Teste

Em um futuro distante, na cadeia de evolução, os símios e os humanos estão cada vez mais semelhantes. Por esse motivo ficou muito difícil distinguir quem é humano e quem é símio.

Você foi contratado para desenvolver um projeto em Node.js, que vai identificar se uma sequência de DNA pertence a um humano ou a um símio.

O projeto consiste em desenvolver uma **API REST**, e disponibilizar **um endpoint HTTP POST "/simian".** Esse endpoint receberá como **parâmetro**, um **JSON** com a sequência de DNA (**Array de Strings)**, onde, cada elemento desse array representa uma linha de uma tabela **quadrada de (NxN)**, Como no exemplo abaixo:

```
POST → /simian
{
   "dna": ["CTGAGA", "CTATGC", "TATTGT", "AGAGGG", "CCCCTA", "TCACTG"]
}
```

A	T	G	C	G	A
C	A	G	T	G	C
T	T	A	T	T	T
A	G	A	C	G	G
G	C	G	T	C	A
T	C	A	C	T	G

Humano

C	T	G	A	G	A
C	T	A	T	G	C
T	A	T	T	G	T
A	G	A	G	G	G
C	C	C	C	T	A
T	C	A	C	T	G

Símio

- Você saberá se um DNA pertence a um símio, se encontrar **2 ou mais sequências de quatro letras iguais** em qualquer direção, horizontal, vertical ou nas diagonais.
- As letras da String **só podem ser: (A, T, C, G)**

A **API** deve retornar um json com **"is_simian": boolean**. Caso você identifique um **símio**, deve ser **true**, caso identifique um **humano**, deve ser **falso**, como no exemplo abaixo:

```
HTTP 200
{"is_simian": true}
```

Desafios

Nível 1:

Desenvolva uma **API** que esteja de acordo com os requisitos propostos acima, que seja capaz de validar uma sequência de DNA e identificar corretamente símios e humanos

Nível 2:

Use um banco de dados de sua preferência para **armazenar** os DNAs verificados pela API. Esse banco deve garantir a unicidade, ou seja, **apenas 1 registro por DNA**.

Disponibilizar um outro **endpoint "/stats"** que responde um **HTTP GET**. A resposta deve ser um Json que retorna as estatísticas de verificações de DNA, onde deve informar a **quantidade de DNA's símios**, **quantidade de DNA's humanos**, **e a proporção de símios para a população humana**. Segue exemplo da resposta:

```
{"count_simian_dna": 40, "count_human_dna": 100: "ratio": 0.4}
```

Nível 3:

Dockerize sua aplicação: Crie um arquivo Dockerfile para criar um container da sua aplicação Node.js e depois gere o docker-compose de uma forma que basta apenas rodarmos: docker-compose up -d que tudo deverá estar funcionando e disponível na porta: 8000.

O que entregar

- Código-fonte
    - Criar um repositório **privado** no Github
    - Adicionar o usuário **perfectflight** como colaborador para que possamos ter acesso ao código.
    - Se o repositório estiver **público**, será automaticamente **desqualificado**.
- Instruções com documentação sobre como executar a API. (README).

Observações:

- Considere a performance, organização do código e boas práticas (clean code e clean architecture).
- Tenha em mente que faremos uma série de testes com matrizes válidas e inválidas.
- O projeto deve conter **testes automáticos**, com uma boa **cobertura do código**.