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

Maceió - Alagoas
2025

SUMÁRIO

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.

```
import socket
import threading
import random
import time

HOST = 'localhost' # editar aqui, se necessário
PORT = 5555

clientes = []
enderecos = []
nomes = []
pontuacoes = {}
vitorias = {}
palavras = []
palavra = ""
palavravisivel = []
erros = 0
maxerros = 6
letrasusadas = []
turno = 0
lock = threading.Lock()

def removercliente(cliente, endereco, nome):
    global turno
    indice = clientes.index(cliente)
    if cliente in clientes:
        clientes.remove(cliente)
    if endereco in enderecos:
        enderecos.remove(endereco)
    if nome in nomes:
        nomes.remove(nome)

    if len(clientes) == 0:
        turno = 0
    elif indice < turno:
        turno -= 1
    elif indice == turno:
        turno = turno % len(clientes)

def enviarajogador(indice, msg): # envia por índice
    try:
        clientes[indice].sendall((msg + "\n").encode())
    except:
        removercliente(clientes[indice], enderecos[indice], nomes[indice])

def broadcast(msg): # envia para todos
    for cliente in clientes:
        try:
            cliente.sendall((msg + "\n").encode())
        except:
            idx = clientes.index(cliente)
            removercliente(cliente, enderecos[idx], nomes[idx])

def pegarpalavras():
    global palavras
    with open("palavras.txt", "r", encoding="utf-8") as f:
        palavras = [linha.strip() for linha in f if linha.strip()]

def escolherpalavra():
    global palavra, palavravisivel, erros, letrasusadas
    palavra = random.choice(palavras).lower()
    palavravisivel = ["_" if c.isalpha() else c for c in palavra]
    erros = 0
    letrasusadas = []

def textos():
    estado = f"Palavra: {' '.join(palavravisivel)}\n"
    estado += f"Erros: {erros}/{maxerros}\n"
    estado += f"Letras usadas: {', '.join(letrasusadas)}\n"
    estado += f"Para desconectar do jogo, digite: sair\n"
    estado += f"Placar:\n"
    for nome in nomes:
        estado += f"- {nome}: {vitorias.get(nome, 0)} vitórias. Tem {pontuacoes.get(nome, 0)} ponto(s) nessa rodada\n"
    return estado

def tratarmsgs(cliente, endereco):
    global turno, erros, palavra, palavravisivel, pontuacoes

    with lock:
        try:
            cliente.sendall("Digite seu nome: ".encode())
            nome = cliente.recv(2048).decode().strip()
            print(f"[NOME RECEBIDO] {nome} de {endereco}")
            nomes.append(nome)
            pontuacoes[nome] = 0
            vitorias[nome] = 0
        except:
            removercliente(cliente, endereco, nome)
            cliente.close()
            return

    cliente.sendall("Conectado ao servidor.\n".encode())

    while True:
        try:
            indice = clientes.index(cliente)

            if indice != turno:
                enviarajogador(indice, f"{textos()}\nAguarde, é a vez de {nomes[turno]}.")
                time.sleep(1)
                continue
            else:
                enviarajogador(indice, f"{textos()}\nSua vez! Envie uma letra ou chute a palavra (+5 pontos):")

            dados = cliente.recv(2048).decode().strip()
            if not dados:
                break

            if len(dados) == 1 and dados.isalpha():
                letra = dados.lower()

                with lock:
                    if letra in letrasusadas:
                        enviarajogador(indice, f"{textos()}\nLetra {letra} já foi usada.")
                        continue

                    letrasusadas.append(letra)

                    if letra in palavra:
                        acertos = 0
                        for i, c in enumerate(palavra):
                            if c == letra and palavravisivel[i] == "_":
                                palavravisivel[i] = letra
                                acertos += 1
                        if acertos > 0:
                            pontuacoes[nome] = pontuacoes.get(nome, 0) + acertos
                        broadcast(f"{textos()}\n{nomes[indice]} acertou a letra '{letra}'!")
                    else:
                        erros += 1
                        broadcast(f"{textos()}\n{nomes[indice]} errou a letra '{letra}'!")

                    if "_" not in palavravisivel:
                        jogadoracerto = nomes[indice]
                        maiorpontuacao = max(pontuacoes.values())
                        vencedores = [nome for nome, pontos in pontuacoes.items() if pontos == maiorpontuacao]

                        for nomevencedor in vencedores:
                            vitorias[nomevencedor] += 1
                        if len(vencedores) == 1:
                            broadcast(f"VITÓRIA! {jogadoracerto} acertou a palavra '{palavra}'.\n{vencedores[0]} ganhou essa rodada com {maiorpontuacao} pontos!\nA próxima rodada começa em 5 segundos.")
                        else:
                            broadcast(f"EMPATE! {jogadoracerto} acertou a palavra '{palavra}'.\n{', '.join(vencedores)} ganharam essa rodada com {maiorpontuacao} pontos!\nA próxima rodada começa em 5 segundos.")
                        pontuacoes = {}
                        turno = (turno + 1) % len(clientes)
                        escolherpalavra()
                        continue

                    if erros >= maxerros:
                        broadcast(f"DERROTA! A palavra era: '{palavra}'. A próxima rodada começa em 5 segundos.")
                        pontuacoes = {}
                        turno = (turno + 1) % len(clientes)
                        escolherpalavra()
                        continue

                    turno = (turno + 1) % len(clientes)

            else:
                with lock:
                    if dados.lower() == palavra.lower():
                        jogadoracerto = nomes[indice]
                        palavravisivel[:] = list(palavra)
                        pontuacoes[jogadoracerto] = pontuacoes.get(jogadoracerto, 0) + 5 # +5 pontos para quem acertou o chute
                        maiorpontuacao = max(pontuacoes.values())
                        vencedores = [nome for nome, pontos in pontuacoes.items() if pontos == maiorpontuacao]

                        for nomevencedor in vencedores:
                            vitorias[nomevencedor] += 1
                        if len(vencedores) == 1:
                            broadcast(f"VITÓRIA! {jogadoracerto} acertou a palavra '{palavra}'.\n{vencedores[0]} ganhou essa rodada com {maiorpontuacao} pontos!\nA próxima rodada começa em 5 segundos.")
                        else:
                            broadcast(f"EMPATE! {jogadoracerto} acertou a palavra '{palavra}'.\n{', '.join(vencedores)} ganharam essa rodada com {maiorpontuacao} pontos!\nA próxima rodada começa em 5 segundos.")
                        pontuacoes = {}
                        turno = (turno + 1) % len(clientes)
                        escolherpalavra()
                        continue
                    else:
                        erros += 1
                        if erros >= maxerros:
                            broadcast(f"DERROTA! A palavra era: '{palavra}'. A próxima rodada começa em 5 segundos.")
                            pontuacoes = {}
                            turno = (turno + 1) % len(clientes)
                            escolherpalavra()
                            continue
                        else:
                            broadcast(f"{textos()}\n{nomes[indice]} tentou chutar a palavra e errou.")

                            turno = (turno + 1) % len(clientes)

        except Exception as e:
            print(f"[ERRO] com o jogador {nome}: {e}")
            break

    removercliente(cliente, endereco, nome)
    cliente.close()
    broadcast(f"{nome} saiu do jogo.")

def main():
    pegarpalavras()
    escolherpalavra()
    servidor = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    try:
        servidor.bind((HOST, PORT))
        servidor.listen()
        print(f"[SERVIDOR] Ouvindo em {HOST}:{PORT}...")
    except:
        return print("\nNão foi possível iniciar o servidor.\n")

    try:
        while True:
            cliente, endereco = servidor.accept()
            print(f"[NOVA CONEXÃO] {endereco}")
            clientes.append(cliente)
            enderecos.append(endereco)

            thread = threading.Thread(target=tratarmsgs, args=(cliente, endereco))
            thread.start()
    except KeyboardInterrupt:
        print("\n[SERVIDOR] Encerrado manualmente.")
        servidor.close()

main()
```

```python
# código fonte do 'client.py':
import socket
import threading
from os import system
import time

HOST = 'localhost' # editar aqui, se necessário
PORT = 5555

def recebermsgs(cliente):
    while True:
        try:
            msg = cliente.recv(2048).decode()
            if not msg:
                break
            if "saiu do jogo." not in msg:
                system("clear||cls")
            print(msg)
            if "segundos" in msg: # se a rodada terminou, espera 5 segundos para iniciar a
próxima
                time.sleep(5)

        except:
            print("[ERRO] Conexão encerrada.")
            break

def main():
    cliente = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    try:
        cliente.connect((HOST, PORT))
    except:
        print("Não foi possível se conectar ao servidor.")
        return

    thread = threading.Thread(target=recebermsgs, args=(cliente,))
    thread.start()

    while True:
        try:
            msg = input()
            if msg.strip().lower() == "sair":
                break
            cliente.sendall(msg.encode())
        except:
            break

    cliente.close()

main()
```