ATIVIDADE PRÁTICA/ RELATÓRIO – PROJETO FINAL – CHAT

REQUISITOS FUNCIONAIS:

- RF1 – O Sistema deve suportar pelo menos 3 dispositivos conectados ao mesmo tempo (incluindo o servidor).
- RF2 – O Sistema sempre deve solicitar o apelido (identificação) do usuário antes dele entrar no chat
- RF3 – Toda vez que um novo usuário entrar no chat, uma mensagem de notificação será enviada para todos os usuários conectados, indicando que essa pessoa entrou no chat.
- RF4 – O Chat deve possuir mensagens broadcast (De um para todos os presentes no chat)
- RF5 - O sistema deve estabelecer e manter uma conexão entre os usuários e o servidor em tempo real, usando sockets
- RF6 – O Chat deve possuir a escolha de mensagem Unicast (Mensagem privada para somente uma pessoa do chat)
- RF7 – Na tela do servidor, deverá ser mostrado o IP e a Porta do usuário que se conectou ao chat
- RF8 – Ao alguém sair do chat, o Sistema deve imprimir uma mensagem para todos os presentes (broadcast), avisando que essa pessoa saiu do chat
- RF9 – O Chat deve possuir uma interface gráfica utilizando TkInter
- RF10 – O Chat deve utilizar protocolo TCP, a partir da função “Socket_Stream”
- RF11 – O Chat deve possuir uma validação de dados e possíveis erros no Socket, exibindo uma mensagem de erro. (Utilizando o tratamento de exceções)
- RF 12 – O chat deve utilizar Threads para realizar o servidor suportar diversos usuários conectados ao mesmo tempo

SOCKETS:

- A biblioteca Sockets foi utilizada para conectar o servidor com os clientes do Chat em “Tempo Real”.
- Para isso, começamos utilizando a função “socket.socket(socket.AF_INET, socket.SOCK_STREAM)”, para criar uma Socket, identificar seu IPV4 e configurar seu protocolo ITCP.
- Utilizamos a função “server.bind” para associar um Socket a uma porta e servidor.
- Também a função “server.listen”, para o servidor escutar as conexões.
- A função “client, address = server.accept()”, para a criação de novos Sockets para cada cliente.

THREADS:

- As Threads são utilizadas para o servidor suportar diversos usuários conectados ao mesmo tempo
- No arquivo Server.py, utilizamos Threads a partir dessa função: “thread = threading.Thread(target=handle, args=(client, nickname))
  thread.start()”, onde o servidor cria uma Thread para cada Cliente que entra, puxando a function “Handle”, e os parâmetros Client(Socket) e Nickname.
  - Já no arquivo Client.py, utilizamos Threads a partir da função: “receive_thread = threading.Thread(target=receive) receive_thread.start()”, para receber as mensagens de cada cliente. Ele puxa a function “Receive”.