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 _____ Д.Е.Горячев
(Группа) (Подпись, дата) (И.О. Фамилия)

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

Москва, 2024

1) Цель лабораторной работы

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

2) Задание

Продолжить изучение Golang и познакомиться с набором стандартных библиотек, используемых для организации сетевого взаимодействия и разработки серверных приложений.

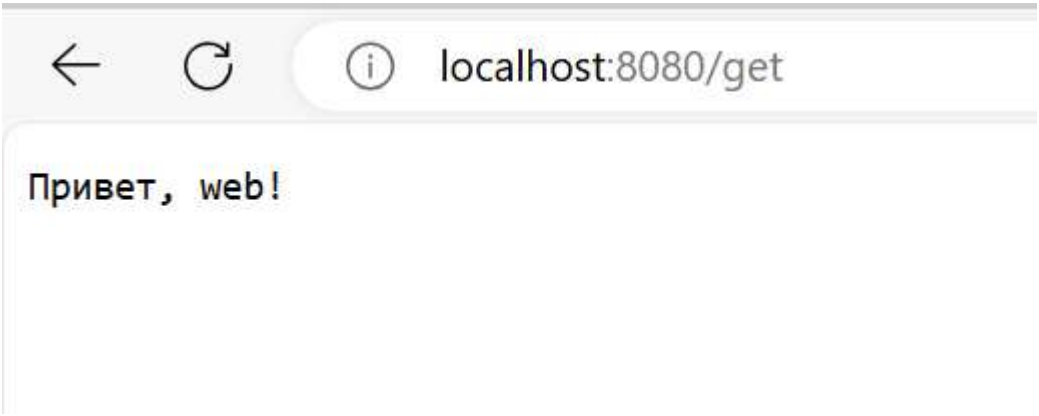
3) Ход работы

1. Реализация веб-сервера /get

Код:

```go
package main

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

// Обработчик HTTP-запросов
func handler(w http.ResponseWriter, r *http.Request) {
    w.Write([]byte("Привет, web!"))
}

func main() {
    // здесь ваш код

    // Регистрируем обработчик для пути "/"
    http.HandleFunc("/get", handler)

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

Результат:



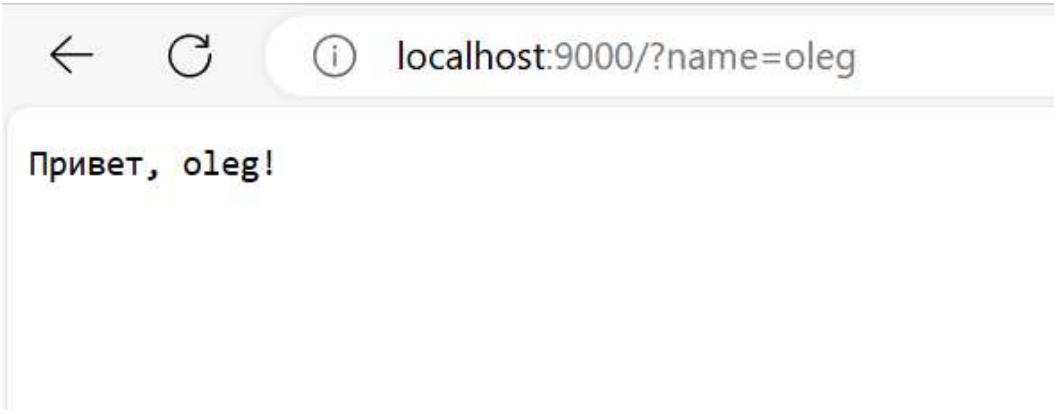
Сервер успешно запускается и возвращает сообщение 'Привет, web!' при обращении к пути '/get'.

2. Реализация веб-сервера /api/user

Код:

```
package main

import (
    "fmt"
    "net/http"
)

// Обработчик HTTP-запросов
func handler(w http.ResponseWriter, r *http.Request) {
    fmt.Println("RawQuery: ", r.URL.String())           // URL с параметрами
    fmt.Println("Name: ", r.URL.Query().Get("name"))    // значение параметра
    fmt.Println("IsExist: ", r.URL.Query().Has("name")) // существует ли такой
параметр
    name := r.URL.Query().Get("name")
    message := fmt.Sprintf("Привет, %s!", name)
    w.Write([]byte(message))
}

func main() {
    // Регистрируем обработчик для пути "/"
    http.HandleFunc("/", handler)

    // Запускаем веб-сервер на порту 9000
    err := http.ListenAndServe(":9000", nil)
    if err != nil {
        fmt.Println("Ошибка запуска сервера:", err)
    }
}
```

Результат:



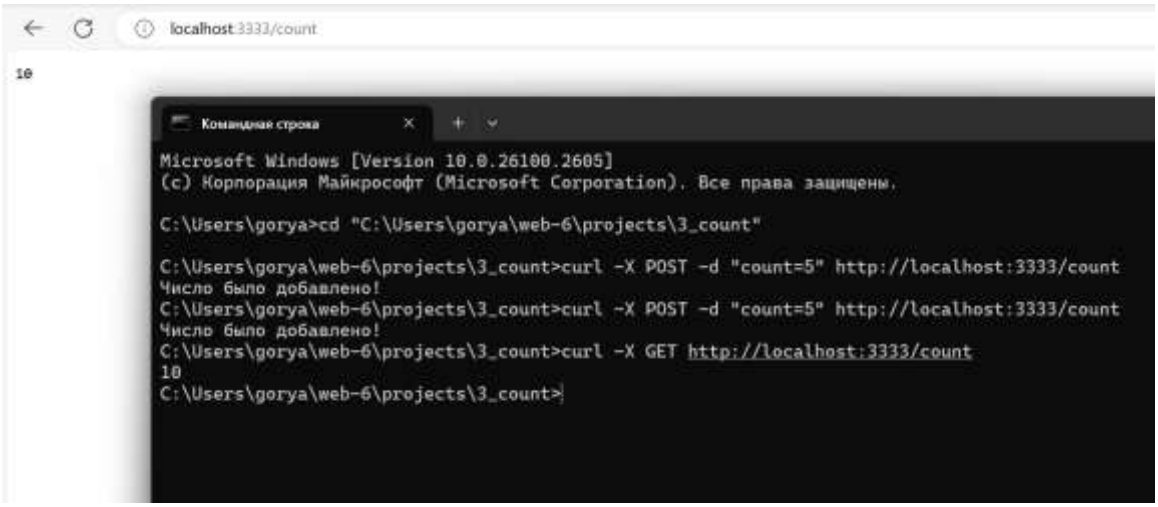
Сервер успешно обрабатывает запросы на пути '/api/user' с параметром 'name', возвращая приветствие.

3. Реализация счётчика /count

```go
package main

import (
    "fmt"
    "log"
    "net/http"
    "strconv"
)

var counter = 0

func handler(w http.ResponseWriter, r *http.Request) {
    if r.Method == "POST" {
        a, err := strconv.Atoi(r.FormValue("count"))
        if err != nil {
            log.Println(err)
            w.WriteHeader(http.StatusBadRequest)
            w.Write([]byte("это не число"))
            return
        }
        counter += a
        message := "Число было добавлено!"
        w.Write([]byte(message))
        return
    } else if r.Method == "GET" {
        w.Write([]byte(strconv.Itoa(counter)))
        return
    }
    w.Write([]byte("Разрешен только метод POST и GET!"))
}

func main() {
    http.HandleFunc("/count", handler)
    err := http.ListenAndServe(":3333", nil)
    if err != nil {
        fmt.Println("Ошибка запуска сервера:", err)
```

```
    }
}
```

Результат:



“curl -X POST -d "count=5" http://localhost:3333/count”

Сервер успешно обрабатывает запросы POST для добавления числа к счётчику и запросы GET для получения текущего значения счётчика.

4) Заключение

В ходе лабораторной работы были изучены основы разработки веб-серверов с использованием языка Golang, а также применены стандартные библиотеки для обработки HTTP-запросов и организации сетевого взаимодействия.

5) Использованные источники

1. https://stepik.org/course/54403/info

2. Coverling/web-6: lab6