

НАПРАВЛЕНИЕ ПОДГОТОВКИ **09.03.01 Информатика и вычислительная техника**

Оглавление

Основы Back-End разработки на Golang.....	1
Цель работы.....	3
Ход работы.....	3
Задание 1.....	3
Условие.....	3
Решение.....	3
Задание 2.....	4
Условие.....	4
Решение.....	4
Задание 3.....	5
Условие.....	5
Решение.....	5
Вывод.....	7
Источники информации.....	7

Цель работы

Изучение основ сетевого взаимодействия и серверной разработки с использованием языка Golang.

Ход работы

Задание 1

Условие

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

Решение

```
package main
// некоторые импорты нужны для проверки
import (
    "fmt"
    "net/http"
)

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

func main() {
    http.HandleFunc("/get", handler)

    // Запускаем веб-сервер на порту 8080
    err := http.ListenAndServe(":8080", nil)
    if err != nil {
        fmt.Println("Ошибка запуска сервера:", err)
    }
}
```

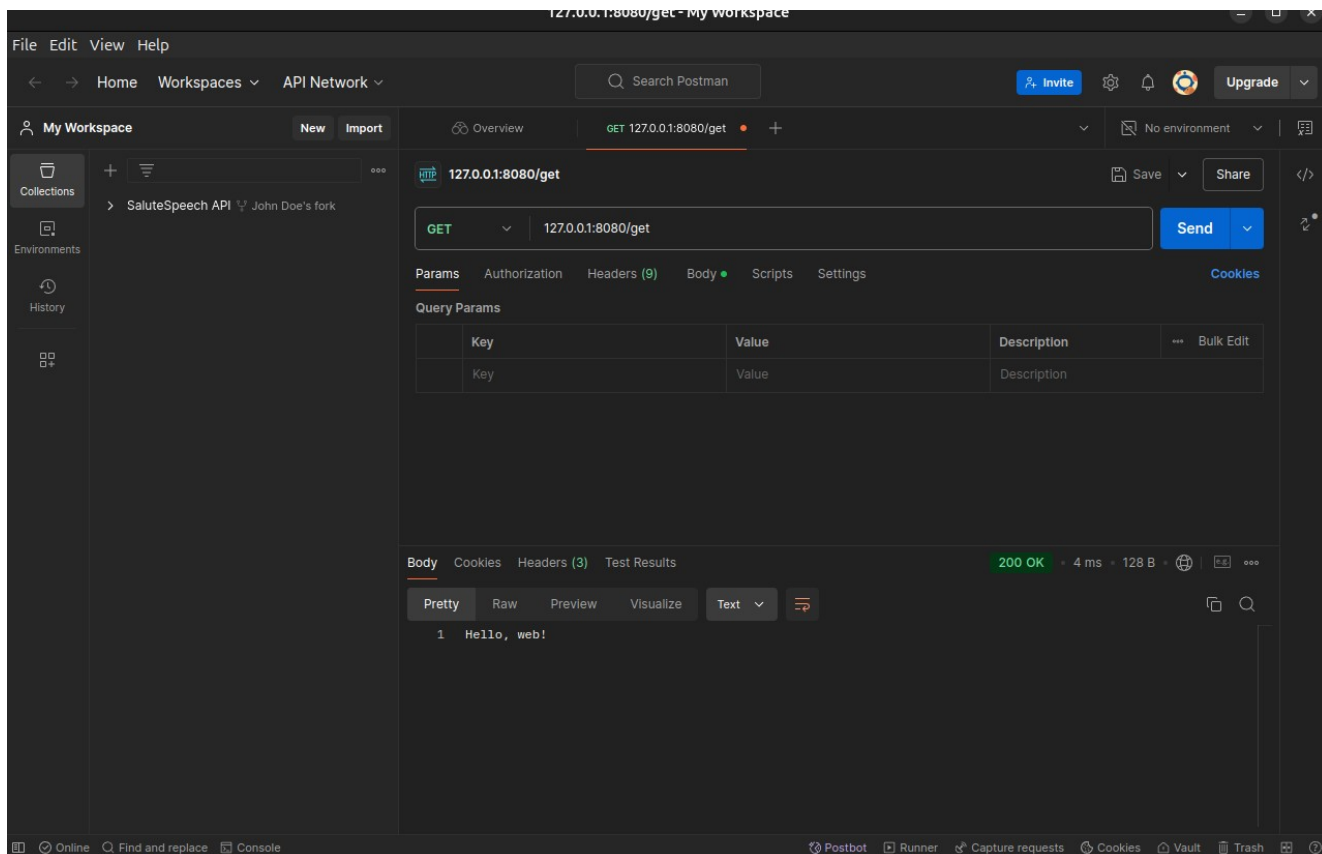
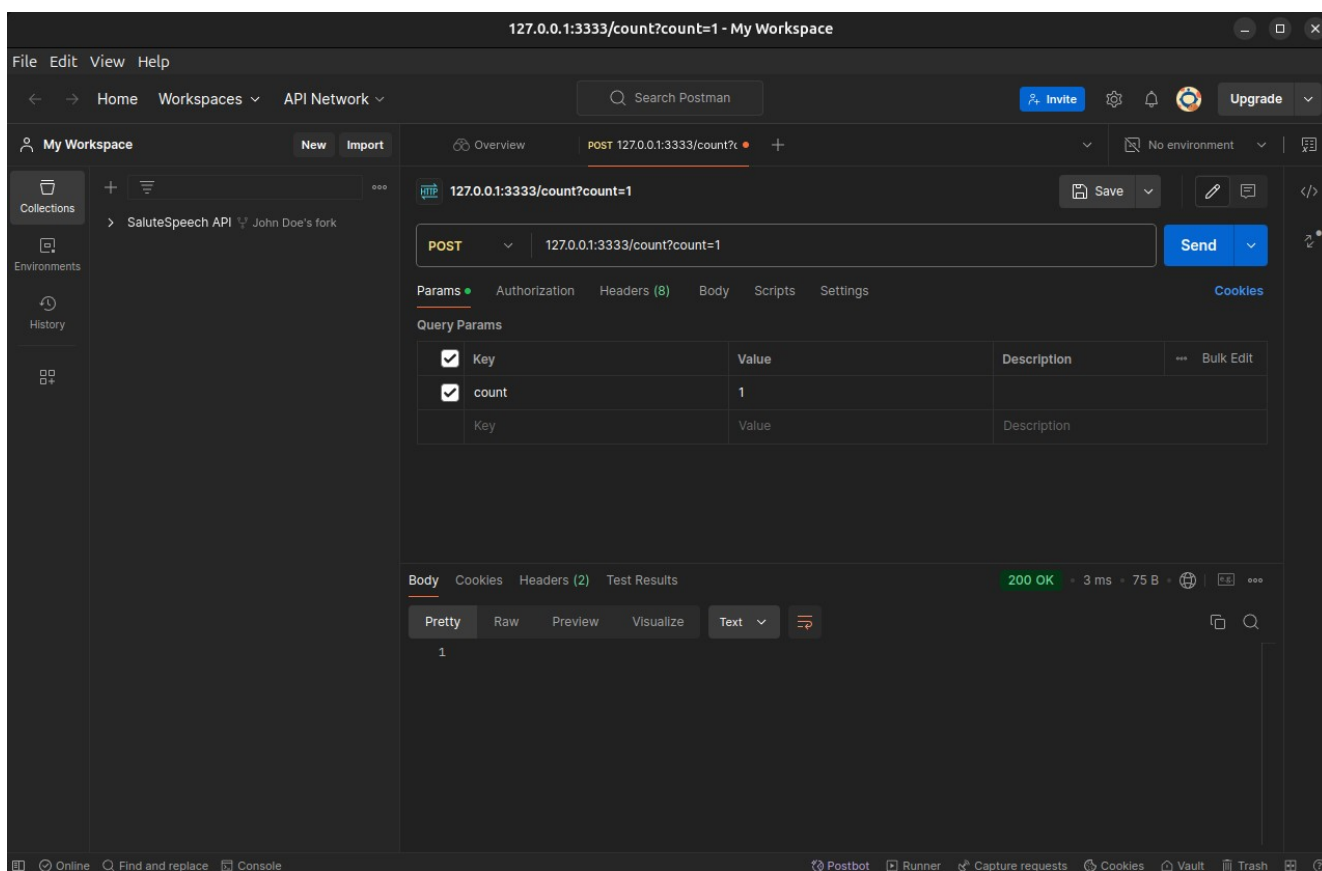


Рисунок 1 - результат отправки запроса



Задание 2

Условие

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

Решение

```
package main
// некоторые импорты нужны для проверки
import (
    "fmt"
    "io"
    "net/http" // пакет для поддержки HTTP протокола
    "os"
    "time"
)

func handler(w http.ResponseWriter, r *http.Request) {
    s := r.URL.Query().Get("name")
    w.Write([]byte("Hello," + s + "!"))
}

func main() {
    http.HandleFunc("/api/user", handler)

    err := http.ListenAndServe(":9000", nil)
    if err != nil {
        fmt.Println("Ошибка запуска сервера:", err)
    }
}
```

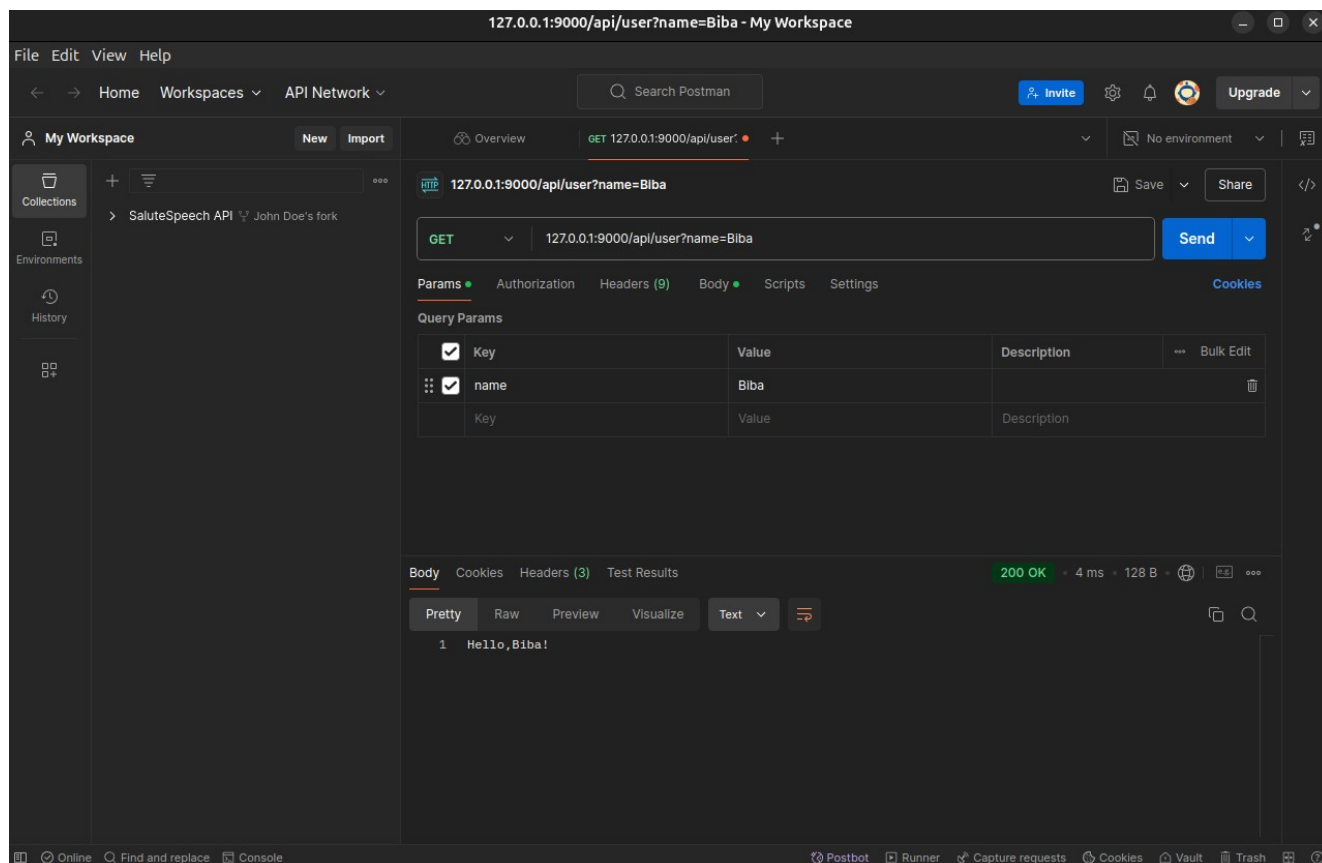


Рисунок 2 - результат отправки запроса

Задание 3

Условие

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).

Решение

```
package main
// некоторые импорты нужны для проверки
import (
    "fmt"
    "io"
    "log"
    "net/http"
    "net/url"
    "os"
    "time"
    "strconv" // вдруг понадобится вам ;)
)

var count1 int = 0

func handler(w http.ResponseWriter, r *http.Request) {
    if r.Method == "GET" {
        w.WriteHeader(http.StatusOK)
        w.Write([]byte(strconv.Itoa(count1)))
        return
    } else if r.Method == "POST" {
        r.ParseForm()
        s := r.FormValue("count")
        if s == "" {
            w.WriteHeader(http.StatusBadRequest)
            w.Write([]byte("это не число"))

            return
        }
        number, err := strconv.Atoi(s)
        if err != nil {
            w.WriteHeader(http.StatusBadRequest)
            w.Write([]byte("это не число"))
        }
    }
}
```

```

    return
}
count1 += number
return
} else {
    w.WriteHeader(http.StatusMethodNotAllowed)
    w.Write([]byte("Метод не поддерживается"))
    return
}
}

func main() {
    http.HandleFunc("/count", handler)

    err := http.ListenAndServe(":3333", nil)
    if err != nil {
        fmt.Println("Ошибка запуска сервера!")
    }
}

```

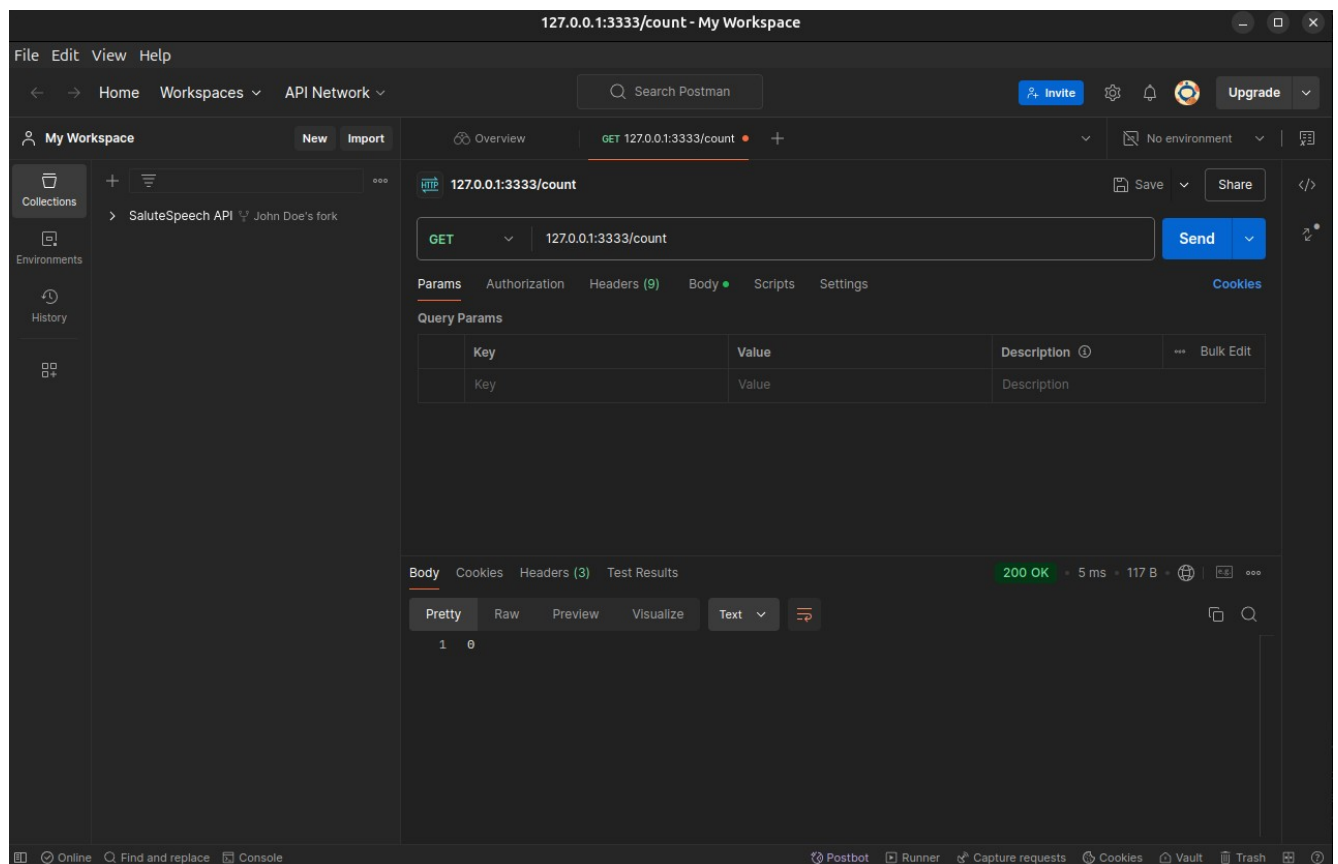


Рисунок 3 - результат отправки первого Get запроса

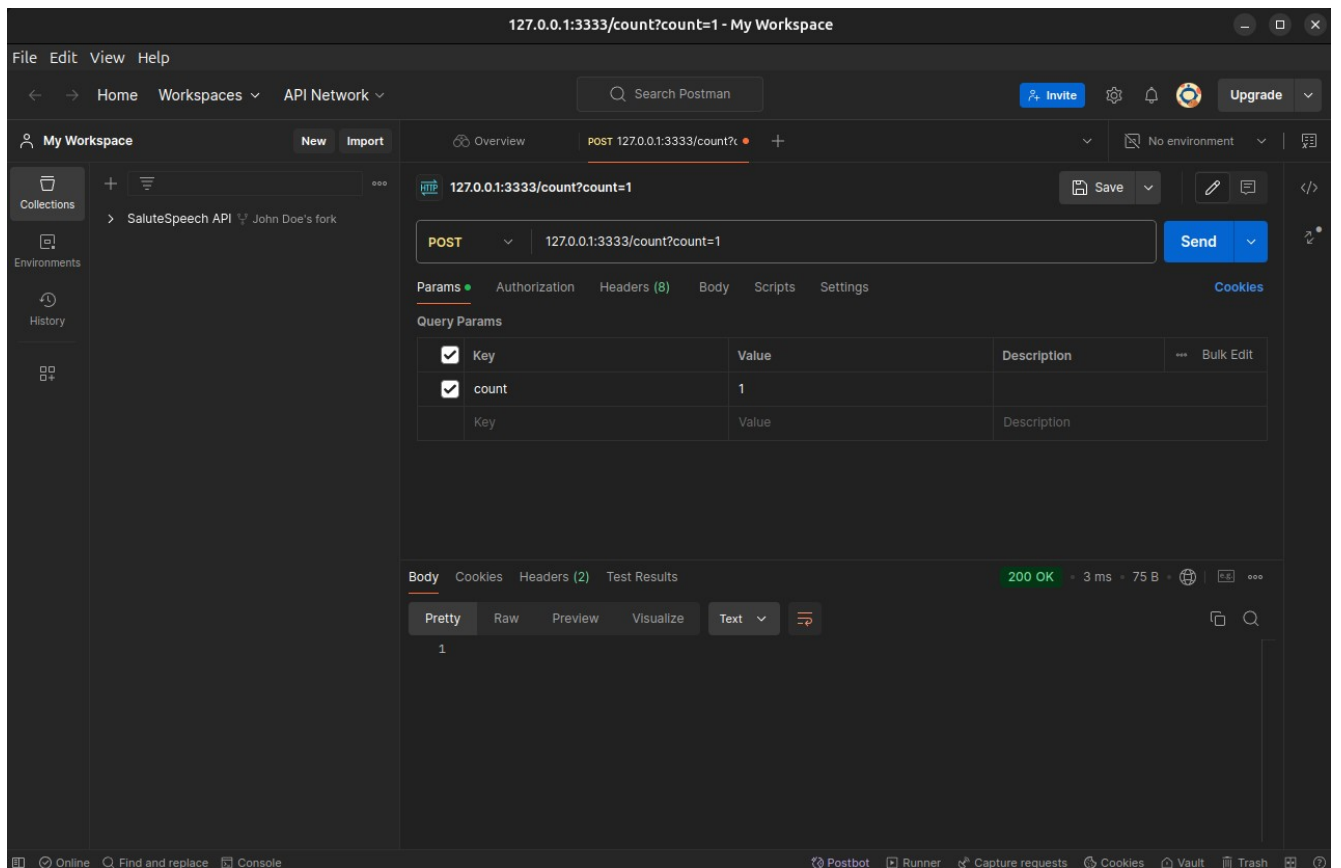


Рисунок 4 - отправка POST запроса с 1 значением

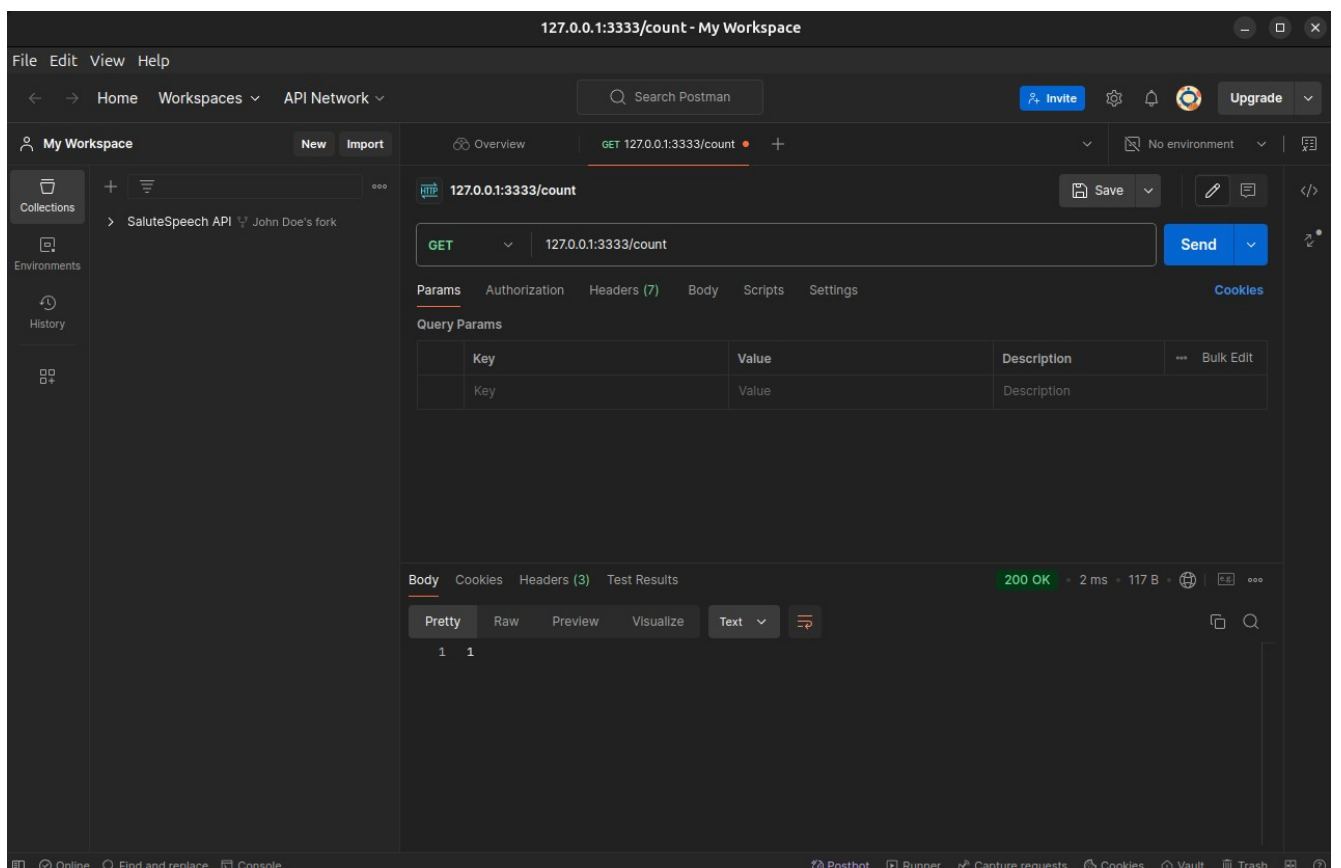


Рисунок 5 - Проверка записи значения

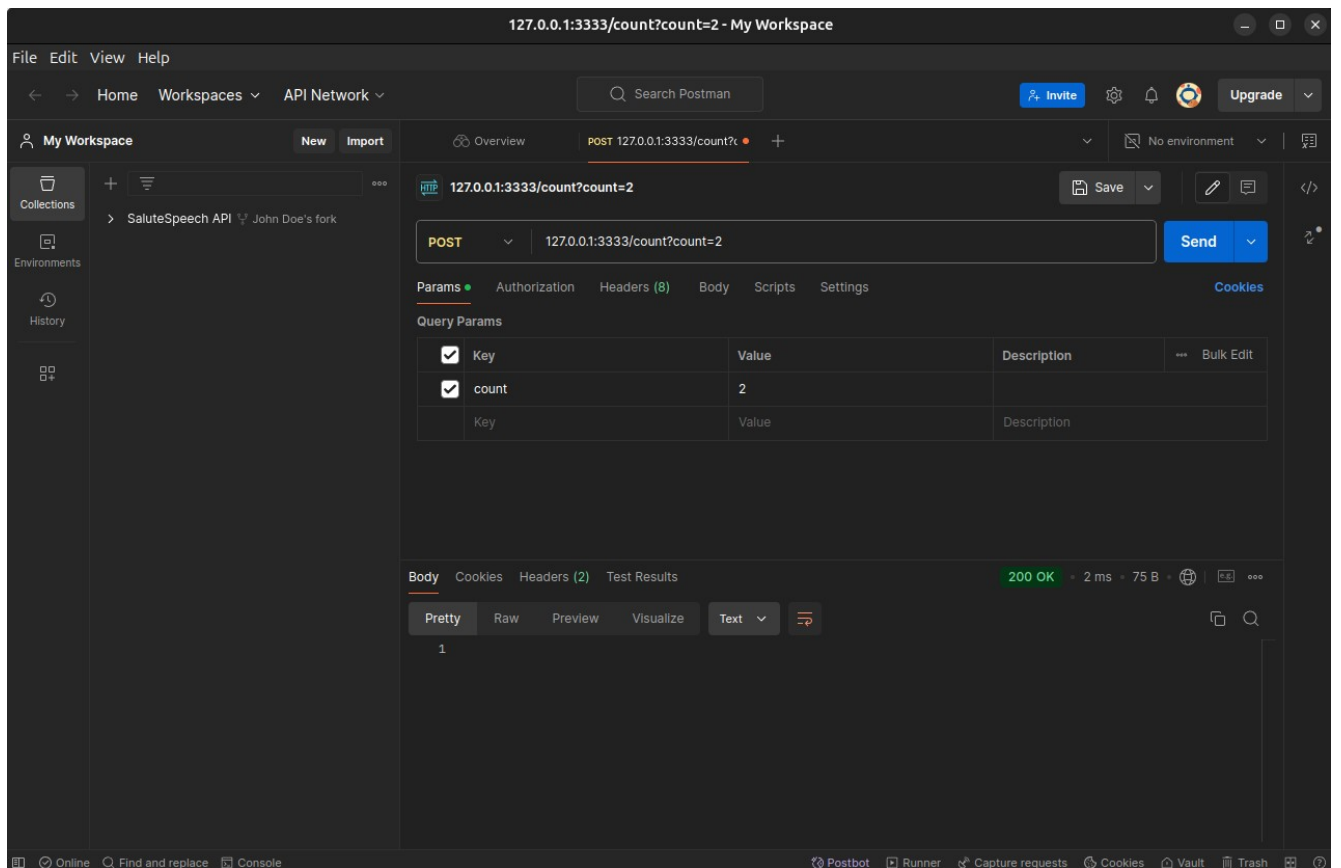


Рисунок 6 - запись второго значения

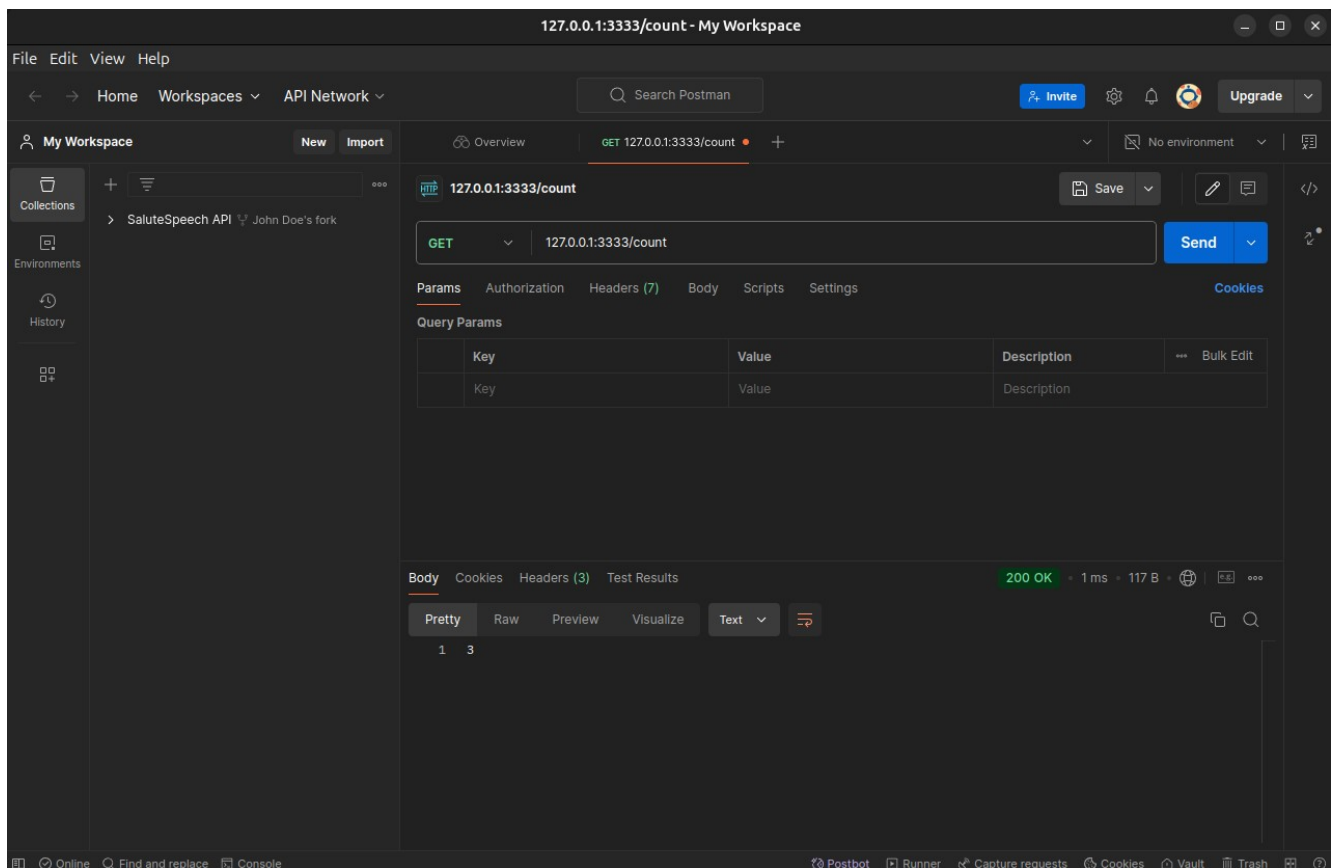


Рисунок 7 - проверка корректности записи значения

Вывод

При выполнении заданий лабораторной работы мы познакомились с основами Back-End разработки на Golang: решили несколько задач, проверили работоспособность программ.

Источники информации

- [Курс Golang на Stepik](#) — источник информации и условий задач