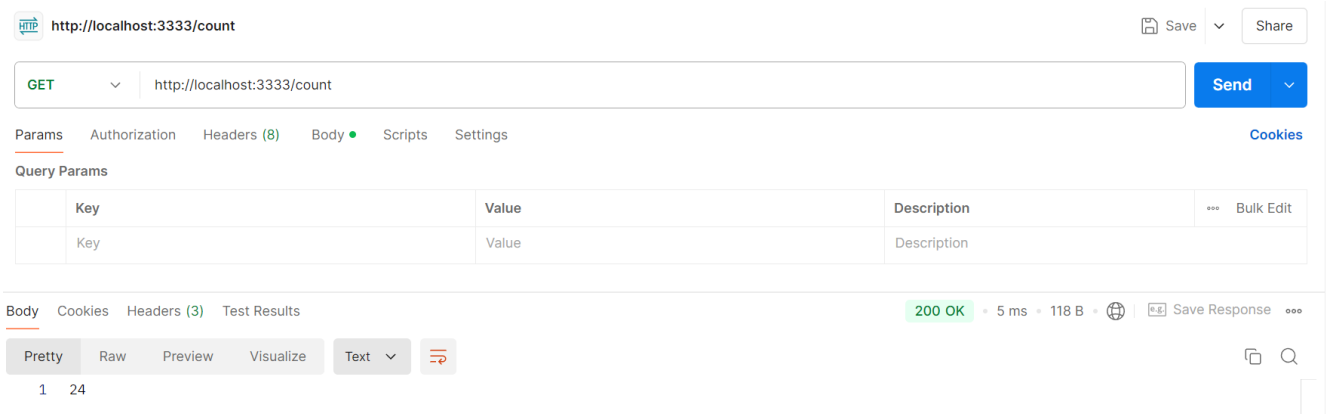

Рисунок 8 - тест 4 задачи 3 (метод GET)

Выводы: В ходе выполнения лабораторной работы были изучены некоторые базовые элементы стандартных библиотек, используемых для организации сетевого взаимодействия и разработки серверных приложений. Полученные данные были закреплены на практике посредством решенеия задач на создание простых веб-серверов. Также созданные веб-сервера были протестированы с помощью Postman.

Список использованных источников:

1) https://stepik.org/course/54403/syllabus
2) https://ru.hexlet.io/blog/posts/postman
3) https://selectel.ru/blog/http-request/