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 № 6.1

по курсу «Компьютерные сети»

«Разработка FTP- клиента и web страниц»

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

Преподаватель Посевин Д. П.

Москва 2024

1 Задание

Задача 1: Реализовать FTP-клиент на сервере.

Задача 2: Протестировать соединение GO FTP-клиента с установленным на сервере students.yss.su FTP-сервером со следующими параметрами доступа:

- ftp-host: students.yss.su
- login: ftpiu8
- passwd: 3Ru7yOTA

Задача 3: Реализовать следующую функциональность:

- создание директории;
- удаление файла;
- получение содержимого директории;
- переход в директорию;
- удаление пустой директории;
- удаление директории рекурсивно.

Задача 4: Реализовать web страницу входа на сервер и web страницу для ввода команды и отображения результата

2 Результаты

Исходный код программы представлен в листингах 1 4.

Листинг 1: client.go

```
1  package main
2
3  import (
4  "encoding/json"
5  "fmt"
6  "github.com/gorilla/websocket"
7  "github.com/jlaffaye/ftp"
8  _ "github.com/jlaffaye/ftp"
9  "log"
10 "net/http"
11 "strings"
12 "time"
13 )
14
```

```
15  var upgrader = websocket.Upgrader{
16    CheckOrigin: func(r *http.Request) bool {
17      return true
18    },
19  }
20  var c *ftp.ServerConn
21  var res string
22  var curDir = "./"
23
24  func main() {
25
26    http.HandleFunc("/work", func(w http.ResponseWriter, r *http.Request
      ) {
27      http.ServeFile(w, r, "./public/work/index.html")
28    })
29    http.Handle("/", http.FileServer(http.Dir("./public/logined")))
30    http.HandleFunc("/login", login)
31    http.HandleFunc("/command", handleCommand)
32    http.HandleFunc("/ws", handleWebSocket)
33    log.Fatal(http.ListenAndServe(":8021", nil))
34
35  }
36
37  func handleWebSocket(w http.ResponseWriter, r *http.Request) {
38    conn, err := upgrader.Upgrade(w, r, nil)
39    if err != nil {
40      log.Println("Error while upgrading WebSocket:", err)
41      return
42    }
43    defer conn.Close()
44    for {
45      if err := conn.WriteJSON(res); err != nil {
46        log.Println("Error whise sendin JSON through WebSocket:", err)
47      }
48      time.Sleep(time.Second)
49    }
50
51  }
52
53  func handleCommand(w http.ResponseWriter, r *http.Request) {
54    if r.Method != http.MethodPost {
55      http.Error(w, "Method not allowed", http.StatusMethodNotAllowed)
56      return
57    }
58
59    type commandReq struct {
```

```
60          Command string `json:"command"`
61      }
62      var req commandReq
63      err := json.NewDecoder(r.Body).Decode(&req)
64      if err != nil {
65          http.Error(w, err.Error(), http.StatusBadRequest)
66          return
67      }
68      cmd := strings.Split(req.Command, " ")
69      command := strings.Split(req.Command, " ")[0]
70      path := ""
71      if len(cmd) == 2 {
72          path = strings.Split(req.Command, " ")[1]
73      }
74
75      log.Println("Get command:", req.Command)
76      if command == "LS" {
77          res = ""
78          list, err := c.List("./")
79          if err != nil {
80              res = err.Error()
81              fmt.Println(err)
82              return
83          }
84          for _, v := range list {
85              res += v.Name + " " + v.Type.String() + "\n"
86
87          }
88      } else if command == "MKDIR" {
89          err = c.MakeDir(path)
90          if err != nil {
91              res = err.Error()
92              fmt.Println(err)
93              return
94          }
95          res = "Directory with name " + path + " was created successfully\n
        "
96      } else if command == "CD" {
97          curDir += path + "/"
98          err = c.ChangeDir(path)
99          if path == ".." {
100             new_path := strings.Split(curDir, "/")
101             curDir = strings.Join(new_path[:len(new_path)-3], "/")
102             curDir += "/"
103         }
104
```

```
105         fmt.Println(curDir)
106         if err != nil {
107           res = err.Error()
108           fmt.Println(err)
109           return
110         }
111         res = "Current directory is " + curDir + "\n"
112       } else if command == "RMDIR" {
113         err = c.RemoveDir(path)
114         if err != nil {
115           res = err.Error()
116           fmt.Println(err)
117           return
118         }
119         res = "Directory with name " + path + " was removed successfully\n
        "
120       } else if command == "RM" {
121         err = c.Delete(path)
122         if err != nil {
123           res = err.Error()
124           fmt.Println(err)
125           return
126         }
127         res = "File with name " + path + " was removed successfully\n"
128       } else if command == "RMREC" {
129         err = c.RemoveDirRecur(path)
130         if err != nil {
131           res = err.Error()
132           fmt.Println(err)
133           return
134         }
135         res = "Directory with name " + path + " was recursively removed
        recusive successfully\n"
136       } else {
137         res = "Unknown command: " + command
138       }
139       fmt.Println(command + " done")
140       w.WriteHeader(http.StatusOK)
141     }
142
143     func login(w http.ResponseWriter, r *http.Request) {
144       log.Println("login try")
145       r.Header.Add("Access-Control-Allow-Origin", "*")
146       w.Header().Set("Access-Control-Allow-Origin", "*")
147       w.Header().Set("Access-Control-Allow-Methods", "POST, GET, OPTIONS,
        PUT, DELETE")
```

```
148     w.Header().Set("Access-Control-Allow-Headers", "Content-Type")
149
150     if r.Method != http.MethodPost {
151       http.Error(w, "Method not allowed", http.StatusMethodNotAllowed)
152       return
153     }
154
155     type data struct {
156       Host     string `json:"host"`
157       Login    string `json:"login"`
158       Password string `json:"password"`
159     }
160
161     var newData data
162     err := json.NewDecoder(r.Body).Decode(&newData)
163     if err != nil {
164       http.Error(w, "bad request format", http.StatusBadRequest)
165       return
166     }
167     fmt.Println(newData.Host, newData.Login, newData.Password)
168
169     err = conFtp(newData.Host, newData.Login, newData.Password)
170     if err != nil {
171       w.WriteHeader(http.StatusUnauthorized)
172       return
173     }
174     res = ""
175     log.Println("Enter successfully")
176     w.WriteHeader(http.StatusOK)
177   }
178   func conFtp(host, login, password string) (err error) {
179     c, err = ftp.Dial(host, ftp.DialWithTimeout(5*time.Second))
180     if err != nil {
181       log.Println(err)
182       return err
183     }
184
185     err = c.Login(login, password)
186     if err != nil {
187       log.Println(err)
188       return err
189     }
190
191     return nil
192   }
193
```

Листинг 2: public/logined/index.html

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-scale
=1.0">
<title>Title</title>
<style>
body {
  background: linear-gradient(135deg, #4c5c96, #2b3467);
  font-family: 'Arial', sans-serif;
  display: flex;
  flex-direction: column;
  align-items: center;
  justify-content: center;
  height: 100vh;
  margin: 0;
  color: #fff;
}

h1 {
  text-align: center;
  margin-bottom: 20px;
}

form.form-example {
  background-color: rgba(255, 255, 255, 0.1);
  padding: 20px;
  border-radius: 8px;
  box-shadow: 0 4px 8px rgba(0, 0, 0, 0.1);
  width: 100%;
  max-width: 400px;
}

div.form-example {
  margin-bottom: 15px;
}

label {
  display: block;
  font-size: 16px;
  margin-bottom: 5px;
}

input {
```

```
45        width: 100%;
46        padding: 10px;
47        font-size: 14px;
48        border: none;
49        border-radius: 5px;
50        background-color: rgba(255, 255, 255, 0.2);
51        color: #fff;
52        margin-right: 10px;
53      }
54
55      input:focus {
56        outline: none;
57        background-color: rgba(255, 255, 255, 0.3);
58      }
59
60      .button-container {
61        width: 100%;
62        max-width: 400px;
63        margin-top: 10px;
64      }
65
66      #sendButton {
67        width: calc(100% - 20px);
68        padding: 12px;
69        font-size: 16px;
70        border: none;
71        border-radius: 5px;
72        background-color: #f05454;
73        color: white;
74        cursor: pointer;
75        box-shadow: 0 4px 8px rgba(0, 0, 0, 0.1);
76        margin: 0 10px;
77      }
78
79      #sendButton:hover {
80        background-color: #c0392b;
81      }
82      </style>
83      </head>
84      <body>
85
86
87      <form action="" method="post" class="form-example">
88      <h1>Login page</h1>
89      <div class="form-example">
90      <label for="host">Enter your host: </label>
```

```
91      <input type="text" name="host" id="host" required />
92      </div>
93      <div class="form-example">
94      <label for="login">Enter your login: </label>
95      <input type="text" name="login" id="login" required />
96      </div>
97      <div class="form-example">
98      <label for="password">Enter your password: </label>
99      <input type="password" name="password" id="password" required />
100     </div>
101
102     </form>
103     <div class="button-container">
104     <button id="sendButton">Login</button>
105     </div>
106     <script type="text/javascript" src="app.js"></script>
107     </body>
108     </html>
109
```

Листинг 3: public/logined/app.js

```
1   let button = document.getElementById("sendButton")
2
3   button.onclick = () => login()
4
5
6   function login() {
7     let host = document.getElementById("host").value
8     let login = document.getElementById("login").value
9     let password = document.getElementById("password").value
10    fetch("/login",{
11      method: "POST",
12      headers: {
13        "Content-Type": "application/json"
14      },
15      body: JSON.stringify({
16        host:host,
17        login:login,
18        password:password
19      },
20      )
21    }).then(response => {
22      if (response.ok){
23        window.location = `http://${window.location.host}/work`
24      }else {
25        if (response.status===401){
```

```
26            alert("Invalid credentials")
27          }
28
29        }
30      }).catch(e =>{
31        console.log("Error: " + e)
32      })
33
34
35    }
36
```

Листинг 4: public/work/index.html

```
1     <!DOCTYPE html>
2     <html lang="en">
3     <head>
4     <meta charset="UTF-8">
5     <meta name="viewport" content="width=device-width, initial-scale
      =1.0">
6     <title>Console</title>
7     <style>
8     body {
9       background: #1a1a1a;
10      color: #f0f0f0;
11      font-family: 'Courier New', Courier, monospace;
12      display: flex;
13      justify-content: center;
14      align-items: center;
15      height: 100vh;
16      margin: 0;
17      flex-direction: column;
18    }
19
20    h1 {
21      margin-bottom: 20px;
22    }
23
24    .need {
25      margin-bottom: 15px;
26    }
27
28    label {
29      font-size: 18px;
30    }
31
32    input.text {
```

```
33        width: 300px;
34        padding: 10px;
35        border: 1px solid #555;
36        border-radius: 5px;
37        background-color: #333;
38        color: #f0f0f0;
39        font-size: 16px;
40      }
41
42      input.text:focus {
43        outline: none;
44        border-color: #66ccff;
45      }
46
47      button#commandButton {
48        padding: 10px 20px;
49        font-size: 16px;
50        color: #fff;
51        background-color: #007acc;
52        border: none;
53        border-radius: 5px;
54        cursor: pointer;
55        margin-top: 10px;
56      }
57
58      button#commandButton:hover {
59        background-color: #005fa3;
60      }
61
62      #data {
63        margin-top: 20px;
64        width: 80%;
65        max-width: 600px;
66        background-color: #252525;
67        padding: 15px;
68        border-radius: 5px;
69        overflow: auto;
70        max-height: 300px;
71        font-size: 14px;
72        color: #e6e6e6;
73      }
74
75      pre {
76        margin: 0;
77        white-space: pre-wrap;
78        word-wrap: break-word;
```

```
79      }
80      </style>
81      </head>
82      <body>
83      <h1>Console</h1>
84      <div class="need">
85      <label for="command">Enter the command: </label>
86      <input type="text" name="command" id="command" class="text" required
        />
87      </div>
88      <button id="commandButton">Enter</button>
89      <div id="data"></div>
90      <script>
91      let cmdBtn = document.getElementById("commandButton");
92      cmdBtn.onclick = () => enter();
93
94      const socket = new WebSocket(`ws://${window.location.host}/ws`);
95
96      socket.onopen = function () {
97        console.log("WebSocket connected successfully");
98      };
99
100     socket.onmessage = function (event) {
101       const data = JSON.parse(event.data);
102       let div = document.getElementById("data");
103       div.innerHTML = `<pre>${data}</pre>`;
104     }
105
106     function enter() {
107       let cmd = document.getElementById("command");
108       fetch("/command", {
109         method: "POST",
110         headers: {
111           "Content-Type": "application/json"
112         },
113         body: JSON.stringify({ command: cmd.value})
114       }).then(response => {
115         if (!response.ok) {
116           alert("Bad command");
117         }
118         cmd.value = ""
119       }).catch(e => {
120         console.log("Error: " + e);
121       });
122     }
123     </script>
```

```
124     </body>
125     </html>
126
```