**Projeto Final**

**Disciplina**: Conectividade em Sistemas Ciberfísicos

**Professor**: Guilherme Schnirmann

**Alunos**: Cauã Kammler, Vinicius Soares da Silva

Link do Github: <https://github.com/ViniciusSoares06/ENTREGA-PROJETO-FINAL.git>

**Requisitos Funcionais do Projeto**

**RF1** – Cada computador deve identificar os outros usuários pelo nome ou apelido.

**RF2** – O sistema deve permitir a comunicação entre no mínimo três computadores conectados ao mesmo roteador.

**RF3** – O usuário deve digitar seu nome antes de começar a conversar no chat.

**RF4** – O sistema deve permitir o envio de mensagens de texto em tempo real.

**RF5** - Mensagens enviadas sem o comando de unicast devem ser broadcast, ou seja, ser enviadas para todos os usuários.

**RF6** - As mensagens irão identificar quem é o remetente pelo nome.

**RF7** – O chat deve possuir interface gráfica.

**RF8** – O servidor deve receber dados de nome, IP e porta do usuário conectado e desconectado.

**RF9** – O cliente deve ter a opção de sair do chat com o comando “/sair”.

**RF10** – O cliente deve ter a opção de mandar uma mensagem privada com o comando “/(nome do destinatário)”.

**Uso de Sockets**

Em nosso código, utilizamos WebSockets com a biblioteca flask\_socketio para comunicação em tempo real entre o servidor e os clientes. O servidor gerencia a conexão dos usuários, permitindo tanto mensagens públicas quanto privadas.

**Broadcast e Unicast**

Broadcast: Uma função recebe os inputs da mensagem pelo javascript e verifica se ela não é a função de sair ou de enviar mensagem privada. Depois, de passar pelas verificações, a função envia o a mensagem com a função emit do flask\_socketio, passando como parâmetro “broadcast=True”, que envia a mensagem para todos os usuários conectados.

Unicast: A função que recebe a mensagem do javascript tem uma verificação que percebe a presença de uma “/” no início da mensagem, iniciando a função de mensagem privada. Durante o processo, a mensagem recebida é modificada para retirar a “/” e o nome do destinatário, reenviando a mensagem com os parâmetros da mensagem em si, dizendo que é privada e quem enviou, e com a sid do usuário destino, armazenada em um dicionário com todos os conectados.

**Uso de Threads**

As bibliotecas do flask e flask\_socketio fazem o uso de threads para permitir que vários loops de diferentes clientes e do servidor não se interrompam e rodem simultaneamente. A cada conexão, as bibliotecas criam uma nova thread para gerenciar cada usuário.