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: Основы Back-End разработки на Golang

Дисциплина: Языки интернет-программирования

Студент ИУ6-33б _____ А.В. Архипов
(Группа) (Подпись, дата) (И.О. Фамилия)

Преподаватель _____ В.Д. Шульман
(Подпись, дата) (И.О. Фамилия)

Москва, 2024

Основы Back-End разработки на Golang

Цель работы — изучение основ сетевого взаимодействия и серверной разработки с использованием языка Golang.

Задание: Выполнить поставленные задачи.

Программа «Hello, web!»

Напишем веб сервер, который по пути /get отдает текст "Hello, web!".

```go
package main

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

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

func main() {
    http.HandleFunc("/get", Poster)
    err := http.ListenAndServe(":8080", nil)
    if err != nil {
        fmt.Println("Ошибка запуска сервера:", err)
    }
}
```

Рисунок 1. Код программы.

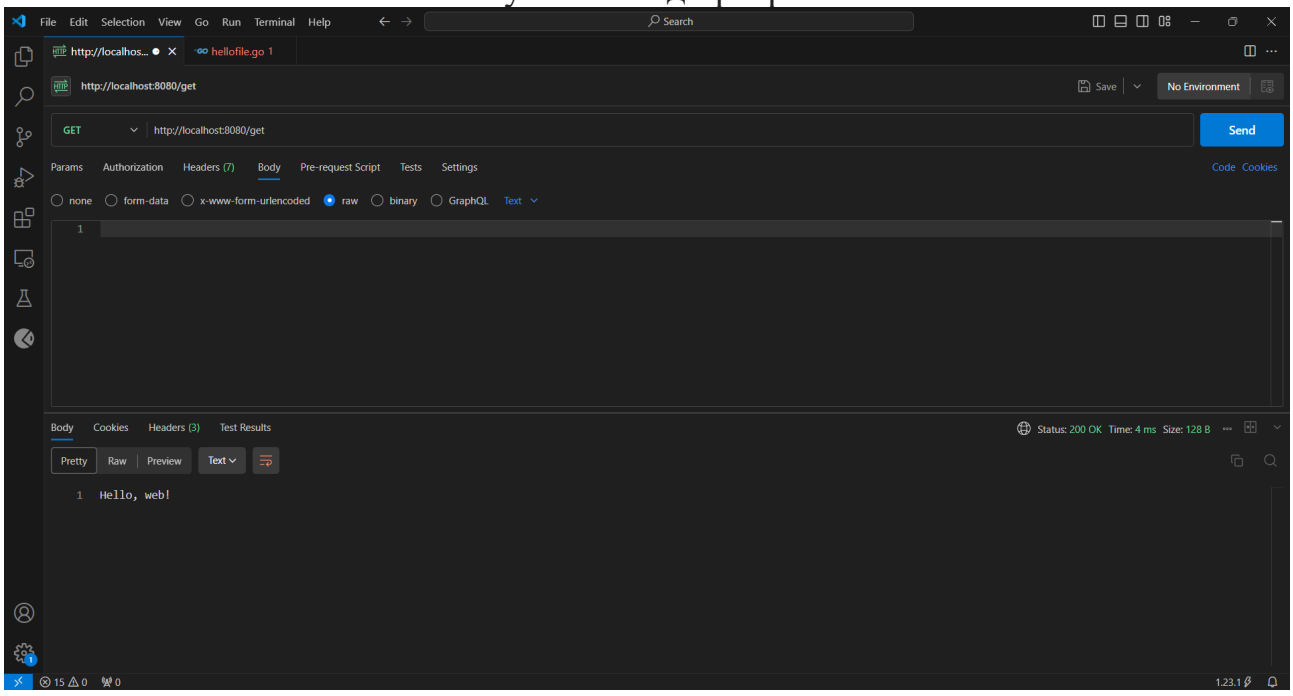


Рисунок 2. Результаты работы сервера.

Программа «Hello, user!»

Напишем веб-сервер который по пути /api/user приветствует пользователя: Принимает и парсит параметр name и делает ответ "Hello,<name>!".

```go
package main

// некоторые импорты нужны для проверки
import (
    "fmt"
    "net/http" // пакет для поддержки HTTP протокола
)

func handler(w http.ResponseWriter, r *http.Request) {
    name := r.URL.Query().Get("name")
    fmt.Fprintf(w, "Hello,%s!", name)
}
func main() {
    http.HandleFunc("/api/user", handler)
    err := http.ListenAndServe(":9000", nil)
    if err != nil {
        fmt.Println("Ошибка запуска сервера:", err)
    }
}
```

Рисунок 3. Код программы.

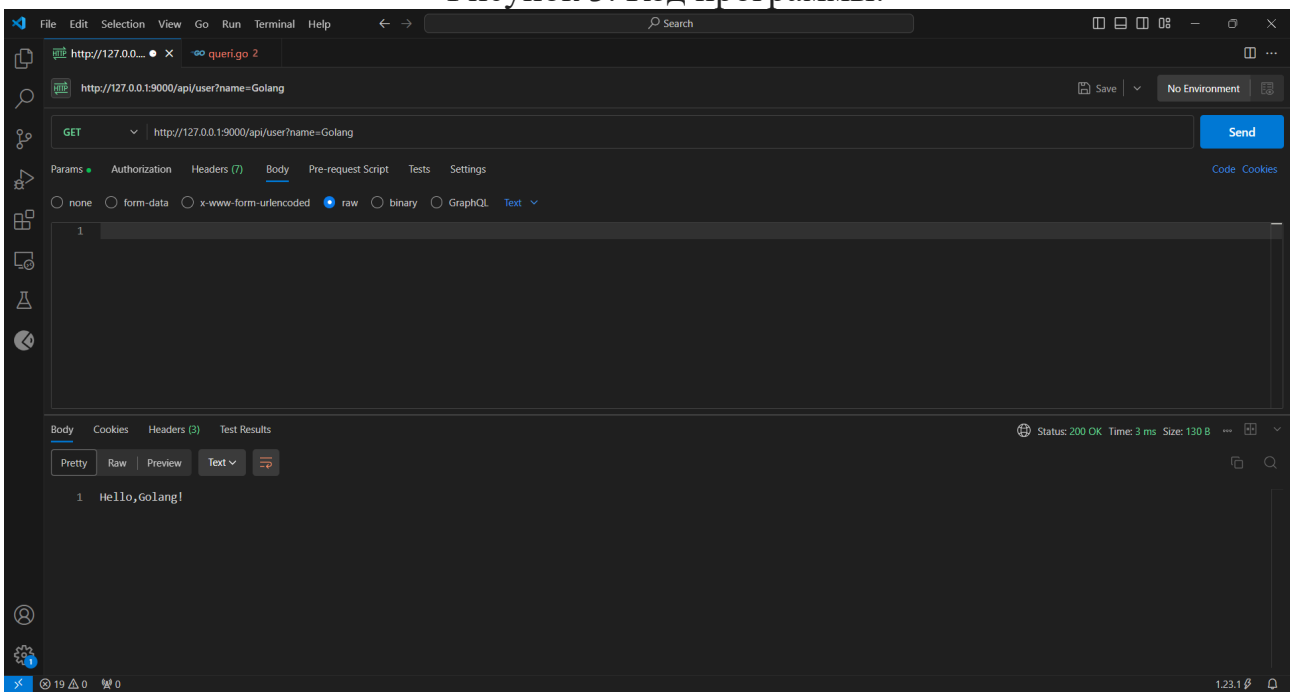


Рисунок 4. Результаты работы сервера.

Программа счетчик

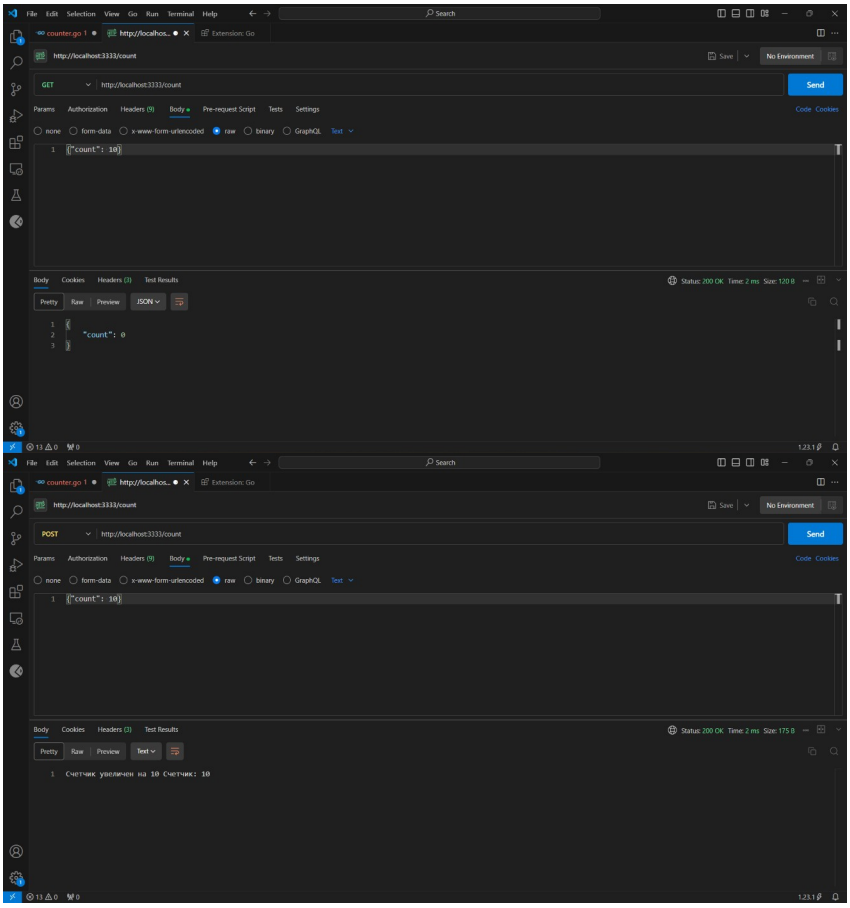
Напиши веб сервер (порт :3333) - счетчик который будет обрабатывать GET (/count) и POST (/count) запросы:

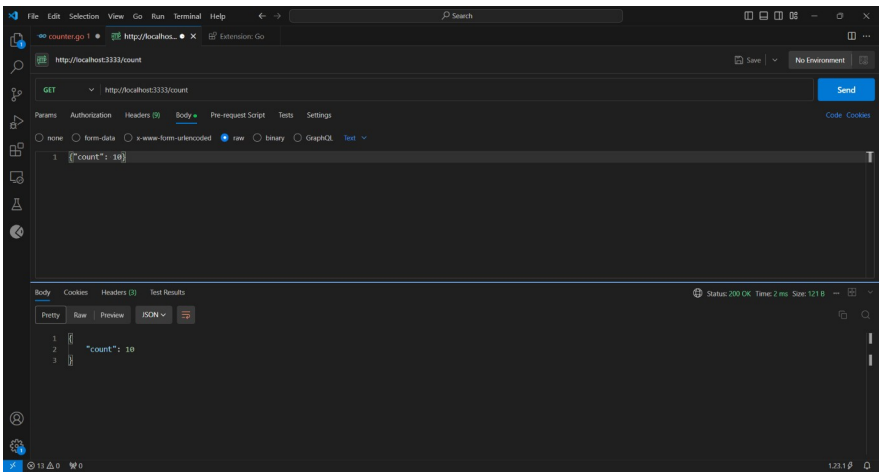
GET: возвращает счетчик

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).

```go
package main

// некоторые импорты нужны для проверки
import (
    "encoding/json"
    "fmt"
    "net/http"
    // "strconv" // вдруг понадобиться вам ;)
)

var counter int

func handler(w http.ResponseWriter, r *http.Request) {
    switch r.Method {
    case http.MethodGet:
        w.Header().Set("Content-Type", "application/json")
        json.NewEncoder(w).Encode(map[string]int{"count": counter})

    case http.MethodPost:
        var req map[string]interface{}
        if err := json.NewDecoder(r.Body).Decode(&req); err != nil {
            http.Error(w, "Ошибка при парсинге запроса", http.StatusBadRequest)
            return
        }
        count, ok := req["count"].(float64)
        if !ok {
            http.Error(w, "это не число", http.StatusBadRequest)
            return
        }
        counter += int(count)
        w.WriteHeader(http.StatusOK)
        fmt.Fprintf(w, "Счетчик увеличен на %d Счетчик: %d", int(count), int(counter))

    default:
        http.Error(w, "Метод не разрешен", http.StatusMethodNotAllowed)
    }
}
func main() {
    http.HandleFunc("/count", handler)
    err := http.ListenAndServe(":3333", nil)
    if err != nil {
        fmt.Println("Ошибка запуска сервера:", err)
    }
}
```

Рисунки 5-6. Код программы.





Рисунки 7-9. Результаты работы сервера.

Заключение: Мы изучили работу серверов и Back-End разработку на Golang.