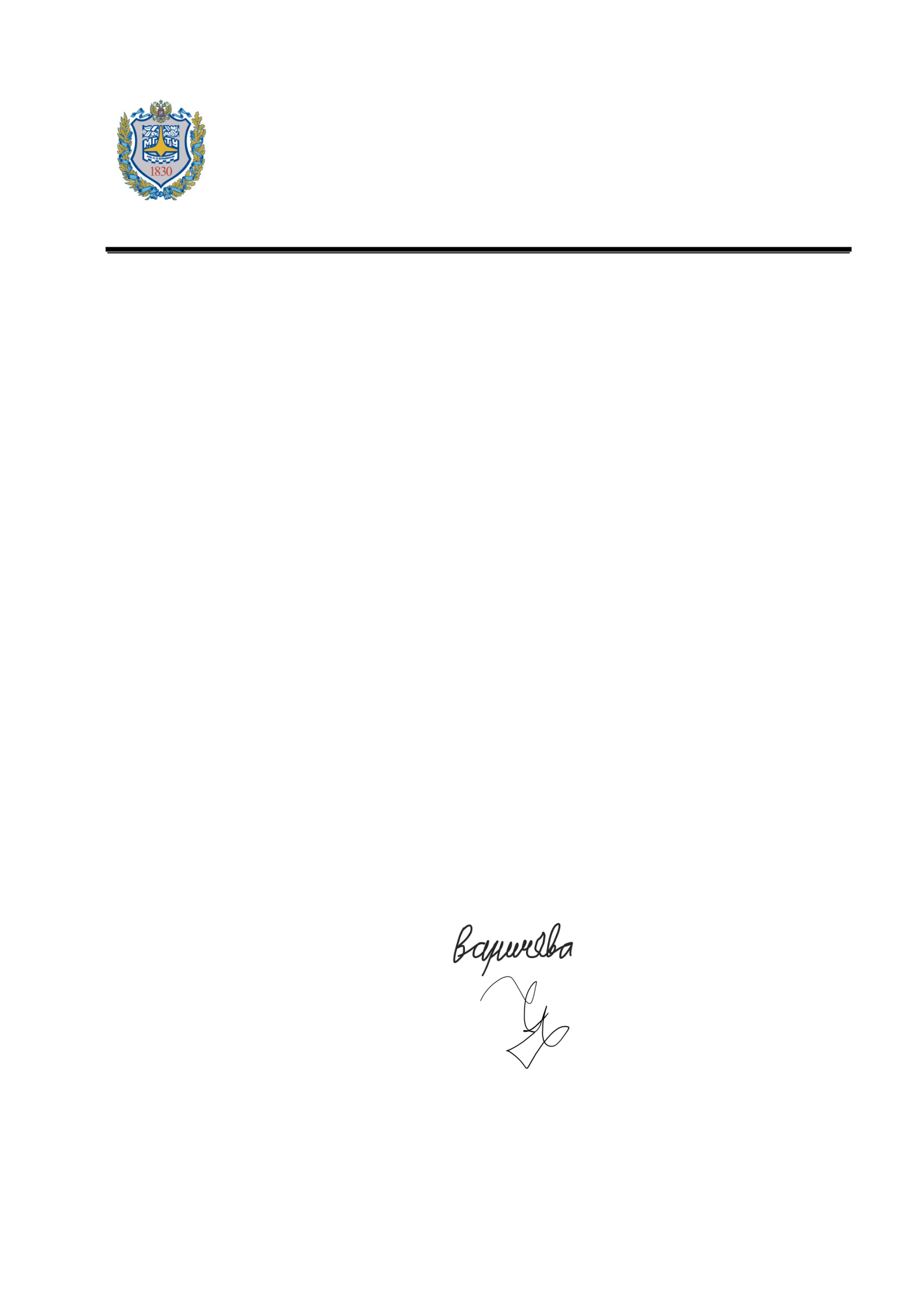
**Министерство науки и высшего образования Российской Федерации Федеральное государственное бюджетное образовательное учреждение высшего образования**

**«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 **09.04.01 Информатика и вычислительная техника**

**ОТЧЕТ**

**по лабораторной работе № 6**

**Название:** Основы Back-End разработки на Golang

**Дисциплина:** Языки интернет-программирования

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Студент | ИУ6-33Б | 19.10.2024 | |  | Д. А. Варичева |
|  | (Группа) |  | (Подпись, дата) |  | (И.О. Фамилия) |
| Преподаватель |  | 19.10.2024 | |  | В. Д. Шульман |
|  |  |  | (Подпись, дата) |  | (И.О. Фамилия) |

Москва, 2024

**Цель работы:**

Изучить основы сетевого взаимодействия и серверной разработки с использованием языка Golang.

**Ход работы:**

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

**Реализация на языке Go**

package main

import (

"fmt"

"net/http"

)

func handler(w http.ResponseWriter, r \*http.Request) { w.Write([]byte("Hello, web!"))

}

func main() {

http.HandleFunc("/get", handler)

err := http.ListenAndServe(":8080", nil)

if err != nil {

fmt.Print("error: server does not start")

}

}

**Тестирование**

Отправка запроса GET:

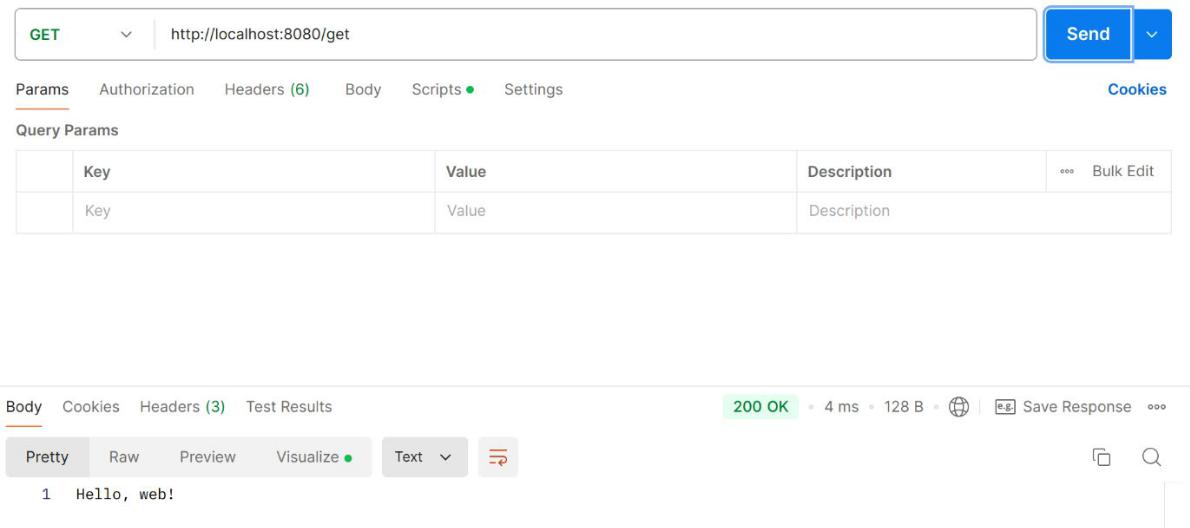


Рисунок 1

***Условие задания 2:*** Напишите веб-сервер который по пути /api/user

приветствует пользователя: принимает и парсит параметр name и делает

ответ "Hello,<name>!"

**Реализация на языке Go**

package main

import (

"fmt"

"net/http"

)

func handler(w http.ResponseWriter, r \*http.Request) { str := r.URL.Query().Get("name")

if str != "" {

w.Write([]byte("Привет, " + str + "!"))

} else {

w.Write([]byte("Попробуй ввести своё имя через query-параметр 'name'"))

}

}

func main() {

http.HandleFunc("/api/user", handler)

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

if err != nil {

fmt.Print("error: server does not start")

}

}

**Тестирование**

1. Отправка запроса серверу без указания параметра «name»:

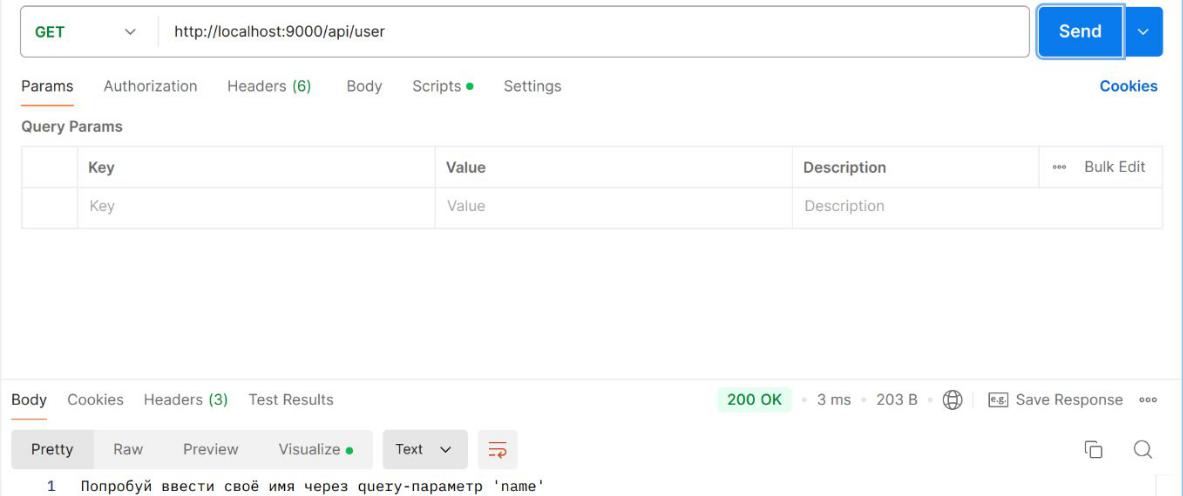


Рисунок 2

1. Отправка запросов серверу с указанием параметра «name» (на рисунках

3и4):

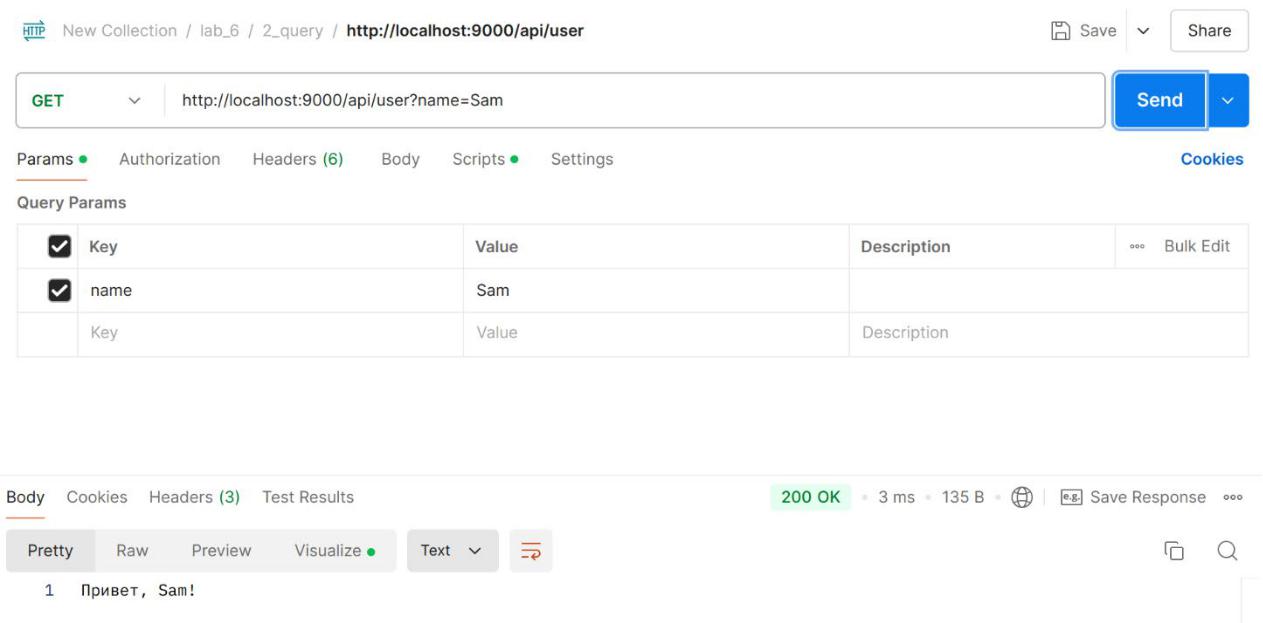


Рисунок 3

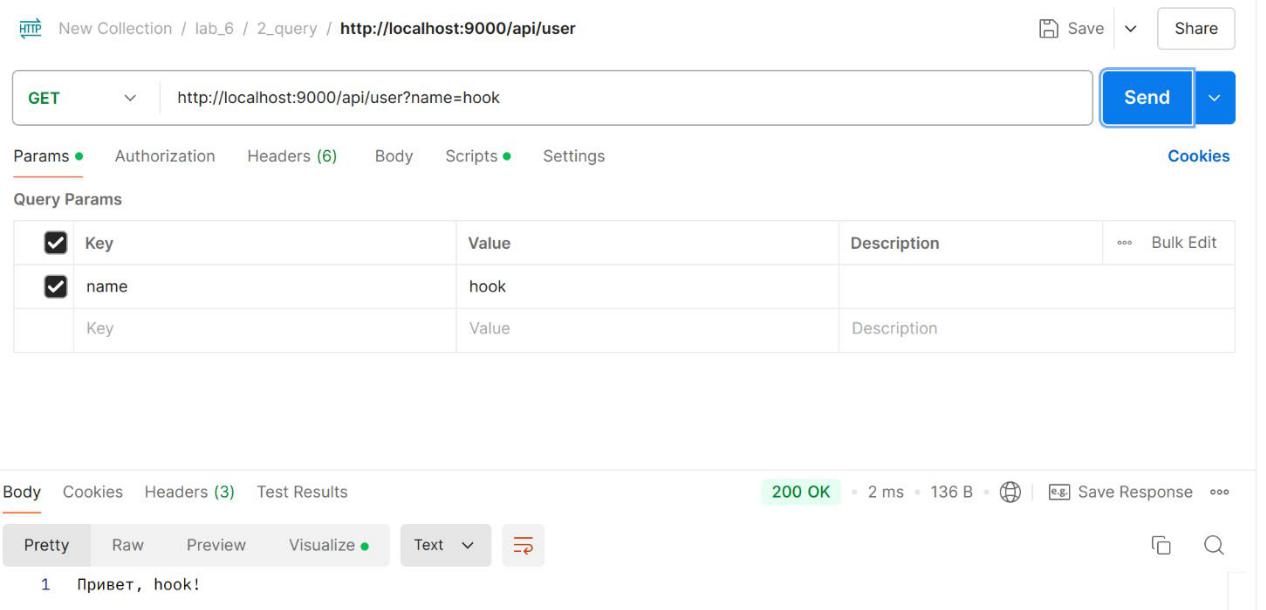


Рисунок 4

***Условие задания 3:*** Напиши веб сервер (**порт :3333**) - счетчик который будет обрабатывать GET (/count) и POST (/count) запросы: **GET**, который возвращает счетчик и **POST**, который увеличивает ваш счетчик на значение (с ключом "count"), получаемое из формы, но если пришло не число, то нужно ответить клиенту: "это не число" со статусом http.StatusBadRequest (400).

**Реализация на языке Go**

package main

import (

"fmt"

"net/http"

"strconv"

)

var counter int = 0

func handler(w http.ResponseWriter, r \*http.Request) { if r.Method == "GET" {

w.Write([]byte("Значение счётчика: " + strconv.Itoa(counter)))

}

if r.Method == "POST" {

r.ParseForm()

adder := r.Form.Get("count")

if val, err := strconv.Atoi(adder); err != nil { w.WriteHeader(http.StatusBadRequest)

w.Write([]byte("ошибка: было введено нечисло\nПопробуйте ввести число для параметра 'count' снова"))

} else {

counter += val

if val > 0 {

w.Write([]byte("Значение счётчика изменено на: +" +

strconv.Itoa(val)))

} else if val == 0 {

w.Write([]byte("Значение счётчика не изменилось, так как параметр 'count' равен 0"))

} else {

w.Write([]byte("Значение счётчика изменено на: " + strconv.Itoa(val)))

}

}

}

}

func main() {

http.HandleFunc("/count", handler)

err := http.ListenAndServe(":3333", nil)

if err != nil {

fmt.Print("error: server does not start")

}

}

**Тестирование**

На рисунках 5- представлены примеры тестов.

1. Отправка запроса GET для получения значения счётчика:

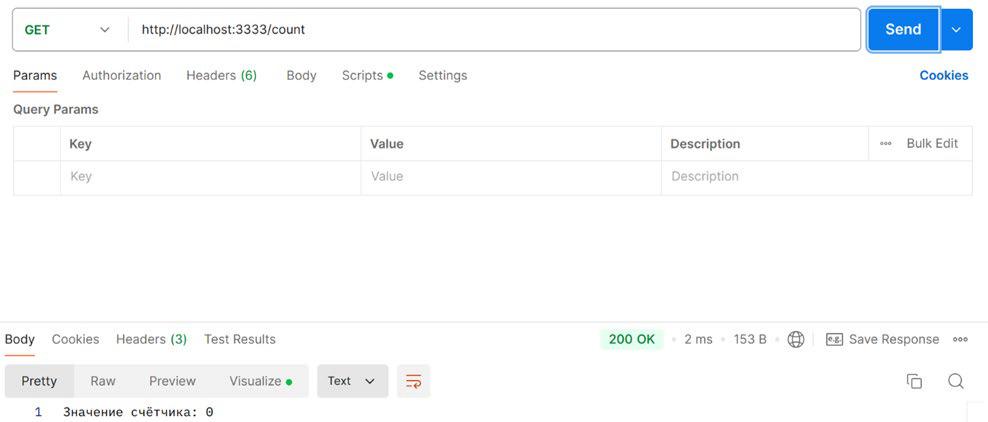


Рисунок 5

1. Отправка запроса POST с указанием параметра «count» = 52 и проверка счётчика с помощью запроса GET:

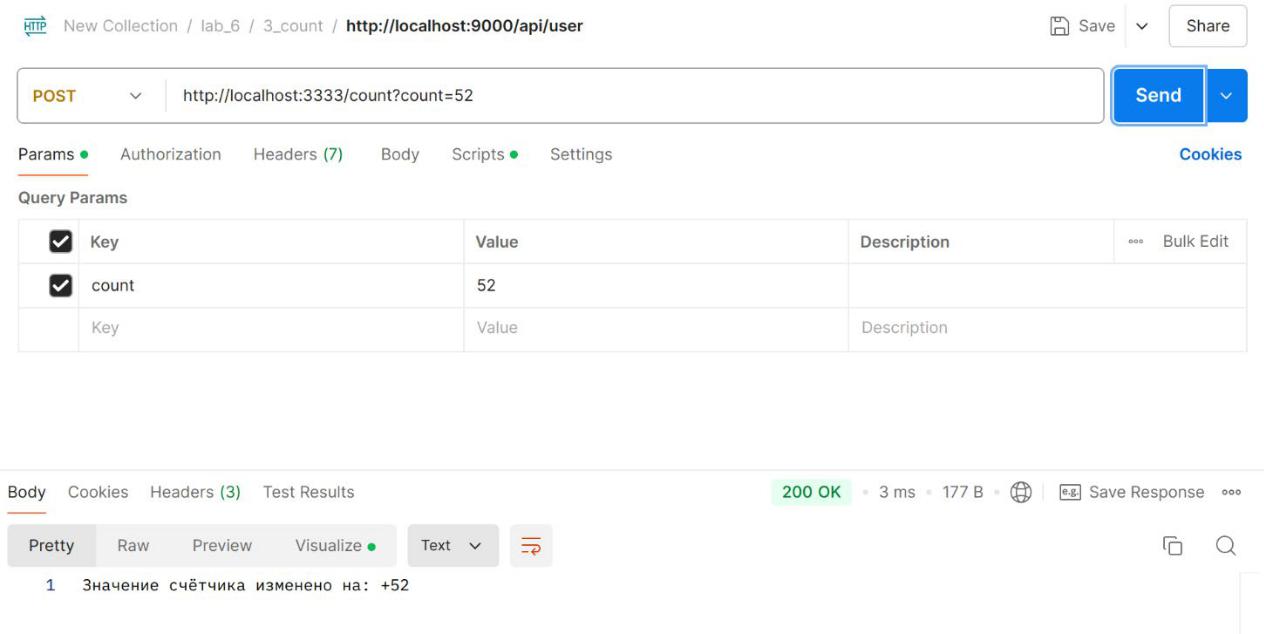


Рисунок 6

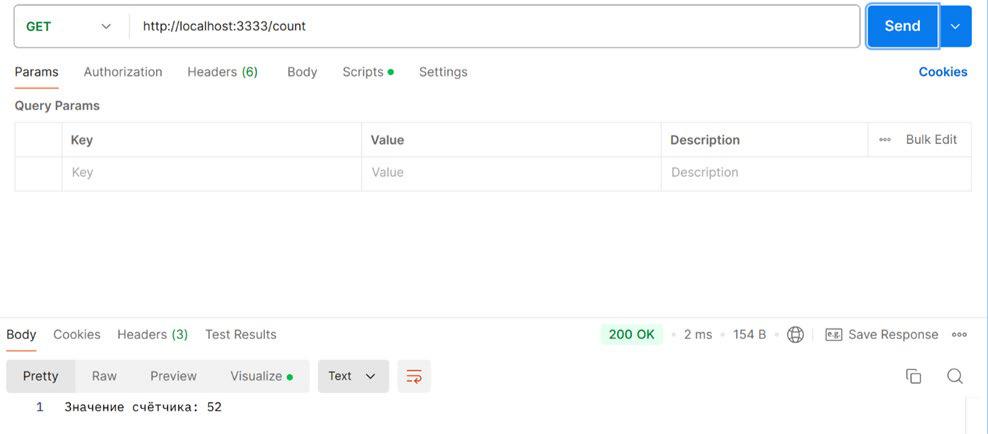


Рисунок 7

1. Отправка запроса POST с указанием параметра «count» = 0:

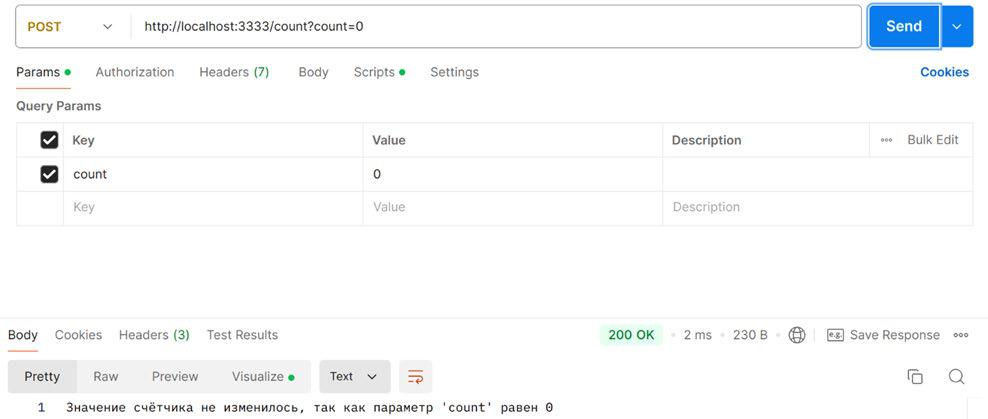


Рисунок 8

1. Отправка запроса POST без указания параметра «count»:

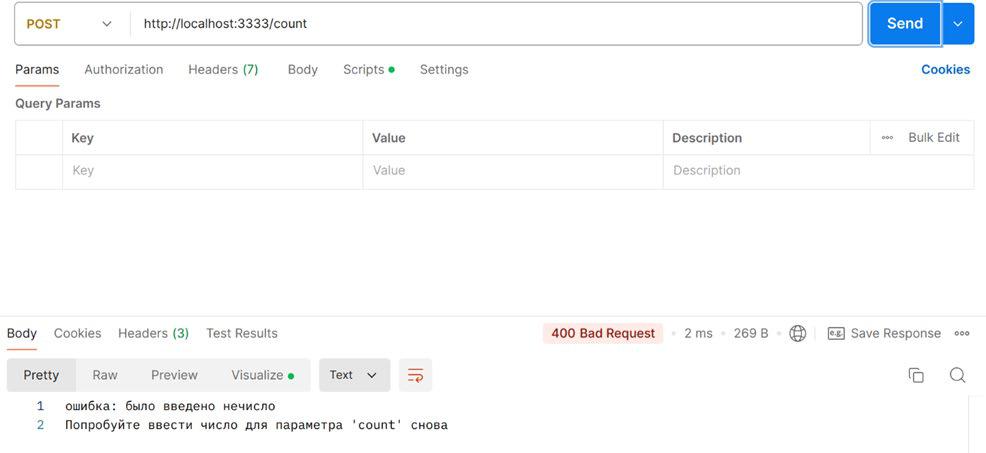


Рисунок 9

**Вывод:**

При выполнении лабораторной работы удалось изучить основы сетевого взаимодействия и серверной разработки с использованием языка Golang, а также разработать три небольших веб-сервера