Федеральное государственное бюджетное образовательное учреждение
высшего профессионального образования
«Московский государственный технический университет
имени Н.Э. Баумана»
(МГТУ им. Н.Э. Баумана)

Факультет: Информатика и системы управления
Кафедра: Теоретическая информатика и компьютерные технологии

Лабораторная работа №2
«Клиент и сервер HTTP»
по курсу: «Компьютерные сети»

Выполнил:
Студент группы ИУ9-32Б
Гнатенко Т. А.

Проверил:
Посевин Д. П.

Москва, 2022

Цели

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

Задачи

Получение списка наиболее популярных историй (top stories) с https://abcnews.go.com/

Решение

Исходный код

download.go

```
package main

import (
    "fmt"
    "net/http"
    "strings"

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

const URL = "abcnews.go.com/"

type Item struct {
    Ref, Title, ImageSrc string
}

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

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

func getChildElement(node *html.Node, tag string) *html.Node
 ↪  {
    if node == nil {
        return nil
    }

    for child := node.FirstChild; child != nil; child =
     ↪  child.NextSibling {
        if isElement(child, tag) {
            return child
        }
    }

    return nil
}

func getElementsByPredicate(node *html.Node, predicate
 ↪  func(*html.Node) bool) []*html.Node {
    var nodes []*html.Node
```

```
    for _, child := range getChildren(node) {
        if predicate(child) {
            nodes = append(nodes, child)
        }
        nodes = append(nodes, getElementsByPredicate(child,
 ↪  predicate)...)
    }
    return nodes
}

func getTitle(doc *html.Node) string {
    nodes := getElementsByPredicate(doc, func(node
 ↪  *html.Node) bool {
        return node.Data == "title"
    })

    if len(nodes) == 0 {
        return ""
    }

    if children := getChildren(nodes[0]); len(children) == 1
     ↪  && isTextNode(children[0]) {
        return children[0].Data
    }

    return ""
}

func getImageSrc(doc *html.Node) string {
    nodes := getElementsByPredicate(doc, func(node
 ↪  *html.Node) bool {
        return getAttribute(node, "class") ==
         ↪  "responsive-img"
    })
    if len(nodes) == 0 {
        return ""
    }
    img := nodes[0].FirstChild
    if img == nil {
        return ""
```

```
    }
    return getAttribute(img, "src")
}

func getItem(url string, doc *html.Node) *Item {
    return &Item{
        Ref:      url,
        Title:    strings.Split(getTitle(doc), " | ")[0],
        ImageSrc: getImageSrc(doc),
    }
}

func getItems(nodes []*html.Node) []*Item {
    var items []*Item

    for _, node := range nodes {
        url := getAttribute(node.FirstChild, "href")
        doc := downloadHtml(url)
        items = append(items, getItem(url, doc))
        log.Info("Got items for HTML page above")
    }

    return items
}

func downloadHtml(url string) *html.Node {
    log.Info("sending request to " + url)
    if response, err := http.Get(url); err != nil {
        log.Error("request to "+url+" failed", "error", err)
    } else {
        defer response.Body.Close()
        status := response.StatusCode
        log.Info("got response from "+url, "status", status)
        if status == http.StatusOK {
            if doc, err := html.Parse(response.Body); err !=
             ↪  nil {
                log.Error("invalid HTML from "+url, "error",
 ↪  err)
            } else {
                log.Info("HTML from " + url + " parsed
 ↪  successfully")
```

```
                return doc
            }
        }
    }

    return nil
}

func downloadNews() []*Item {
    log.Info("sending request to " + URL)
    if response, err := http.Get("https://" + URL); err !=
     ↪  nil {
        log.Error("request to "+URL+" failed", "error", err)
    } else {
        defer response.Body.Close()
        status := response.StatusCode
        log.Info("got response from "+URL, "status", status)
        if status == http.StatusOK {
            if doc, err := html.Parse(response.Body); err !=
             ↪  nil {
                log.Error("invalid HTML from "+URL, "error",
 ↪  err)
            } else {
                log.Info("HTML from " + URL + " parsed
 ↪  successfully")
                nodes := getElementsByPredicate(doc,
 ↪  func(node *html.Node) bool {
                    return getAttribute(node, "class") ==
                     ↪  "ContentList"
                })
                nodes = getChildren(nodes[0])
                fmt.Println(nodes)
                return getItems(nodes)
            }
        }
    }

    return nil
}
```

server.go

```
package main

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

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

const INDEX_HTML = `
    <!doctype html>
    <html lang="ru">
        <head>
            <meta charset="utf-8">
            <title>Наиболее популярные истории с
 ↪  news.com.au</title>
            <link

 ↪  href="https://cdn.jsdelivr.net/npm/bootstrap@5.0.2/dist/css/bootstrap.
            rel="stylesheet"
            integrity="sha384-
 ↪  EVSTQN3/azprG1Anm3QDgpJLIm9Nao0Yz1ztcQTwFspd3yD65VohhpuuCOmLASjC"
            crossorigin="anonymous"
            />
        </head>
        <body class="bg-dark">
            <div class="text-primary text-center pt-3 mb-0
 ↪  h1"> популярные истории </div>
            <div class="row">
            <div class="col-3">
            </div>
            <div class="col-5 text-center py-5">
                {{if .}}
                    {{range .}}
                        <div class="mb-4">
                            <a href="{{.Ref}}" class="btn
 ↪  btn-light text-center mx-5 w-100">{{.Title}}</a>
                            <img src="{{.ImageSrc}}"></img>
                            <br/>
```

```
                        </div>
                    {{end}}
                {{else}}
                    Не удалось загрузить истории!
                {{end}}
            </div>
            <div class="col-3">
            </div>
            </div>
        </body>
    </html>
    `

var indexHtml =
    template.Must(template.New("index").Parse(INDEX_HTML))

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

func main() {
    http.HandleFunc("/", serveClient)
    log.Info("starting listener")
    log.Error("listener failed", "error",
    http.ListenAndServe("127.0.0.1:8080", nil))
}
```

Вывод

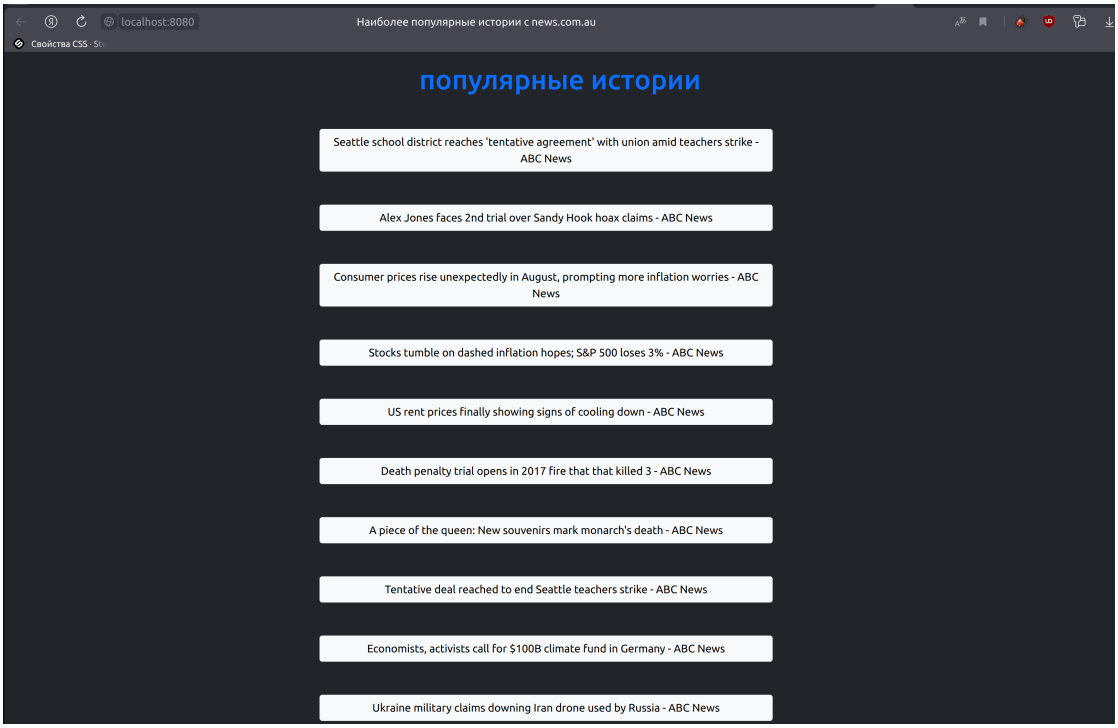

Рис. 1: Получение наиболее популярных историй с сайта