Московский государственный технический университет им. Н.Э. Баумана

Факультет ИУ

Кафедра ИУ5

Курс «Основы информатики »

Отчет по Домашнему заданию

Выполнил студент группы ИУ5-33Б: Лачина Е.А. Подпись и дата:

Проверил преподаватель каф.: Гапанюк Ю. Е. Подпись и дата:

HTTP Client Server

Инструментарий:

Golang - go version go1.23.3 darwin/arm64

Visual studio code

Для тестирования запросов использовался API testing

platform Postman

Функционал:

Обработка реквестов сервером производится с использованием функций-хэндлеров:

getRoot, getHello - стандартный запросы

createUser - создает нового пользователя, добавляя его id в массив пользователей.

getUser - запрос информации о пользователе, в соответствии с его id.

deleteUser -удаляет пользователя с соответствующим id.

Middleware:

loggingMiddleware - лог пользователя, отправившего запрос.

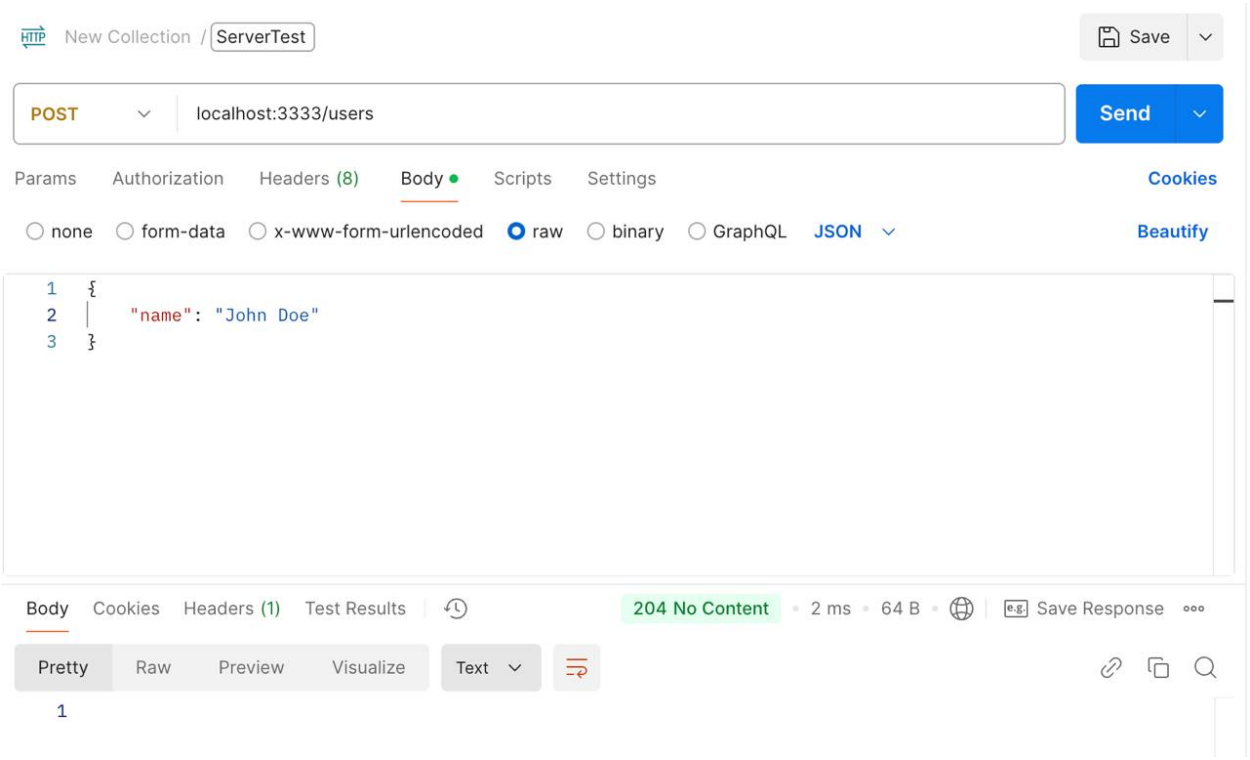
Code Rrealisation

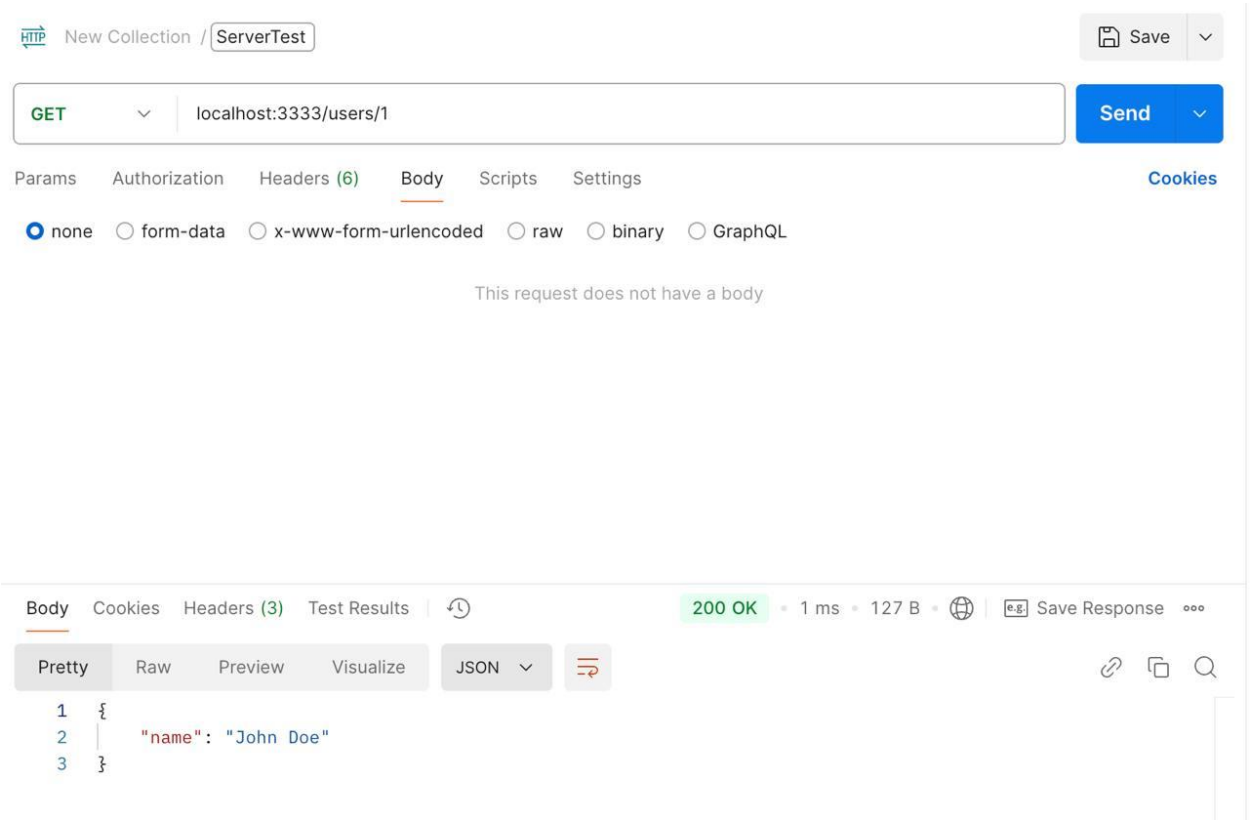
```go
package main

import (
    "encoding/json"
    "errors"
    "fmt"
    "io"
    "log"
    "net/http"
    "os"
    "strconv"
    "sync"
)

type User struct {
    Name string `json:"name"` //Receive in Json format
}

var userCache = make(map[int]User) //HashMap of users

var cacheMutex sync.RWMutex

// Handler
func getRoot(w http.ResponseWriter, r *http.Request) {
    fmt.Printf("got / request\n")
    io.WriteString(w, "This is my website!\n")
}

func getHello(w http.ResponseWriter, r *http.Request) {
    fmt.Printf("got /hello request\n")
    io.WriteString(w, "Hello, HTTP!\n")
}

func createUser(w http.ResponseWriter, r *http.Request) {
    var user User
    err := json.NewDecoder(r.Body).Decode(&user)
    if err != nil {
        http.Error(w, err.Error(), http.StatusBadRequest)
        return
```

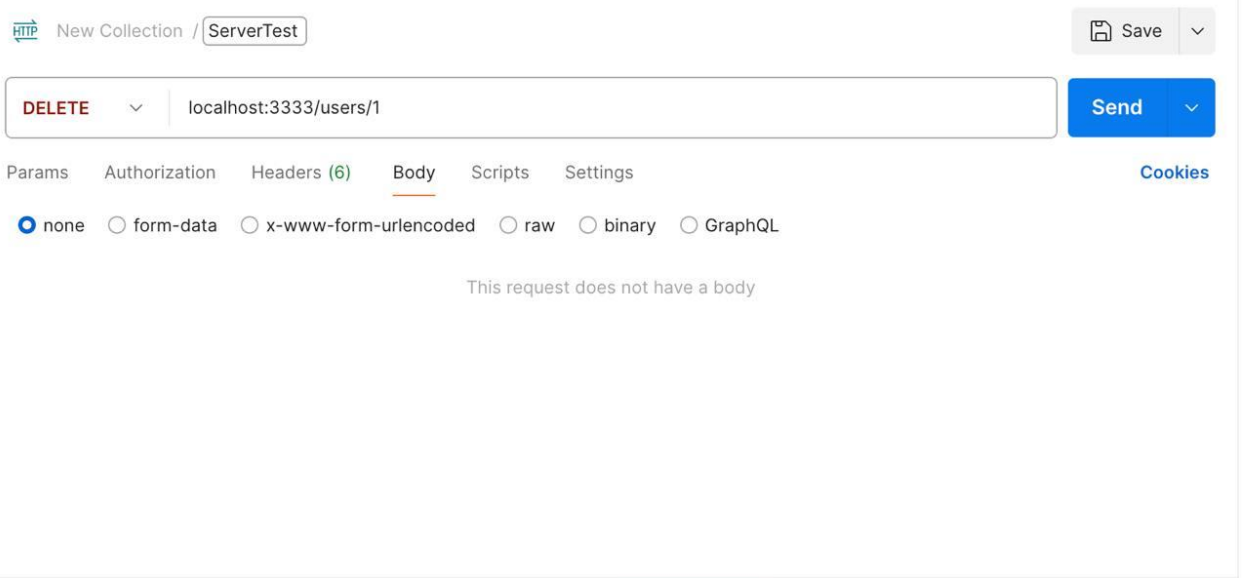
```
    }

    if user.Name == "" {
        http.Error(w, "User name is required",
http.StatusBadRequest)
        return
    }

    cacheMutex.Lock()
    userCache[len(userCache)+1] = user
    cacheMutex.Unlock()

    w.WriteHeader(http.StatusNoContent)
}

func getUser(w http.ResponseWriter, r *http.Request) {
    id, err := strconv.Atoi(r.PathValue("id"))
    if err != nil {
        http.Error(w, err.Error(), http.StatusBadRequest)
        return
    }

    cacheMutex.RLock()
    user, ok := userCache[id]
    cacheMutex.RUnlock()
    if !ok {
        http.Error(w, "User not found", http.StatusNotFound)
        return
    }

    w.Header().Set("Content-Type", "application/json")
    jUser, err := json.Marshal(user)
    if err != nil {
        http.Error(w, err.Error(),
http.StatusInternalServerError)
    }
    w.WriteHeader(http.StatusOK)
    w.Write(jUser)

}

func deleteUser(w http.ResponseWriter, r *http.Request) {
```

```
    id, err := strconv.Atoi(r.PathValue("id"))
    if err != nil {
        http.Error(w, err.Error(), http.StatusBadRequest)
        return
    }

    if _, ok := userCache[id]; !ok {
        http.Error(w, "User not found", http.StatusBadRequest)
        return
    }

    cacheMutex.Lock()
    delete(userCache, id)
    cacheMutex.Unlock()

    w.WriteHeader(http.StatusNoContent)
}
// Middleware
func loggingMiddleware(next http.HandlerFunc) http.HandlerFunc {
    return func(w http.ResponseWriter, r *http.Request) {
        log.Printf("User %s Hit Endpoint", r.FormValue("user"))
        next(w, r)
    }
}

func main() {
    mux := http.NewServeMux()
    mux.HandleFunc("/", loggingMiddleware(getRoot))
    mux.HandleFunc("/hello", loggingMiddleware(getHello))
    mux.HandleFunc("POST /users", loggingMiddleware(createUser))
    mux.HandleFunc("GET /users/{id}", loggingMiddleware(getUser))
    mux.HandleFunc("DELETE /users/{id}",
loggingMiddleware(deleteUser))

    err := http.ListenAndServe(":3333", mux)
    if errors.Is(err, http.ErrServerClosed) {
        fmt.Printf("server closed\n")
    } else if err != nil {
        fmt.Printf("error starting server: %s\n", err)
        os.Exit(1)
    }
}
```

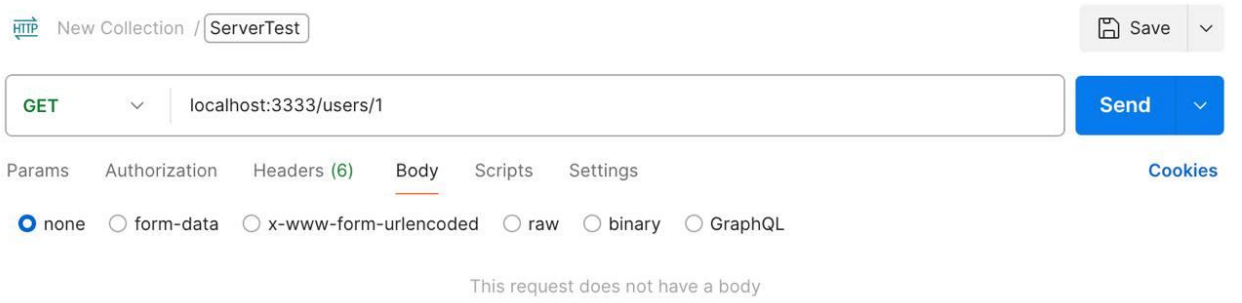
Tests:

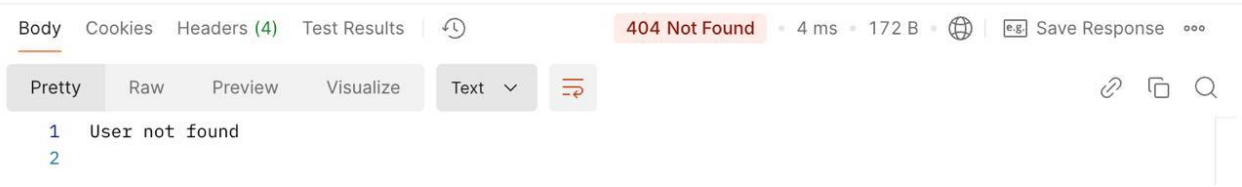

New Collection / ServerTest
Save
POST
localhost:3333/users
Send
Params
Authorization
Headers (8)
Body
Scripts
Settings
Cookies
none
form-data
x-www-form-urlencoded
raw
binary
GraphQL
JSON
Beautify
1 {
2 "name": "John Doe"
3 }
Body
Cookies
Headers (1)
Test Results
204 No Content
2 ms
64 B
Save Response
Pretty
Raw
Preview
Visualize
Text
1

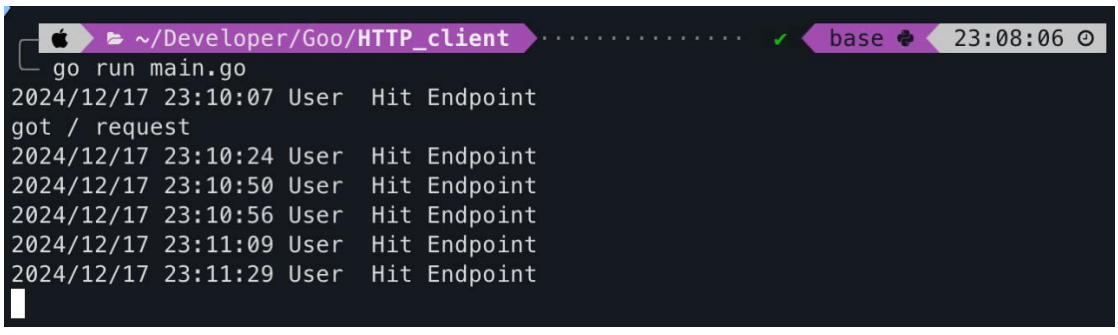

New Collection / ServerTest
Save
GET
localhost:3333/users/1
Send
Params
Authorization
Headers (6)
Body
Scripts
Settings
Cookies
none
form-data
x-www-form-urlencoded
raw
binary
GraphQL
This request does not have a body
Body
Cookies
Headers (3)
Test Results
200 OK
1 ms
127 B
Save Response
Pretty
Raw
Preview
Visualize
JSON
1 {
2 "name": "John Doe"
3 }


New Collection / ServerTest
Save
DELETE
localhost:3333/users/1
Send
Params
Authorization
Headers (6)
Body
Scripts
Settings
Cookies
none
form-data
x-www-form-urlencoded
raw
binary
GraphQL
This request does not have a body


Body
Cookies
Headers (1)
Test Results
204 No Content
1 ms
64 B
Save Response
Pretty
Raw
Preview
Visualize
Text
1


New Collection / ServerTest
Save
GET
localhost:3333/users/1
Send
Params
Authorization
Headers (6)
Body
Scripts
Settings
Cookies
none
form-data
x-www-form-urlencoded
raw
binary
GraphQL
This request does not have a body


Body
Cookies
Headers (4)
Test Results
404 Not Found
4 ms
172 B
Save Response
Pretty
Raw
Preview
Visualize
Text
1 User not found
2


~/Developer/Goo/HTTP_client
base
23:08:06
go run main.go
2024/12/17 23:10:07 User Hit Endpoint
got / request
2024/12/17 23:10:24 User Hit Endpoint
2024/12/17 23:10:50 User Hit Endpoint
2024/12/17 23:10:56 User Hit Endpoint
2024/12/17 23:11:09 User Hit Endpoint
2024/12/17 23:11:29 User Hit Endpoint