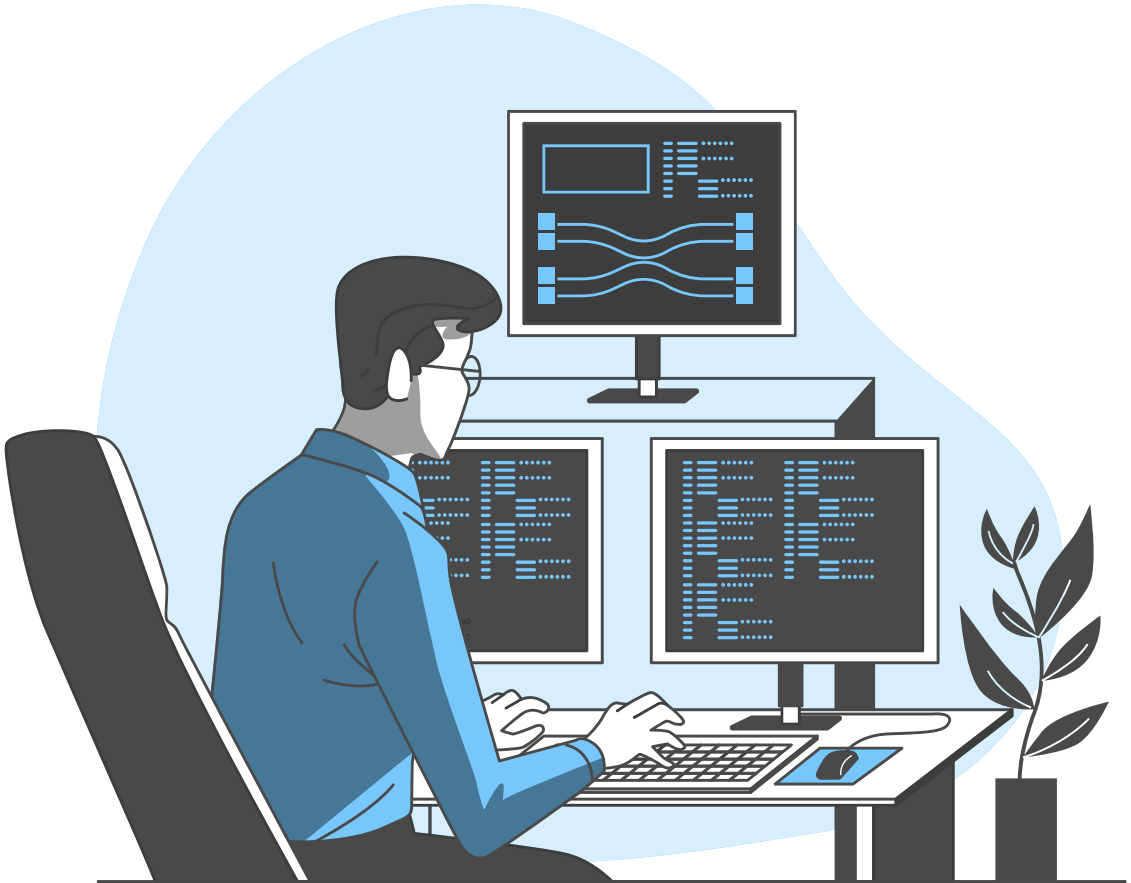


Distributed Black Jack

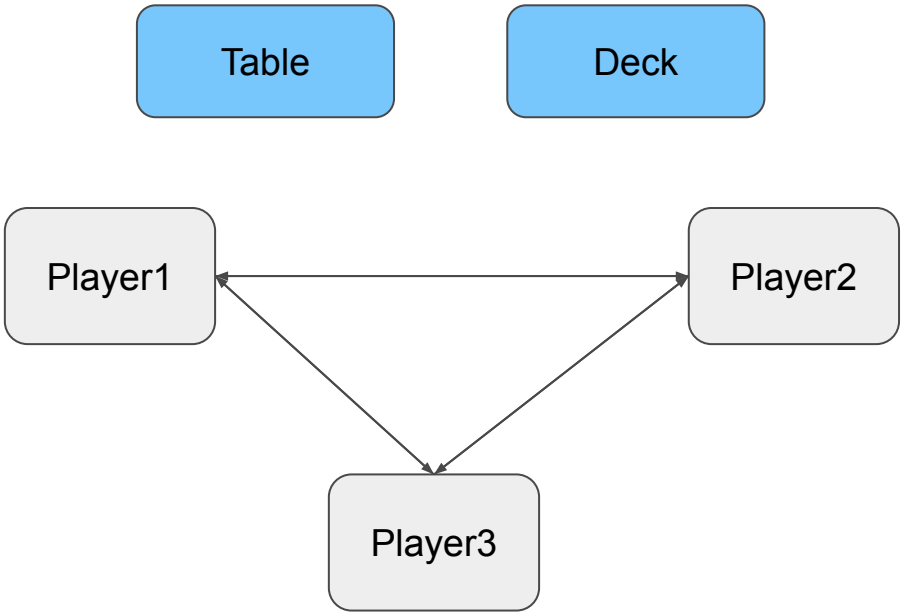
CD 2023

Maria Abrunhosa 107658
Marta Inácio 107826

Arquitetura

- Para desenvolver um jogo de BlackJack distribuído recorremos a:
  - Organização descentralizada, peer-to-peer;
  - Servidor para distribuição das cartas que atende um jogador de cada vez;
  - Servidor para guardar as cartas de cada jogador ;
  - Competição dos jogadores entre si.

Arquitetura



Classes implementadas

Classes

01

Class PaxosNode

Classe encarregue de definir se foi possível chegar a um consenso

02

Class Message

Classe que define o comando da mensagem

03

Class PlayMessage

Classe que define uma mensagem de jogada com a porta do jogador

04

Class NextMessage

Classe que define uma mensagem para passar a jogada a outro jogador, com a porta de quem acabou de jogar e pode enviar o score deste

Classes

05 Class WinMessage

Classe que define uma mensagem de vitória com a porta do jogador vencedor

06 Class DefeatMessage

Classe que define uma mensagem de derrota com a porta do jogador que perdeu

07 Class FairMessage e UnfairMessage

Classe que define uma mensagem a dizer se houve batota no jogo ou não, consoante o hash recolhido

08 Class BlackJackProto e BlackJackBadFormat

Classes de protocolo

Protocolo

- a troca de mensagens foi feita por mensagens JSON

Nome	Objetivo	Destino	Mensagem	Command
Play	Avisa que um jogador está a jogar	Peer	{"command": "{self.command}", "player": "{self.self_port}"}	play
Next	Avisa o jogador seguinte que é ele a jogar	Peer	{"command": "{self.command}", "player": "{self.self_port}"} ou {"command": "{self.command}", "player": "{self.self_port}", "score": "{self.score}"}	next
Win	Avisa os jogadores que venceu	Peer	{"command": "{self.command}", "player": "{self.self_port}"}	win

Protocolo

- a troca de mensagens foi feita por mensagens JSON

Nome	Objetivo	Destino	Mensagem	Command
Fair	Avisar o peer que as cartas do redis e de jogo são iguais	Peer	{"command": "{self.command}", "player": "{self.self_port}"}	fair
Unfair	Avisar o peer que as cartas do redis e de jogo não são iguais	Peer	{"command": "{self.command}", "player": "{self.self_port}"	unfair
Defeat	Avisa os jogadores que perdeu	Peer	{"command": "{self.command}", "player": "{self.self_port}"}	defeat

Funções mais importantes implementadas

Funções

check_fair_result(sock, player, ports, hash)

Com o hash das cartas que já saíram durante o jogo, avalia se houve ou não batota

...

get_card()

Vai buscar uma carta ao baralho através da mensagem “GC”

...

Funções

get_hashCards()

Vai buscar todas as cartas até ao momento jogadas através da mensagem “HC”

...

update_redis(port, cards)

Atualiza o redis com as cartas retiradas do baralho e guarda as cartas tiradas por um jogador identificadas pela porta deste

...

Funções

playalone() e play()

Simulam o jogo de BlackJack com 1, 2 e 3 jogadores

...

interact_with_user1(cards)

Interage com o utilizador de forma a que o jogador consiga inserir o que pretende fazer na jogada: (H)it, (S)tand, W(in) ou (D)efeat

...

FIM