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 «Теоретическая информатика и компьютерные технологии»

Лабораторная работа № 2

по курсу «Компьютерные сети»

«Клиент и сервер HTTP»

Вариант 15

Студент группы ИУ9-32Б Павлов И. П.

Преподаватель Посевин Д. П.

Москва 2023

1 Цель работы

Целью данной работы является создание HTTP-клиента и HTTP-сервера на языке Go.

2 Условие

Получение списка заголовков новостей с https://kruzhok.org/news

3 Реализация server.go

```
package main

import (
        _ "embed"
        "html/template"
        "net/http"

        log "github.com/mgutz/logxi/v1"
)

//go:embed templates/index.html
var indexHTML string
var tIndex = template.Must(template.New("index.html").Parse(indexHTML))

func serveClient(w http.ResponseWriter, r *http.Request) {
        path := r.URL.Path
        log.Info("got request", "Method", r.Method, "Path", path)
        if path != "/" && path != "/index.html" {
                log.Error("invalid path", "Path", path)
                w.WriteHeader(http.StatusNotFound)
                return
        }
        data := downloadNews()
        if data == nil {
                log.Error("Empty data", "error", data)
                return
        }
        if err := tIndex.Execute(w, data); err != nil {
                log.Error("HTML creation failed", "error", err)
                return
        }
        log.Info("response sent to client successfully")
}

func main() {
        http.HandleFunc("/", serveClient)
        log.Info("starting listener")
        log.Error("listener failed", "error", http.ListenAndServe("127.0.0.1:9000", nil))
}
```

4 Реализация downloader.go

```
package main

import (
        "net/http"

        log "github.com/mgutz/logxi/v1"
        "golang.org/x/net/html"
)

type Item struct {
        Title string
        Link  string
}

func getAttr(node *html.Node, key string) string {
        for _, attr := range node.Attr {
                if attr.Key == key {
                        return attr.Val
                }
        }
        return ""
}

func getChildren(node *html.Node) []*html.Node {
        var children []*html.Node
        for c := node.FirstChild; c != nil; c = c.NextSibling {
                children = append(children, c)
        }
        return children
}

func isElem(node *html.Node, tag string) bool {
        return node != nil && node.Type == html.ElementNode && node.Data == tag
}

func isText(node *html.Node) bool {
        return node != nil && node.Type == html.TextNode
}

func isDiv(node *html.Node, class string) bool {
        return isElem(node, "div") && getAttr(node, "class") == class
}

func readItem(item *html.Node) *Item {
        cs := getChildren(item)
        left := cs[1]
        right := cs[3]
        if isDiv(left, "news-left") && isDiv(right, "news-right") {
                csr := getChildren(right)
                title := csr[3].FirstChild
                link := csr[9]
                if isText(title) && getAttr(link, "class") == "readmore" {
                        return &Item{
                                Title: title.Data,
                                Link:  getAttr(link.FirstChild, "href"),
                        }
```

```
                }

        }
        return nil
}

func search(node *html.Node) []*Item {
        if isDiv(node, "container") {
                var items []*Item
                for c := node.FirstChild; c != nil; c = c.NextSibling {
                        if isDiv(c, "row news") {
                                if item := readItem(c); item != nil {
                                        items = append(items, item)
                                }
                        }
                }
                return items
        }
        for c := node.FirstChild; c != nil; c = c.NextSibling {
                if items := search(c); items != nil {
                        return items
                }
        }
        return nil
}

func downloadNews() []*Item {
        log.Info("sending request to kruzhok.org/news")
        if response, err := http.Get("https://kruzhok.org/news"); err != nil {
                log.Error("request to kruzhok.org/news failed", "error", err)
        } else {
                defer response.Body.Close()
                status := response.StatusCode
                log.Info("got response from kruzhok.org/news", "status", status)
                if status == http.StatusOK {
                        if doc, err := html.Parse(response.Body); err != nil {
                                log.Error("invalid HTML from kruzhok.org/news", "error", err)
                        } else {
                                log.Info("HTML from kruzhok.org/news parsed successfully")
                                return search(doc)
                        }
                }
        }
        return nil
}
```

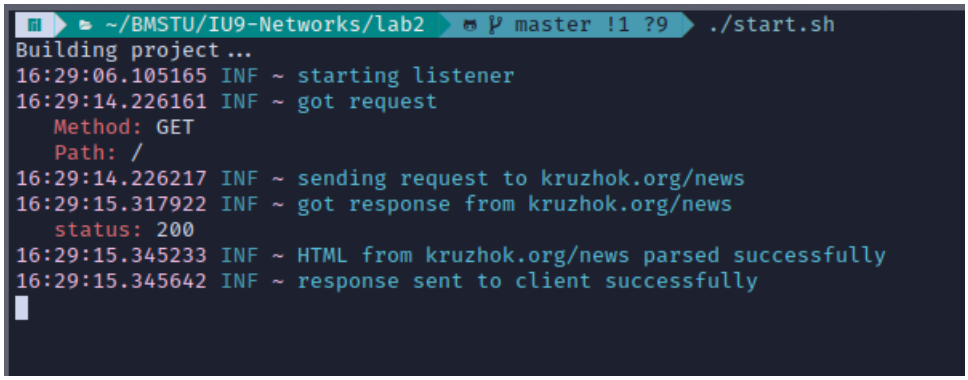

Рис. 1: Билд сервера и получение запроса

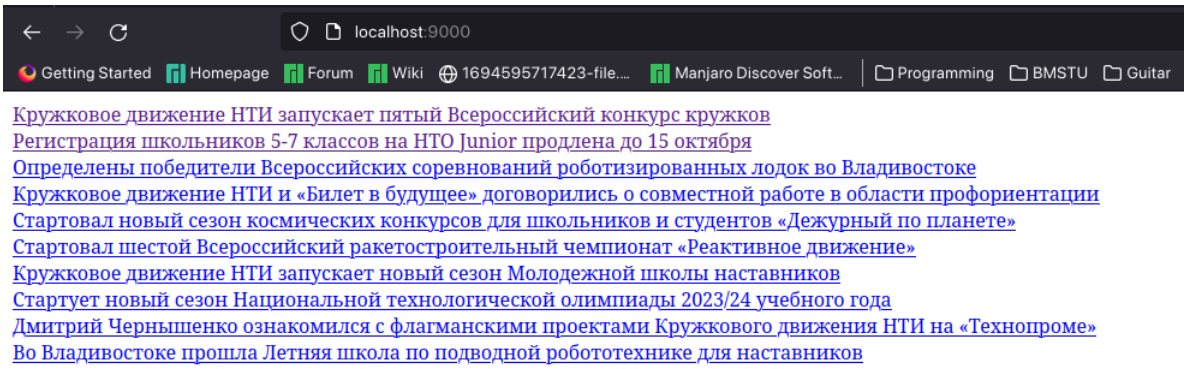

Рис. 2: Вывод заголовков новостей на сервере