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»

Студент группы ИУ9-32Б Лавров Р. Д.

Преподаватель Посевин Д. П.

Москва 2025

1 Задание

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

2 Результаты

Получение курса указанной пользователем валюты за указанную дату с https://www.sberometer.ru/cbr/ **??**.

Листинг 1: Пример реализации класса

```
package main

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

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

var ITEMS []*Item

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

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

type Item struct {
  Date, USD, EUR, CNY string
}

func getText(node *html.Node) string {
```

```
33    if node == nil {
34      return ""
35    }
36    if node.Type == html.TextNode {
37      return strings.TrimSpace(node.Data)
38    }
39    for c := node.FirstChild; c != nil; c = c.NextSibling {
40      if txt := getText(c); txt != "" {
41        return txt
42      }
43    }
44    return ""
45  }
46
47  func readItem(tr *html.Node) *Item {
48    var cols []string
49    for c := tr.FirstChild; c != nil; c = c.NextSibling {
50      if isElem(c, "td") {
51        cols = append(cols, getText(c))
52      }
53    }
54    if len(cols) < 4 {
55      return nil
56    }
57    return &Item{
58      Date: cols[0],
59      USD:  cols[1],
60      EUR:  cols[2],
61      CNY:  cols[3],
62    }
63  }
64
65  func search(node *html.Node) []*Item {
66    if isElem(node, "tbody") {
67      var items []*Item
68      for c := node.FirstChild; c != nil; c = c.NextSibling {
69        if isElem(c, "tr") {
70          if item := readItem(c); item != nil {
71            items = append(items, item)
72          }
73        }
74      }
75      ITEMS = items
76      return items
77    }
78    for c := node.FirstChild; c != nil; c = c.NextSibling {
```

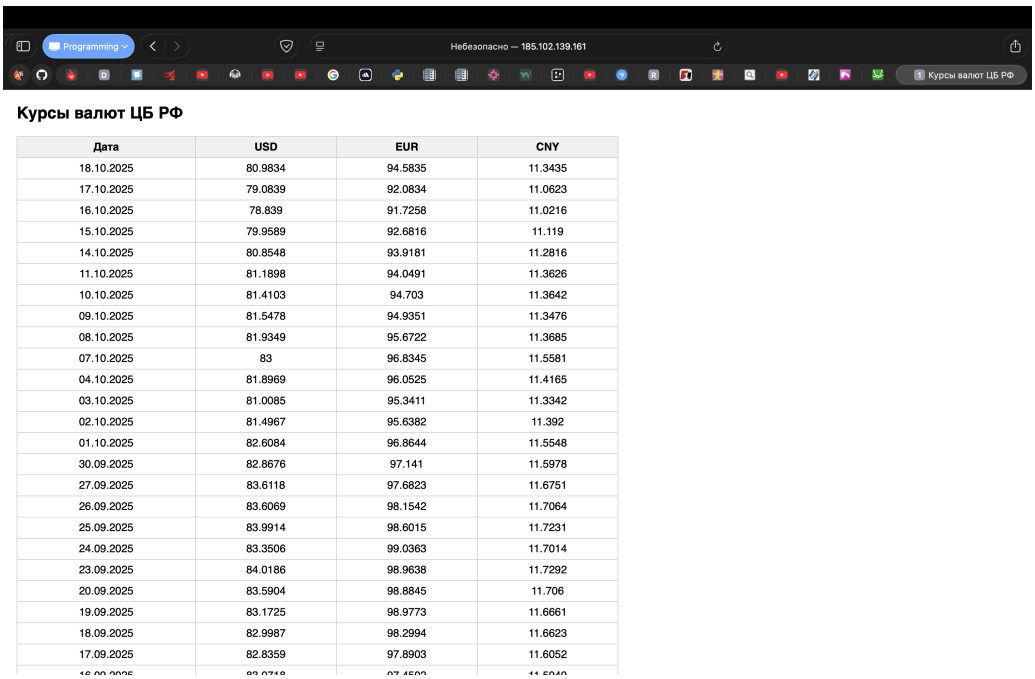
```
    if items := search(c); items != nil {
      return items
    }
  }
  return nil
}

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

const INDEX_HTML = `
<!doctype html>
<html lang="ru">
<head>
    <meta charset="utf-8">
    <titleКурсы валют ЦБ РФ</title>
    <style>
        body { font-family: sans-serif; margin: 20px; }
        table { border-collapse: collapse; width: 60%; }
        th, td { border: 1px solid #ccc; padding: 6px 12px; text-align: center; }
        th { background-color: #f0f0f0; }
    </style>
</head>
<body>
    <h2Курсы валют ЦБ РФ</h2>
    {{if .}}
    <table>
        <tr>
            <thДата></th>
```

```
            <th>USD</th>
            <th>EUR</th>
            <th>CNY</th>
        </tr>
        {{range .}}
        <tr>
            <td>{{.Date}}</td>
            <td>{{.USD}}</td>
            <td>{{.EUR}}</td>
            <td>{{.CNY}}</td>
        </tr>
        {{end}}
    </table>
    {{else}}
        <pНе> удалось загрузить данные!</p>
    {{end}}
</body>
</html>
`

const USD_HTML = `
<!doctype html>
<html lang="ru">
<head>
    <meta charset="utf-8">
    <titleКурсы> валют ЦБ РФ</title>
    <style>
        body { font-family: sans-serif; margin: 20px; }
        table { border-collapse: collapse; width: 60%; }
        th, td { border: 1px solid #ccc; padding: 6px 12px; text-align: center; }
        th { background-color: #f0f0f0; }
    </style>
</head>
<body>
    <h2Курсы> валют ЦБ РФ</h2>
    {{if .}}
    <table>
        <tr>
            <thДата></th>
            <th>USD</th>
        </tr>
        {{range .}}
        <tr>
            <td>{{.Date}}</td>
            <td>{{.USD}}</td>
        </tr>
```

```
        {{end}}
    </table>
    {{else}}
        <pНе> удалось загрузить данные!</p>
    {{end}}
</body>
</html>
‘

const EUR_HTML = ‘
<!doctype html>
<html lang="ru">
<head>
    <meta charset="utf-8">
    <titleКурсы> валют ЦБ РФ</title>
    <style>
        body { font-family: sans-serif; margin: 20px; }
        table { border-collapse: collapse; width: 60%; }
        th, td { border: 1px solid #ccc; padding: 6px 12px; text-align: center; }
        th { background-color: #f0f0f0; }
    </style>
</head>
<body>
    <h2Курсы> валют ЦБ РФ</h2>
    {{if .}}
    <table>
        <tr>
            <thДата></th>
            <th>EUR</th>
        </tr>
        {{range .}}
        <tr>
            <td>{{.Date}}</td>
            <td>{{.EUR}}</td>
        </tr>
        {{end}}
    </table>
    {{else}}
        <pНе> удалось загрузить данные!</p>
    {{end}}
</body>
</html>
‘

const CNY_HTML = ‘
<!doctype html>
```

```
<html lang="ru">
<head>
    <meta charset="utf-8">
    <titleКурсы> валют ЦБ РФ</title>
    <style>
        body { font-family: sans-serif; margin: 20px; }
        table { border-collapse: collapse; width: 60%; }
        th, td { border: 1px solid #ccc; padding: 6px 12px; text-align: center; }
        th { background-color: #f0f0f0; }
    </style>
</head>
<body>
    <h2Курсы> валют ЦБ РФ</h2>
    {{if .}}
    <table>
        <tr>
            <thДата></th>
            <th>CNY</th>
        </tr>
        {{range .}}
        <tr>
            <td>{{.Date}}</td>
            <td>{{.CNY}}</td>
        </tr>
        {{end}}
    </table>
    {{else}}
        <pНе> удалось загрузить данные!</p>
    {{end}}
</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" {
    response.WriteHeader(http.StatusNotFound)
    return
  }
  items := downloadRates()
  if err := indexHtml.Execute(response, items); err != nil {
    log.Error("HTML creation failed", "error", err)
  } else {
```

```
263         log.Info("response sent  successfully ")
264     }
265 }
266
267 func serveUSD(response http.ResponseWriter, request *http.Request) {
268     var indexHtml2 = template.Must(template.New("index").Parse(USD_HTML))
269     if err := indexHtml2.Execute(response, ITEMS); err != nil {
270         log.Error("HTML creation failed", "error", err)
271     } else {
272         log.Info("response sent  successfully ")
273     }
274 }
275
276 func serveEUR(response http.ResponseWriter, request *http.Request) {
277     var indexHtml2 = template.Must(template.New("index").Parse(EUR_HTML))
278     if err := indexHtml2.Execute(response, ITEMS); err != nil {
279         log.Error("HTML creation failed", "error", err)
280     } else {
281         log.Info("response sent  successfully ")
282     }
283 }
284
285 func serveCNY(response http.ResponseWriter, request *http.Request) {
286     var indexHtml2 = template.Must(template.New("index").Parse(CNY_HTML))
287     if err := indexHtml2.Execute(response, ITEMS); err != nil {
288         log.Error("HTML creation failed", "error", err)
289     } else {
290         log.Info("response sent  successfully ")
291     }
292 }
293
294 func main() {
295     http.HandleFunc("/", serveClient)
296     http.HandleFunc("/usd", serveUSD)
297     http.HandleFunc("/eur", serveEUR)
298     http.HandleFunc("/cny", serveCNY)
299     log.Info("starting  server  on :6060")
300     fmt.Println(" listener  failed ", "error", http.ListenAndServe(":6060", nil))
301 }
```



Дата	USD	EUR	CNY
18.10.2025	80.9834	94.5835	11.3435
17.10.2025	79.0839	92.0834	11.0623
16.10.2025	78.839	91.7258	11.0216
15.10.2025	79.9589	92.6816	11.119
14.10.2025	80.8548	93.9181	11.2816
11.10.2025	81.1898	94.0491	11.3626
10.10.2025	81.4103	94.703	11.3642
09.10.2025	81.5478	94.9351	11.3476
08.10.2025	81.9349	95.6722	11.3685
07.10.2025	83	96.8345	11.5581
04.10.2025	81.8969	96.0525	11.4165
03.10.2025	81.0085	95.3411	11.3342
02.10.2025	81.4967	95.6382	11.392
01.10.2025	82.6084	96.8644	11.5548
30.09.2025	82.8676	97.141	11.5978
27.09.2025	83.6118	97.6823	11.6751
26.09.2025	83.6069	98.1542	11.7064
25.09.2025	83.9914	98.6015	11.7231
24.09.2025	83.3506	99.0363	11.7014
23.09.2025	84.0186	98.9638	11.7292
20.09.2025	83.5904	98.8845	11.706
19.09.2025	83.1725	98.9773	11.6661
18.09.2025	82.9987	98.2994	11.6623
17.09.2025	82.8359	97.8903	11.6052

Рис. 1 — Результат работы

3 Вывод

Я научился собирать данные с сайтов с помощью функционала языка Golang