|  |  |
| --- | --- |
| Gerb-BMSTU_01 | **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)**

**Отчет**

**по лабораторной работе № 6**

**Дисциплина:** Языки интернет-программирования

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Студент | ИУ6-32Б |  |  | А. В. Тимохин |
|  | (Группа) |  | (Подпись, дата) | (И.О. Фамилия) |
|  |  |  |  |  |
| Преподаватель |  |  |  | В. Д. Шульман |
|  |  |  | (Подпись, дата) | (И.О. Фамилия) |

Москва, 2024

Цель: освоить работу с веб-серверами, решить несколько задач на эту тему.

Задача 1:

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

package main

import (

    "fmt"

    "net/http"

)

func main() {

    http.HandleFunc("/get", func(w http.ResponseWriter, r \*http.Request) {

        fmt.Fprint(w, "Hello, web!")

    })

    fmt.Println("Starting server on :8080")

    if err := http.ListenAndServe(":8080", nil); err != nil {

        fmt.Println("Error starting server:", err)

    }

}

Запуск программы в консоли представлен на рисунке 1:



Рисунок 1 – запуск программы

Результат работы в postman:

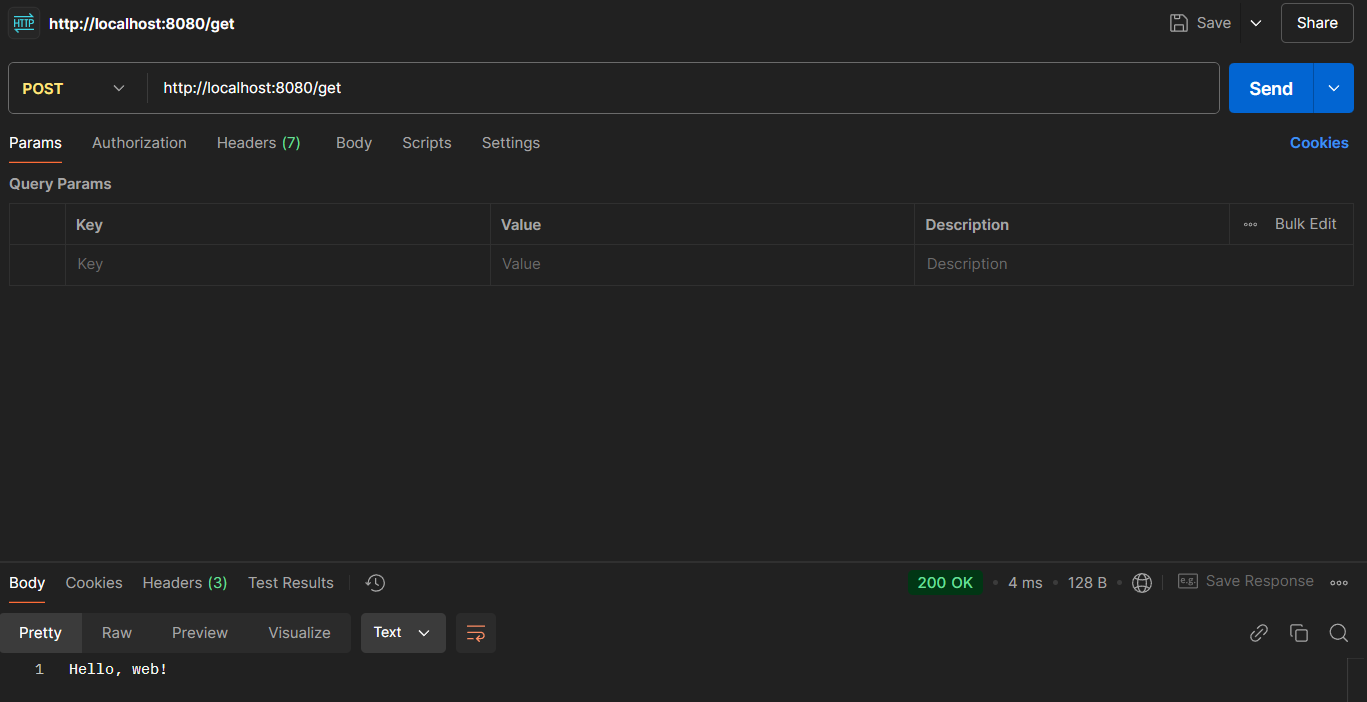


Рисунок 2 – результат в postman

При неправильном вводе команды:

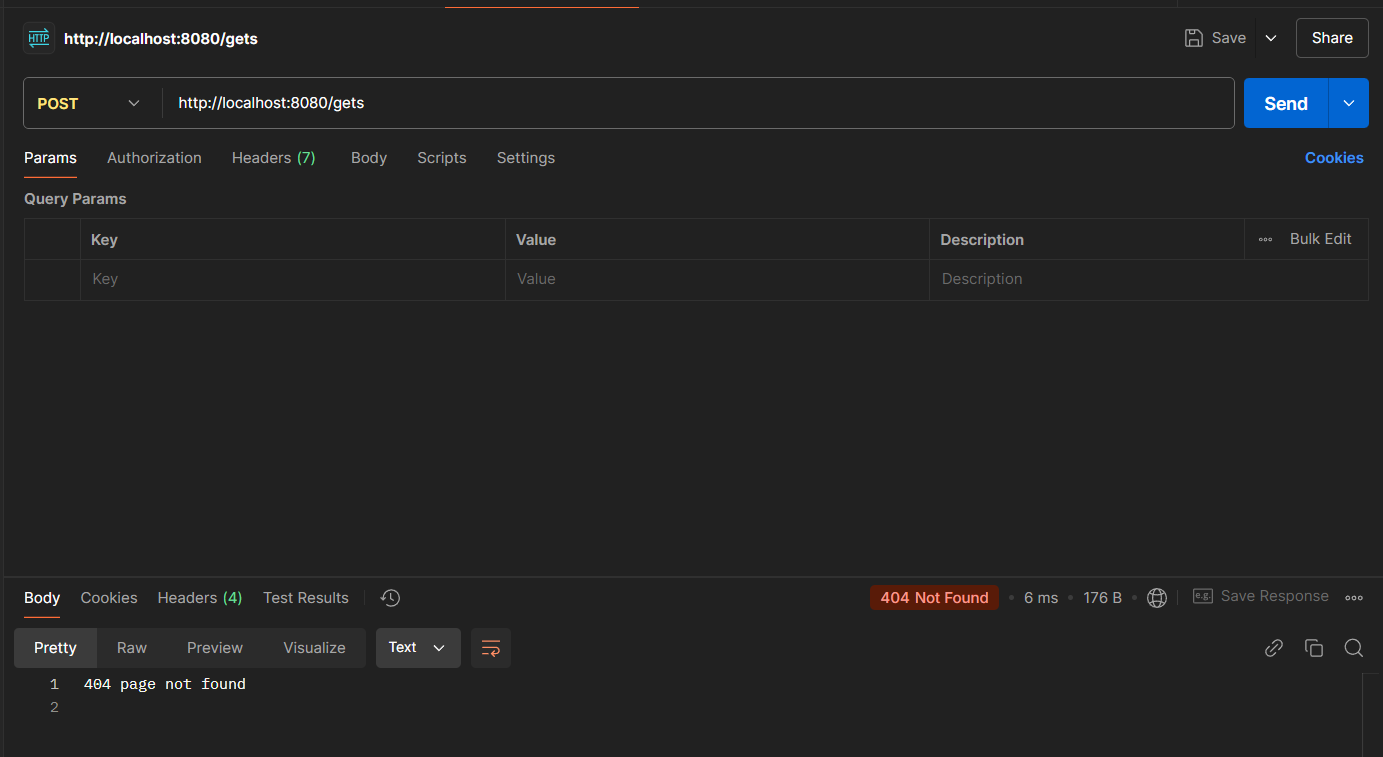


Рисунок 3 – неправильный результат в postman

Задача 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 (

    "fmt"

    "net/http"

)

func main() {

    http.HandleFunc("/api/user", func(w http.ResponseWriter, r \*http.Request) {

        name := r.URL.Query().Get("name")

        if name == "" {

            http.Error(w, "Parameter 'name' is required", http.StatusBadRequest)

            return

        }

        response := fmt.Sprintf("Hello,%s!", name)

        fmt.Fprintln(w, response)

    })

    fmt.Println("Сервер запущен на порту :9000")

    err := http.ListenAndServe(":9000", nil)

    if err != nil {

        fmt.Println("Ошибка запуска сервера:", err)

    }

}

Пример запуска программы в postman представлен на рисунке 4:

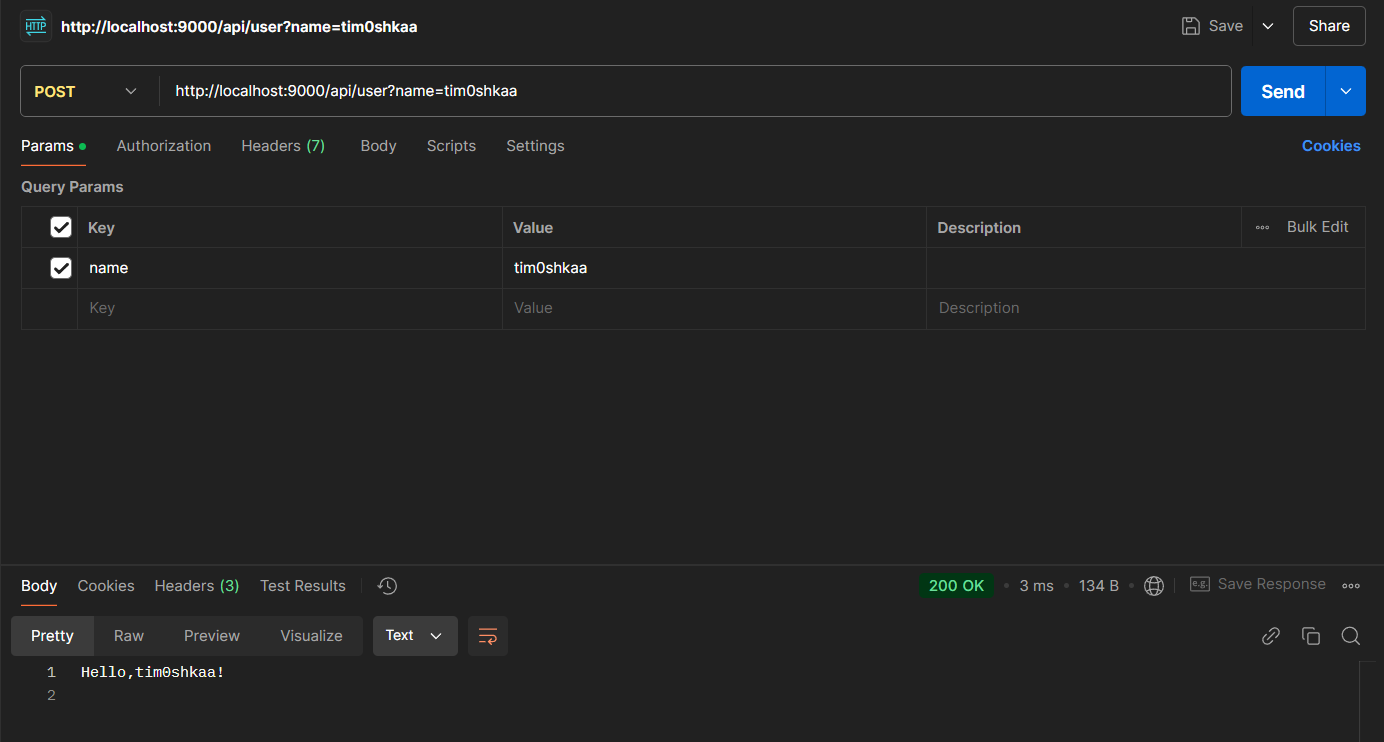


Рисунок 4 – пример работы в postman

Задача 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 (

    "fmt"

    "net/http"

    "strconv"

    "sync"

)

var (

    counter int        // Переменная-счетчик

    mu      sync.Mutex // Мьютекс для потокобезопасных операций

)

func main() {

    http.HandleFunc("/count", countHandler)

    // Запускаем сервер на порту 3333

    fmt.Println("Сервер запущен на порту :3333")

    if err := http.ListenAndServe(":3333", nil); err != nil {

        fmt.Println("Ошибка запуска сервера:", err)

    }

}

// Обработчик для маршрута /count

func countHandler(w http.ResponseWriter, r \*http.Request) {

    switch r.Method {

    case http.MethodGet: // Обрабатываем GET запрос

        mu.Lock()

        fmt.Fprintf(w, "Текущий счетчик: %d", counter)

        mu.Unlock()

    case http.MethodPost: // Обрабатываем POST запрос

        if err := r.ParseForm(); err != nil { // Парсим данные формы

            http.Error(w, "Ошибка обработки формы", http.StatusBadRequest)

            return

        }

        fmt.Print("hello")

        // Получаем значение из ключа "count"

        countValue := r.FormValue("count")

        // Преобразуем строку в число

        countInt, err := strconv.Atoi(countValue)

        if err != nil { // Если преобразование прошло с ошибкой

            http.Error(w, "это не число", http.StatusBadRequest)

            return

        }

        // Увеличиваем счетчик

        mu.Lock()

        counter += countInt

        mu.Unlock()

        fmt.Fprintf(w, "Счетчик увеличен на %d. Текущий счетчик: %d", countInt, counter)

    default: // Если метод не поддерживается

        http.Error(w, "Метод не поддерживается", http.StatusMethodNotAllowed)

    }

}

Пример работы программы представлен на рисунках 5-7:

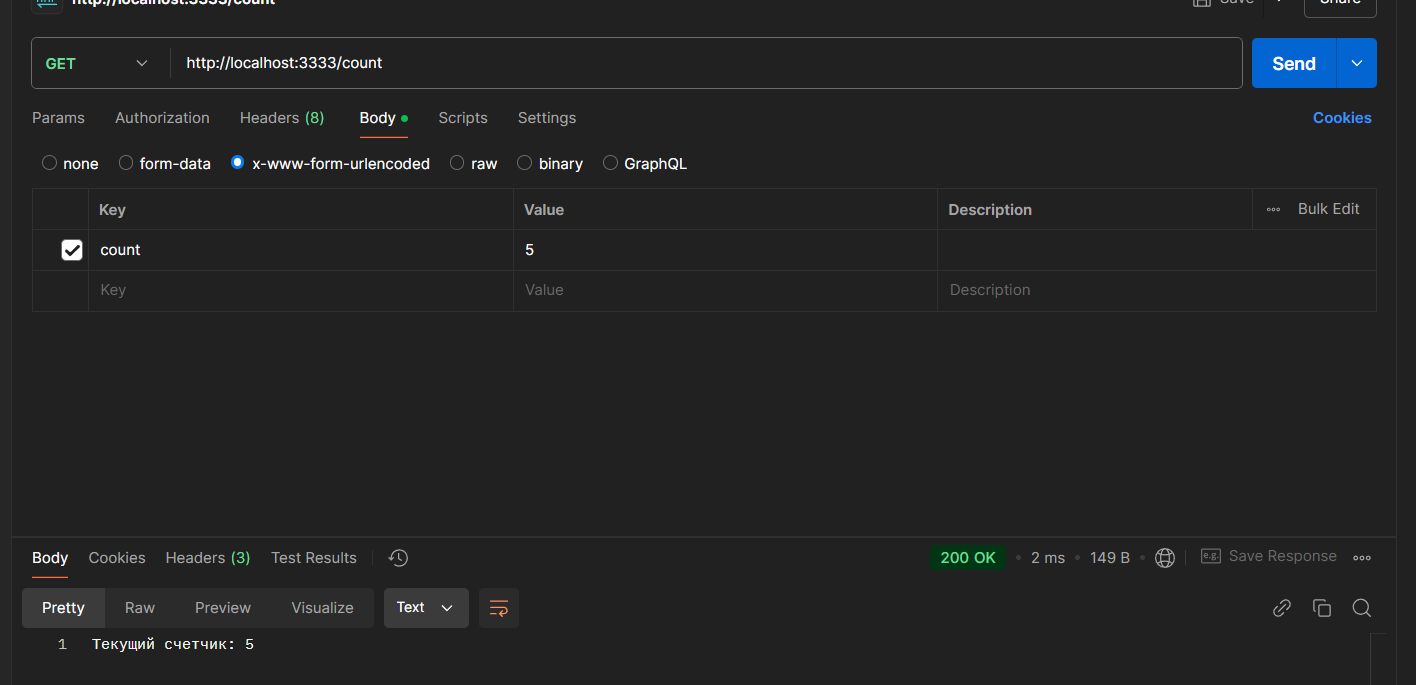


Рисунок 5 – пример работы на postman

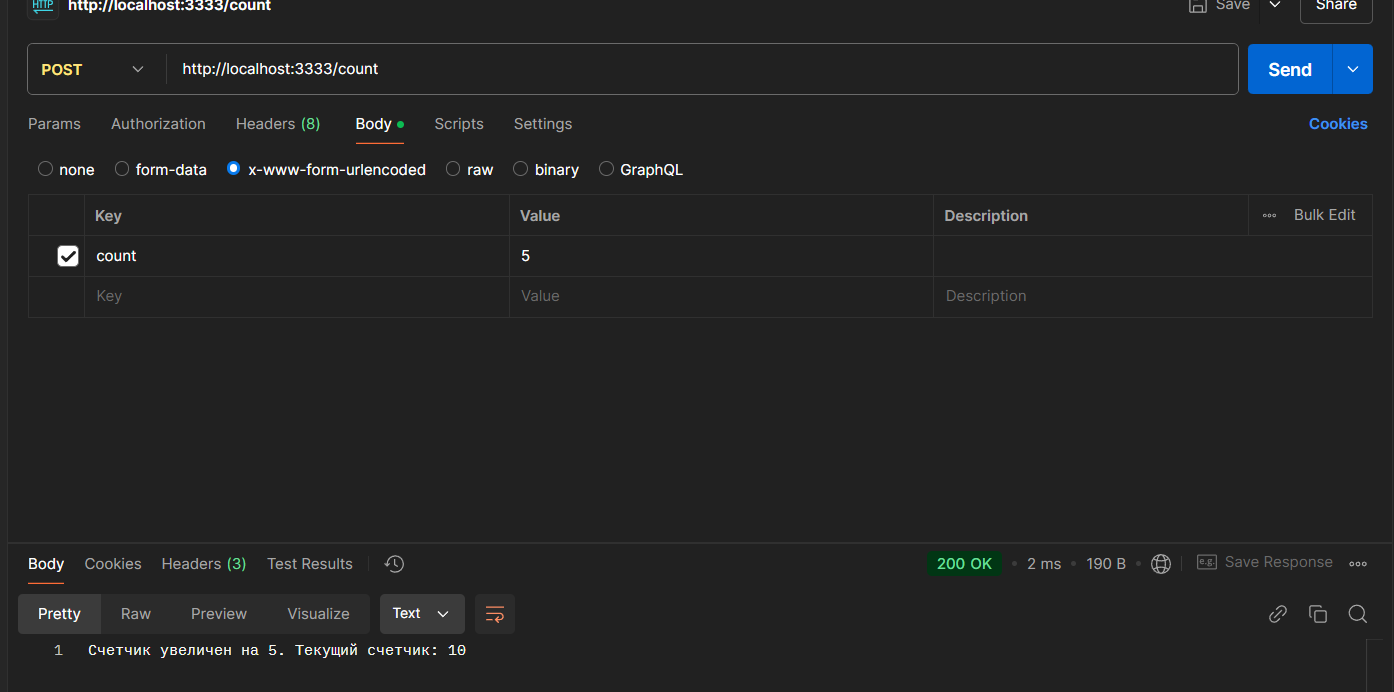


Рисунок 6 – пример работы на postman

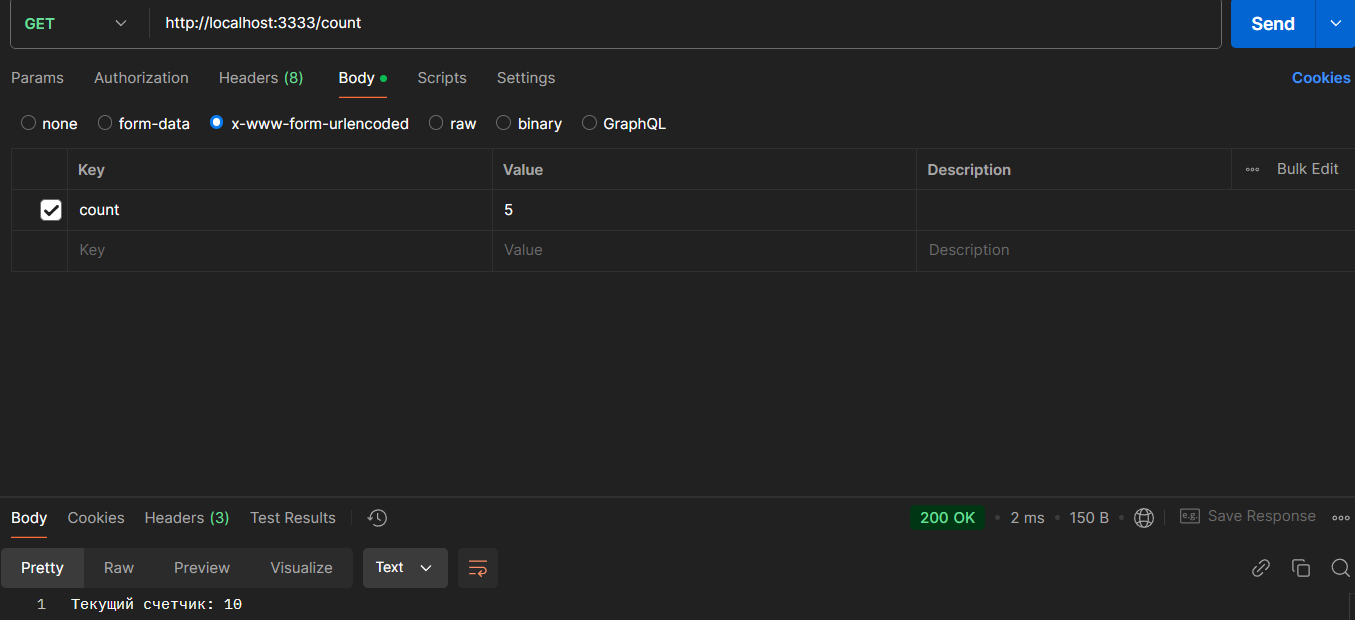


Рисунок 7 – пример работы на postman

Вывод: я освоил работу с веб-сервером и выполнил несколько задач на эту тему.