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 № 4

по курсу «Компьютерные сети»

«Proxy-server на go»

Студент группы ИУ9-32Б Тараканов В. Д..

Преподаватель Посевин Д. П.

Москва 2024

1 Задание

Нужно разработать proxy-server на языке go для отображения сайтов

2 Результаты

Исходный код программы представлен в листингах 2– ??.

Листинг 1: client.go

```
1    package main
2
3  package main
4
5  import (
6  "database/sql"
7  "fmt"
8  "html/template"
9  "log"
10 "net/http"
11 "time"
12
13 _ "github.com/go-sql-driver/mysql"
14 "github.com/gorilla/websocket"
15 )
16
17 type Article struct {
18   ID       int
19   Title    string
20   Content string
21 }
22
23 var upgrader = websocket.Upgrader{
24   CheckOrigin: func(r *http.Request) bool {
25     return true
26   },
27 }
28
29 func dbConn() (db *sql.DB, err error) {
30   dsn := "iu9networkslabs:Je2dTYr6@tcp(students.yss.su:3306)/
     iu9networkslabs"
31   db, err = sql.Open("mysql", dsn)
32   if err != nil {
33     return nil, err
34   }
```

```
35      err = db.Ping()
36      if err != nil {
37        return nil, err
38      }
39      return db, nil
40    }
41
42    func fetchArticles() ([]Article, error) {
43      db, err := dbConn()
44      if err != nil {
45        return nil, err
46      }
47      defer db.Close()
48
49      rows, err := db.Query("SELECT id, title, content FROM iu9Tarakanov")
50      if err != nil {
51        return nil, err
52      }
53      defer rows.Close()
54
55      var articles []Article
56      for rows.Next() {
57        var article Article
58        if err := rows.Scan(&article.ID, &article.Title, &article.Content)
      ; err != nil {
59          return nil, err
60        }
61        articles = append(articles, article)
62      }
63
64      return articles, nil
65    }
66
67    func wsHandler(w http.ResponseWriter, r *http.Request) {
68      conn, err := upgrader.Upgrade(w, r, nil)
69      if err != nil {
70        http.Error(w, "Failed to upgrade connection", http.
      StatusInternalServerError)
71        return
72      }
73      defer conn.Close()
74
75      for {
76        articles, err := fetchArticles()
77        if err != nil {
78          log.Println("Error fetching articles:", err)
```

```
79          return
80        }
81
82        err = conn.WriteJSON(articles)
83        if err != nil {
84          log.Println("Error writing to websocket:", err)
85          return
86        }
87
88        time.Sleep(5 * time.Second)
89      }
90    }
91
92    func main() {
93      http.HandleFunc("/ws", wsHandler)
94      http.HandleFunc("/", func(w http.ResponseWriter, r *http.Request) {
95        tmpl, err := template.New("index").Parse(`
96        <!DOCTYPE html>
97        <html lang="en">
98        <head>
99        <meta charset="UTF-8">
100       <meta name="viewport" content="width=device-width, initial-scale
      =1.0">
101       <title>Articles</title>
102       <script>
103       let socket = new WebSocket("ws://" + window.location.host + "/ws")
      ;
104       socket.onmessage = function(event) {
105         let articles = JSON.parse(event.data);
106         let container = document.getElementById("articles");
107         container.innerHTML = "";
108         articles.forEach(article => {
109           let articleDiv = document.createElement("div");
110           let title = document.createElement("h2");
111           title.textContent = article.Title;
112           let content = document.createElement("p");
113           content.textContent = article.Content;
114           articleDiv.appendChild(title);
115           articleDiv.appendChild(content);
116           articleDiv.appendChild(document.createElement("hr"));
117           container.appendChild(articleDiv);
118         });
119       };
120       </script>
121       </head>
122       <body>
```

```
123         <h1>Articles</h1>
124         <div id="articles"></div>
125         </body>
126         </html>
127         `)
128         if err != nil {
129           http.Error(w, "Failed to create template", http.
      StatusInternalServerError)
130           return
131         }
132         tmpl.Execute(w, nil)
133       })
134
135       fmt.Println("Server is running on 9786")
136       log.Fatal(http.ListenAndServe(":9786", nil))
137     }
138
```

Листинг 2: server.go

```
1  package main
2
3  import (
4  "database/sql"
5  "fmt"
6  "github.com/SlyMarbo/rss"
7  _ "github.com/go-sql-driver/mysql"
8  "log"
9  "strings"
10 "time"
11 )
12
13 var database *sql.DB
14
15 type news struct {
16   title       string
17   description string
18 }
19
20 func insertIntoDb(new news) {
21   db, err := sql.Open("mysql", "iu9networkslabs:Je2dTYr6@tcp(students.
       yss.su:3306)/iu9networkslabs")
22   if err != nil {
23     log.Println(err)
24   }
25   database = db
26   defer database.Close()
```

```
27    query := database.QueryRow("SELECT EXISTS(SELECT `id` FROM `
      iu9Tarakanov` WHERE `title`=? OR `content`=?);", new.title, new.
      description)
28    var isExists bool
29
30    query.Scan(&isExists)
31
32    if !isExists {
33      database.Exec("INSERT INTO `iu9Tarakanov` (`title`, `content`)
      VALUES (?, ?);", new.title, new.description)
34    } else {
35      database.Exec("UPDATE `iu9Tarakanov` SET `content`=? WHERE `title
      `=?;", new.description, new.title)
36      database.Exec("UPDATE `iu9Tarakanov` SET `title`=? WHERE `content
      `=?;", new.title, new.description)
37    }
38
39  }
40
41  func rssparser() {
42    rssObject, err := rss.Fetch("https://news.rambler.ru/rss/Namibia/")
43    if err == nil {
44      fmt.Printf("Title           : %s\n", rssObject.Title)
45      fmt.Printf("Description     : %s\n", rssObject.Description)
46      fmt.Printf("Link            : %s\n", rssObject.Link)
47      fmt.Printf("Number of Items : %d\n", len(rssObject.Items))
48      for v := range rssObject.Items {
49        item := rssObject.Items[v]
50        new_news := news{}
51        new_news.title = strings.ReplaceAll(item.Title, "\u00A0", " ")
52        new_news.description = strings.ReplaceAll(item.Summary, "\u00A0",
      " ")
53        insertIntoDb(new_news)
54        fmt.Println()
55        fmt.Printf("Item Number : %d\n", v)
56        fmt.Printf("Title       : %s\n", item.Title)
57        fmt.Printf("Description : %s\n", item.Summary)
58
59      }
60    } else {
61      fmt.Println(err)
62    }
63  }
64
65  func main() {
66    for {
```

```
67        rssparser()
68        time.Sleep(time.Second * 5)
69    }
70
71 }
72
```