Propósito

Este desafio tem como objetivo avaliar sua forma de resolver problemas. Portanto, a estrutura e a qualidade do código, bem como as tecnologias e técnicas utilizadas vão dizer muito sobre a metodologia utilizada.

Este desafio avaliará as seguintes habilidades:

- Entendimento e processamento de séries temporais financeiras.
- Eficiência no armazenamento dos dados
- Construção de uma API REST que responda às perguntas-chave, definidas nas especificações abaixo

Sobre o Desafio

A Easynvest é uma corretora de valores que ajuda milhares de clientes a investir seu dinheiro de uma forma fácil e ágil. Nossos produtos oferecem uma plataforma online e mobile de negociação de títulos públicos entre pessoas físicas e o programa do Tesouro Direto do governo. Esses títulos possuem uma característica de baixo risco, atraindo pessoas com perfil de investimento mais conservador.

Dentro do Tesouro existem dois tipos de título que podem ser adquiridos:

- **Pré-fixados**: nos quais você sabe exatamente a rentabilidade que irá receber se mantiver o título até a data de vencimento.
- **Pós-fixados**: nos quais a rentabilidade até o vencimento variam de acordo com um indexador (geralmente a taxa básica de juros Selic ou a inflação - IPCA). Nesse modelo, a rentabilidade é composta por um valor base mais a variação do indexador.

Existem diversos títulos oferecidos dentro dessas categorias. Você pode conhecer mais sobre eles aqui.

Seu primeiro trabalho consiste em:
1) Processar o arquivo (retirado do site do Tesouro Nacional).
Esse arquivo contém o histórico de venda e resgate dos diversos títulos do Tesouro
2) Armazenar as informações obtidas da melhor forma possível.
Sinta-se à vontade para escolher a melhor forma de guardar esses dados, desde que você siga as orientações colocadas na seção “Stack de Tecnologias”.

Alguns conceitos úteis:
- Venda : Ato do governo dar um título para uma pessoa física em troca de um valor monetário.
- Resgate: Ato do governo pegar o título de volta após pagar uma pessoa física.

Seu segundo trabalho será construir uma API REST que execute as funções abaixo. Atente-se para as estruturas de Request, Parameters e Body descritas em cada uma das chamadas

1. Adicionar um valor monetário a um título

Request: POST /titulo_tesouro

Parameters:

- categoria_titulo: obrigatório {NTN-B, LTF, ...}
- mês: obrigatório
- ano: obrigatório
- ação: obrigatório {venda, resgate}
- valor: obrigatório

Response: fica a seu critério

2. Remover um valor monetário de um título

Request: DELETE /titulo_tesouro/{id}

Response: fica a seu critério

3. Atualizar um valor monetário para um título específico

Request: PUT ou PATCH ou UPDATE /titulo_tesouro/{id}

Parameters:

- mês: opcional
- ano: opcional
- ação: opcional
- valor: obrigatório

Response: fica a seu critério

4. Mostrar o histórico de um título específico

Request: GET /titulo_tesouro/{id}

Parameters:

- data_inicio: opcional
- data_fim: opcional

- group_by: {ano} opcional. Ao preencher esse parâmetro, a API deve retornar a soma dos valores agrupados por ano

Response:

```
{
    id: 5,
    categoria_titulo: "NTN-B",
    historico: [
        {
            mes: 2, // caso o parâmetro group_by seja implementado, remover esse attr
            ano: 2013,
            valor_venda: "2.349,00",
            valor_resgate: "2.349,00"
        },
        ...
    ]
}
```

5. Comparar o histórico de dois ou mais títulos

Request: GET /titulo_tesouro/comparar/

Parameters:

- ids: obrigatório. Lista de Ids dos títulos a serem comparados
- data_inicio: opcional
- data_fim: opcional
- group_by: {ano} opcional. Ao preencher esse parâmetro, a API deve retornar a soma dos valores agrupados por ano.

Response:

```
{
    ano: 2013,
    mes: 2, // caso o parâmetro group_by seja implementado, remover esse attr
    categoria_titulo: NTN-B,
    valores: [
        {
            id: 5,
            categoria_titulo: "NTN-B",
            valor_venda: "2.349,00",
            valor_resgate: "2.349,00"
        },
        {
            id: 13,
            categoria_titulo: "LTF",
            valor_venda: "2.349,00",
```

```
            valor_resgate: "2.349,00"
        },
        ...
    ]
}
```

6. Buscar valores de venda de um título em determinado período

Request: GET /titulos_tesouro/venda/{id}

Parameters:

- data_inicio: opcional
- data_fim: opcional
- group_by: {ano} opcional. Ao preencher esse parâmetro, a API deve retornar a soma dos valores agrupados por ano.

Response: fica a seu critério

7. Buscar valores de resgate de um título em determinado período

Request: GET /titulos_tesouro/resgate/{id}

Parameters:

- data_inicio: opcional
- data_fim: opcional
- group_by: {ano} opcional. Ao preencher esse parâmetro, a API deve retornar a soma dos valores agrupados por ano.

Response: fica a seu critério

Stack de Tecnologias

- Linguagem de backend: Python ou R.
- Armazenamento: Arquivos de texto, arquivos binários, tecnologia de bancos de dados open source (relacionais ou não-relacionais).

Não há restrições para uso de bibliotecas que possam otimizar o seu código. ***But, be careful!*** Não esperamos que você use apenas código dos outros. Queremos conhecer bastante o código que você é capaz de desenvolver.

Não basta apenas você saber executar o projeto

É importante que seu projeto tenha uma documentação consistente que permita a qualquer pessoa instalar as dependências necessárias e executá-lo localmente.

Dica: ter o seu projeto em um repositório online é uma boa ideia, afinal de contas sempre trabalhamos em time.

O que esperamos ver ao final?

Nosso time está bastante curioso para ver e analisar o seu código. E queremos compartilhar com você alguns dos pontos que avaliaremos no seu projeto:

1. **Interpretação e metodologia de resolução:** mostrar de forma clara, por meio da estrutura do código e da documentação, a estratégia adotada para a resolução do problema, bem como todas as premissas assumidas e suas razões.
2. **Manutenibilidade**: é um código legível e de fácil manutenção. Segue premissas claras de padronização de código.
1. **Design**: como você separou as responsabilidades e por quê.
2. **Qualidade**: é bem importante que seu projeto possua alguns testes unitários. Lembre-se que dados públicos ou de fontes desconhecidas sempre podem conter inconsistências.
3. **Desempenho**: escreveu um código que tem uma boa performance? Não esperamos a solução ótima, mas é interessante saber identificar pontos de melhoria e otimização.

O tempo estimado de criação desse projeto é de 5 a 6 horas.

BOA SORTE! =)