

## 1 Proposta

A ideia do trabalho é desenvolver sockets para xadrez online. Cada jogador terá um tempo de 3 minutos para desenvolver suas jogadas (partida do tipo blitz) e a partida termina ou quando o tempo de um jogador acaba ou quando tem um cheque mate, um stalemate, repetição de três movimentos, etc.

O protocolo escolhido foi o TCP, para desenvolver um sistema do tipo híbrido, que terá interação cliente-servidor e interação cliente-cliente.

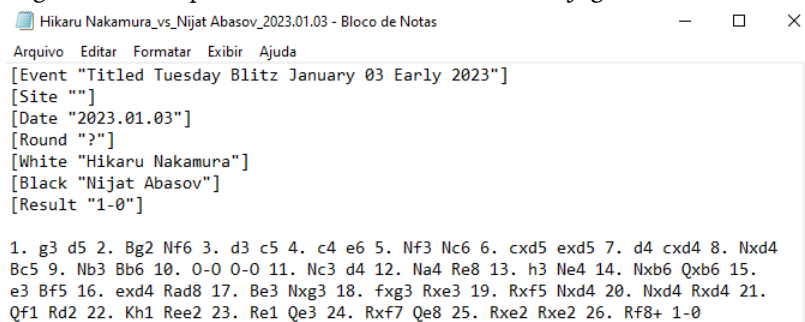
O jogador se conecta ao servidor e vai escolher seu estado de atividade (procurando um jogo ou não) e vai esperar em uma sala até aparecer um outro jogador para jogar uma partida blitz. Ao ter dois jogadores procurando por uma partida, o servidor será responsável por parear os dois jogadores para que eles joguem, a partir daí o servidor não terá nenhuma interferência no jogo, a partida em si vai se desenvolver no sistema cliente-cliente, ou peer to peer.

Os jogadores vão trocar as jogadas através de mensagens de texto. O servidor, ao parear, vai decidir de forma aleatória qual jogador será as brancas e qual será as pretas. Cada jogador terá seus três minutos no total para fazer suas jogadas.

Vou chamar o jogador das brancas de P1 e o das pretas de P2. O jogo se desenvolve da seguinte forma: o P1 começa e manda uma jogada para o P2, ele pode demorar o tempo que quiser para fazer essa jogada, o tempo de jogo dele não começa a contar ainda, ao mandar a jogada, o tempo do P2 começa a contar para fazer sua jogada, ele manda a jogada para o P1, seu tempo para de contar e o tempo do P1 começa a rodar, o P1 manda a jogada para o P2, seu tempo para e o tempo de P2 começa a rodar e assim por diante.

As mensagens enviadas serão analisadas antes do envio para ver se está dentro do padrão de jogadas do xadrez, que pode ser vista na Figura 1 [1]. Vou tentar produzir esse resultado para cada jogador, mas não garanto nada. Teria que ser desenvolvido dentro do próprio cliente o algoritmo para produzir esse arquivo de texto. Se colocar no servidor vai tirar a parte puramente cliente-cliente da aplicação.

Figura 1: Exemplo de um PGN resultado de um jogo oficial de xadrez.



```
[Event "Titled Tuesday Blitz January 03 Early 2023"]
[Site ""]
[Date "2023.01.03"]
[Round "?"]
[White "Hikaru Nakamura"]
[Black "Nijat Abasov"]
[Result "1-0"]

1. g3 d5 2. Bg2 Nf6 3. d3 c5 4. c4 e6 5. Nf3 Nc6 6. cxd5 exd5 7. d4 cxd4 8. Nxd4
Bc5 9. Nb3 Bb6 10. O-O O-O 11. Nc3 d4 12. Na4 Re8 13. h3 Ne4 14. Nxb6 Qxb6 15.
e3 Bf5 16. exd4 Rad8 17. Be3 Nxe3 18. fxe3 Rxe3 19. Rxf5 Nxd4 20. Nxd4 Rxd4 21.
Qf1 Rd2 22. Kh1 Re2 23. Re1 Qe3 24. Rxf7 Qe8 25. Rxe2 Rxe2 26. Rf8+ 1-0
```

As mensagens comuns são as seguintes:

- A primeira parte representa a peça que está sendo movida:
  - A mensagem começa com uma letra. Essa letra pode ser maiúscula (R, N, B, Q, K) ou não ter letra nenhuma, que representa um peão.
- A segunda parte é uma indicação da casa alvo que a peça movida está indo:
  - Representada pela coluna (a, b, c, d, e, f, g, h) e pela linha (1, 2, 3, 4, 5, 6, 7, 8)
  - Ela pode ser precedida ou não por um x, que indica que a peça movida está capturando a peça que está casa alvo

Tem algumas mensagens extras especiais:

- Tem a jogada especial que é a única que pode mover duas peças ao mesmo tempo:

- A mensagem O-O representa o roque curto, que move a torre do lado do rei com o rei (o kingside castle)
- A mensagem O-O-O representa o roque longo, que move a torre do lado da dama com o rei (o queenside castle)
- As jogadas normais podem ter um sufixo:
  - Se seguidas por um +, indica uma jogada que coloca o rei oponente em cheque
  - Se seguidas por um #, indica uma jogada que dá um xeque mate e termina o jogo.

A aplicação não terá uma verificação da jogadas em si, não vai ver se o cavalo andou em L ou se o bispo foi na diagonal. Não vai ver se o movimento faz sentido de acordo com a posição das peças no tabuleiro (até porque nem terá um tabuleiro), ela só vai verificar se a mensagem enviada está dentro dos padrões descritos anteriormente. Não vai, também, verificar se as jogadas especiais são possíveis, ele vai apenas enviar as jogadas predefinidas de um jogador para o outro.

A partida vai acabar quando um dos jogadores mandarem uma jogada de cheque mate (com vitória do jogador que enviou) ou até acabar o tempo de jogo de um dos jogadores (com a derrota do jogador que ficou sem tempo disponível).

## 2 Código

O código teve como base uma página do Instituto Federal de Santa Catarina que fala sobre sockets TCP com java [2].

No servidor tem uma duas listas que armazenam os clientes conectados e os clientes prontos para jogar. Cada cliente que entra no servidor tem uma thread própria, controlada pela classe `ClienteHandler`. O cliente que se conecta recebe uma mensagem pedindo o nome do jogador, depois o status, 1 para ficar online e pronto para uma partida, 2 para ficar offline e apenas conectado no servidor e EXIT para desconectar do servidor. Ele segue uma lógica de pedir por um nome de jogador, depois pede para um status e, caso seja um status offline ele pede novamente o status do usuário, caso opte pelo status online ele coloca o jogador na lista de prontos e entra em uma busca por um oponente. Ao encontrar um oponente ele vai colocar os dois em contato e o jogo rola na apenas entre os dois clientes, sem intervenção do servidor. Ao terminar o jogo a conexão entre os cliente morre e eles voltam a se conectar com o servidor e a pedir um status para buscar por um jogo ou sair do servidor.

Tive muito problemas para fazer com que as mensagens sejam trocadas de cliente para cliente após emparelhar ambos para a partida. Eu decidi implementar no cliente uma espécie de mini servidor que seria responsável por criar um socket que se comunicaria com o outro cliente. Então todo cliente também pode ser um servidor para se conectar com outro cliente para a partida de xadrez. O primeiro cliente que entrar no emparelhamento com outros jogadores seria o servidor da comunicação e o segundo seria o cliente enquanto o jogo estivesse rolando. Após o término do jogo o socket de comunicação do jogo morre e ambos voltam para o servidor original, onde eles vão escolher de novo um status e começar uma nova partida com outro jogador.

A escolha de quem será as brancas e quem será as pretas é por um sorteio, quem tiver com as brancas terá o primeiro movimento (setando seu turno como *true*), e aí o turno troca sempre depois de enviar uma mensagem ou receber uma mensagem.

Não consegui implementar o tempo de jogo e a verificação de jogadas. Gastei todo o tempo tentando solucionar a comunicação Cliente-Cliente. Pelo modelo que adotei, acho que essas implementações seriam no próprio cliente, o cliente teria uma função de verificação de jogadas para ver se a jogada é válida antes mesmo de enviar para o oponente e o tempo de jogo também seria no próprio cliente, o tempo começa ao receber a primeira mensagem (de acordo com um jogo de xadrez, as brancas tem a primeira jogada livre da contagem de tempo) e ele pausa ao enviar e recomeça ao receber a mensagem seguinte e assim por diante. Aí se o tempo acaba antes de enviar uma mensagem o jogo termina.

As mensagens trocadas são as seguintes:

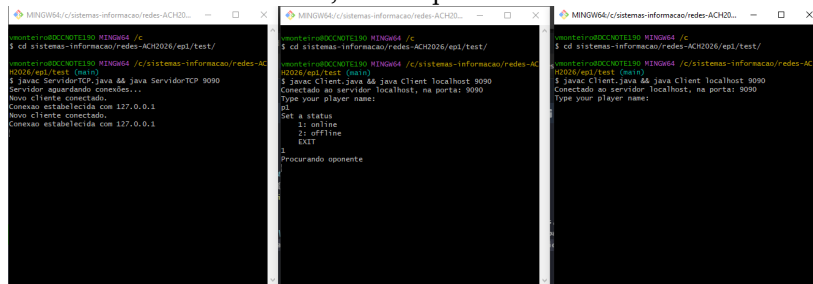
1. Mensagem para o jogador escolher um nome de identificação
2. Mensagem para o jogador escolher seu status, que pode ser online ou offline ou desconectar do servidor
3. Entra em uma partida

4. O servidor faz um sorteio de quem será as peças brancas e quem será as peças negras, ele envia uma mensagem apontando qual cliente será qual o cliente que for as brancas terá sua variável *turno* definida como *true*
5. ao mandar uma mensagem com a partida termina
6. volta para a mensagem de status

### 3 Testes

Os testes são feitos compilando e rodando o servidor e depois compilar os clientes, quantos forem necessários para o teste. Para o servidor uso **javac ServidorTCP.java** e **java ServidorTCP 9090** e para os clientes **javac Client.java** e **java Client localhost 9090**. Eu abro pelo GitBash mesmo, uma janela do GitBash para cada servidor/cliente, Figura 2

Figura 2: Janelas do GitBash rodando os testes, a da esquerda é servidor e a do meio e da direita são clientes.



Alguns dos testes feitos que deram certo:

1. Diferentes nomes para os jogadores são aceitos
2. As três possibilidades de status, com 1 sendo online e entrando na procura por um oponente, a 2 basicamente manda de volta o status (pensei depois que essa opção é meio desnecessário, poderia ser só deixar ali o pedido de status até o cliente decidir buscar uma partida, mas resolvi deixar assim para enviar para ter uma mensagem extra no pacote) e a *EXIT* para desconectar do servidor
3. Teste com dois clientes conectados e eles entram no jogo
4. Teste com 3 clientes, após o término de um jogo, um terceiro jogador conecta e ele joga com o primeiro que entrar online dos dois primeiros

Testes feito que não deu certo:

1. Se um terceiro jogador se conectar ao servidor e ficar online ele vai achar uma partida com um jogador que já está em uma partida. Isso ocorre por que o jogador só é retirado da lista de prontos depois de terminar a partida. A minha ideia principal de solução, que ainda não consegui implementar, é a sincronização de threads para que ambas estejam em um ponto que eu posso retirar da lista de jogadores prontos sem o risco de confundir os jogadores. Tentei me basear na referência [3], mas não consegui fazer a tempo.

### Referências

- [1] chess.com. *Descrição de um PGN de uma partida de xadrez*. URL: <https://www.chess.com/terms/chess-pgn#:~:text=Conclusion-,What%20Is%20PGN%20In%20Chess%3F,most%20chess%20programs%20support%20it..> (accessed: 28.11.2023).
- [2] Instituto Federal de Santa Catarina. *Trabalhando com sockets TCP em Java*. URL: [https://wiki.sj.ifsc.edu.br/index.php/Trabalhando\\_com\\_sockets\\_TCP\\_em\\_Java](https://wiki.sj.ifsc.edu.br/index.php/Trabalhando_com_sockets_TCP_em_Java). (accessed: 02.12.2023).
- [3] Tutorialspoint. *Java - Thread Synchronization*. URL: [https://www.tutorialspoint.com/java/java\\_thread\\_synchronization.htm](https://www.tutorialspoint.com/java/java_thread_synchronization.htm). (accessed: 12.12.2023).