



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 № 9**

**Название:** Golang & Echo

**Дисциплина:** Языки интернет программирования

Студент

ИУ6-32Б

(Группа)

30.11.2024

(Подпись, дата)

Заушников Л.И.

(И.О. Фамилия)

Преподаватель

30.11.2024

(Подпись, дата)

В.Д. Шульман

(И.О. Фамилия)

Москва, 2024

**Цель работы** — получить первичных навыков использования веб-фреймворков в Backend-разработке на Golang.

1. Было произведено ознакомление с материалами для подготовки перед выполнением лабораторной работы
2. Был сделан форк репозитория в GitHub (рисунок 1), копия была клонирована локально, была создана от мастера ветка dev и было произведено переключение на неё.

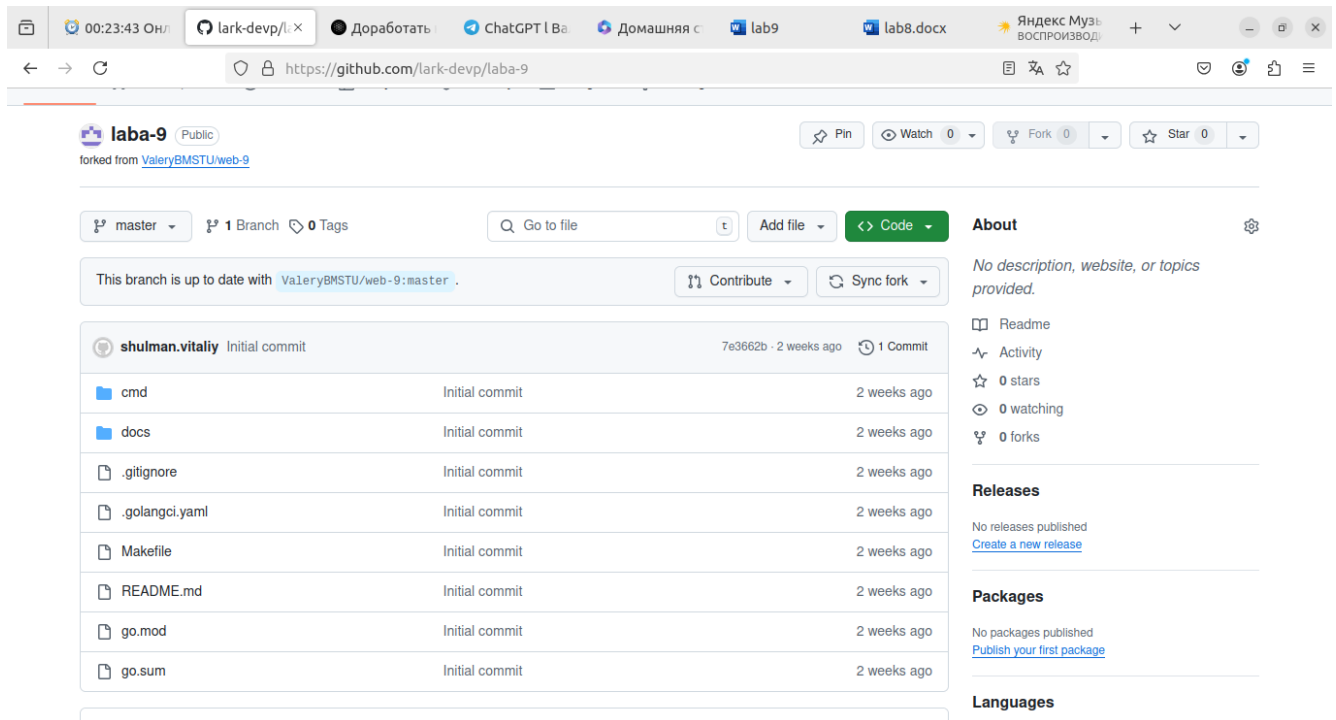


Рисунок 1 - Форкнутый репозиторий

Далее были доработаны и протестированы 3 сервиса, использовался фреймворк Echo, все коды были проверены линтами.

**Сервис hello**  
package main

```
import (  
    "log"  
    "net/http"
```

```
    "github.com/labstack/echo/v4"  
)
```

```
// Структура для приема JSON данных  
type Response struct {  
    Message string `json:"message"`  
}
```

```
func helloHandler(c echo.Context) error {  
    response := Response{Message: "Hello, web!"}
```

```

return c.JSON(http.StatusOK, response)
}

func main() {
// Создание нового экземпляра Echo
e := echo.New()

// Логирование
e.Logger.SetHeader("${time_rfc3339}    ${remote_ip}    ${method}    ${url}
${status}")

// Роутинг
e.GET("/get", helloHandler)

// Обработка ошибок
e.HTTPErrorHandler = func(err error, c echo.Context) {
code := http.StatusInternalServerError
if he, ok := err.(*echo.HTTPError); ok {
code = he.Code
}

c.JSON(code, map[string]interface{}{
"error": err.Error(),
})
}

// Запуск сервера
if err := e.Start(":8080"); err != nil {
log.Fatalf("Ошибка запуска сервера: %v", err)
}
}

```

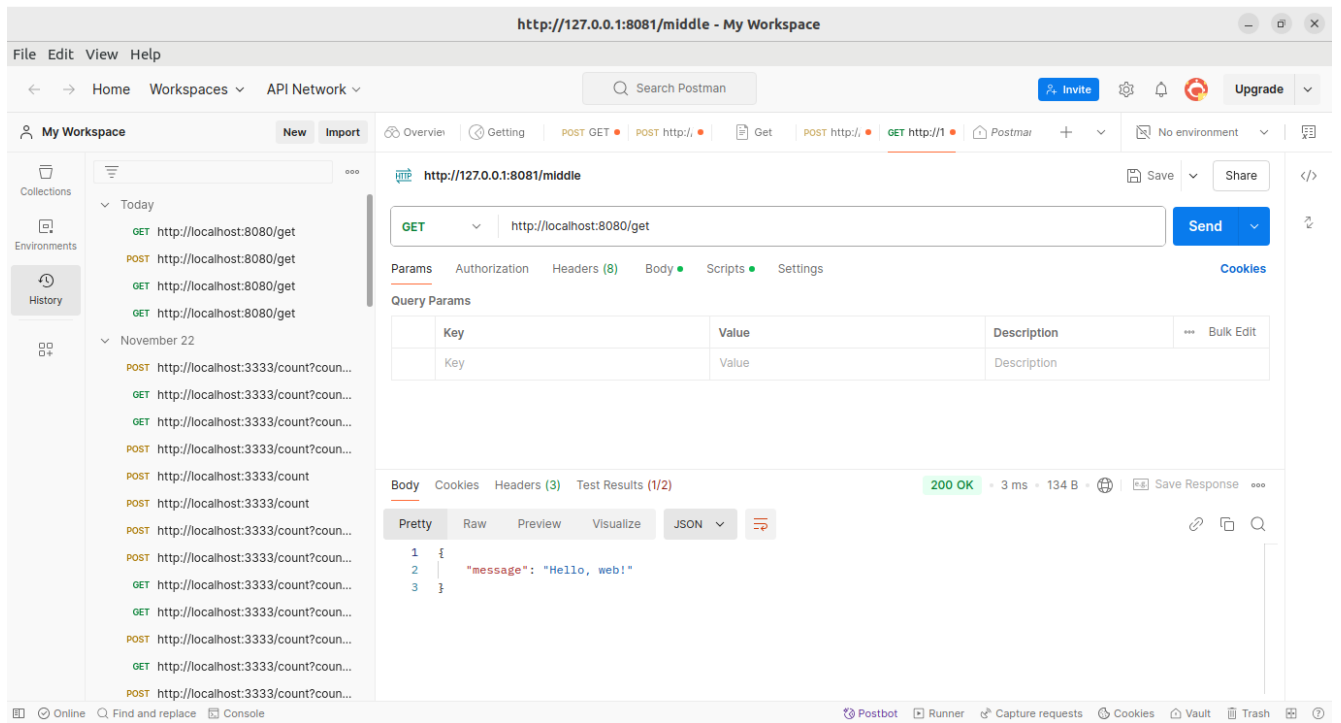


Рисунок 2 - Тестирование сервиса hello

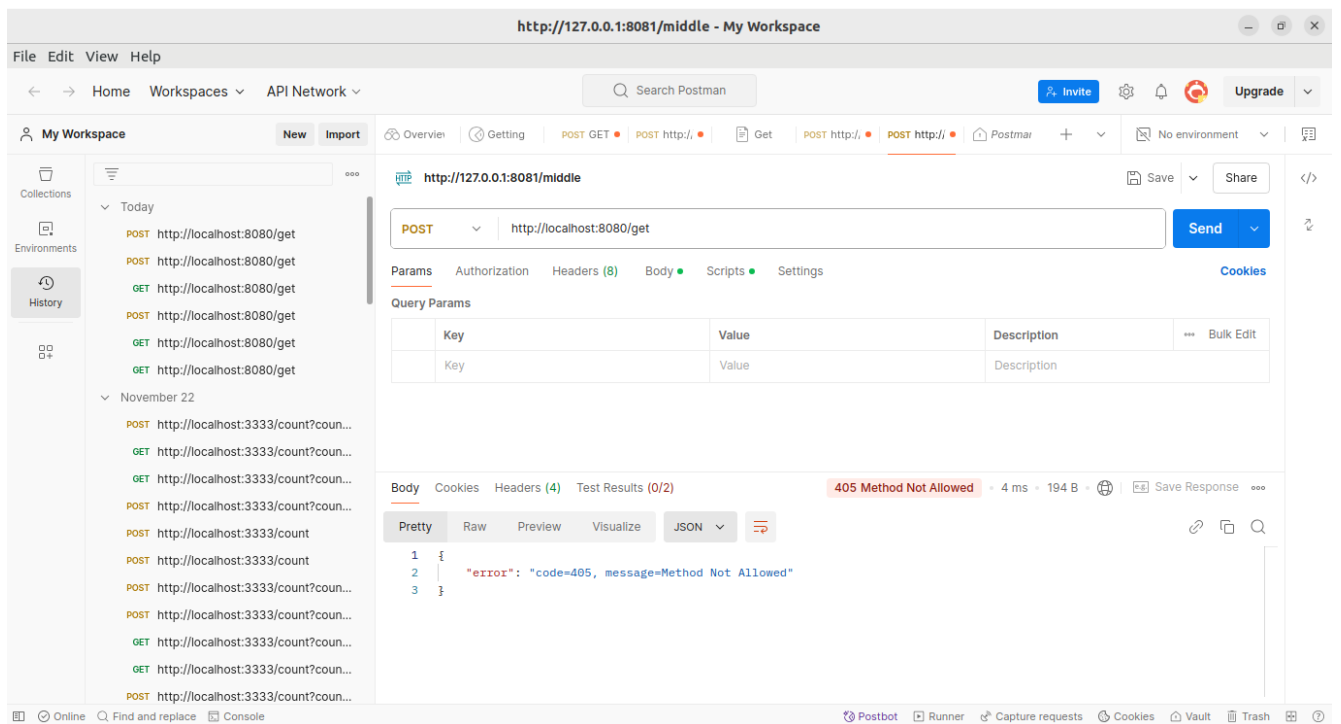
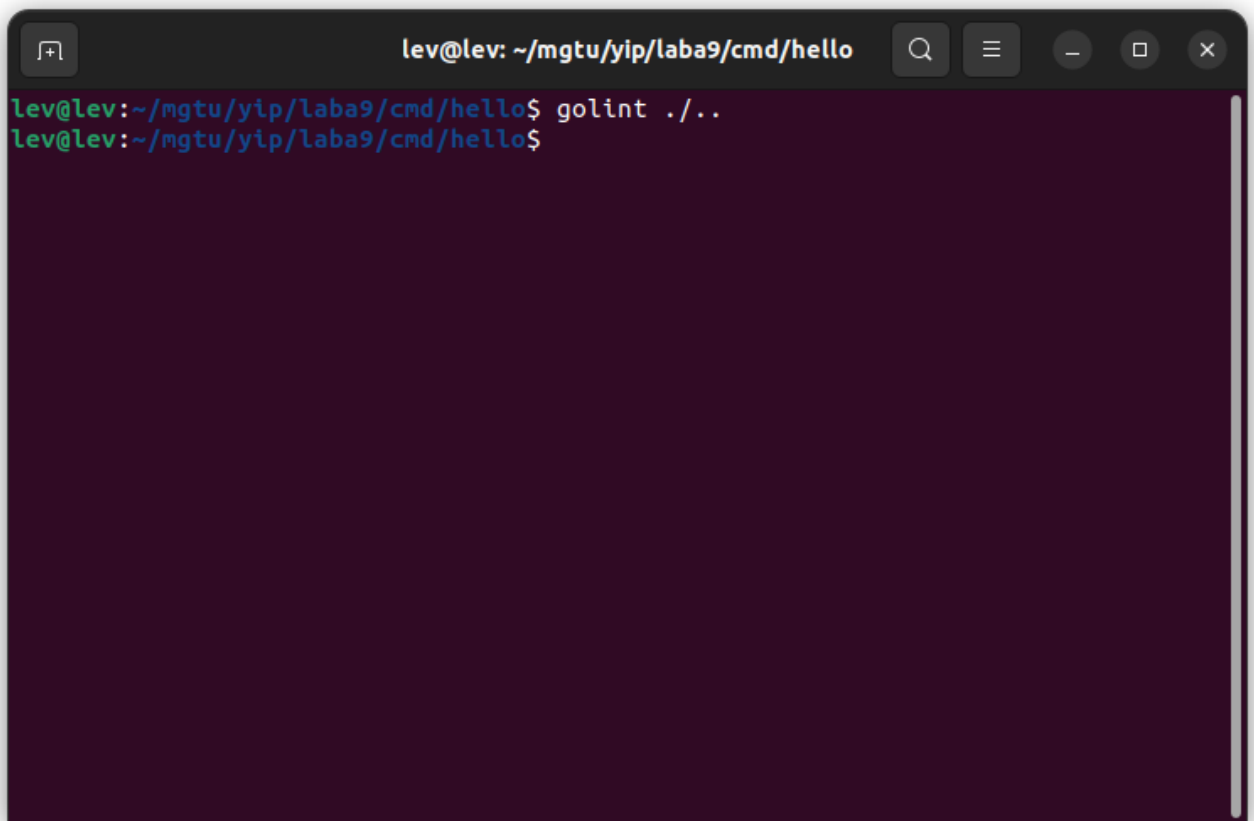


Рисунок 3 - Тестирование сервиса hello

A terminal window with a dark background. The title bar shows the user 'lev' at host 'lev' in the directory '~/mgtu/yip/laba9/cmd/hello'. The terminal shows two lines of text: 'lev@lev:~/mgtu/yip/laba9/cmd/hello\$ golint ./..' followed by a new prompt 'lev@lev:~/mgtu/yip/laba9/cmd/hello\$'.

```
lev@lev: ~/mgtu/yip/laba9/cmd/hello
lev@lev:~/mgtu/yip/laba9/cmd/hello$ golint ./..
lev@lev:~/mgtu/yip/laba9/cmd/hello$
```

Рисунок 4 - Проверка кода сервиса hello линтами

## Сервис query

```
package main

import (
    "net/http"

    "github.com/labstack/echo/v4"
)

// Структура для JSON-ответа
type Response struct {
    Message string `json:"message"`
}

func handler(c echo.Context) error {
    name := c.QueryParam("name")

    // Проверка, было ли передано имя
    if name == "" {
        return c.JSON(http.StatusBadRequest, Response{Message: "Пожалуйста, введите ваше имя с помощью параметра 'name'."})
    }
}
```

```

response := Response{Message: "Hello, " + name + "!"}
return c.JSON(http.StatusOK, response)
}

func main() {
// Создание нового экземпляра Echo
e := echo.New()

// Логирование
e.Logger.SetHeader("${time_rfc3339}    ${remote_ip}    ${method}    ${url}
${status}")

// Роутинг
e.GET("/api/user", handler)

// Обработка ошибок
e.HTTPErrorHandler = func(err error, c echo.Context) {
code := http.StatusInternalServerError
if he, ok := err.(*echo.HTTPError); ok {
code = he.Code
}

c.JSON(code, map[string]interface{}{
"error": err.Error(),
})
}

// Запуск сервера
if err := e.Start(":9000"); err != nil {
e.Logger.Fatalf("Ошибка запуска сервера: %v", err)
}
}

```

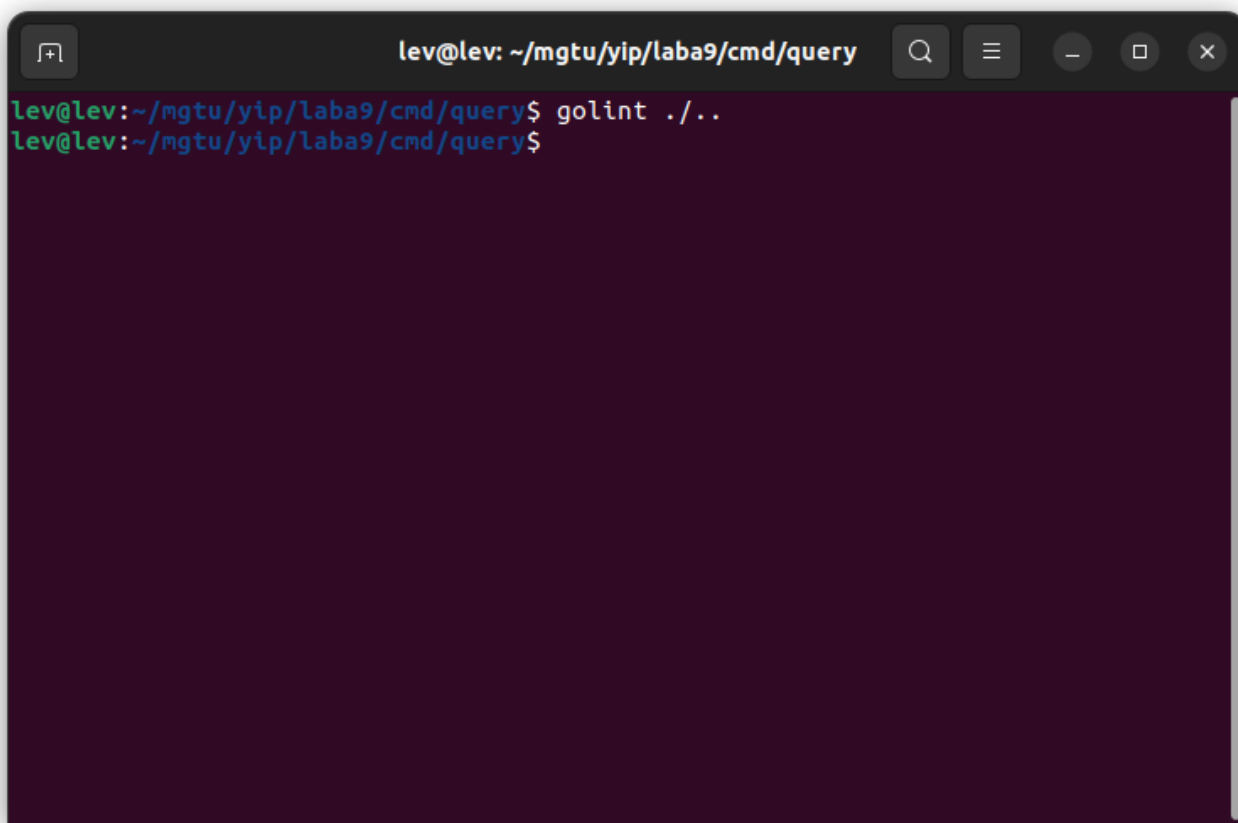


Рисунок 5 - Проверка кода сервиса query линтами

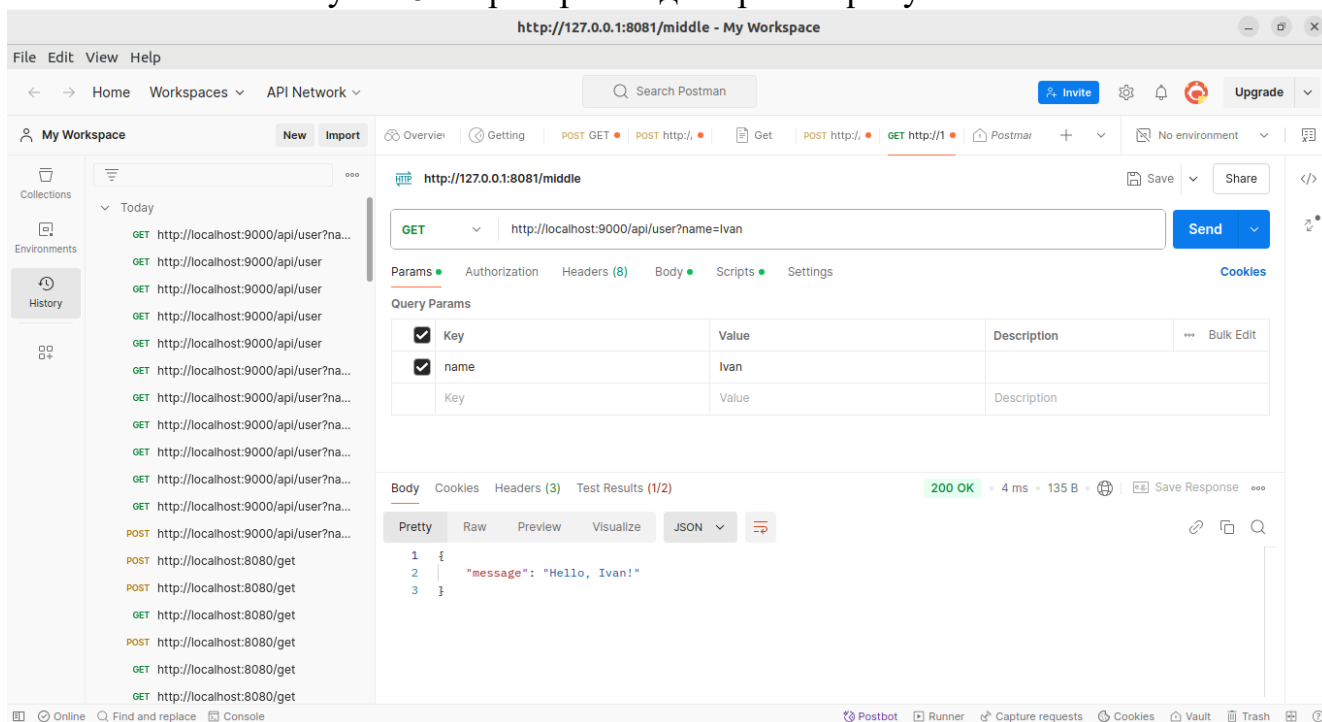


Рисунок 6 - Тестирование сервиса query

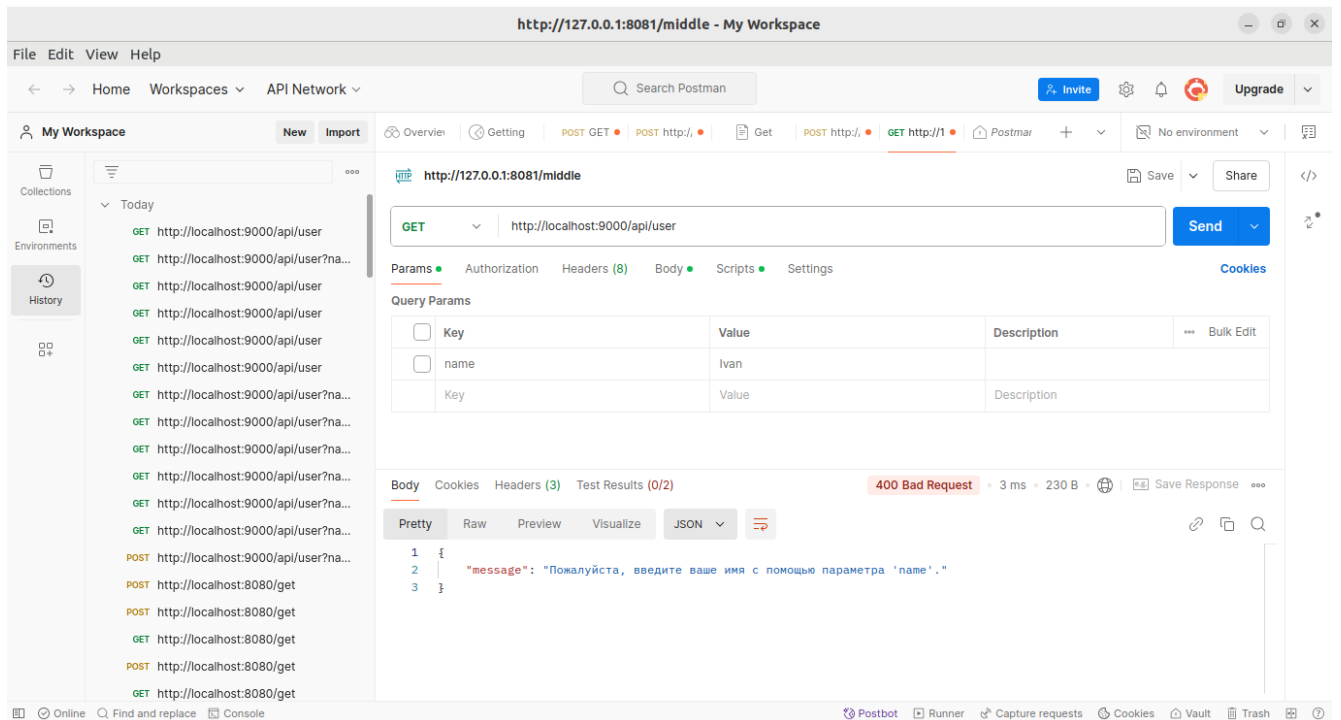


Рисунок 7 - Тестирование сервиса query

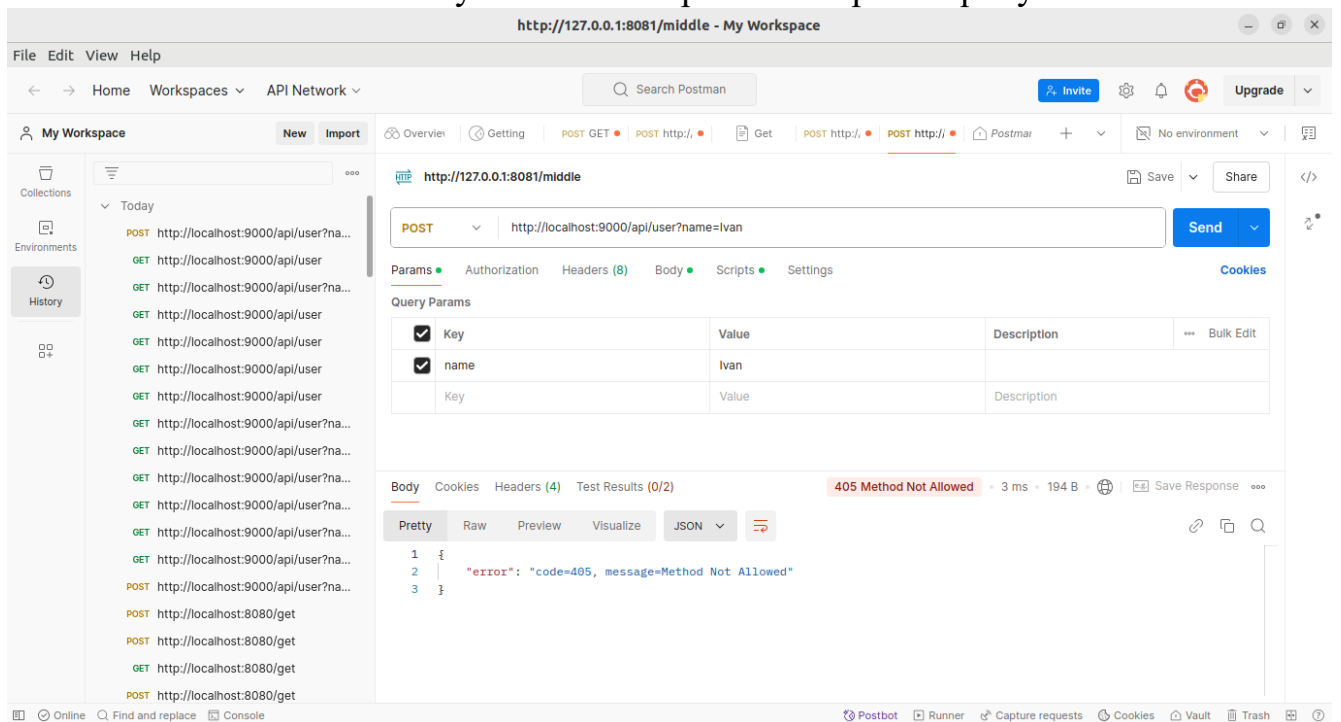


Рисунок 8 - Тестирование сервиса query

## Сервис count

package main

```
import (
    "net/http"
```

```
"github.com/labstack/echo/v4"
```

```
"github.com/labstack/echo/v4/middleware"
```



)

```
var counter int = 0
```

```
// Структура для JSON-ответов
```

```
type CountResponse struct {  
    Count int `json:"count"`  
}
```

```
type CountRequest struct {  
    Count int `json:"count"`  
}
```

```
func countHandler(c echo.Context) error {  
    switch c.Request().Method {  
    case http.MethodGet:  
        return c.JSON(http.StatusOK, CountResponse{Count: counter})  
    case http.MethodPost:  
        var req CountRequest  
        if err := c.Bind(&req); err != nil { //Это метод, который принимает указатель  
на переменную (обычно структуру), в которую будут помещены данные, и  
пытается заполнить ее данными из запроса.  
            return c.JSON(http.StatusBadRequest, map[string]string{"error": "это не  
число"})  
        }  
        counter += req.Count  
        return c.JSON(http.StatusOK, map[string]string{"message": "Значение счётчика  
обновлено"})  
    default:  
        return c.JSON(http.StatusMethodNotAllowed, map[string]string{"error": "Метод  
не поддерживается"})  
    }  
}
```

```
func main() {  
    e := echo.New()
```

```
// Добавление middleware для логирования
```

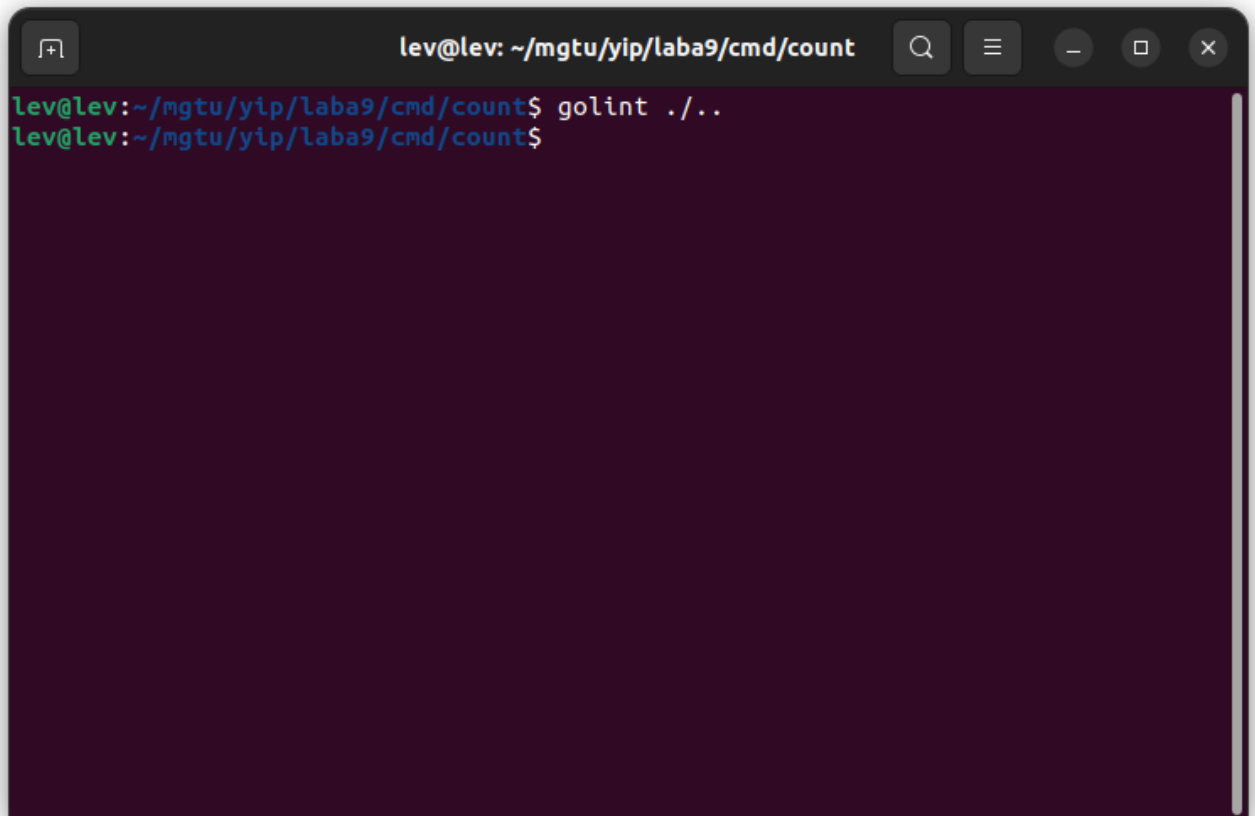
```
e.Use(middleware.Logger())  
e.Use(middleware.Recover())
```

```
// Определение маршрута
```

```
e.GET("/count", countHandler)  
e.POST("/count", countHandler)
```

```
// Запуск сервера
```

```
if err := e.Start(":3333"); err != nil {  
    e.Logger.Fatal("Ошибка запуска сервера:", err)  
}  
}
```

A terminal window with a dark background and light-colored text. The window title bar shows the user 'lev' at host 'lev' in the directory '~/mgtu/yip/laba9/cmd/count'. The terminal shows two lines of text: a command 'golint ../..' and the resulting prompt, indicating the command was executed successfully without output.

```
lev@lev: ~/mgtu/yip/laba9/cmd/count  
lev@lev:~/mgtu/yip/laba9/cmd/count$ golint ../..  
lev@lev:~/mgtu/yip/laba9/cmd/count$
```

Рисунок 9 - Проверка кода сервиса count линтами

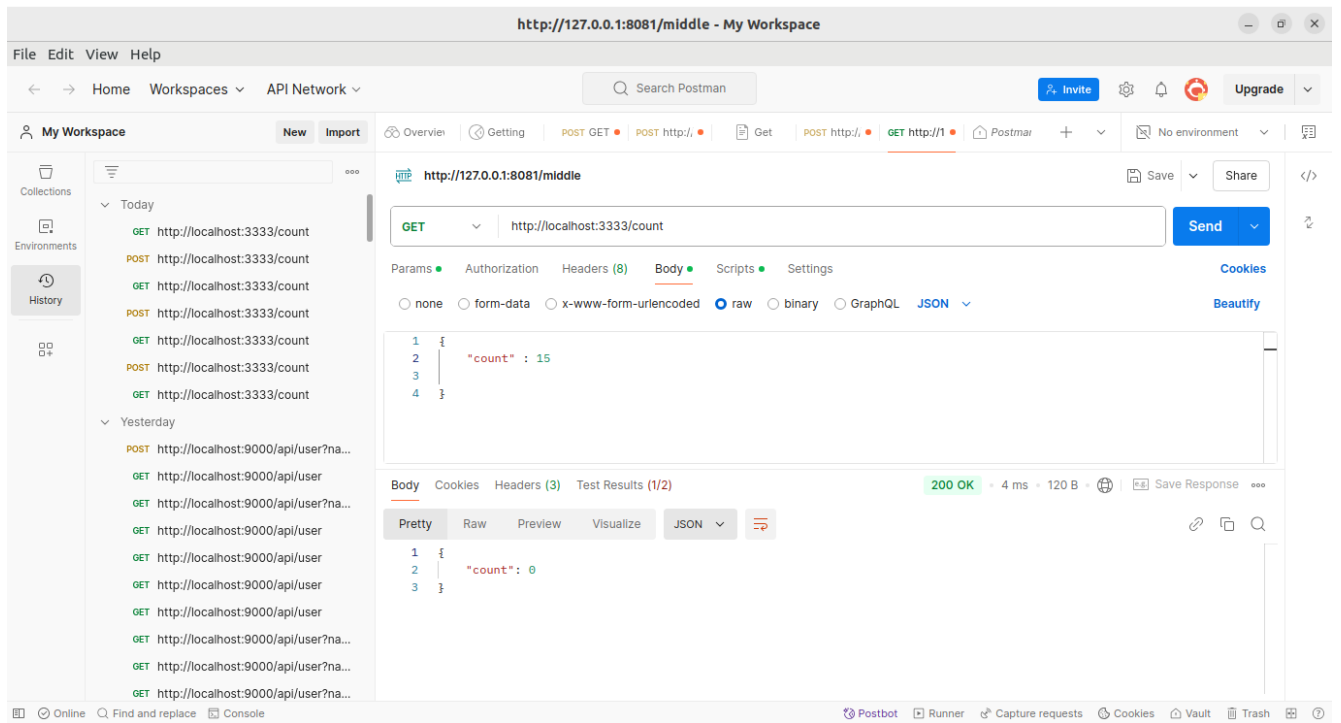


Рисунок 10 - Тестирование сервиса count

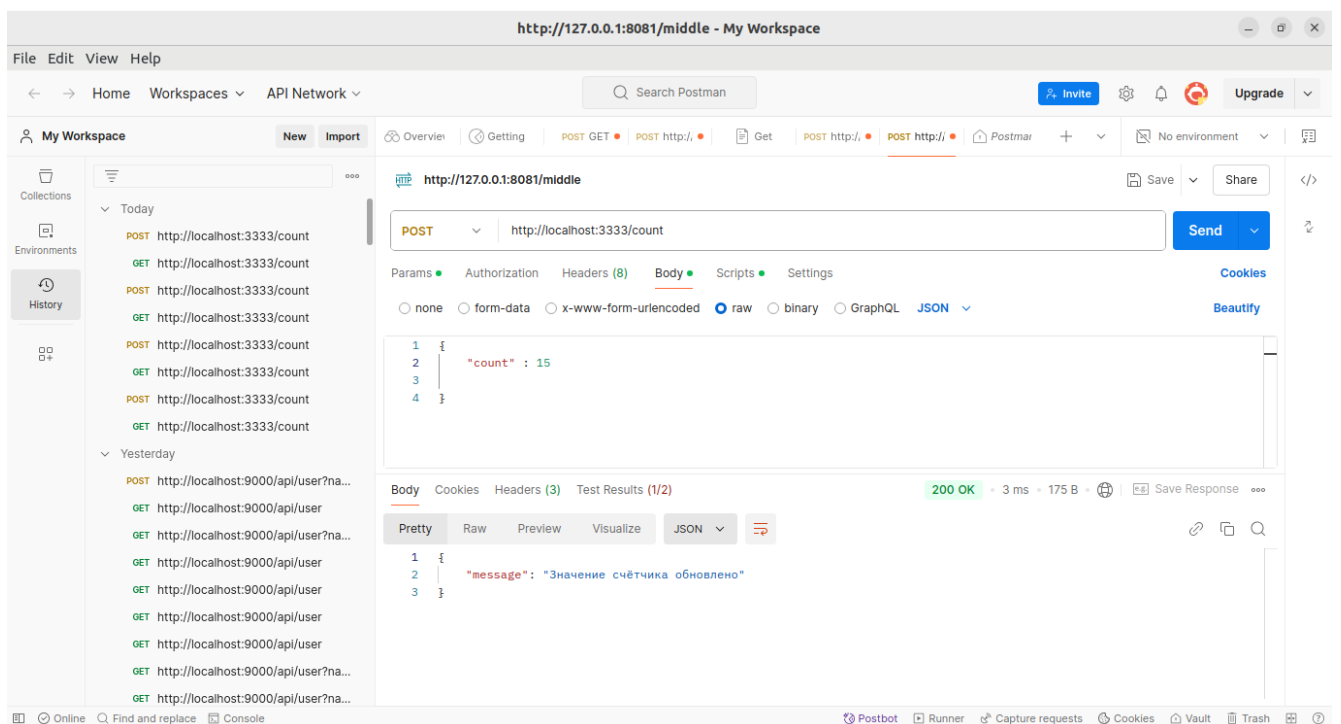


Рисунок 11 - Тестирование сервиса count

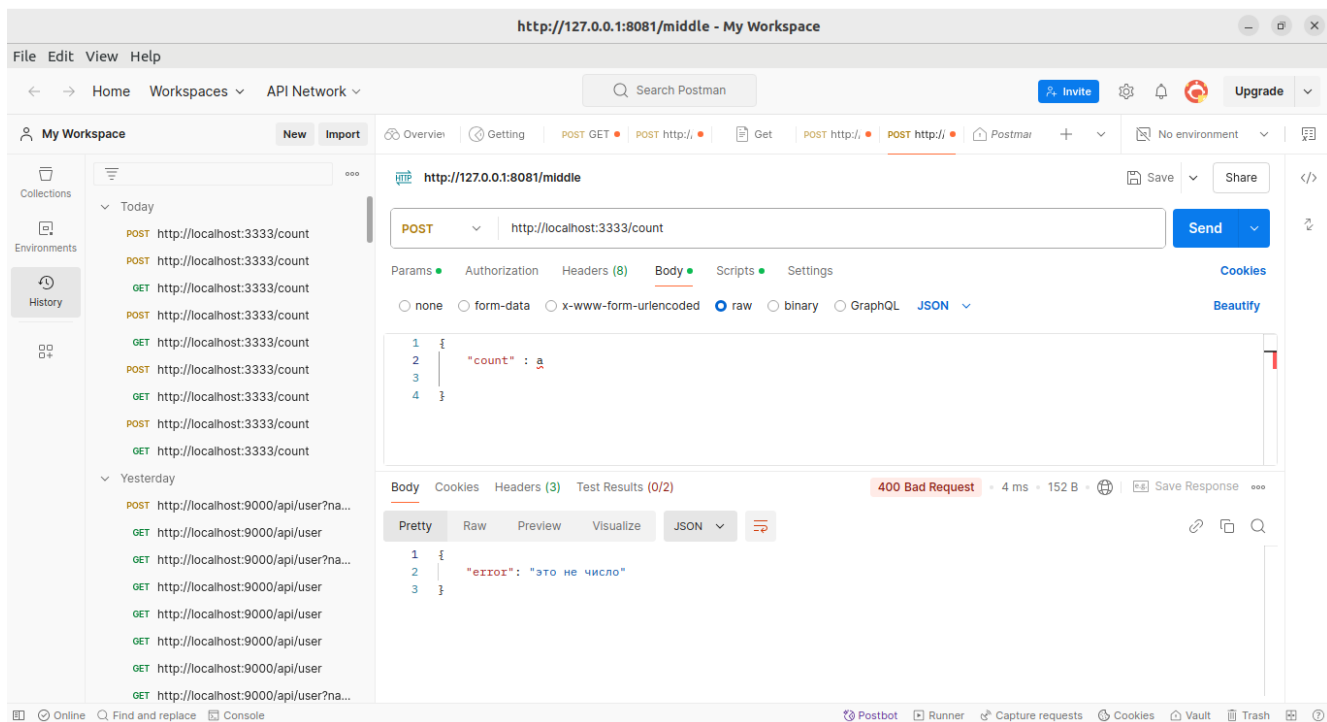


Рисунок 12 - Тестирование сервиса count

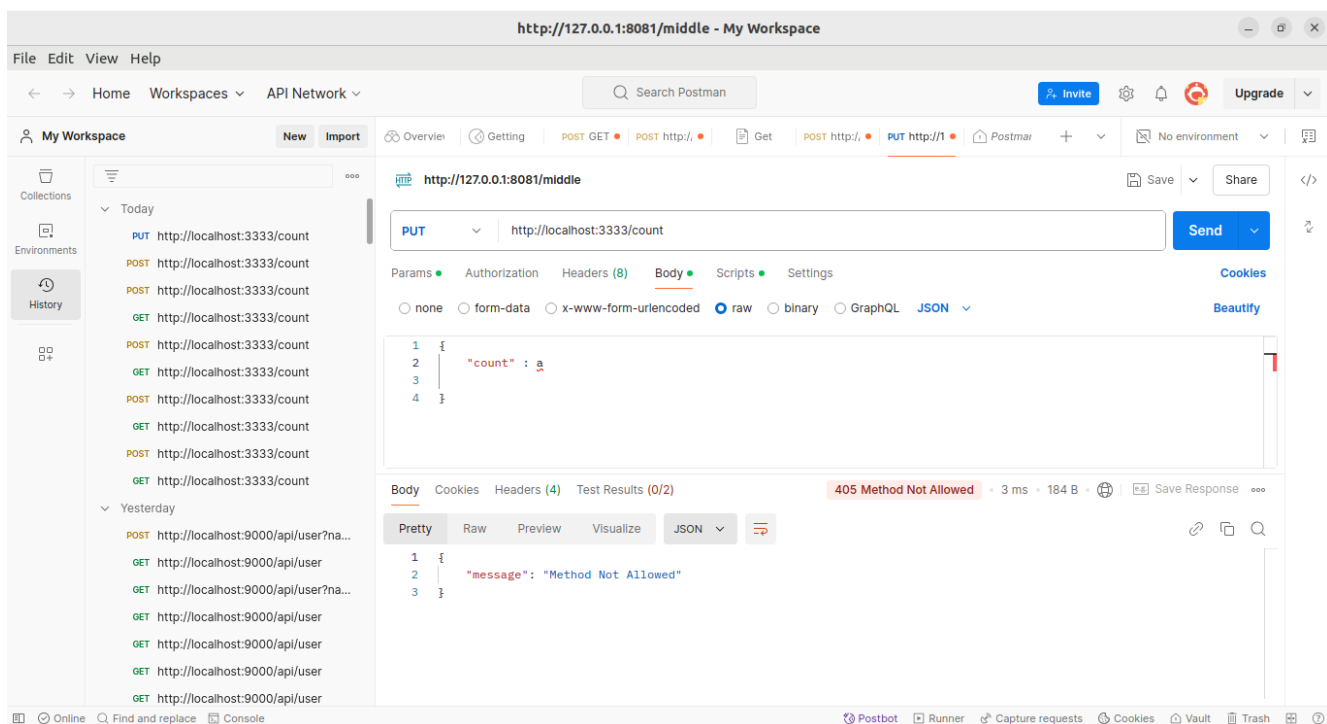


Рисунок 13 - Тестирование сервиса count

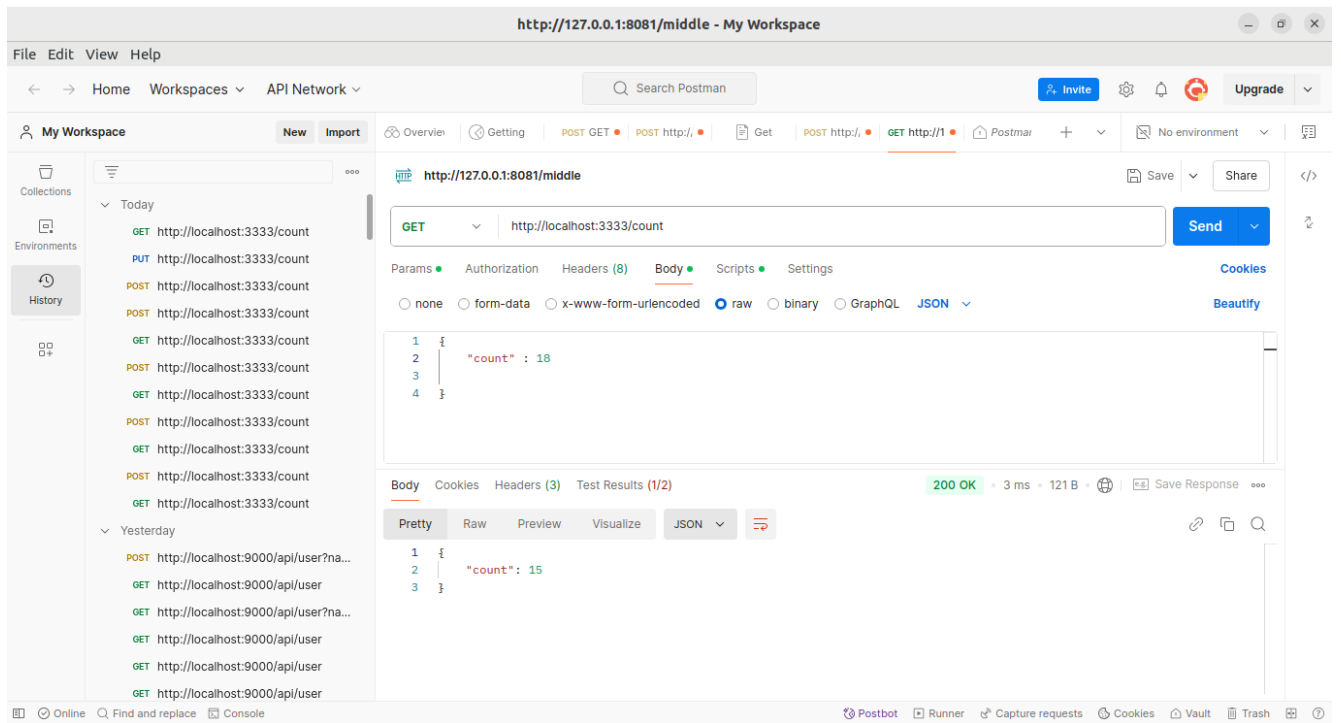


Рисунок 14 - Тестирование сервиса count

**Заключение:** были получены и закреплены первичные навыки использования веб-фреймворков в BackEnd-разработке на Golang.

**Список использованных источников:**

1. <https://echo.labstack.com/docs/quick-start>
2. <https://github.com/labstack/echo>