

Exercício 2

Cliente/Servidor em Go



A aplicação – Biblioteca

- ❑ Servidor 1 (autenticação): realiza o *login* do cliente, gera um *token* para o mesmo que fica salvo e verifica se um determinado token é válido.
- ❑ Servidor 2 (biblioteca): retorna uma lista de livros.
- ❑ O servidor 2 se comunica com o 1 para validar o *token* do cliente para ele poder visualizar os livros.

auth

```
auth.go  auth_service.go x  client.go  library.go  requests.go  ...  auth_service.go x

106 func handleUDPConnection(conn net.PacketConn) {
107     defer conn.Close()
108
109     for {
110         fmt.Printf("Auth Serving %s\n", conn.LocalAddr().String())
111
112         bytesAction := make([]byte, bufferSize)
113         nBytes, addr, err := conn.ReadFrom(bytesAction)
114         if err != nil {
115             fmt.Println(err)
116             return
117         }
118
119         action := string(bytesAction[:nBytes])
120         action = utils.FormatString(action)
121
122         conn.WriteTo(utils.EncodeString("ok"), addr) // send ok connection
123
124         bytesRequest := make([]byte, bufferSize)
125         nBytes, addr, err = conn.ReadFrom(bytesRequest)
126         if err != nil {
127             return
128         }
129
130         switch action {
131         case "login":
132             var msgRequest messages.UserAuthRequest
133
134             err = json.Unmarshal(bytesRequest[:nBytes], &msgRequest)
135             if err != nil {
136                 bytes, _ := json.Marshal(&messages.UserAuthResponse{
137                     err: err.Error()
138                 })
139                 conn.WriteTo(utils.Encode(bytes), addr)
140                 continue
141             }
142             bytes := processLogin(msgRequest)
143             conn.WriteTo(utils.Encode(bytes), addr)
144         case "isLoggedIn":
145             var msgRequest messages.IsLoggedInAuthRequest
146             err = json.Unmarshal(bytesRequest[:nBytes], &msgRequest)
147             if err != nil {
148                 bytes, _ := json.Marshal(&messages.IsLoggedInAuthResponse{
149                     err: err.Error()
150                 })
151                 conn.WriteTo(utils.Encode(bytes), addr)
152                 continue
153             }
154             bytes := processIsLoggedIn(msgRequest)
155             conn.WriteTo(utils.Encode(bytes), addr)
156         }
157     }
158 }

58 func handleConnection(conn net.Conn) {
59     for {
60         fmt.Printf("Auth Serving %s\n", conn.LocalAddr().String())
61
62         action, err := bufio.NewReader(conn).ReadString('\n')
63         if err != nil {
64             return
65         }
66
67         action = strings.Replace(action, "\n", "", 1)
68         conn.Write(utils.EncodeString("ok")) // send ok connection
69
70         bytesRequest, err := bufio.NewReader(conn).ReadBytes('\n')
71         if err != nil {
72             return
73         }
74
75         switch action {
76         case "login":
77             var msgRequest messages.UserAuthRequest
78
79             err = json.Unmarshal(bytesRequest, &msgRequest)
80             if err != nil {
81                 bytes, _ := json.Marshal(&messages.UserAuthResponse{
82                     err: err.Error()
83                 })
84                 conn.Write(utils.Encode(bytes))
85                 continue
86             }
87             bytes := processLogin(msgRequest)
88             conn.Write(utils.Encode(bytes))
89         case "isLoggedIn":
90             var msgRequest messages.IsLoggedInAuthRequest
91
92             err = json.Unmarshal(bytesRequest, &msgRequest)
93             if err != nil {
94                 // adicionar erro ao messages.IsLoggedInAuthResponse
95                 bytes, _ := json.Marshal(&messages.IsLoggedInAuthResponse{
96                     err: err.Error()
97                 })
98                 conn.Write(utils.Encode(bytes))
99                 continue
100             }
101             bytes := processIsLoggedIn(msgRequest)
102             conn.Write(utils.Encode(bytes))
103         }
104     }
105 }
```

Ln 174, Col 1 Tab Size: 4 UTF-8 CRLF Go 6

library

```
auth.go  auth_service.go  library_service.go  utils.go  ...  library_service.go
102
103 func handleConnection(conn net.Conn) {
104     for {
105         fmt.Printf("Library Serving %s\n", conn.LocalAddr().String())
106
107         action, err := bufio.NewReader(conn).ReadString('\n')
108         if err != nil {
109             return
110         }
111
112         action = strings.Replace(action, "\n", "", 1)
113
114         conn.Write(utils.EncodeString("ok")) // send connection ok
115
116         bytesRequest, err := bufio.NewReader(conn).ReadBytes('\n')
117         if err != nil {
118             return
119         }
120
121         switch action {
122             case "list":
123                 var msgRequest messages.ServiceRequest
124
125                 err = json.Unmarshal(bytesRequest, &msgRequest)
126                 if err != nil {
127                     bytes, _ := json.Marshal(&messages.Error{Error: "Invalid request"})
128                     conn.Write(utils.Encode(bytes))
129                     continue
130                 }
131
132                 bytes := processList(msgRequest)
133                 conn.Write(utils.Encode(bytes))
134             }
135     }
136 }
137
138 func processList(request messages.ServiceRequest) []byte {
139     authConn, err := utils.OpenConnection(library.Protocol, library.Address)
140     if err != nil {
141         return nil
142     }
143 }

59 func handleUDPConnection(conn net.PacketConn) {
60     defer conn.Close()
61
62     for {
63         fmt.Printf("Library Serving %s\n", conn.LocalAddr().String())
64         bytesAction := make([]byte, bufferSize)
65
66         nBytes, addr, err := conn.ReadFrom(bytesAction)
67         if err != nil {
68             return
69         }
70
71         _, err = conn.WriteTo(utils.EncodeString("ok"), addr)
72         if err != nil {
73             return
74         }
75
76         action := string(bytesAction[:nBytes])
77         action = strings.Replace(action, "\n", "", 1)
78
79         bytesRequest := make([]byte, bufferSize)
80
81         nBytes, addr, err = conn.ReadFrom(bytesRequest)
82         if err != nil {
83             return
84         }
85
86         switch action {
87             case "list":
88                 var msgRequest messages.ServiceRequest
89
90                 err = json.Unmarshal(bytesRequest[:nBytes], &msgRequest)
91                 if err != nil {
92                     bytes, _ := json.Marshal(&messages.Error{Error: "Invalid request"})
93                     conn.WriteTo(utils.Encode(bytes), addr)
94                     continue
95                 }
96
97                 bytes := processList(msgRequest)
98                 conn.WriteTo(utils.Encode(bytes), addr)
99     }
}
```

Ln 1, Col 1 Tab Size: 4 UTF-8 CRLF Go 6

main

```
14
15 var client c.Client
16 var user library.User
17
18 func main() {
19     protocol := os.Args[1]
20     authPort := os.Args[2]
21     servicePort := os.Args[3]
22
23     go auth.StartServer(protocol, authPort)
24     go library.StartServer(protocol, servicePort, authPort)
25
26     client = c.Client{
27         Protocol: protocol,
28         AuthPort: authPort,
29         ServicePort: servicePort,
30     }
31
32     reader := bufio.NewReader(os.Stdin)
33
34     for {
35         fmt.Println("Escolha uma das opções abaixo: ")
36         fmt.Println("1 - Login")
37         fmt.Println("2 - Listar livros")
38
39         option, _ := reader.ReadString('\n')
40         option = utils.FormatString(option)
41
42         switch option {
43             case "1":
44                 login()
45             case "2":
46                 books()
47             default:
48                 fmt.Printf("%s é uma opção inválida.\n", option)
49         }
50     }
51 }
52
53 func login() { ...
75 }
```

ok

```
PS C:\Users\1555 MX7\go\src\middleware2\middleware\src> go run .\main.go tcp 1234 2345
Escolha uma das opções abaixo:
1 - Login
2 - Listar livros
1
Login: edjan
Password: michiles
Auth Serving 127.0.0.1:1234
Auth Serving 127.0.0.1:1234
User logged in successfully
Escolha uma das opções abaixo:
1 - Login
2 - Listar livros
2
Library Serving 127.0.0.1:2345
Auth Serving 127.0.0.1:1234
Auth Serving 127.0.0.1:1234
Library Serving 127.0.0.1:2345
[Name: Medicina Interna de Harrison - 2 Volumes
Description: Apresentando os extraordinários avanços ocorridos em todas as áreas da medicina, esta nova edição do Harrison foi amplamente revisada para oferecer uma
atualização completa sobre a patogênese das doenças, ensaios clínicos, técnicas de diagnóstico, diretrizes clínicas baseadas em evidências, tratamentos já estabeleci
dos e métodos recentemente aprovados
PublishDate: 21 nov 2016
Author: Dennis L. Kasper, Stephen L. Hauser, J. Larry Jameson, Anthony S. Fauci, Dan L. Longo, Joseph Loscalzo
Categories: [Medicina Especialidades]

Name: Netter Atlas de Anatomia Humana 7ª edição
Description: É um dos nomes mais fortes mundialmente na área de Anatomia, reconhecido pela didática e clareza de suas ilustrações. Figuras modernas, que, em um volum
e, apresenta todo o corpo humano em descrições detalhadas e clinicamente relevantes
PublishDate: 8 dez 2018
Author: Frank H. Netter
Categories: [Medicina Anatomia]

Name: Cultura Inglesa. Go Beyond - Caixa com Workbook: Student's Pack With Workbook
Description: Go Beyond is an exciting 6-level American English course for teenagers learning English. The course covers CEFR levels A1+ through to B2, + all levels b
eing based on mapping of the requirements of the CEFR and international exams.
PublishDate: 2 out 2018
Author: Rebecca Robb Benne
Categories: [Inglês e outras línguas Educação Didáticos]

]
```

user not found

```
PS C:\Users\1555 MX7\go\src\middleware2\middleware\src> go run .\main.go tcp 1234 2345
```

```
Escolha uma das opções abaixo:
```

```
1 - Login
```

```
2 - Listar livros
```

```
1
```

```
Login: xablau
```

```
Password: xablau
```

```
Auth Serving 127.0.0.1:1234
```

```
Auth Serving 127.0.0.1:1234
```

```
User not found
```

```
Escolha uma das opções abaixo:
```

```
1 - Login
```

```
2 - Listar livros
```

```
█
```

user not logged in

```
Escolha uma das opções abaixo:  
1 - Login  
2 - Listar livros  
2  
Library Serving 127.0.0.1:2345  
Auth Serving 127.0.0.1:1234  
Auth Serving 127.0.0.1:1234  
Library Serving 127.0.0.1:2345  
User not logged in  
Escolha uma das opções abaixo:  
1 - Login  
2 - Listar livros  
█
```