

Relatório de trabalho prático

Relatório do Trabalho Pratico CD

Luís Martins

Aluno nº 16980

Carlos RIbeiro

Aluno nº 16986

Trabalho realizado sob a orientação de:

Michael Matias

**Comunicação de Dados**

**Licenciatura em Engenharia de Sistemas Informáticos**

Barcelos, Maio de 2020

Índice

[1 Introdução 1](#_Toc41861574)

[1.1 Descrição do Tema a Desenvolver 1](#_Toc41861575)

[1.2 Projetos 1](#_Toc41861576)

[2 Desenvolvimento 2](#_Toc41861577)

[2.1. Descrição da Estrutura do Programa 2](#_Toc41861578)

[3 Conclusão 9](#_Toc41861579)

[Bibliografia 11](#_Toc41861580)

Lista de Figuras

Figura 1: Método que regista novo utilizador e coloca esse utilizador no chat

Figura 2: Exemplo do servidor a receber dados relacionados ao novo utilizador

Figura 3: Processo do Log-in

Figura 4: Exemplo de uma “thread” a ser incializada usando o método “ChatNovo”

Figura 5: Parte do método “ChatNovo” em que vemos o servidor a receber uma mensagem

Figura 6: Todas as Listas usadas

Figura 7: Client a tentar conectar-se ao servidor com ip “192.168.43.159”

Figura 8: Client a enviar os parâmetros de registo para o servidor

Figura 9: Criação de uma thread que recebe mensagens no “background”

Figura 10: Loop que se encarrega de enviar mensagens para o servidor

# Introdução

## 1.1 Descrição do Tema a Desenvolver

Face ao mais recente episódio de Covid-19, foi decretado estado de emergência com várias medidas, entre elas, regimes de teletrabalho e e-learning.

Neste projeto visamos criar uma aplicação que permite que um utilizador se conecte a um servidor para falar com outros utilizadores sendo eles, alunos ou professores, e que também permita a inserção de comandos para mudar de salas como também para outras coisas.

### 1.2 Projetos

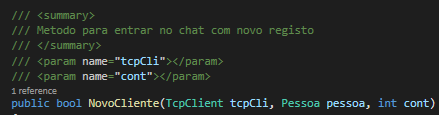
* **Server –** Projeto que dá “host” ao servidor que vai receber clientes e também controla as ações do mesmo.
* **Client –** Projeto que cria cliente que vai conectar ao servidor e pode enviar mensagens para o mesmo, como também receber mensagens ao mesmo tempo usando uma “thread”.
* **Objetos –** Biblioteca que define um objeto “Pessoa” que descreve uma pessoa normal.

# Desenvolvimento

### 2.1. Descrição da Estrutura do Programa

**Servidor:**

A classe main do projeto “Servidor” serve primariamente para receber a informação que o cliente envia em relação à sua intenção (fazer registo de utilizador novo ou fazer log in de utilizador existente), e depois invocar o método da classe “ControlaCliente” respetivo:



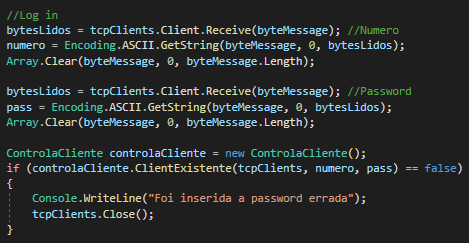
*Figura 1 - Método que regista novo utilizador e coloca esse utilizador no chat*

Vemos acima o método utilizado no “cenário 1” ou seja, quando o Client quer registar um utilizador novo, neste caso o servidor vai receber os dados relacionados ao utilizador novo usando métodos “.Receive()” e com estes dados vai criar um novo objeto do tipo “Pessoa” que também serve de parâmetro para o método “NovoCliente”



*Figura 2 - Exemplo do servidor a receber dados relacionados ao novo utilizador (neste caso o nome)*

No “cenário 2” vemos o cliente a tentar fazer “Log-in” para um utilizador já existente, para realizar esta ação, o servidor recebe um número e uma password do Client, depois, como no primeiro cenário, vai ser invocado outro método que permite ao cliente fazer Log-in caso o numero que foi inserido se encontre na lista de utilizadores registados:



*Figura 3 - Processo do Log-in*

**Controla Cliente:**

Sempre que existe uma nova conexão, é criada uma instância da classe “ControlaCliente”, como o nome diz esta classe é dedicada a controlar cada instância de Client que é criada na ocorrência em que haja uma nova conexão no servidor.

Dependendo do método que foi invocado na parte final da classe main do Servidor, vai ser criado uma Thread para que o Client consiga enviar mensagens constantemente.



*Figura 4 - Exemplo de uma “thread” a ser incializada usando o método “ChatNovo”*

O método “ChatNovo” é o mesmo que vai permitir ao servidor receber mensagens tal como enviar essa mensagem para os outros Clients que estão no mesmo lobby usando um método chamado “Broadcast”, e, usando um “switch - case” conseguimos também identificar se foi chamado um comando por um dos Clients

Uma imagem com captura de ecrã, sentado, mesa, preto

Descrição gerada automaticamente

*Figura 5 - Parte do método “ChatNovo” em que vemos o servidor a receber uma mensagem*

Nesta classe são usadas listas para servirem de lobbies como também uma lista para anotar todos os utilizadores registados, ou seja cada curso tem um lobby daí cada curso tem uma lista de composta por “TcpClient” e outra por objetos “Pessoa” em que se vão ser adicionados e removidas TcpClients e Pessoas à medida que Clients mudem de lobbies ou se desconectam do servidor:

Uma imagem com captura de ecrã

Descrição gerada automaticamente

*Figura 6 - Todas as Listas usadas*

**Client:**

Por últimos esta classe descreve uma instância de um Client a tentar se conectar com um servidor:

Uma imagem com captura de ecrã, ecrã, homem, telefone

Descrição gerada automaticamente

*Figura 7 - Client a tentar conectar-se ao servidor com ip “192.168.43.159”*

Após se conectar vai escolher uma de duas opções, registar um novo utilizador (opção número 1) ou fazer Log – in (opção número 2).

Se escolher a opção 1 vai ser apresentado com vários campos para preencher dados sobre o utilizador novo, sendo esses dados enviados para o servidor ao mesmo tempo usando o método “.Send()”:

Uma imagem com laranja, preto, propriedade, jogador

Descrição gerada automaticamente

*Figura 8 - Client a enviar os parâmetros de registo para o servidor*

Depois esta ação ser executada é criada um Thread nova que vai correr um método chamado “RecebeMsg” que se vai encarregar de receber mensagens ao mesmo tempo que o Client envia mensagens:

Uma imagem com ecrã, propriedade

Descrição gerada automaticamente

*Figura 9 - Criação de uma thread que recebe mensagens no “background”*

Depois da Thread ser criada é feito um loop que se vai encarregar de enviar mensagens para o servidor, apenas parando quando a mensagem for “/sair”:

Uma imagem com ecrã, monitor, portátil, sentado

Descrição gerada automaticamente

*Figura 10 - Loop que se encarrega de enviar mensagens para o servidor*

# Conclusão

Em conclusão, com a realização deste trabalho aumentamos o nosso conhecimento da linguagem C#, como também as varias formas que esta linguagem pode ser usada em situações diferentes, neste caso, no ato de comunicação entre utilizadores.

**Repositório GitHub:** <https://github.com/BasilNight/TP1-CD-16980-16986>

# Bibliografia

* Código disponibilizado no Moodle
* https://stackoverflow.com