Relatório – EP Jogo da Forca em Rede (ACH2026)

1. Introdução

Esse trabalho faz parte da disciplina ACH2026 – Redes de Computadores.
O objetivo foi criar um jogo da forca que funciona pela rede, com um servidor que coordena o jogo e clientes que participam adivinhando a palavra.
O protocolo usado foi o “Hangman Protocol”, que define como as mensagens são trocadas entre o servidor e os jogadores usando TCP.

---

2. Decisões de projeto

O projeto foi feito em Python porque é mais simples de lidar com sockets e tem menos burocracia pra rodar.
Foi usado um servidor (server.py) e um cliente (client.py).
O servidor controla todo o jogo: escolhe o mestre, recebe os palpites e avisa o estado pra todo mundo.
Os clientes mandam as jogadas e mostram o desenho da forca na tela.

O servidor guarda os jogadores em um dicionário {nome : conexão} e usa listas e conjuntos pra controlar:

- Quem joga em cada rodada
- Letras e palavras que já foram usadas
- O estado da palavra (com '_' nas letras ainda não descobertas)

---

3. Dificuldades enfrentadas

A parte mais chata foi lidar com o fluxo das mensagens TCP, porque recv() e send() não garantem que a mensagem chega completa. Tive que entender bem o uso do \r\n pra saber quando uma mensagem terminava.
Também deu trabalho organizar a ordem dos turnos e garantir que só um jogador recebesse o “YOURTURN” por vez.
Outra parte foi implementar todas as mensagens de erro certinho, porque o enunciado tinha várias situações diferentes pra tratar.

---

4. Método usado

O desenvolvimento foi feito em partes.
Primeiro testei só a conexão cliente-servidor.
Depois adicionei a escolha do mestre e a palavra.
Na sequência implementei os turnos e as jogadas (letra e palavra).

Por fim, fiz o tratamento de erros e deixei o cliente com o desenho da forca e a lista de palpites errados.
Tudo foi testado no terminal, abrindo três janelas (um servidor e dois clientes).

---

5. Testes

Os testes foram todos manuais.
Rodei o servidor com 'python server.py 2' e abri dois clientes com 'python client.py Alice' e 'python client.py Bob'.
Testei os casos normais e também alguns erros, tipo mandar letra repetida, tentar sair com \q e mandar palavra errada.
Quando o jogo acabava, o servidor mandava o "GAMEOVER" e encerrava tudo certinho.

---

6. Melhorias possíveis

Se fosse pra deixar o jogo melhor:

- Daria pra aceitar palavras com hífen só mudando a regex na validação da palavra.
- Poderia mostrar uma mensagem “Servidor cheio” se mais gente tentasse entrar.
- Outra ideia seria permitir várias rodadas sem precisar reiniciar o servidor.
- E também dava pra guardar uma pontuação pros jogadores.

---

7. Uso de IA

Usei a IA (ChatGPT) em alguns momentos específicos durante o desenvolvimento. No começo, usei pra entender melhor como funcionava a criação de sockets em Python e como fazer o cliente e o servidor trocarem mensagens usando TCP. Depois, usei novamente quando fiquei travado no tratamento de erros — especialmente quando o servidor estava fechando todas as conexões por engano ao receber uma mensagem inválida. A IA me ajudou a perceber que eu precisava tratar cada conexão individualmente em vez de encerrar o servidor inteiro.

Também usei a IA pra gerar o arquivo README.md, principalmente pra organizar as instruções de execução e deixar o formato parecido com o exigido pelo enunciado. As respostas foram usadas como base, mas todas as partes de código e testes foram feitos e ajustados manualmente.

---

8. Conclusão

O EP funcionou bem. O servidor e os clientes se comunicam direitinho, o protocolo foi seguido e o jogo roda de forma estável.
Serviu pra entender bem como funciona a troca de dados via TCP e como montar um protocolo simples de aplicação.
Deu um pouco de trabalho, mas valeu a pena.