Code Challenge: Autorizador

O desafio é implementar uma função que autoriza transações para uma conta específica seguindo uma série de regras predefinidas.

Entrada

Os **parâmetros** da função são:

- A transação a ser autorizada
- O estado da conta do cliente

A **transação** é feita para um comerciante (`merchant`), possui um determinado valor (`amount`) e o horário em que foi realizada (`time`).

O **estado da conta** do cliente inclui as seguintes informações:

- Se a conta está ativa (`active`)
- O saldo disponível da conta (`availableLimit`)
- O histórico de transações daquela conta (`history`)

Saída

O estado atual da conta junto de quaisquer violações da lógica de negócios. Se não houverem violações no processamento da operação, o campo `violations` deve retornar um vetor vazio [].

Regras de negócios

A autorização de uma transação deve seguir as seguintes regras:

- Nenhuma transação deve ser aceita para uma conta inativa: `account-not-active`
- O valor da **primeira** transação não deve exceder 90% do limite: `first-transaction-above-threshold`
- O valor da transação não deve exceder o limite disponível: `insufficient-limit`
- Não deve haver mais que 3 transações de qualquer comerciante em um intervalo de 2 minutos: `high-frequency-small-interval`
- Não deve haver mais que 1 transação similar (mesmo valor e comerciante) no intervalo de 2 minutos: `doubled-transaction`

Se todas as regras forem atendidas, o valor da transação autorizada deve ser subtraído do saldo da conta e o histórico de transações da conta deve ser atualizado.

Exemplo de uso

Os seguintes pseudo códigos ilustram o uso da função.

Caso de transação autorizada

```
transaction = {
    amount: 10,
    merchant: "Burger King",
    time: Date.now()
}
account = {
    active: true,
    availableLimit: 100,
    history: []
}

result = authorize(transaction, account)

result.account == {
    active: true,
    availableLimit: 90,
    history: [{
        amount: 10,
        merchant: "Burger King",
        time: 1629298219336
    }]
}
result.violations == []
```

Caso de transação rejeitada

```
transaction = {
    amount: 100,
    merchant: "Paris 6",
    time: Date.now()
}
account = {
    active: false,
    availableLimit: 100,
    history: []
}

result = authorize(transaction, account)

result.account == {
    active: false,
    availableLimit: 100,
    history: []
}
result.violations == ["account-not-active", "first-transaction-above-threshold"]
```

Dicas

Algumas dicas podem ser úteis para facilitar a execução do desafio. Tenha em mente que elas estão aqui apenas como sugestões e não precisam ser obrigatoriamente seguidas.

- Considere iniciar a implementação de uma regra de negócio e depois expanda para as demais violações;
- Considere em estruturar uma solução que seja extensível ao invés de querer apenas implementar todas as violações sugeridas;
- Considere implementar a solução como se estivesse implementando a lógica de negócio de um serviço. Evite se preocupar com fatores externos a lógica principal;
- Teste a sua solução!