



**UNIVERSIDADE FEDERAL DE ALAGOAS
INSTITUTO DE COMPUTAÇÃO (IC)**

Ciência da Computação
Redes de Computadores 2024.2

Relatório de Projeto

Jogo da Forca em Python, com sockets e threads

ERICK DE LIMA MASCARENHAS
JADIEL HENRIQUE CALADO LINO
NEILTON GABRIEL GONÇALVES LUCIANO
PEDRO DE CARVALHO CEDRIM

SUMÁRIO

Principais funcionalidades.....	3
Servidor (server.py).....	3
Cliente (client.py).....	3
Protocolos e conceitos envolvidos.....	4
TCP/IP.....	4
Sockets.....	4
Threads.....	4
Modelo cliente-servidor.....	4
Possíveis melhorias.....	5
Dificuldades encontradas.....	6
Diversos clientes simultâneos.....	6
Tratamento de desconexões.....	6
Interface visual.....	6
Código fonte:.....	7
palavras.txt.....	7
server.py e client.py.....	7

Principais funcionalidades

O projeto implementa um jogo da forca multijogador, utilizando uma conexão entre um servidor e um ou mais clientes, tendo duas partes que funcionam em conjunto:

Servidor (server.py)

Ele é responsável por gerenciar as conexões dos clientes, usando socket e threads para lidar com conexões simultâneas, além de receber informações dos clientes, enviar respostas para eles e guardar alguns dados de cada um, como o nome de usuário. É também ele que cuida da lógica do jogo, contando com uma função que escolhe uma palavra aleatória de um arquivo, para ser adivinhada a cada rodada, ele controla os turnos dos jogadores, armazena a quantidade de erros, as letras já usadas, as pontuações de cada jogador, e até determina quem venceu a rodada.

O servidor envia constantemente mensagens sobre o estado atual do jogo para todos os clientes, pedindo por uma entrada, avisando ao cliente que ainda não é a vez dele, informando o resultado final da rodada, ou até dizer que um jogador se desconectou. Além disso, quando um jogador se desconecta, ele ajusta os turnos conforme necessário, passando a vez para evitar problemas, se for o caso.

Cliente (client.py)

O cliente é a interface de texto que aparece no terminal, conectada com o servidor também por socket. Essa interface mostra todas as mensagens recebidas pelo servidor, como o estado atualizado geral do jogo, e permite que o usuário digite entradas, como letras ou palavras. Para diminuir a poluição visual e aprimorar a experiência do jogador, o cliente limpa a tela a cada mensagem que recebe, antes de a exibir. Por fim, ele permite que o jogador se desconecte através de um comando simples, lida com essa desconexão, e até sincroniza o jogo após cada rodada, dando 5 segundos para o início da próxima.

Protocolos e conceitos envolvidos

O projeto utiliza os seguintes protocolos e conceitos de rede:

TCP/IP

Usando os dois protocolos em conjunto para a comunicação entre o servidor e o cliente, uma conexão confiável é fornecida, ao garantir que os dados sejam entregues na ordem certa e sem perdas, para os dispositivos identificados na rede através da conexão.

Sockets

Tanto o cliente quanto o servidor criam sockets para enviar e receber dados um do outro. Enquanto o servidor usa um socket para ouvir as conexões de entrada (`servidor.listen()`), o cliente usa sockets para se conectar ao servidor (`cliente.connect((HOST, PORT))`).

Threads

O servidor utiliza threads para lidar com múltiplas conexões simultâneas, onde cada cliente conectado é gerenciado em uma thread individual. Dessa forma, o servidor consegue atender a vários jogadores ao mesmo tempo.

Modelo cliente-servidor

Com esse modelo, o servidor cuida da lógica do jogo e gerencia os clientes, os quais se conectam para participar do jogo. Dessa forma, as informações que são enviadas de um cliente a outro passam antes pelo servidor.

Possíveis melhorias

Baseando-se no jogo da forca, implementamos sistemas como pontuação e cooperação, mas há outras melhorias que poderiam ter sido realizadas, como:

1. Interface gráfica com música, efeitos sonoros e um bate-papo.
2. Sistema para escolher a categoria das palavras, dar dicas e alterar nível de dificuldade das palavras.
3. Autenticação de usuário usando contas, guardando estatísticas de cada jogador e criando uma tabela de classificação de jogadores com mais vitórias.
4. Segurança e criptografia, para proteger a comunicação entre servidor e cliente, evitando trapaças e hacks.
5. Possibilidade de reconexão, para jogadores que forem desconectados abruptamente.

Todas essas melhorias visam melhorar a experiência do jogador, através de uma experiência mais atraente com visuais e sons, mais dinâmica com interações entre jogadores e competição para se manter em classificações altas, mais justa com acessibilidade para pessoas de todas as idades, possibilidade de reconexão e tratamento de trapaças, e mais segura com conexões criptografadas e sistemas contra hackers.

Entretanto, grande parte dessas melhorias fogem do nosso escopo para esse projeto, pois não temos experiência nem conhecimento para poder implementá-las. Por isso, priorizamos melhorar a experiência do jogador o máximo possível usando o conhecimento que temos sobre Python.

Dificuldades encontradas

Durante o desenvolvimento do projeto, encontramos algumas dificuldades, as quais estão citadas abaixo:

Diversos clientes simultâneos

Alguns problemas surgiram no processo, pois no começo estávamos tendo dificuldades na forma de usar threads, mas assim que entendemos como usar locks, ficou mais fácil compartilhar o estado do jogo e manter tudo sincronizado entre todos os clientes.

Tratamento de desconexões

Clientes desconectados podem gerar problemas ao servidor, e até fazer ele parar de funcionar por completo. Tratamos esse problema e corrigimos os erros que isso poderia gerar da melhor forma possível, mas dependendo da situação, algum *bug* despercebido ainda pode ocorrer e fazer o servidor enfrentar algum problema e finalizar sozinho.

Interface visual

Tentamos aprimorar a interface, mas não conseguimos obter um resultado original, por isso nos mantivemos usando a interface apenas baseada em texto, diretamente pelo terminal. Isso facilitou o trabalho, por ser mais fácil de garantir a clareza das mensagens exibidas dessa forma.

Código fonte:

palavras.txt

<https://www.ime.usp.br/~pf/dicios/br-sem-acentos.txt>

Utilizamos como fonte o arquivo acima, que conta com aproximadamente 240.000 palavras. Para diminuir a quantidade total de palavras, reduzir a redundância e facilitar a experiência do jogador, filtramos o arquivo, mantendo apenas palavras de 5 a 15 caracteres, e removendo nomes próprios e a maioria dos verbos que não estavam no infinitivo. Dessa forma, restaram aproximadamente 15.000 palavras, um número adequado para evitar que a mesma palavra seja escolhida repetidamente.

server.py e client.py

Ambos os códigos estão anexados separadamente abaixo, nesse PDF.

