
Relatório

Tecnologias de Distribuição e Integração

# Internet Relay Chat (.NET Remoting)

Mestrado Integrado em Engenharia Informática e Computação

Abril 2020

Amadeu Pereira up201605646@fe.up.pt

Nuno Lopes up201605337@fe.up.pt

# Conteúdo

# 1 Introdução

Este trabalho foi realizado no âmbito da unidade curricular de Tecnologias de Distribuição e Integração pertencente ao Mestrado Integrado em Engenharia Informática e Computação. O projeto consiste na elaboração, utilizando a linguagem C#, de um sistema de IRC (*Internet Relay Chat*) baseado em .NET Remoting e com uma interface gráfica (GUI) nos clientes, usando Gtk#. Este sistema permite que utilizadores registados num servidor comum (aplicação servidor) consigam comunicar entre si através da aplicação cliente, que é a mesma para cada um dos utilizadores.

No decorrer deste relatório será descrito ao pormenor a arquitetura e funcionalidades bem como será mostrado o resultado final do projeto com capturas das janelas.

# 2 Arquitetura

Este projeto teve como base uma arquitetura servidor-cliente que permite a comunicação entre estas duas entidades, tanto através de *Remote Objects* como pela subscrição a *events*.

## 2.1 Common

Esta biblioteca (.dll) contém todas as informações que são comuns ao servidor e ao cliente.

Nela encontram-se as interfaces implementadas pelos *Remote Objects* (falado em baixo) bem como as estruturas de dados (*Serializable*) que são partilhadas entre o cliente e o servidor, nomeadamente a classe *User* e a classe *Message*.

## 2.2 Server

Esta aplicação encontra-se dividida em dois módulos. O módulo *Database*, responsável pela base de dados do servidor e o módulo *Services* onde se encontram todos os *Remote Objects*, cada um representando um serviço oferecido pelo servidor.

No primeiro módulo encontra-se a classe *ServerUser* que é uma subclasse de *User* (definida no Common). Esta contém informações mais específicas ao servidor sobre um utilizador, como por exemplo o facto de ele se encontrar *online* ou não. Neste módulo também se encontra a *BDManager* que está responsável por fazer a gestão de todos os utilizadores registados no sistema bem como garantir a sua persistência. Para garantir esta persistência optamos por fazer algo simples, em que todos os utilizadores se encontram guardados num ficheiro XML (*users.xml)*.

No segundo módulo estão implementados dois *Remote Objects*: *Authentication* e *ChatManager*.

O primeiro permite que o utilizador se consiga registar, fazer *login*, *logout* e

pedir todos os utilizadores *online* naquele momento. Também se encontra aqui implementado um evento (*OnlineChanged*) ao qual os *users* dão *subscribe* para serem notificados sempre que algum utilizador faz *login* ou *logout* da aplicação.

O segundo *Remote Object* (*ChatManager*) apenas mantém informação sobre todos os *chats* ativos entre utilizadores. A sua utilidade é referida na secção 2.3.

## 2.3 Client

Esta aplicação encontra-se dividida em 4 módulos.

O módulo *ServerServices* implementa abstrações (*AuthServer* e *ChatManagerServer*) para utilizar os serviços fornecidos pelo servidor.

O módulo *Utils* contém a classe *RemoteNew* que é um mecanismo para instanciar um Objeto Remoto através da sua interface, utilizando o ficheiro de configurações.

O módulo *Services*, tal como no servidor, é onde se encontram todos os *Remote Objects*, cada um representando um serviço, oferecidos pelo cliente.

No ficheiro *Request.cs* encontram-se dois Objetos Remotos implementados: *Request* e *RequestCallback*. Estes permitem que um cliente mande um *request* de um *chat* para outro cliente e, ao fazer tal pedido, tem que lhe enviar também o seu *RequestCallback* para que o utilizador que recebeu o *request* possa informar quem pediu sobre a sua decisão: se decidiu aceitar (*Accepted*) ou recusar (*Refused*) tal pedido.

O *ChatService* permite a troca de mensagens diretamente entre clientes e também permite que um cliente informe outro que saiu de um determinado *chat*.

O *GroupRequest* permite que um utilizador convide outro para um *chat* que já esteja aberto com outras pessoas e também permite que o cliente que está a ser convidado possa informar todos os outros clientes que se encontram naquele chat sobre a sua decisão. O cliente sabe quais são os utilizadores que se encontram num dado chat utilizando o serviço *ChatManager* fornecido pelo servidor. Neste caso não seria possível utilizar um *callback*, como no *Request*

falado anteriormente, porque a resposta não é só para uma pessoa mas sim para múltiplas.

No último módulo, *Windows*, encontra-se a implementação da parte gráfica de cada janela bem como do *WindowManager* que facilita o controlo das janelas e ajuda na troca de informação entre elas.

# 3 Funcionalidades

## 3.1 Registo

No inicio da aplicação, o utilizador pode escolher registar-se no caso de não possuir conta. Para efetuar o registo, o utilizador tem de definir um *username*, o seu nome verdadeiro e uma *password*. No caso de o *username* já existir no servidor é mostrada uma mensagem de erro ao utilizador.

## 3.2 Autenticação

Os utilizadores que possuem uma conta podem autenticar-se no sistema, introduzindo o *username* e a respetiva *password*. Se o *username* não existir ou a *password* introduzida não corresponder ao *username* aparece um erro a alertar o utilizador, permitindo assim a correção e consequentemente um *login* com sucesso. Após um *login* com sucesso, um ecrã novo aparece onde o utilizador consegue ver os utilizadores que estão *online* e os pedidos de conversação.

## 3.3 Lista de utilizadores *online*

Depois de o utilizador se autenticar aparece uma janela onde é possível ver uma lista com os utilizadores que estão *online*, sendo possível para cada um deles enviar um pedido de conversação. Esta lista é atualizada em tempo real, ou seja sempre que um utilizador terminar a sessão, vai ser removido da lista das pessoas *online* dos outros utilizadores.

## 3.4 Convidar utilizadores a participar num *chat*

Sempre que um utilizador quer falar com outro envia um pedido de conversação. Este pedido pode ser aceite ou recusado pelo outro utilizador, sendo que se for aceite surge uma janela de *chat* para ambos os utilizadores. No caso do pedido ser rejeitado, é removido da lista de pedidos e o utilizador que fez o pedido

é notificado.

## 3.5 Conversação entre dois utilizadores

Após um pedido de conversação ser aceite uma janela de *chat* surge em ambos os utilizadores. Nesta janela os utilizadores podem trocar mensagens entre si.

## 3.6 Conversação entre múltiplos utilizadores

Após a iniciação da conversação entre dois utilizadores é possível adicionar mais utilizadores presentes na lista *online* que se encontra no canto inferior direito da janela de *chat*. Um pedido de *group chat* é enviado ao novo utilizador e no caso de ser aceite este aparece como utilizador presente em todos os *chats* dos outros utilizadores que já estavam presentes. De salientar que quando um pedido de *group chat* é enviado a um utilizador, nenhum outro consegue enviar um pedido ao mesmo utilizador até este aceitar ou recusar o primeiro pedido recebido.

## 3.7 Permanência de *group chat* após saída de um participante

Quando um utilizador sai de um *group chat* o mesmo mantém-se ativo entre os restantes participantes.

## 3.8 Fechar *chat* sem participantes

Quando um *chat* fica apenas com um participante a janela desse *chat* é fechada uma vez que o utilizador não está a comunicar com mais ninguém.

## 3.9 Possibilidade de haver múltiplos *chats* ao mesmo tempo

Não existe nenhuma limitação para o número de *chats* que um utilizador pode ter ativos, ou seja é possível um utilizador ter várias conversações com o mesmo utilizador se assim o desejar.

## 3.10 Enviar ficheiros

Para além de ser possível enviar mensagens de texto também é possível enviar qualquer tipo de ficheiros. Para isso existe um botão com um *icon* de pasta que ao ser carregado aparece uma janela para o utilizador selecionar o ficheiro a ser enviado. Depois de o ficheiro ser enviado, qualquer utilizador do *chat* pode carregar nele e consequentemente fazer a sua transferência, sendo que antes de ser efetuada é necessário selecionar o local onde o ficheiro vai ser guardado.

## 3.11 Terminar sessão

Em qualquer momento, um utilizador pode terminar a sua sessão, carregando no botão *Logout* ou fechando todas as janelas ativas. Quando isto acontece todos os outros utilizadores são notificados e consequentemente é removido de todos os *chats* em que estiver presente.

# 4 Testes Funcionais

Numa primeira iteração do projeto foram feitos testes referentes ao serviço de autenticação do servidor. Estes testes foram necessários de forma a garantir que um utilizador não consegue fazer *login* mais que uma vez seguida, que precisa de ter as credenciais corretas para conseguir se conseguir autenticar e que nenhum cliente se pode registar na aplicação com um *username* já utilizado.

A próxima fase de testes foi a verificação da comunicação entre o cliente e o servidor, nomeadamente na receção de todos os utilizadores *logged in* e na subscrição do evento do servidor para ser notificado sempre que há alterações nos *users* autenticados, atualizando consecutivamente, a sua interface gráfica.

Seguidamente, foi testado o pedido de mensagem de um utilizador para outro e a capacidade de resposta por parte deste. Nesta altura também foi necessário testar o facto de um utilizador, tendo feito um pedido, não conseguir fazer outra vez esse pedido antes de o primeiro ter sido respondido. Por outro lado, também se testou a impossibilidade de nós fazermos um pedido quando alguém já nos fez. Estes testes foram sempre acompanhados pela verificação de que a interface gráfica era atualizada conforme estas restrições.

Ao mesmo tempo dos testes anteriores tivemos que nos certificar que após um pedido ser aceite/recusado ele era removido de ambos os envolventes, tanto a nível interno como a nível visual.

Após aceitar um pedido de conversação de outro utilizador para um *chat*, foi preciso verificar que em ambos os intervenientes era criada uma nova janela que permitisse a comunicação entre os dois. Ao nível do servidor, também tivemos que nos certificar que, quando uma conversação é criada, o *ChatManager* é atualizado com a informação que estes dois clientes estão a utilizar o *chat* em específico. Em relação a este tópico, tivemos também que testar que quando um utilizar sai de um *chat* com outra pessoa ou simplesmente faz *logout* a janela correspondente no outro usuário era fechada.

Em relação aos *chats* com mais que duas pessoas, tivemos que verificar que, a nível visual era possível ver todos os *users online* e que esta lista era atualizada

em tempo real sempre que havia um novo *login*/*logout*. Também foi verificada a incapacidade de vários utilizadores do mesmo *chat* convidarem a mesma pessoa e que a resposta a tal pedido era enviada a todos os participantes da conversação.

Tanto ao nível dos *chats* privados como dos *group chats* foi necessário verificar se as mensagens eram enviadas corretamente para todos os intervenientes.

# 5 Modo de Funcionamento

Instruções para correr tanto o servidor como o cliente podem ser encontradas na secção 6. Primeiramente é necessário executar o servidor e só depois é que se podem executar os clientes. Assim que um cliente é executado aparece uma janela onde o utilizador se pode autenticar ou, no caso de não ter conta, efetuar o registo carregando no botão ”Register”. Após o registo aparece novamente a janela para efetuar a autenticação.

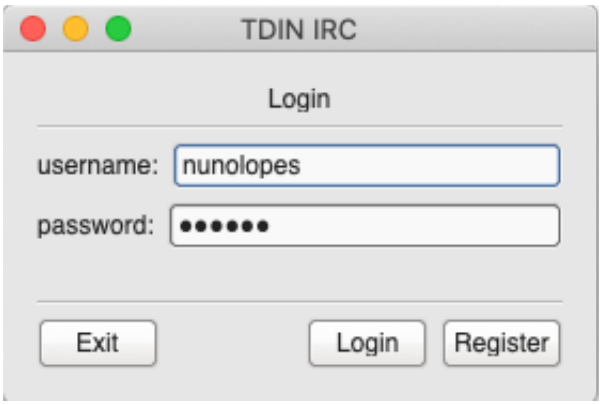


Figura 1: Ecrã de autenticação

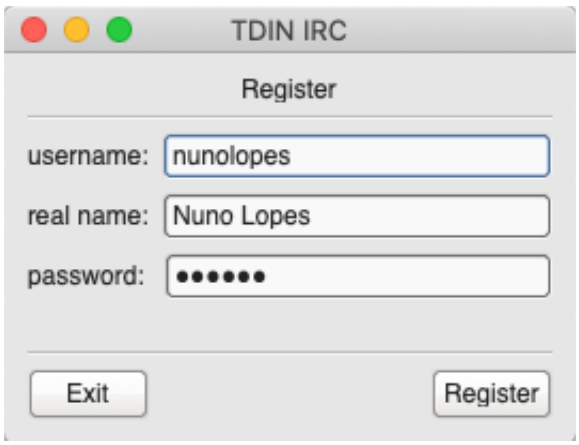


Figura 2: Ecrã de registo

Depois de um utilizador se autenticar surge uma janela onde o utilizador pode ver os outros utilizadores *online* bem como os pedidos para iniciar conversações enviados pelos utilizadores. Estes pedidos de conversação podem ser para *chats* entre dois utilizadores ou para *chats* de grupos, com vários utilizadores. Quando um pedido é enviado ou recebido por um utilizador o botão para fazer um pedido

novo ao mesmo utilizador fica inativo até que o primeiro pedido seja aceite ou recusado. É possível criar quantos *chats* quisermos para cada um dos utilizadores. Tanto os pedidos de conversação como a lista de utilizadores *online* são atualizados em tempo real, por isso sempre que um utilizador terminar sessão ou um novo se autenticar a lista é atualizada.

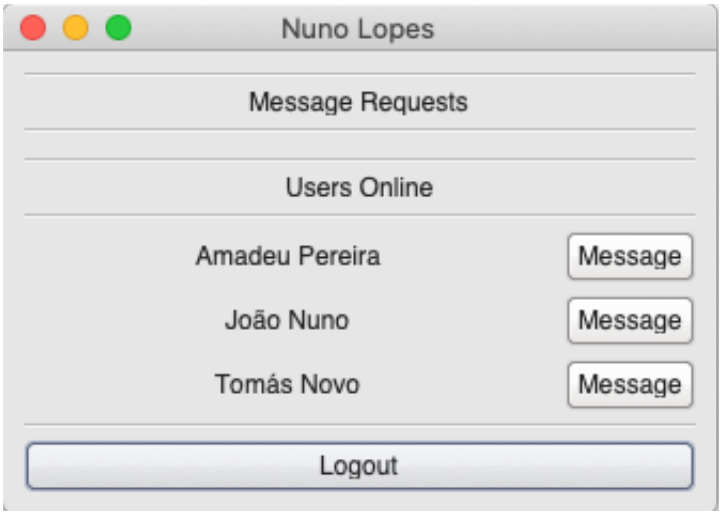


Figura 3: Janela de utilizadores e pedidos de conversação vista da perspetiva do utilizador 'Nuno Lopes'

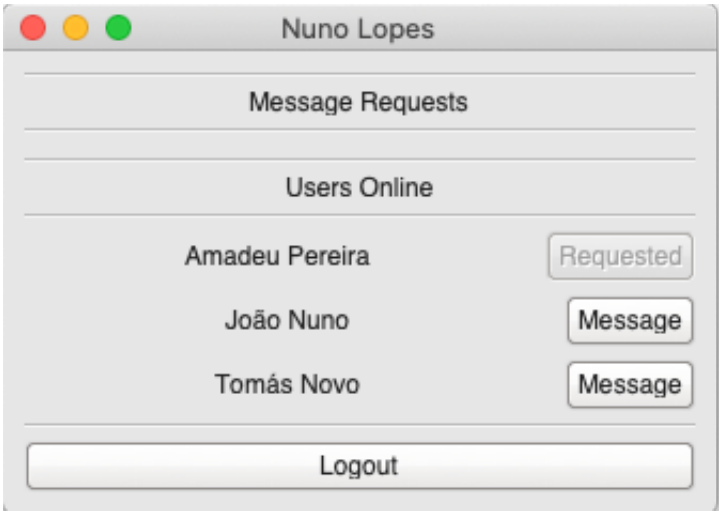


Figura 4: Pedido de conversação enviado por 'Nuno Lopes' ao 'Amadeu Pereira'

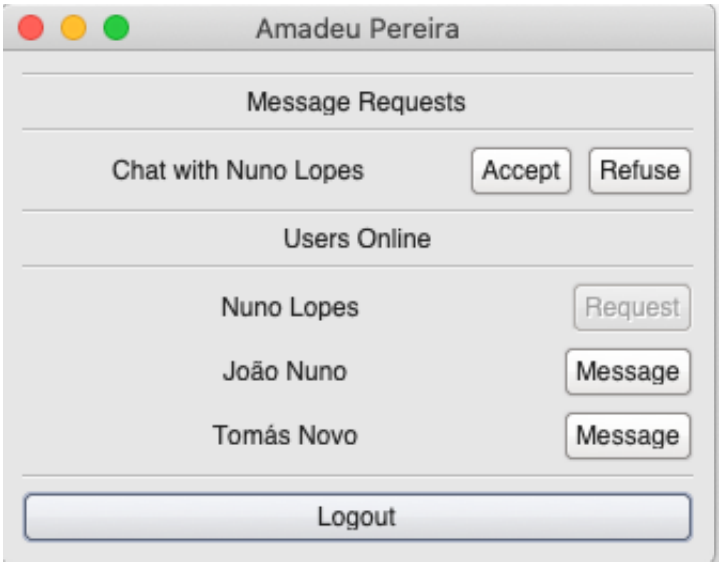


Figura 5: Pedido de conversação recebido no 'Amadeu Pereira' enviado por 'Nuno Lopes'

Quando um pedido de conversação é aceite aparece em cada utilizador uma janela de *chat* para os utilizadores comunicarem privadamente entre si.

Nesta janela é possível ver uma caixa de texto onde são mostradas as mensagens trocadas, uma caixa de texto onde é possível ver os utilizadores presentes no *chat*, uma caixa de introdução de texto para escrever as mensagens a enviar, um botão para enviar ficheiros e uma lista de utilizadores que estão *online* mas que não estão no *chat* em específico.

Todos as componentes deste ecrã são atualizadas em tempo real.

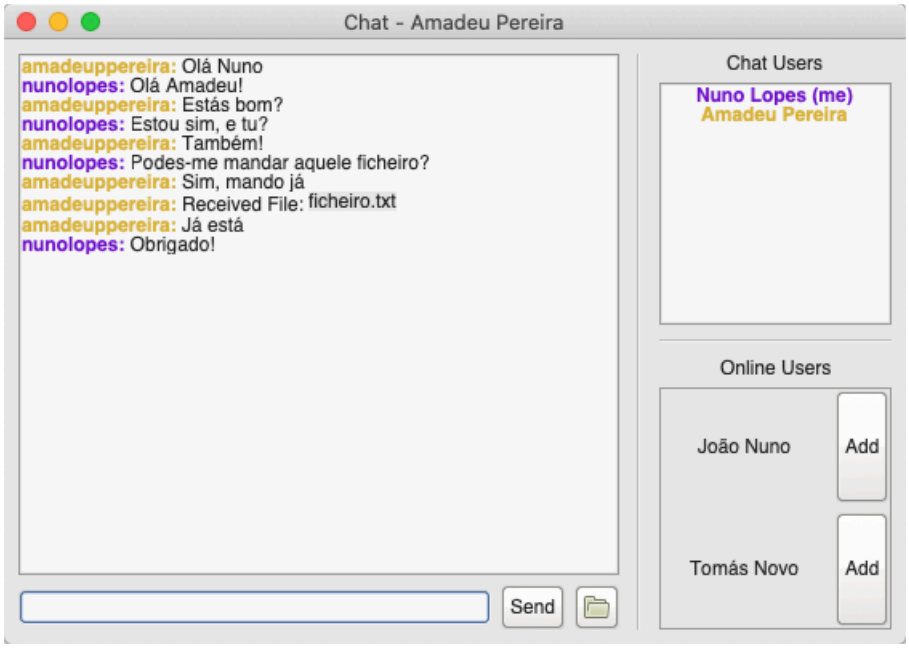


Figura 6: Ecrã de conversação com 'Amadeu' na perspetiva do 'Nuno'

Quando um utilizador carrega no botão para enviar um ficheiro aparece uma janela onde é possível escolher o ficheiro a ser enviado.

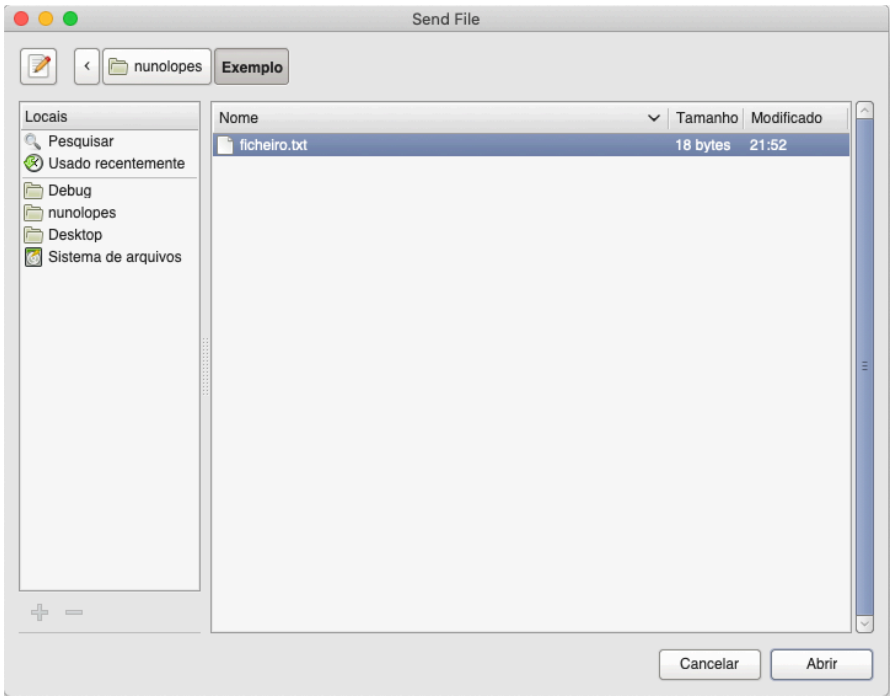


Figura 7: Ecrã para selecionar ficheiro a ser enviado

Quando um utilizador quer guardar um ficheiro basta carregar nele dentro da caixa de texto que contém as mensagens trocadas e uma janela aparece para

selecionar o local do seu computador onde quer guardar o ficheiro.

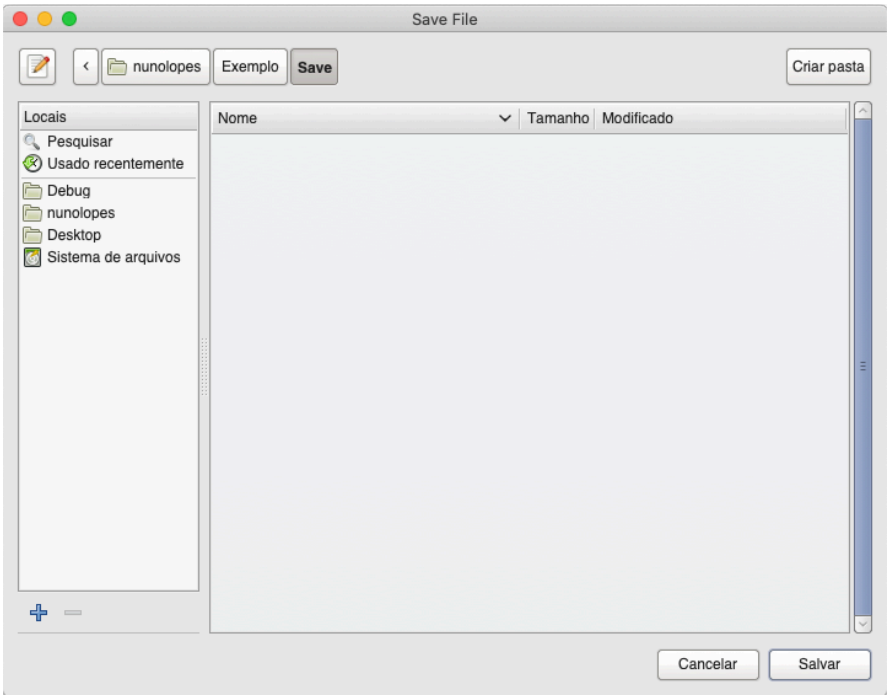


Figura 8: Ecrã para selecionar local onde guardar o ficheiro enviado

Para cada utilizador *online* que não está presente no *chat* é possível enviar um pedido de conversação, sendo que se este for aceite o *chat* passa de uma conversação entre dois utilizadores para uma conversação entre vários utilizadores.

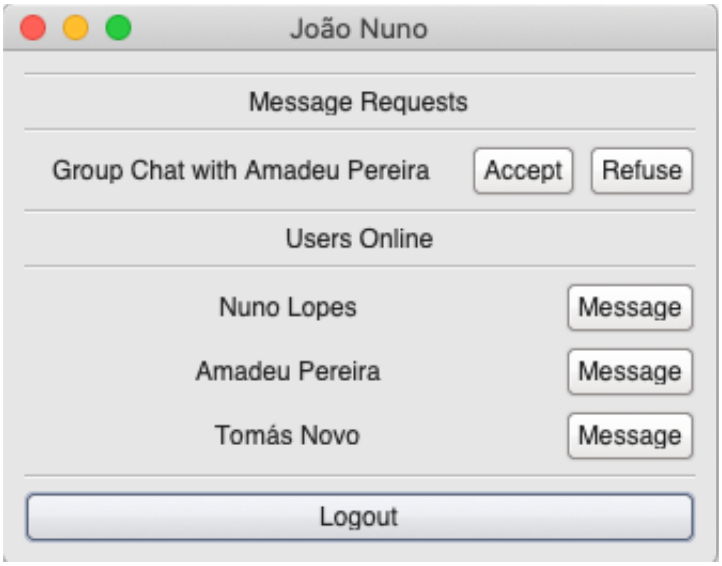


Figura 9: Pedido de conversação em grupo enviado pelo 'Amadeu' ao 'João'

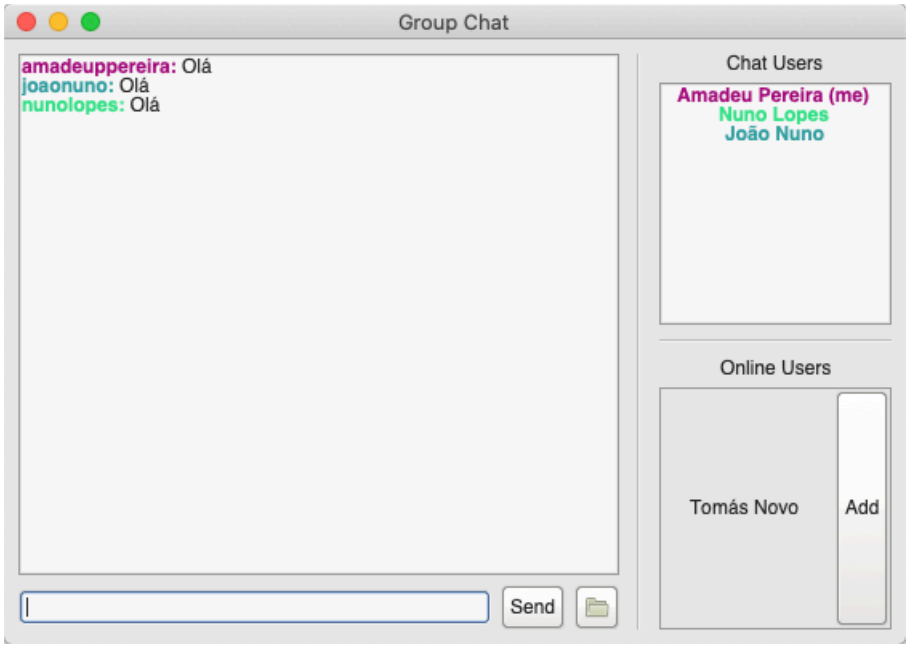


Figura 10: Ecrã de conversação em grupo na perspetiva do 'Amadeu'

# 6 Instruções

Antes de executar os programas é preciso ter a biblioteca de Gtk# instalada no computador, bem como as bibliotecas de .NET que são instaladas com o Visual Studio. Para instalar o Gtk# basta ir ao *link* que está presente em (2). No caso do Windows basta instalar usando o ficheiro *gtk-sharp-2.12.45.msi* que está dentro da pasta *Compiled*.

Existem dois executáveis para correr projeto, o *Server.exe* dentro da pasta *Compiled/Server* que inicia uma instância do servidor e o executável *Client.exe* dentro da pasta *Compiled/Client* que inicia uma instância de um cliente. Primeiro tem de se executar apenas uma instância do servidor e depois podem-se executar múltiplas instâncias do cliente. Também existe a possibilidade de se importar o projeto com o Visual Studio, podendo assim fazer uma nova compilação do código e correr os programas sem utilizar um executável.

Uma vez que o programa foi desenvolvido em ambiente de MAC OS existe a possibilidade de haver erros de dependências por não estarem presentes noutros sistemas operativos.

# 7 Conclusão

Com este projeto o grupo reconhece a importância de uma boa arquitetura no processo de desenvolvimento para se obter um bom resultado final. Foram adquiridas competências básicas do domnínio de .NET Remoting, assim como da linguagem C#.

A aplicação desenvolvida permite que diferentes utilizadores comuniquem entre si, recorrendo a conversações diretas ou através de *group chats* onde pode existir a troca de mensagens entre múltiplos utilizadores em simultâneo. O servidor persiste a informação dos utilizadores para que não seja necessário fazer o registo de utilizadores sempre que se iniciar o programa.

Sendo assim, podemos concluir que os requisitos do projeto foram todos cumpridos e para além disso ainda foram implementadas algumas funcionalidades adicionais.

# 8 Recursos

## 8.1 Bibliografia

[1] Página da UC de Tecnologias de Distribuição e Integração.
Acedido em março de 2020.
https://paginas.fe.up.pt/~apm/TDIN/

[2] C# documentation.
Acedido em abril de 2020.
https://docs.microsoft.com/en-us/dotnet/csharp/

[3] .NET Framework documentation.
Acedido em abril de 2020.
https://docs.microsoft.com/en-us/dotnet/framework/

## 8.2 Software utilizado

[1] Visual Studio Community 2019 for Mac.
Versão 8.5.3 (build 16)
https://visualstudio.microsoft.com/vs/

[2] GTK# for .NET.
Installer for running Gtk#-based applications on Microsoft .NET.
https://www.mono-project.com/download/stable/#download-win