# Relatório

## Introdução

Este relatório tem como objetivo descrever, analisar e explicar o funcionamento de uma aplicação de cliente-servidor desenvolvida em Python, utilizando a biblioteca `socket`. A aplicação consiste em dois scripts: um que implementa um servidor TCP, e outro que atua como cliente, permitindo a realização de um pequeno jogo de adivinhação numérica.  
  
O foco é demonstrar o uso básico de comunicação entre máquinas em rede através do protocolo TCP/IP, reforçando conceitos como: conexão, envio e receção de dados, tratamento de exceções e interação cliente-servidor.

## Descrição do Código

### Servidor

O script do servidor realiza as seguintes operações:

1. Inicialização do socket:  
 - Criação de um socket TCP/IP.  
 - Bind para aceitar conexões em todas as interfaces (0.0.0.0) na porta 12345.  
 - O método listen() coloca o servidor em estado de escuta.

2. Aceitação da conexão:  
 - O servidor aguarda e aceita uma conexão de cliente.  
 - addr guarda o endereço IP e a porta do cliente.

3. Interação com o cliente:  
 - O servidor começa solicitando que o cliente adivinhe um número de 1 a 10.  
 - Ao receber a resposta, o servidor tenta convertê-la para inteiro.  
 - Se a resposta for inválida (ex: letras), o servidor envia uma mensagem de erro.  
 - O número correto está fixado como 10. Se o cliente acertar, o jogo termina com uma mensagem de sucesso. Caso contrário, o servidor solicita nova tentativa.

4. Encerramento:  
 - O servidor fecha a conexão com o cliente e finaliza o socket.

### Cliente

O script do cliente tem como propósito conectar-se ao servidor, responder ao desafio e reagir às mensagens recebidas.

1. Inicialização e conexão:  
 - Cria um socket TCP/IP e conecta-se ao servidor no IP e porta indicados.

2. Envio de mensagem inicial:  
 - O cliente envia uma mensagem qualquer ao servidor, como parte do protocolo de início.

3. Loop de interação:  
 - O cliente aguarda mensagens do servidor.  
 - Quando recebe a mensagem de adivinhação, solicita input do usuário.  
 - Envia a resposta ao servidor.  
 - Se a resposta for correta, o cliente finaliza.

4. Encerramento:  
 - Caso o servidor desconecte ou envie a mensagem final, o cliente termina a sua execução e fecha o socket.

## Funcionamento do Jogo

O servidor escolhe o número 10 como resposta correta. O cliente recebe a mensagem: "Adivinhe o número (1 a 10):" e envia um número como resposta.  
- O servidor avalia a tentativa e responde com:  
 - "Parabéns! Acertou!" e o jogo termina.  
 - "Errou o número. Tente novamente..." o jogo continua.  
 - "Por favor, envia um número válido." → em caso de erro de input.

## ****Sugestões de Melhoria****

**Segurança e robustez:** Adicionar tratamento de exceções com try/except em blocos como accept(), recv() e send(), garantindo que o programa não encerre abruptamente.

**Estrutura modular:** Separar a lógica em funções para melhorar a organização e legibilidade do código.

## Conclusão

A aplicação proposta cumpre satisfatoriamente o seu propósito: Implementar e demonstrar o funcionamento de uma comunicação cliente-servidor usando Python. Apesar da simplicidade, o projeto serve como uma ótima base para estudos de redes, podendo ser facilmente expandido para cenários mais complexos, como jogos multiplayer, chats, servidores web simples, entre outros.  
  
A experiência obtida neste exercício contribui para o entendimento prático dos conceitos de conexão, escuta, troca de mensagens, e estrutura de rede com sockets, temas fundamentais no desenvolvimento de aplicações distribuídas.