Videoconferência Distribuída

Disciplina de Redes II

2023.2

Pedro Henrique Ximenes
Marina Piragibe

Profª Débora Saade

Documentação do código:

Servidor.py:

Atributos:

1)self.HOST: Armazena o endereço IP ao qual o servidor está vinculado.

2)self.PORTA: Armazena o número da porta na qual o servidor está ouvindo as conexões dos clientes.

3)self.listaSockets: Lista que mantém os sockets dos clientes conectados ao servidor.

4)self.listaClientes: Lista que mantém objetos “Cliente” que representam os clientes conectados ao servidor.

5)self.ativo: É uma variável booleana que indica se o servidor está ativo (True) ou não (False).

Métodos:

1):verificarCliente(self, cliente): Verifica se um cliente com atributos específicos já está na lista de clientes do servidor. Se o cliente não estiver na lista, ele é registrado e adicionado à lista. Retorna True se o cliente for registrado com sucesso, caso contrário, retorna False.

2):conexaoClienteServidor(self, socketCliente, cliente): Lida com a interação contínua entre o servidor e um cliente específico. Ela chama a função menuServidor do módulo Utils que é executada até que conexão seja encerrada.

3):recebeMensagem(self, clientSocket): Recebe uma mensagem do cliente através do socket e a desserializa usando a biblioteca pickle. Ela retorna a mensagem recebida.

4):enviaMensagem(self, clientSocket, msg): Envia uma mensagem para o cliente através do socket, serializando a mensagem usando a biblioteca pickle.

Cliente.py:

Atributos:

1)self.nome: Armazena um nome, string, o qual será salvo no servidor.
2)self.IP: Armazena o número da porta na qual o cliente está vinculado.
3)self.porta: Armazena o número da porta na qual o cliente receberá informações.
4)self.ativo: Boolean que fica o loop para o cliente acessar o menu no Utils.

Métodos:

1):recebeMensagem(self, clientSocket): Usada para receber mensagens do servidor. Ela fica em um loop até que uma mensagem seja recebida (ou seja, até que msg seja diferente de None). A mensagem é desserializada usando o módulo pickle e, em seguida, retornada.
2):enviaMensagem(self, clientSocket, msg): Usada para enviar mensagens para o servidor. A mensagem é serializada usando o módulo pickle e enviada através do socket. Há um pequeno atraso de 0.2 segundos após o envio da mensagem.

Utils.py:

Atributos:

Na maioria das vezes, recebe parâmetros e executa funções nos outros arquivos.

Métodos:

1):recebeCliente(): Solicita ao usuário que insira seu nome, IP e porta e cria um objeto do tipo Cliente com essas informações. Retornando o objeto Cliente criado.

2):imprimeListaClientes(listaClientes): Recebe uma lista de objetos do tipo Cliente chamada listaClientes como argumento.

Itera pela lista de clientes e imprime o nome e o IP de cada cliente. Usado no servidor para listar os clientes conectados.

3):menuCliente(conexao, cliente): Implementa um menu de interação para o lado do cliente. Permite ao cliente escolher entre várias opções, interagindo com o servidor para receber informações ou realizar ações específicas, vejamos elas:

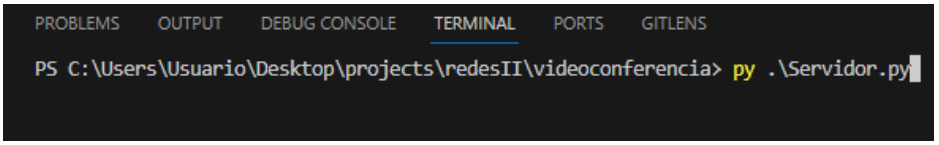
- ➔ 1 - Lista todos usuários cadastrados: Pede para o servidor a lista dos clientes conectados e imprime logo em seguida;
- ➔ 2 - Busca pelo nome de um usuário: Digita o nome do cliente que deseja buscar e espera a resposta do servidor. Se for diferente de [] achou um cliente e imprime ele na tela;
- ➔ 3 - Busca pelo IP de um usuário: Digita o IP do cliente que deseja buscar e espera a resposta do servidor. Se for diferente de [] achou um cliente e imprime ele na tela;
- ➔ 4 - Desligar conexão e sair: Pede para o servidor o desligamento, fecha a conexão e sai do loop de menu;

4):menuServidor(servidor, socketCliente, cliente): Implementa um menu de interação para o lado do servidor. Aguarda a interação do cliente, ou seja, um pedido, para realizar alguma operação/ação, vejamos elas:

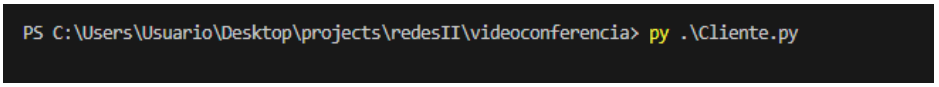
- ➔ Lista todos usuários cadastrados: Envia para o cliente a lista dos clientes conectados;
- ➔ Busca pelo nome de um usuário: aguarda o recebimento do nome do cliente que deseja ser achado e procura na lista de clientes, verificando o nome informado. Caso ache, envia esse cliente para a requisição. Caso contrário, Envia uma mensagem vazia;
- ➔ Busca pelo nome de um usuário: aguarda o recebimento do IP do cliente que deseja ser achado e procura na lista de clientes, verificando o nome informado. Caso ache, envia esse cliente para a requisição. Caso contrário, Envia uma mensagem vazia;
- ➔ Desligar conexão e sair: Desliga a conexão existente do socket, retira esse cliente das listas existentes e retorna um boolean False para sair do loop daquela thread recorrente;

Como usar:

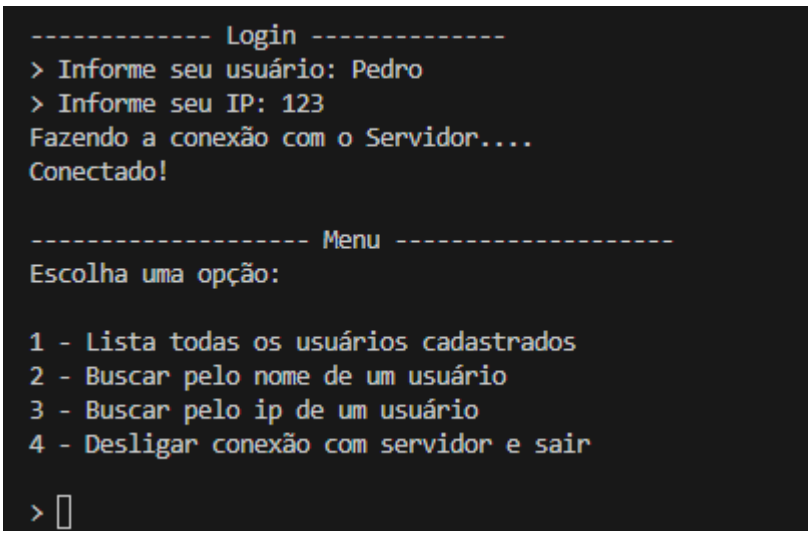
1) Inicialmente execute o Servidor.py, como no comando abaixo:



2) Verifique se o servidor foi inicializado. Caso positivo, execute o Cliente.py, como no comando abaixo em outro Terminal:



3) Logo após, cadastre seu cliente e siga as instruções do **menuCliente**:



Política de Divisão de Tarefas:

Fizemos boa parte dessa primeira entrega em uma chamada no discord. Pensamos e debatemos juntos algumas formas de implementar e corrigir os erros que surgiram. Com base nos commits a divisão de tarefas ficou assim:

Pedro: Criação das threads do servidor, lógica para listagem dos clientes, busca por nome e deleção da conexão cliente e servidor. Conserto na lógica das threads e refatoração do código, por meio de uma limpa e melhorias no código.

Marina: Criação do repositório no github, criação dos arquivos base do cliente e do servidor. Criação de um menu interativo e uma comunicação via objetos no python.Verificar se o usuário já está cadastrado no servidor. Criação do atributo porta e refatoração do código, por meio de uma limpa e melhorias no código.

Repositório:

Link do GitHub: https://github.com/MarinaPiragibe/videoconferencia