Aplicação de “bate-papo distribuído”

Atividade 1:

Usaremos a arquitetura em camadas, dividida segundo o modelo clássico de 3 camadas (aplicação, processamento e dados). O servidor intermediará a comunicação entre dois clientes, sendo responsável por *echoar* a mensagem do cliente A para o B e vice-versa.

1. **Descrição dos componentes do sistema**:

   Função de inicialização do servidor: responsável pela abertura do socket, bind da porta e host escolhidos e retorno do socket pronto para uso.

   Função de atendimento das requisições de entrada/conexão(servidor): responsável por retornar a um novo cliente (recém conectado) a lista de usuários conectados. A lista será retornada por *default*, não havendo necessidade do cliente requisitar essa funcionalidade.

   Função de atendimento das requisições de troca de mensagens(servidor): responsável por repassar a mensagem a enviada do cliente A para o cliente B (similar a um servidor de *echo* com remetente e destinatário diferentes).

   Função de criação do log do histórico de mensagens(servidor): responsável pela criação de um arquivo de log com todas as mensagens trocadas pelos usuários no dia em questão.

   Função de recuperação do histórico de mensagens(servidor): responsável pela exibição do histórico de mensagens trocados pelos usuários no dia em questão.

   Objeto/Classe Cliente: possui todos as informações (atributos) do cliente e todas suas funcionalidades (envio e recebimento de mensagens) implementadas em métodos.

   Função de inicialização da conexão cliente-servidor(cliente): responsável pela abertura do socket e a conexão, em si, com o servidor.

Função de envio de mensagens(cliente):
responsável pelo envio das mensagens do cliente para o servidor, que será responsável por enviar para o cliente “final”.

Função de recebimento de mensagens(cliente):
responsável pelo recebimento das mensagens do cliente final, que na verdade foi repassada pelo servidor.

Função de criação do log do histórico de mensagens(cliente): responsável pela criação de um arquivo de log com todas as mensagens trocadas pelo usuário A com o B.

Função de recuperação do histórico de mensagens(cliente): responsável pela exibição do histórico de mensagens trocados pelo usuário A com o B.

2. **Descrição dos conectores**: a maior parte da comunicação se dará por validações das informações (atributos) do objeto Cliente, ou seja, o nome e endereço do destinatário da mensagem serão atributos essenciais para coordenar corretamente o fluxo de mensagens e a recuperação dos respectivos históricos (não é desejável que um cliente obtenha o histórico de mensagens de outro). Conclui-se, então, que o objeto Cliente será um parâmetro de quase todas as funções.

3. **Dados trocados entre os componentes:** mensagens a serem trocadas entre os clientes, informações sobre o cliente (nome e IP), tipo de requisição ao servidor (envio de mensagem, recuperação do histórico e identificação dos usuários onlines).

Atividade 2:

A arquitetura é tipicamente centralizada, uma vez que todo processamento (troca de mensagens) deve passar pelo servidor. A maioria das funcionalidades são implementadas pelo servidor, com exceção da manutenção do histórico das conversas, que é implementado por ambos os lados, de maneiras ligeiramente distintas.

1. **Como os componentes serão agrupados**: foi sinalizado na Atividade 1 ao lado de cada componente, identificando, entre parênteses, a entidade que implementará essa funcionalidade (cliente ou servidor).

2. **Onde os componentes serão implementados**: idealmente, se dispuséssemos de recursos necessários, o servidor seria centralizado em uma única máquina (ou distribuído horizontalmente para paralelismo de processamento), enquanto que cada cliente seria obrigatoriamente um nó. Na prática, uma única máquina será usada, rodando uma instância de servidor e, no mínimo, duas instâncias de cliente (usuários finais do bate-papo).
3. **Como os componentes irão interagir**: se os componentes pertencerem à mesma entidade (cliente ou servidor) a comunicação será feita via passagem de parâmetro, caso contrário, será usada a troca de mensagens padrão da biblioteca *socket*. Um identificador crucial da operação principal (troca de mensagem do bate-papo) será o objeto cliente e seus atributos (nome e IP), coordenando o destinatário da mensagem. Além disso, será usada uma palavra-chave (melhor explicitada na atividade 3) indicando o tipo de operação requisitada (envio de mensagens, recuperação de histórico, entre outras).

Atividade 3:

O modelo do protocolo utilizado será semelhante ao funcionamento do HTTP. Será indicada, primeiramente, a operação a ser feita, seguida do destinatário e, por fim, o parâmetro da requisição (aqui consideraremos que há um único parâmetro, **no máximo**). Todos os componentes da mensagem serão separados por espaço. Exemplo onde Lucas envia uma mensagem para Pedro:

send_message Pedro Olá, tudo bem?
A mensagem será recebida da seguinte forma pelo destinatário Pedro:
Lucas: Olá, tudo bem?

1. Há, apenas, dois tipos de mensagem:
    a. send_message <destinatário> <mensagem>
    b. history <destinatário>
2. Descrita no enunciado da Atividade 3
3. Uma vez conectado ao servidor de bate-papo, o cliente receberá uma mensagem do servidor com os clientes conectados. A partir daí, qualquer cliente conectado poderá enviar uma mensagem a qualquer outro cliente, cujo envio será intermediado pelo servidor. Além disso, cada cliente poderá, a qualquer momento, enviar uma mensagem ao servidor solicitando o histórico de mensagens de algum cliente específico.