Árvore genealógica

O Problema

Nesse desafio deve-se criar uma API que expõe endpoints *HTTP* para a listagem, leitura, criação, edição e deleção (B.R.E.A.D.) de entidades do tipo "pessoa" (*Person*).

```
{
  "name": "string"
}
```

E entidades do tipo "relações de paternidade" (**Relationship**).

```
{
 "parent": "Person.identifier",
 "children": "Person.identifier"
}
```

> **Importante**: As entidades podem ter mais campos que os do exemplo , mas devem conter os campos mostrados acima.

A API deve prover um endpoint que retorna a árvore genealógica de um indivíduo contendo todos os ascendentes possíveis até o seu nível. Por exemplo, suponha a seguinte família:

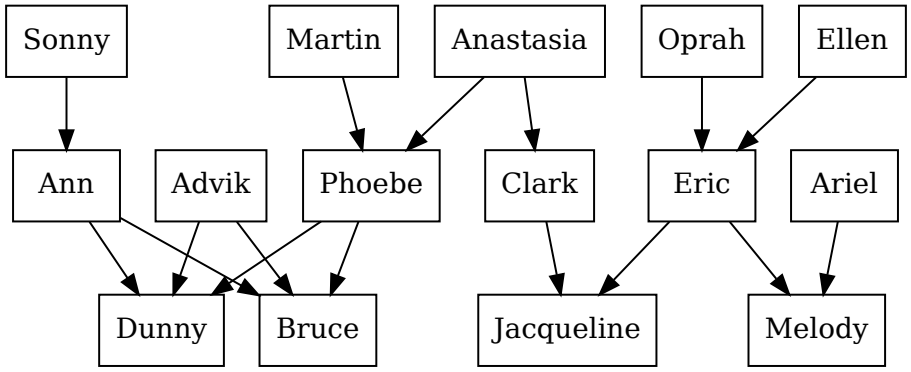

Figure 1: Árvore completa.

Se procurarmos pelo identificador de `Bruce`, a API deve retornar: `Bruce`, `Ann`, `Advik`, `Phoebe`, `Sonny`, `Martin` e `Anastasia`
Já se buscarmos pelo identificador de `Phoebe`, devem retornar os membros `Phoebe`, `Martin` e `Anastasia`.

O retorno deve prover todas as informações para que quem o consuma consiga construir a árvore, minimizando informações redundantes. Por exemplo:

```
{
  "members":[
    {
      "name": "Phoebe",
      "relationships": [
        {
          "name": "Martin",
          "relationship": "parent"
        },
        {
          "name": "Anastasia",
          "relationship": "parent"
        }
      ]
    },
    {
      "name": "Martin",
      "relationships": []
    },
    {
      "name": "Anastasia",
      "relationships": []
    }
  ]
}
```

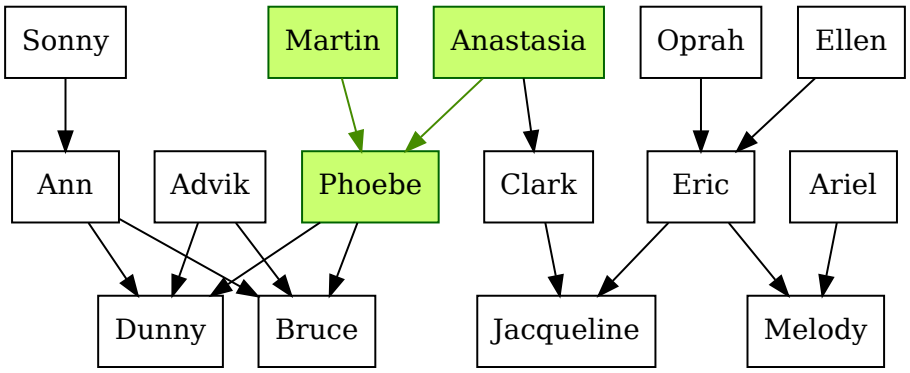

Figure 2: Árvore da Phoebe.

Importante: O retorno não precisa seguir o exemplo dado.

Importante: A árvore pode ter sua profundidade limitada, mas se sua implementação seguir por esse caminho, o valor de limite deve ser configurável ou por uma variável de ambiente, ou como parâmetro da requisição.

Entrega

O desafio deve ser entregue em um repositório GIT com a evolução do projeto. O repositório deve conter além do código, um Dockerfile para a construção de uma imagem do container da aplicação e um arquivo Docker Compose, que contenha tudo necessário para que a aplicação execute normalmente (como: banco de dados, variáveis de ambiente, serviços auxiliares, etc.). Além de um arquivo de documentação explicitando o uso da API, este arquivo pode ser o próprio README do projeto ou algo como um arquivo OpenAPI, a documentação da API deve ser exaustiva, ou seja, todo endpoint assim como sua entrada e possíveis retornos devem estar documentados.

Importante: A história e evolução do projeto é importante.

Extras

A implementação das seguintes funcionalidades são requisitos excitantes, porém não necessários. Estes requisitos são mais complicados do que o projeto original, mas dão a oportunidade do candidato mostrar habilidades específicas:

- Adição dos nós de filhos, irmãos, tios, tio-avós, etc., Ou seja, a árvore deve contem os filhos de todos os nós da árvore original. Exemplo: Ao buscar pelo identificador de `Phoebe`, deve-se retornar: `Phoebe`, `Dunny`, `Bruce`, `Clark`, `Martin` e `Anastasia`.

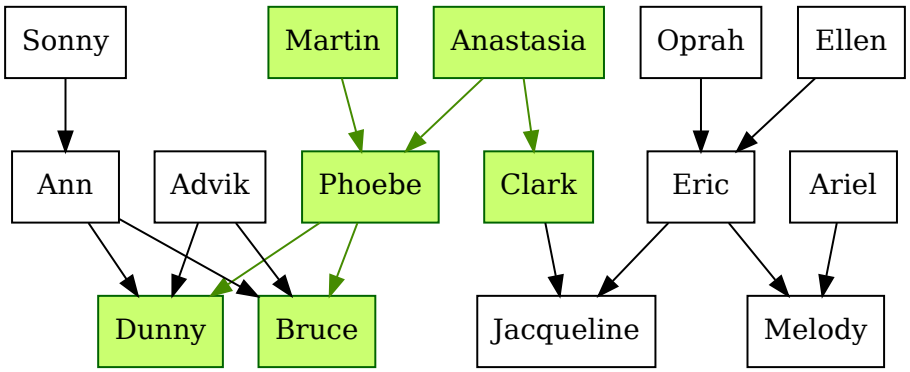

Figure 3: Árvore estendida da Phoebe.

- O cliente deve poder escolher o formato da entrada e saida da API seguindo o padrão de negociação de conteúdo. Possíveis valores seriam: `application/json`, `application/xml` e `application/yaml`

- Inferência de parentescos.
  Por exemplo, ao realizar a busca por `Bruce` e `Jacqueline` deve-se retornar `cousins`.
- Calculo do Bacon's Number entre duas pessoas.
  Por exemplo: O *Bacon's Number* entre `Bruce` para `Jacqueline` é quatro, pois são necessárias quatro arestas para alcançar a `Jacqueline` partindo do `Bruce`, são elas: `Bruce -> Phoebe`, `Phoebe -> Anastasia`, `Anastasia -> Clark` e `Clark -> Jacqueline`).

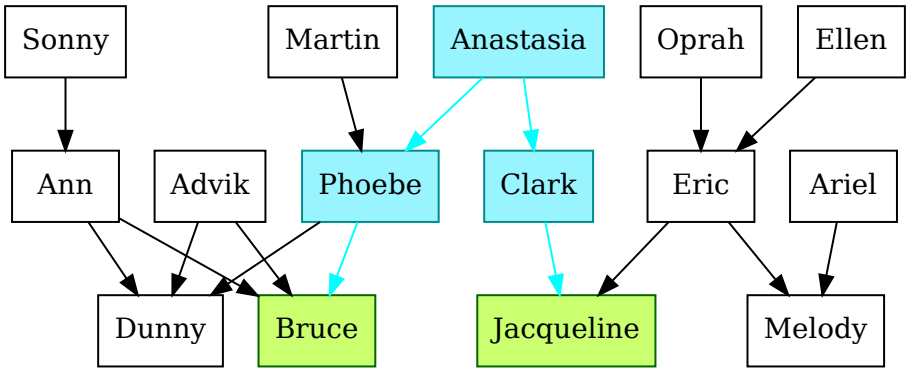

Figure 4: Bacon's Number entre Bruce e Jacqueline.

- Adição da relação do tipo `spouse` entre dois membros que compartilham filhos. (Essa adição deve modificar o *Bacon's Number* entre eles, se implementado).

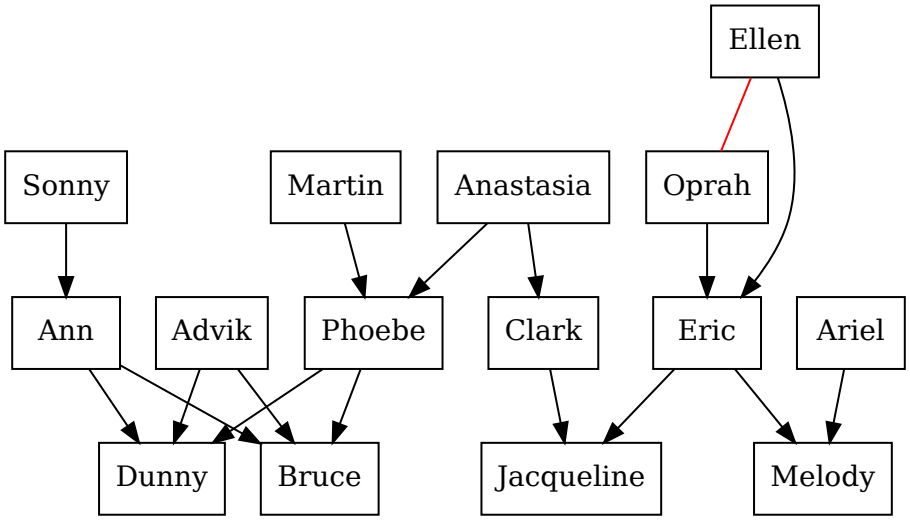

Figure 5: Árvore com a relação de spouce entre Ellen e Oprah.

- Impedir prole incestuosa, isso pode ser feito com a seguinte restrição, uma inserção de parentesco entre A e B pode ser feita se, e somente se, A não aparece na árvore de B e B não aparece na árvore de A, salvo onde o pai já é pai de algum irmão do nó. Para essa lidar com essa restrição pode ser definido um valor mínimo de distância, para que relações entre membros de uma mesma árvore suficientemente distantes possam ter um filho, neste caso esse valor deve ser configurável por uma variável de ambiente.