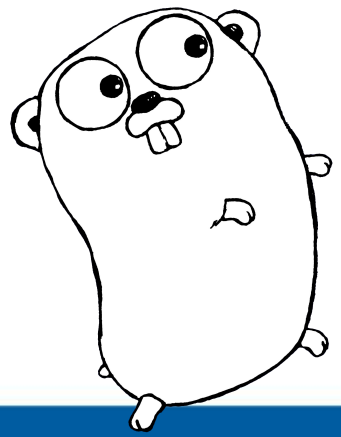
INFTalk - RPC

Gislainy Crisóstomo
gislainycrisostomo@gmail.com
Pabllo Felipe
pabllofelipe2@gmail.com

Implementação

```
//server
import (
    "fmt"
    "net"
    "net/rpc"
    "sync"
)
```

```
//client
import (
    "fmt"
    "net/rpc"
    "bufio"
    "log"
    "os"
    "strings"
    "sync"
    "time"
)
```



Package rpc

(...)

- o método tem dois argumentos, ambos tipos exportados (ou armazenados).
- o segundo argumento do método é um ponteiro.
- o método tem erro de tipo de retorno.

```
func (t *T) MethodName(argType T1, replyType *T2) error
```

O primeiro argumento do método representa os argumentos fornecidos pelo chamador; o segundo argumento representa os parâmetros de resultado a serem devolvidos ao chamador

Server - Struct

```
type Nothing bool

type ChatServer struct {
    users []string
    messageQueue map[string][]string
    mutex sync.Mutex
}
type Message struct {
    Nickname string
    Text string
}
```

Server - main()

```
func main() {
    cs := new(ChatServer)
    cs.messageQueue = make(map[string][]string)
    rpc.Register(cs)
    ln, err := net.Listen("tcp", ":9999")
    if err != nil {
        fmt.Println(err)
        return
    }
    for {
        c, err := ln.Accept()
        if err != nil {
            continue
        }
        go rpc.ServeConn(c)
    }
}
```

Server - Methods (1)

```
func (chat *ChatServer) CreateUser(nickname string, reply *string)
error {
    chat.mutex.Lock()
    defer chat.mutex.Unlock()
    for _, value := range chat.users {
        if value == nickname {
            *reply = "Already user\n"
            return nil
        }
    }
    chat.users = append(chat.users, nickname)
    chat.messageQueue[nickname] = nil;
    for k, _ := range chat.messageQueue {
        if(k != nickname) {
            chat.messageQueue[k] = append(chat.messageQueue[k],
nickname+ " has joined.")
        }
    }
    *reply = "User create with success\n"
    return nil
}
```

Server - Methods (2)

```
func (chat *ChatServer) ConnectedUsersList(nothing
*Nothing, reply *string) error {
    for _, value := range chat.users {
        *reply += value + "\n"
    }
    return nil
}
```

Server - Methods (3)

```
func (chat *ChatServer) CheckMessages(nickname string,
reply *[]string) error {
    chat.mutex.Lock()
    defer chat.mutex.Unlock()
    *reply = chat.messageQueue[nickname]
    chat.messageQueue[nickname] = nil
    return nil
}
```

Server - Methods (4)

```
func (chat *ChatServer) SendMessage(message Message,
reply *string) error {
    chat.mutex.Lock()
    defer chat.mutex.Unlock()
    for k, _ := range chat.messageQueue {
        chat.messageQueue[k] =
append(chat.messageQueue[k], message.Nickname + ": " +
message.Text)
    }
    return nil
}
```

Server - Methods (5)

```
func (chat *ChatServer) Quit(nickname string,  reply
*string) error {
    chat.mutex.Lock()
    defer chat.mutex.Unlock()
    delete(chat.messageQueue, nickname)
    for i := range chat.users {
        if chat.users[i] == nickname {
            chat.users = append(chat.users[:i],
chat.users[i+1:]...)
        }
    }
    for k, v := range chat.messageQueue {
        chat.messageQueue[k] = append(v, nickname+" has
logged out.")
    }
    *reply = "User " + nickname + " has logged out."
    return nil
}
```

Client - Struct

```
type Client struct {
    nickname string
    Connection   *rpc.Client
}
type Message struct {
    Nickname   string
    Text string
}
var wg sync.WaitGroup
```

Client - main()

```
func main() {
    var client *Client = &Client{}
    client.CreateConnection()
    client.Help()
    go client.CheckMessages()
    client.Input()
}
```

Client - Methods (1)

```
func (c *Client) CreateConnection() {
    connection, err := rpc.Dial("tcp", "127.0.0.1:9999")
    if err != nil {
      log.Fatalln(err)
        return
    }
    c.Connection = connection
}
```

Client - Methods (2)

```
func (c *Client) CheckMessages() {
    var reply []string
    for {
        err :=
c.Connection.Call("ChatServer.CheckMessages",
c.nickname, &reply)
        if err != nil {
            log.Fatalln("Chat has been shutdown.
Goodbye.")
        }
        for i := range reply {
            log.Println(reply[i])
        }
        time.Sleep(time.Second)
    }
}
```

Client - Methods (3)

```
func (c *Client) Input() {
    for {
        reader := bufio.NewReader(os.Stdin)
        str, err := reader.ReadString('\n')
        if err != nil {
            wg.Done()
            break
        }
        if strings.HasPrefix(str, CMD_CREATE) {
            c.CreateUser(str)
        } else if strings.HasPrefix(str, CMD_LIST) {
            c.ConnectedUsersList()
        } else if strings.HasPrefix(str, CMD_QUIT) {
            c.Quit()
        } else if strings.HasPrefix(str, CMD_HELP) {
            c.Help()
        } else if len(str) > 1 && len(c.nickname) > 0  {
            c.SendMessage(str)
        } else if( len(c.nickname) == 0) {
            fmt.Println("Create a user with ==> " + CMD_CREATE)
        }
    }
}
```

Client - Methods (4)

```
func (c *Client) CreateUser(str string) {
    var message string
    nickname :=
strings.TrimSuffix(strings.TrimPrefix(str, CMD_CREATE + "
"), "\n")
    c.nickname = nickname
    err := c.Connection.Call("ChatServer.CreateUser",
nickname, &message)
    if err != nil {
        wg.Done()
    }
    fmt.Print(message)
}
```

Client - Methods (5)

```
func (c *Client) ConnectedUsersList() {
    var message string
    err :=
c.Connection.Call("ChatServer.ConnectedUsersList", true,
&message)
    if err != nil {
        wg.Done()
    }
    fmt.Print(message)
}
```

Client - Methods (6)

```
func (c *Client) SendMessage(str string) {
    var message string
    text := Message{
        Nickname: c.nickname,
        Text: str,
    }
    err := c.Connection.Call("ChatServer.SendMessage",
text, &message)
    if err != nil {
        wg.Done()
    }
    fmt.Print(message)
}
```

Client - Methods (7)

```
func (c *Client) Quit() {
    var message string
    err := c.Connection.Call("ChatServer.Quit",
c.nickname, &message)
    if err != nil {
        wg.Done()
    }
    fmt.Print(message)
}
```

Demostração com vídeo

Dúvidas?