Desafio - Fluxo de caixa via fila

Objetivos gerais

Avaliar o candidato (a) nos conceitos de orientação a objeto, qualidade de código, aplicação de padrões, resiliência, disponibilidade, performance, capacidade de monitoramento, testes em suas diversas formas e o bom uso do git (clareza dos commits, divisão do trabalho e melhores práticas).

Descrição

Uma empresa possui uma API para realizar lançamentos financeiros e gerir seus pagamentos, recebimentos e os encargos. Com o crescimento do volume de lançamentos foi escolhido o modelo de filas com o RabbitMQ para aumentar a capacidade de processamento.

Como analista seu trabalho é desenvolver:

- Uma Api de lançamentos financeiros;
    - Cada tipo de lançamento financeiro deve ser publicado em sua própria fila, ou seja exite uma fila de pagamentos e outra de recebimento;
    - Deve existir uma fila que consolida todos os laçamentos financeiros processados com sucesso, para compor o fluxo de caixa.
- Um consumidor que tenha responsabilidade de consolidar todas os lançamentos do mês em um fluxo de caixa.
- Uma API que retorne o fluxo de caixa do dia e dos próximos 30 dias, já com os os lançamentos futuros;
- É necessário que os dados enviados para lançamento sejam válidados;
- Deseja-se calcular a posição de caixa, que basicamente, mostra o saldo dos lançamentos do dia e se é positivo ou negativo em relação ao dia anterior.

Regras Gerais

- Dados dos lançamentos devem ser validados;
- Não é permitido fazer lançamentos no passado;
- A empresa tem um limite para ficar negativo (como se fosse um cheque especial de cartão), que não poder superar o valor de R$ 20.000,00, ou seja no dia a conta **não pode ficar mais de R$ 20.000,00, negativos**. Caso ultrapasse não será possivel fazer novos lançamentos no dia e qualquer tentativa de lançamento deve falhar e o motivo deve ser informado.
- Caso o valor da conta esteja negativo sobre o mesmo incidem juros de 0.83% ao dia.

Input de Dados

Lançamento financeiro(formato json)

O payload dos lançamentos financeiros

```
{
```

```
    "tipo_da_lancamento":"Tipos de lançamento (pagamento e recebimento)",
    "descricao": "Qualquer descrição sobre o pagamento",
    "conta_destino": "Conta do destinatário",
    "banco_destino": "Banco do destinatário",
    "tipo_de_conta": "Se a conta é corrente ou poupança",
    "cpf_cnpj_destino": "Numero formatado do CPF ou CNPJ",
    "valor_do_lancamento": "Valor em reais do lançamento no formato (R$
0.000,00)",
    "encargos": "Valor em reais dos encargos do lançamento no formato (R$
0.000,00)",
    "data_de_lancamento": "Data em que a lançamento o ocorreu no formato (dd-
mm-aaaa)"
}
```

Fluxo de caixa (formato json)

O retorno da chamada ao recurso de fluxo de caixa, que deve retornar todos os lançamentos do dia e lançamentos dos próximos 30 dias. Abaixo exibimos um exemplo que mostra os dados de um dos dias, esse dados se repete ao longo de 30 dias.

```
[
    {
        "data":"01-01-2018",
        "entradas": [{
          "data": "Data em que a entrada foi lançada no fluxo de caixa no
formato (dd-mm-aaaa)",
          "valor": "Valor da entrada em reais no formato (R$ 0.000,00)"
        }],
        "saidas":[{
          "data": "Data em que a saida foi lançada no fluxo de caixa no
formato (dd-mm-aaaa)",
          "valor": "Valor da saida em reais no formato (R$ 0.000,00)"
        }],
        "encargos":[{
          "data": "Data em que o encargo foi lançada no fluxo de caixa no
formato (dd-mm-aaaa)",
          "valor": "Valor do encargo em reais no formato (R$ 0.000,00)"
        }],
        "total": "Total de lançamentos do dia em reais no formato (R$
0.000,00)",
        "posicao_do_dia": "Comparando com o dia anterior se houve aumento do
caixa ou queda em percentual no formato (xxx.x%)",
    }
]
```

Pré-requisitos

- **Todo código deve estar no Github do candidato e deve ser informado ao término**

- Filas devem ser controladas usando RabbitMQ
- .Net Core ou .Net Full Framework (Livre escolha)
- Uso de WebApi;
- XUnit para testes;
- Caso escolha ter algum tipo de persistência fora a propria fila, faça com que esta persistência funcione sem a necessidade de qualquer instalação e/ou configuração prévia.

Diferenciais

- Usar uma configuração de ambiente com container(Docker).
- Usar documentação automática para a API.

OBSERVAÇÕES: Usar o pacote **RabbitMQ.Client nativo**, não é necessário fazer frontend, apenas documentar como utilizar e visualizar os dados da api usando um rest client (Postman por exemplo), **não é necessário criar filas dinamicamente, elas podem ser pré configuradas. É necessário apenas informa-las na documentação do projeto**.

Dicas

- Surpreenda-nos e lembre-se menos é mais!
- Nomes são uma das coisas mais importantes! 😃
- Lembre-se que o seu código é um espelho da sua qualidade!

Desejamos que divirta-se e aproveite ao máximo esse momento de desafio! 😃