## **LuizaLabs - Desafio técnico - Vertical Logística**

Você está na etapa do desafio técnico, Parabéns por ter chegado até aqui!

Neste desafio, queremos conhecer suas habilidades técnicas em foco prático e aplicado na resolução de um problema com nuances aproximadas do seu dia-a-dia em nosso time. Aqui conheceremos seu estilo de código, aptidões técnicas, seus hard skills e, sobretudo, a capacidade de resolução de problemas =) .

### O desafio

Temos uma demanda para integrar dois sistemas. O sistema legado que possui um arquivo de pedidos desnormalizado, precisamos transformá-lo em um arquivo json normalizado. E para isso precisamos satisfazer alguns requisitos.

### Objetivo do desafio

Faça um sistema que receba um arquivo **via API REST** e processe-o para ser retornado via **API REST**.

### Entrada de dados

O arquivo do sistema legado possui uma estrutura em que cada linha representa uma parte de um pedido. Os dados estão padronizados por tamanho de seus valores, respeitando a seguinte tabela:

| campo | tamanho | tipo |
| --- | --- | --- |
| id usuário | 10 | numérico |
| nome | 45 | texto |
| id pedido | 10 | numérico |
| id produto | 10 | numérico |
| valor do produto | 12 | decimal |
| data compra | 8 | numérico ( formato: yyyymmdd ) |

Observação: todos os campos **numéricos** são completados com ‘0’ à esquerda. Os demais com espaço à esquerda. Ponto importante, a formatação das colunas sempre será igual.

Dados de exemplo (a primeira linha não consta no arquivo):

|-userId--|--------------userName----------------------|-orderId-|-prodId--|---value---|-date--|

0000000002 Medeiros00000123450000000111 256.2420201201

0000000001 Zarelli00000001230000000111 512.2420211201

0000000001 Zarelli00000001230000000122 512.2420211201

0000000002 Medeiros00000123450000000122 256.2420201201

### Saída de dados

A saída de dados deverá ser disponibilizada via api REST considerando a estrutura base de payload de response:

[

{

"user\_id":1,

"name":"Zarelli",

"orders":[

{

"order\_id":123,

"total":"1024.48",

"date":"2021-12-01",

"products":[

{

"product\_id":111,

"value":"512.24"

},

{

"product\_id":122,

"value":"512.24"

}

]

}

]

},

{

"user\_id":2,

"name":"Medeiros",

"orders":[

{

"order\_id":12345,

"total":"512.48",

"date":"2020-12-01",

"products":[

{

"product\_id":111,

"value":"256.24"

},

{

"product\_id":122,

"value":"256.24"

}

]

}

]

}

]

Considere a consulta geral de pedidos e, também, a inclusão de filtros:

* id do pedido;
* intervalo de data de compra (data início e data fim);

### Arquivos

Os arquivos estão em anexo ao e-mail com o desafio técnico!

## **Key words**

* **Testes**
* Lógica
* **Simplicidade**
* SOLID
* Linguagem (**não estamos falando de framework**)
* Automação (Ex: Build, Coverage)
* Desenho da API
* Git

## **TL;DR**

A modelagem / arquitetura do sistema fica a seu critério, bem como a seleção e o uso de Frameworks e linguagem fica de livre escolha, é importante focar na **simplicidade**.

Deixe claro na documentação, pode ser no readme, as escolhas utilizadas, ao que tange tecnologia e padrões arquiteturais aplicados na resolução.

Independente da solução, é sempre legal colocar no **README** sua maneira de execução.

Sobre o output (Retorno da API REST), segue o mesmo princípio, pode usar a maneira mais benéfica de persistência, exemplo: arquivo, banco de dados, stream, etc…

O mais legal desse desafio é sua versatilidade e ver a **lógica implementada da leitura e tratamento dos dados**.