Alan Soares Lima – Gerente de Projeto
Projeto Kariba

Contrato API Gateway (REST)

MÓDULO COMPLETO DO JOGO BASE

Descrição: **MÍNIMA ESTRUTURA BACKEND NECESSÁRIA** para o funcionamento original do jogo, listando as rotas (para lambdas) e os métodos (GET, POST, PUT, DELETE, etc) de cada uma.

startGame (**POST** recebendo JSON de um gameId)

- Cria o jogo em estado inicial: registra na tabela de cartas o id do jogo, id da carta e quantidade, e na tabela do tabuleiro registra o id do jogo, a posição e a quantidade de cartas presente nele, já na tabela de partida, é registrado o gameId, de quem é a vez, qual rodada está, o status da partida, ids de conexão e quem foi o vencedor. A lambda deve retornar apenas status 200 e “Jogo criado com sucesso” no body.

Exemplo de rota:
https://nfba1cx8uf.execute-api.sa-east-1.amazonaws.com/production/startGame

Exemplo de payload esperado do frontend:

```
{
   "gameId": '1'
}
```

exemplo tabela: cards

gameId	cardId	quantity
1	1	8
1	2	8
1	3	8
1	4	8
1	5	8
1	6	8
1	7	8
1	8	8

exemplo tabela: board

gameId	position	quantity
1	1	0
1	2	0
1	3	0
1	4	0
1	5	0
1	6	0
1	7	0
1	8	0

exemplo tabela: match

gameId	round	gameState	player1State	player2State	player1conId	player2conId	winner
1	0	waiting	waiting	waiting	None	None	None

exemplo tabela: score

gameId	connectionId	playerNumber	collectedCards
1	None	1	0
1	None	2	0

dealCards (**POST** recebendo os connectionId)

- Distribui no início do jogo aleatoriamente entre os dois jogadores as cartas de 1 a 8 (5 para cada um, lembrando que o jogo tem 64 cartas totais, 8 de cada), removendo a quantidade de cada carta da tabela de cartas e adicionando na tabela de deck do jogador, lembrar de receber no post os connectionId dos dois jogadores pelo event. Ao final, a lambda deve retornar status 200 e “Deck criado com sucesso” no body.

Exemplo de rota:
https://nfba1cx8uf.execute-api.sa-east-1.amazonaws.com/production/dealCards

Exemplo de payload esperado do frontend:

```
{
   "players": [
      {
         "connectionId": "eHTRvfKVGjQCJTA="
      },
      {
         "connectionId": "eHTlmc7LmjQCEoA"
      }
   ]
}
```

exemplo tabela: cards

gameId	cardId	quantity
1	1	6
1	2	8
1	3	5
1	4	8
1	5	8
1	6	6
1	7	7
1	8	7

exemplo tabela: deck

gameId	connectionId	cardId	quantity
1	eHTRvfKVGjQCJTA=	1	1
1	eHTRvfKVGjQCJTA=	2	0
1	eHTRvfKVGjQCJTA=	3	2
1	eHTRvfKVGjQCJTA=	4	0
1	eHTRvfKVGjQCJTA=	5	0
1	eHTRvfKVGjQCJTA=	6	1
1	eHTRvfKVGjQCJTA=	7	0
1	eHTRvfKVGjQCJTA=	8	1
1	eHTlmc7LmjQCEoA=	1	1
1	eHTlmc7LmjQCEoA=	2	0
1	eHTlmc7LmjQCEoA=	3	1
1	eHTlmc7LmjQCEoA=	4	0
1	eHTlmc7LmjQCEoA=	5	0
1	eHTlmc7LmjQCEoA=	6	1
1	eHTlmc7LmjQCEoA=	7	1
1	eHTlmc7LmjQCEoA=	8	1

getMatch (**GET** /{gameId})

- Consulta a tabela da partida, enviando uma mensagem ao websocket com todos os dados da tabela, a fim de iniciar o jogo e determinar qual jogador começa jogando. A lambda retorna 200 e “Partida encontrada” no body.

Exemplo de rota:
https://nfba1cx8uf.execute-api.sa-east-1.amazonaws.com/production/getMatch/1

Exemplo de mensagem a ser enviada ao websocket:

```
Data = json.dumps({
    'action': 'getMatch',
    'match': {
        'gameId': 1,
        'round': 0,
        'gameState': 'waiting',
        'player1State': 'waiting',
        'player2State': 'waiting',
        'player1conId': None,
        'player2conId': None,
        'winner': None
    }
})
```

exemplo tabela: match

gameId	round	gameState	player1State	player2State	player1conId	player2conId	winner
1	0	waiting	waiting	waiting	None	None	None

getConnections (**GET** ALL)

- Consulta tabela de conexões, manda para o websocket uma mensagem com os connectionId (no caso, dois), retorna 200 e “Consulta de conexões realizada com sucesso” no body.

Exemplo de rota:
https://nfba1cx8uf.execute-api.sa-east-1.amazonaws.com/production/getConnections

Exemplo de mensagem a ser enviada ao websocket:

```
# Conjunto de connectionsIds
connectionsIds = {'CONEXAO1', 'CONEXAO2'}

# Construindo uma lista de objetos, cada um com a chave 'connectionId'
Data = json.dumps({
   'action': 'getPlayers',
   'players': [{'connectionId': connId} for connId in connectionsIds]
})
```

startMatch (**POST** recebendo um JSON com os connectionId)

- Inicia a partida atualizando a tabela de partida, inserindo os valores das conexões, atualizando o round para 1 e o gameState para “in progress”, além de mudar o state do player 1 para “playing” e na tabela de pontuação inserir as conexões e o número do jogador. A lambda deve retornar status 200 e “Sucesso ao iniciar a partida” no body.

Exemplo de rota:
https://nfba1cx8uf.execute-api.sa-east-1.amazonaws.com/production/startMatch

Exemplo de payload esperado do frontend:

```
{
   "players": [
      {
         "connectionId": "eHTlmc7LmjQCEoA="
      },
      {
         "connectionId": "eHTRvfKVGjQCJTA="
      }
```

```
    ]
}
```

exemplo tabela: score

gameId	connectionId	playerNumber	collectedCards
1	eHTlmc7LmjQCEoA=	1	0
1	eHTRvfKVGjQCJTA=	2	0

exemplo tabela: match

gameId	round	gameState	player1State	player2State	player1conId	player2conId	winner
1	1	In progress	playing	waiting	eHTlmc7LmjQCEoA=	eHTRvfKVGjQCJTA=	None

getGameState (**GET** /{gameId})

- Consulta as tabelas: cards, deck, board, match e score através do gameId, e constrói uma mensagem a ser enviada ao front completa, com todas as informações do jogo. A lambda deve retornar status 200 e “Sucesso ao consultar estado atual do jogo” no body.

Exemplo de rota:
https://nfba1cx8uf.execute-api.sa-east-1.amazonaws.com/production/getGameState/1

Exemplo de mensagem a ser enviada ao websocket (esse exemplo seria depois de iniciar a partida, antes da primeira jogada ser feita):

```
Data = json.dumps({
    "action": "gameState",
    "gameState": {
        "cards": {
            "gameId": 1,
            "cards": [
                {"cardId": 1, "quantity": 6},
                {"cardId": 2, "quantity": 8},
                {"cardId": 3, "quantity": 5},
                {"cardId": 4, "quantity": 8},
                {"cardId": 5, "quantity": 8},
                {"cardId": 6, "quantity": 6},
                {"cardId": 7, "quantity": 7},
                {"cardId": 8, "quantity": 6}
            ]
        },
        "deck": {
            "gameId": 1,
            "players": [
                {
```

```
            "connectionId": "eHTRvfKVGjQCJTA=",
            "deck": [
               {"cardId": 1, "quantity": 1},
               {"cardId": 2, "quantity": 0},
               {"cardId": 3, "quantity": 2},
               {"cardId": 4, "quantity": 0},
               {"cardId": 5, "quantity": 0},
               {"cardId": 6, "quantity": 1},
               {"cardId": 7, "quantity": 0},
               {"cardId": 8, "quantity": 1}
            ]
         },
         {
            "connectionId": "eHTlmc7LmjQCEoA=",
            "deck": [
               {"cardId": 1, "quantity": 1},
               {"cardId": 2, "quantity": 0},
               {"cardId": 3, "quantity": 1},
               {"cardId": 4, "quantity": 0},
               {"cardId": 5, "quantity": 0},
               {"cardId": 6, "quantity": 1},
               {"cardId": 7, "quantity": 1},
               {"cardId": 8, "quantity": 1}
            ]
         }
      ]
   },
   "board": {
      "gameId": 1,
      "positions": [
         {"position": 1, "quantity": 0},
         {"position": 2, "quantity": 0},
         {"position": 3, "quantity": 0},
         {"position": 4, "quantity": 0},
         {"position": 5, "quantity": 0},
         {"position": 6, "quantity": 0},
         {"position": 7, "quantity": 0},
         {"position": 8, "quantity": 0}
      ]
   },
   "match": {
      "gameId": 1,
      "round": 1,
      "gameState": "in progress",
      "player1State": "playing",
      "player2State": "waiting",
```

```
        "player1conId": "eHTlmc7LmjQCEoA=",
        "player2conId": "eHTRvfKVGjQCJTA=",
        "winner": None
      },
      "score": {
        "gameId": 1,
        "players": [
          {
            "connectionId": "eHTlmc7LmjQCEoA=",
            "playerNumber": 1,
            "collectedCards": 0
          },
          {
            "connectionId": "eHTRvfKVGjQCJTA=",
            "playerNumber": 2,
            "collectedCards": 0
          }
        ]
      }
    }
})
```

makeMove (**POST** recebendo um JSON com gameId, position, quantity e o connectionId de quem fez a jogada)

- Realiza uma atualização na tabela board registrando as cartas jogadas no tabuleiro e decrementa da tabela deck (as cartas jogadas do determinado jogador). A lambda deve retornar status 200 e “Jogada realizada com sucesso” no body.

Exemplo de rota:
https://nfba1cx8uf.execute-api.sa-east-1.amazonaws.com/production/makeMove

Exemplo de payload esperado do frontend:

```
{
   "gameId": 1,
   "position": 3,
   "quantity": 2,
   "player": {"connectionId": "eHTRvfKVGjQCJTA="}
}
```

(Nos exemplos abaixo, o jogador da conexão “eHTRvfKVGjQCJTA=” jogou todas as duas cartas id 3 na posição 3)

exemplo tabela: board

gameId	position	quantity
1	1	0
1	2	0
1	3	*2*
1	4	0
1	5	0
1	6	0
1	7	0
1	8	0

exemplo tabela: deck

gameId	connectionId	cardId	quantity
1	eHTRvfKVGjQCJTA=	1	1
1	eHTRvfKVGjQCJTA=	2	0
1	eHTRvfKVGjQCJTA=	3	*0*
1	eHTRvfKVGjQCJTA=	4	0
1	eHTRvfKVGjQCJTA=	5	0
1	eHTRvfKVGjQCJTA=	6	1
1	eHTRvfKVGjQCJTA=	7	0
1	eHTRvfKVGjQCJTA=	8	1
1	eHTlmc7LmjQCEoA=	1	1
1	eHTlmc7LmjQCEoA=	2	0
1	eHTlmc7LmjQCEoA=	3	1
1	eHTlmc7LmjQCEoA=	4	0
1	eHTlmc7LmjQCEoA=	5	0
1	eHTlmc7LmjQCEoA=	6	1
1	eHTlmc7LmjQCEoA=	7	1
1	eHTlmc7LmjQCEoA=	8	1

*Obs. * (asterisco) apenas para ênfase na alteração, não deve existir no banco de dados.*

pickCard (**POST** recebendo gameId e connectionId)

- Completa deck do jogador aleatoriamente com cartas de 1 a 8 (tem que totalizar 5 em quantidade na mão do jogador) consultando a tabela de cartas do jogo e vendo qual a quantidade de cada carta disponível, removendo a quantidade de cada carta da tabela de cartas e adicionando na tabela de deck do jogador, lembrar de receber no post os gameId e connectionId do jogador a ser completado o deck pelo event. A lambda deve retornar status 200 e “Cartas compradas com sucesso” no body.

Exemplo de rota:
https://nfba1cx8uf.execute-api.sa-east-1.amazonaws.com/production/pickCard

(Nos exemplos abaixo, o jogador pegou uma carta 2 e uma carta 5, completando as 5 cartas necessárias na mão, já que na jogada anterior do exemplo o jogador baixou duas cartas)

exemplo tabela: cards

gameId	cardId	quantity
1	1	6
1	2	*7*
1	3	5
1	4	8
1	5	*7*
1	6	6
1	7	7
1	8	7

exemplo tabela: deck

gameId	connectionId	cardId	quantity
1	eHTRvfKVGjQCJTA=	1	1
1	eHTRvfKVGjQCJTA=	2	*1*
1	eHTRvfKVGjQCJTA=	3	0
1	eHTRvfKVGjQCJTA=	4	0
1	eHTRvfKVGjQCJTA=	5	*1*
1	eHTRvfKVGjQCJTA=	6	1
1	eHTRvfKVGjQCJTA=	7	0
1	eHTRvfKVGjQCJTA=	8	1
1	eHTlmc7LmjQCEoA=	1	1
1	eHTlmc7LmjQCEoA=	2	0
1	eHTlmc7LmjQCEoA=	3	1
1	eHTlmc7LmjQCEoA=	4	0
1	eHTlmc7LmjQCEoA=	5	0
1	eHTlmc7LmjQCEoA=	6	1
1	eHTlmc7LmjQCEoA=	7	1
1	eHTlmc7LmjQCEoA=	8	1

*Obs. * (asterisco) apenas para ênfase na alteração, não deve existir no banco de dados.*

updateMatch (**PUT** recebendo JSON com o gameId)

- Faz uma atualização na tabela match, para isso precisa consultar os valores da tabela, ver se quem estava jogando era o jogador1, caso for o jogador1 anteriormente, deve alterar o playerState (antes quem estava jogando era o player1, agora ele deve estar em “waiting” e quem joga é o player2), caso o jogador2 seja o resultado da última jogada, deve atualizar ele para “waiting” e o player1 para “playing” e vice e versa, além de atualizar o round para 2 (ou 3 e assim por diante) quando o último jogador a jogar tiver sido o player2, já que se o

último jogador a jogar foi o 2, significa que na próxima partida é outra rodada. A lambda deverá retornar status 200 e “Sucesso ao atualizar a partida” no body.

Exemplo de rota:
https://nfba1cx8uf.execute-api.sa-east-1.amazonaws.com/production/updateMatch

Exemplo de payload esperado do frontend:

```
{
   "gameId": 1
}
```

exemplo tabela: match

gameId	round	gameState	player1State	player2State	player1conId	player2conId	winner
1	1	In progress	waiting	*playing*	eHTlmc7LmjQCEoA=	eHTRvfKVGjQCJTA=	None

*Obs. * (asterisco) apenas para ênfase na alteração, não deve existir no banco de dados.*

collectCard (**PUT** recebendo JSON com o gameId e connectionId responsável por espantar o outro animal)

- Quando uma pilha de cartas chegar a 3 ou mais, atualiza o board "espantando" as cartas do animal próximo mais fraco, no exemplo abaixo seria coletar as cartas da posição 2 e armazenar na tabela de pontuação. a lambda deve retornar state 200 e “Animal próximo mais fraco espantado com sucesso” no body.

DETALHE IMPORTANTE: o elefante (8) espanta qualquer animal, mas o rato (1) espanta o elefante (8), após 3 cartas. o rato (1) é o único que espanta o elefante (8).

Exemplo de rota:
https://nfba1cx8uf.execute-api.sa-east-1.amazonaws.com/production/collectCard

Exemplo de payload esperado do frontend:

```
{
   "gameId": 1,
   "player": {"connectionId": "eHTlmc7LmjQCEoA="}
}
```

(Cenário hipotético, 3 cartas iguais na posição 3, onde o jogador a jogar a terceira carta foi a conexão “eHTlmc7LmjQCEoA=”, a carta mais fraca próxima era a da posição 2)

exemplo tabela: board

gameId	position	quantity
1	1	0
1	2	*2* vira 0 (mais fraca próx)
1	3	3 (3 ou mais cartas)

1	4	0
1	5	0
1	6	2
1	7	0
1	8	1

exemplo tabela: score

gameId	connectionId	playerNumber	collectedCards
1	eHTlmc7LmjQCEoA=	1	*2* (próximas chamadas devem sempre somar a esse valor ou ao de baixo)
1	eHTRvfKVGjQCJTA=	2	0

*Obs. * (asterisco) apenas para ênfase na alteração, não deve existir no banco de dados.*

endGame (**PUT** recebendo JSON com gameId e connectionId do usuário com mais cartas coletadas, isso pode ser obtido através da rota getGameState, nos parâmetros de score)

- Atualiza a partida na tabela match, encerrando a mesma, para isso precisa atualizar os campos gameState para “finished”, player1State e player2State para “stopped” e por fim atualizar o campo winner para o player vencedor. A lambda deve retornar status 200 e “Partida encerrada” no body.

Exemplo de rota:
https://nfba1cx8uf.execute-api.sa-east-1.amazonaws.com/production/endGame

Exemplo de payload esperado do frontend:

```
{
   "gameId": 1,
   "player": {"connectionId": "eHTlmc7LmjQCEoA="}
}
```

exemplo tabela: score

gameId	connectionId	playerNumber	collectedCards
1	eHTlmc7LmjQCEoA=	1	40
1	eHTRvfKVGjQCJTA=	2	24

exemplo tabela: match

gameId	round	gameState	player1State	player2State	player1conId	player2conId	winner
1	20	finished	stopped	stopped	eHTlmc7LmjQCEoA=	eHTRvfKVGjQCJTA=	1