Documentação para
Aplicação Distribuída de Bate-Papo

João Pedro Lopes Murtinho - DRE: 116040336

O sistema permitirá que múltiplos clientes se conectem a um servidor remoto para troca de mensagens diretas com outros usuários.

Ao se conectar, o usuário terá acesso a uma lista contendo todos outros usuários disponíveis para troca de mensagens. Ele poderá então decidir se tornar disponível para envio e recebimento de mensagens através de um comando de "*login"*.

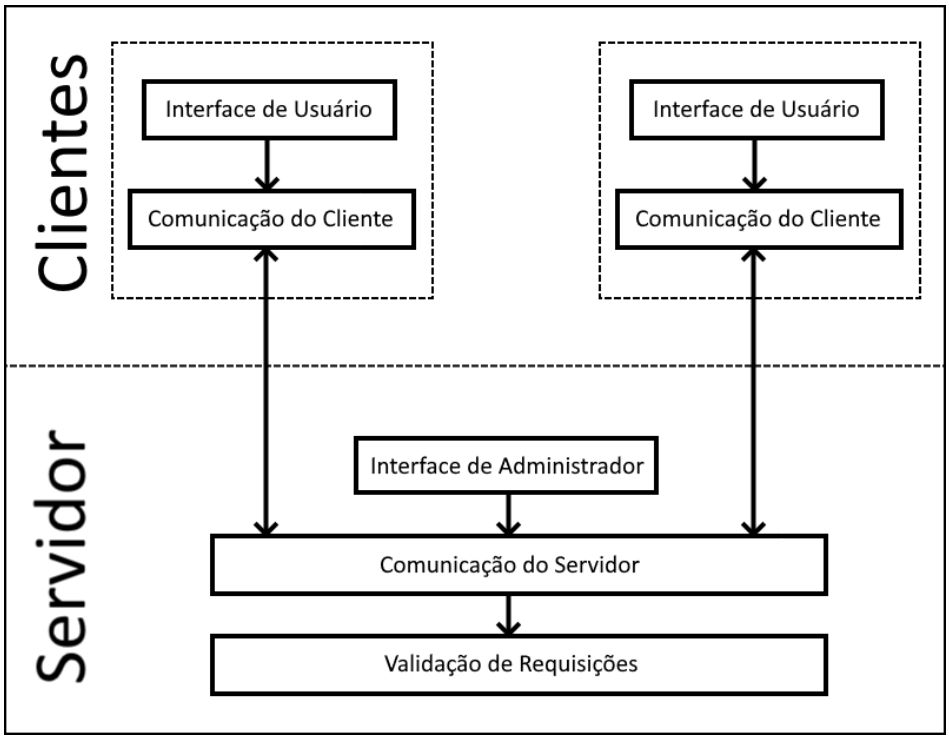
Após isso, o usuário será capaz de inserir a qualquer momento em seu console um comando no formato: "*@user message"* para enviar uma mensagem a outro usuário que esteja listado como disponível no sistema. Essas mensagens não serão públicas para todos os usuários, sendo somente visíveis ao destinatário e ao administrador no lado do servidor.

Outros comandos incluem "*userlist"* e "*logout"*. O primeiro mostra ao usuário a lista atualizada de usuários logados no servidor, enquanto o segundo o torna indisponível para troca de mensagens, o deslogando do servidor e tirando ele da lista pública de usuários disponíveis.

Uma interface semelhante à dos usuários também é oferecida no lado do servidor para uso de um administrador. Ele poderá visualizar a lista de clientes conectados (incluindo aqueles não logados), enviar mensagens diretas a usuários, desligar o servidor, realizar o *broadcast* de uma mensagem para todos os clientes e deslogar usuários através do comando *"kick"*.

Arquitetura de Software e Sistema:

O estilo arquitetural de software será desenvolvido em **camadas**, conforme descrito no diagrama abaixo:



Usuários e o administrador irão interagir com a aplicação através de seus respectivos componentes de interface na camada mais superior do sistema. Estes componentes irão enviar à camada de comunicação diretamente abaixo os dados que serão transmitidos já com sua devida formatação respeitando o protocolo da camada de aplicação. A exceção dessa regra ocorre no lado do servidor, no fluxo de execução de tratamento de requisição de clientes, onde o processamento e tratamento de requisições ocorrem dentro da camada de comunicação do servidor.

Assim, os componentes da camada de comunicação ficarão encarregados somente de: estabelecer a conexão, realizar a conversão dos dados recebidos para a stream de bytes que será encaminhada via rede, e de detectar alguma eventual desconexão e tratá-la de acordo.

Ao alcançar a camada de validação de requisições no servidor, os dados recebidos são então interpretados em possíveis diferentes comandos. No caso de requisição de um comando inválido, esta camada irá somente retornar ao cliente uma mensagem de erro; Caso contrário retornará à camada superior de comunicação do servidor com a resposta adequada à requisição: Podendo ser uma mensagem a ser enviada a outro cliente ou um retorno a um comando (como o de listar usuários disponíveis) ao cliente que o requisitou.

A implementação das camadas se dará por meio de diferentes classes representando cada componente do sistema, sempre respeitando a regra de que componentes de camadas superiores (como definido no modelo) somente farão requisições à componentes da camada diretamente inferior à dele. Estas camadas também foram separadas em seus próprios arquivos devido à extensão do código.

O sistema, como descrito no diagrama acima, possuirá uma **arquitetura centralizada**. Utilizando-se de um servidor mediador para as requisições dos diversos clientes conectados a ele. Cada um desses clientes terá um conjunto idêntico de componentes do sistema, responsáveis por receber e enviar mensagens ao servidor, além de servir como interface para o usuário. No servidor central remoto, estarão armazenados os componentes responsáveis por realizar a validação e tratamento de requisições dos clientes conectados, além de prover uma interface para receber comandos locais de administrador e se comunicar com os usuários via rede.

Protocolo da Camada de Aplicação:

Toda mensagem trocada entre clientes e servidor será uma string codificada em bytes para envio. Essa string sempre possuirá um header de 5 caracteres, sendo os 4 primeiros dígitos representando o tamanho da mensagem a ser enviada (para um máximo de 9999 caracteres), seguida pelo caractere ‘:’. Com essa informação, o componente destinatário da mensagem, ao decodificar os bytes recebidos do header, sempre saberá o quanto deverá ser lido para compor a mensagem enviada por inteiro para tratamento.

Exemplos de mensagens antes de ser codificada para envio:

- 0014:@otheruser Oi.
- 0071:broadcast AVISO: O servidor irá desligar em 30 minutos para manutenção!

A seguir estão listadas as diferentes mensagens que serão tratadas de forma distinta pela camada de aplicação. Note que o header foi omitido na descrição de sintaxe de cada uma delas, mesmo sempre estando presente.

Mensagens Cliente-Servidor:

- **Mensagens de Login:** Mensagem enviada por um cliente ao servidor logo após estabelecida a conexão com o mesmo.
    - **Sintaxe:** login *username*

      Com *username* sendo o nome desejado que aparecerá para outros clientes conectados na lista de usuários disponíveis do servidor para envio de mensagens.
- **Mensagens de Cliente:** Mensagem enviada por um cliente, endereçada a outro cliente, ao servidor.
    - **Sintaxe:** @*user message*

      Com *user* sendo o usuário para qual a mensagem *message* será encaminhada pelo servidor.
- **Mensagens de Comando:** Mensagem enviada por um cliente ao servidor, requisitando alguma funcionalidade provida por ele (como por exemplo a listagem com todos os outros usuários disponíveis para mensagem).
    - **Sintaxe:** *command*

      Mensagens enviadas ao servidor sem um prefixo definido serão interpretadas como um comando pelo mesmo.

Mensagens Servidor-Cliente:

- **Mensagens de Servidor:** Mensagem enviada pelo servidor a um cliente para exibir ao usuário. Esta mensagem alcança o cliente já formatada pronta para ser mostrada na interface, podendo ser o retorno de um comando passado ao servidor, uma mensagem de outro cliente ou uma mensagem de erro.
    - **Sintaxe:** *message*

      A aplicação do lado do cliente não fica a cargo de interpretar mensagens recebidas do servidor que não tenham prefixo definido, somente de passá-la ao usuário diretamente por meio da interface.
- **Mensagens de Encerramento de Conexão:** Mensagem enviada pelo servidor a um cliente indicando a ele que sua conexão foi encerrada pelo servidor. Isso pode ser causado pelo fato do administrador ter desligado o servidor ou por ele ter manualmente removido o usuário em questão através do comando *“kick”*. Ao receber essa mensagem, o cliente irá exibir uma mensagem avisando ao usuário e irá terminar o processo da aplicação local.
    - **Sintaxe:** $QUIT

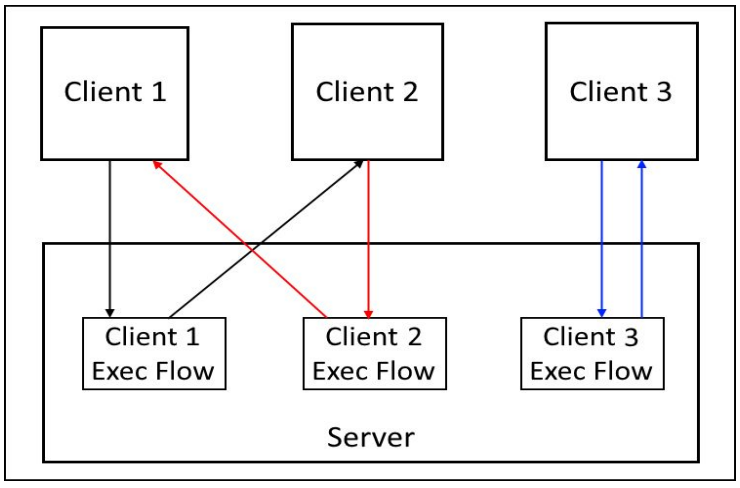
Regras:

Após o servidor ser inicializado, clientes ao se conectarem receberão uma mensagem contendo uma instrução sobre o comando de '*help'* que listará todos os comandos disponíveis. Usuários poderão então visualizar a lista pública de usuários logados no servidor ou realizar seu próprio login. Depois de logado, ele será incluído na lista pública de usuários disponíveis e terá acesso a funcionalidade de chat, podendo receber e enviar mensagens a outros usuários.

O servidor separará cada cliente conectado em um diferente fluxo de execução, cada um tratando requisições específicas de seu respectivo cliente. Esses diferentes fluxos terão acesso a uma variável compartilhada que conterá a lista (implementada como um dicionário) de usuários conectados disponíveis. Do lado dos clientes não haverá essa concorrência, ele estará pronto a qualquer momento para responder às duas entradas através função *select()*:

- Ler do usuário e enviar os dados lidos para seu fluxo de execução correspondente do lado do servidor.
- Receber dados através de seu socket. Estes dados podem ter se originado de fluxos correspondentes a outros clientes (no caso de mensagens diretas) ou não (no caso de retorno de comandos requisitados previamente por ele próprio).

Quando um cliente tenta enviar uma mensagem a outro, essa mensagem é enviada ao servidor, onde é validada (checado se o destinatário está presente na lista) e reformatada (para conter o nome de usuário do remetente) antes de ser enviada ao cliente final. Se a validação falhar, uma mensagem de erro é enviada àquele que originalmente tentou mandar a mensagem. Algo similar acontece no caso de uma requisição de comando ao servidor, em que ocorrerá uma validação, processamento e um retorno ao cliente com a resposta que ele aguarda ou uma mensagem de erro.



O diagrama ao lado ilustra o comportamento descrito acima. Com as setas pretas e vermelhas representando envios de mensagens entre dois usuários, enquanto as setas azuis representam um comando requisitado ao servidor e seu retorno ao próprio cliente.

Comandos de administrador são tratados de maneira análoga, com a diferença sendo que estes partirão do fluxo de execução principal do servidor, onde serão processados, validados e enviados para os clientes relevantes ao comando.