TEC502 - TP01 - Concorrência e Conectividade

Coordenador: Paulo Henrique
Quadro: David Neves
Mesa: Claudio Daniel

Sessão 1

14 de agosto de 2025

Fatos

- **Escolha do jogo é livre.**
  Definiremos um jogo que atenda ao cenário de **duelos 1×1** (ex.: o próprio "jogo de damas" citado nas ideias) para validar a comunicação e as regras básicas. Resultado esperado: regras do jogo escolhidas em uma página, com condições de vitória/derrota e fluxo de turno.

- **Utilização de Docker.**
  O ambiente de execução e testes será containerizado. Resultado esperado: `Dockerfile`(s) e `docker-compose.yml` mínimos para subir rapidamente as instâncias necessárias aos testes.

- **Frameworks liberados (exceto para comunicação).**
  Podemos usar bibliotecas e frameworks auxiliares, **mas a comunicação deve usar a API nativa de sockets** do SO/linguagem escolhida. Resultado esperado: seção no README indicando o que é nativo (rede) e o que é framework (UI, logs etc.).

Ideias

- **Várias salas; cada sala com dois jogadores.**
  Estrutura de “salas” para isolar partidas 1×1. Cada sala mantém estado próprio (jogadores, turno, tabuleiro/mão). Critério de pronto: criar/entrar/sair de sala e impedir mais de 2 jogadores por sala.

- **Partidas com adversário aleatório ou definido.**
  Dois modos de pareamento: **fila aleatória** e **convite/código da sala**. Critério de pronto: comandos/mensagens distintos para “entrar por fila” e “entrar por código”.

- **Sistema de cartas (referência Fortnite/FIFA).**
  Existirá um conjunto de cartas e ações de “obter/usar”. Foco inicial: **operações atômicas** de obtenção/uso para não gerar inconsistência. Critério de pronto: operações de “pegar carta” e “descartar/usar” sem duplicidades.

- **Jogo de damas** como protótipo.
  Serve para validar rapidamente a mecânica de turno/movimento e a troca de mensagens. Critério de pronto: partida completa de damas numa sala.

- **Uso de *mutex* para concorrência de cartas.**
  Quando houver disputa por cartas/recursos, aplicar exclusão mútua. Critério de pronto: seção técnica descrevendo onde o *mutex* é adquirido/liberado e um teste que prova ausência de corrida.

- **Evitar microsserviços.**
  Manter solução simples na primeira versão, reduzindo pontos de falha e sobrecarga operacional. Critério de pronto: um processo principal para orquestrar salas e um processo/jogador por instância de teste, conforme necessário.

Questões

- **Concorrência nas cartas:** em quais trechos exigiremos exclusão mútua? Como provar ausência de corrida (logs, contadores, testes repetidos)?

- **Protocolos e uso:** qual protocolo de transporte e formato de mensagem usaremos com sockets nativos? Como padronizar códigos de erro e *acks*?

- **Interface do jogo:** qual será o nível mínimo (CLI simples? pequenas janelas?) apenas para validar regras e mensagens?

- **Biblioteca nativa de sockets:** quais chamadas/bloqueios/timeouts são necessárias? Usaremos modo bloqueante, não bloqueante ou multiplexação?

- **Linguagem:** qual linguagem facilita o uso direto de sockets nativos pela equipe e atende aos requisitos acima?

Metas

- **Estudar e aplicar Docker.**
  Entregar ambiente *up* com um comando (`docker compose up`), imagens pequenas e instruções para build/run.

- **Utilizar dois contêineres comunicando-se por sockets.**
  Prova de conceito: **dois contêineres trocando mensagens** (ex.: “ping/pong” e, depois, comandos do jogo/sala). Critérios: latência estável no ambiente local e logs claros de envio/recebimento.