Teste INMETA – Back-End Pleno

API de Gerenciamento de documentação de colaboradores

Seu objetivo é desenvolver uma API RESTful com foco na criação e acompanhamento do fluxo de **documentação obrigatória** de colaboradores. Cada colaborador será vinculado a tipos de documentos específicos que são obrigatórios para envio.

Especificações funcionais

Exemplo de fluxo

Colaborador X foi vinculado á dois tipos de documentos, **CPF** e **Carteira de Trabalho**, portanto ele está com dois documentos pendentes de envio. O usuário da aplicação enviou o documento **CPF** do **colaborador X**, então agora o **colaborador X** está apenas com o documento **Carteira de Trabalho** pendente para envio.

Funcionalidades esperadas

- Cadastro de colaborador
- Atualização de colaborador
- Cadastro de tipo de documento
- Vinculação e desvinculação de um colaborador com tipos de documentos
  - Deve ser possível vincular e desvincular mais de um tipo de documento por vez
- Enviar um documento relacionado ao tipo de documento e ao colaborador
  - Não é necessário enviar um arquivo com conteúdo, apenas a representação dele como um documento já é suficiente
- Obter o status da documentação de um colaborador específico, mostrando quais foram enviados e quais ainda estão pendentes de envio
- Listar todos os documentos pendentes de todos os colaboradores
  - Incluir paginação
  - Incluir filtros opcionais por colaborador e por tipo de documento

Especificações técnicas

Entidades principais (sugestão)

- Employee
  - id
  - name
  - document (cpf)
  - hiredAt

- DocumentType
  - id
  - name
- Document
  - id
  - name
  - status
  - employeeId
  - documentTypeId

> 💡 Fique a vontade para criar mais entidades, adicionar ou renomear campos caso seja necessário dentro da sua ideia de resolução do desafio, acima são apenas sugestões de entidades para exemplificar melhor o desafio.

Tecnologias

A aplicação deve ser uma **API RESTful** de entrada/saída JSON desenvolvida em **Node.JS** com **Typescript**. Fique a vontade para escolher algum framework, segue abaixo algumas sugestões:

- TS.ED
- Nest.JS
- Fastify
- Express

Para persistência dos dados você tem a liberdade para escolher qualquer banco de dados além da biblioteca utilizada na comunicação com o mesmo.

Restrições

- O projeto deve estar no GitHub
- Não deve ser feito fork de nenhum outro projeto
- Apenas o seu usuário deve realizar commits no projeto

Extras (opcional)

Vamos propor alguns desafios extras caso você queira implementa-los. Nenhum desses requisitos é obrigatório, mas são desejados e podem ser um diferencial.

1. **Testes automatizados:** sejam unitários e/ou funcionais, testes automatizados são importantes e ajudam a evitar problemas no futuro.
2. **Tratamento de Erros:** como você trataria os erros da sua aplicação?
3. **Documentação do Sistema:** sua aplicação provavelmente precisa ser construída antes de ser executada. Você consegue documentar como outra pessoa que pegou sua aplicação pela primeira vez pode construir e executa-la?
4. **Deploy:** você consegue realizar o deploy da sua aplicação?

Como revisamos e o que avaliamos

Seu teste será analisado por pelo menos dois de nossos engenheiros. Levamos em consideração seu nível de experiência.

Valorizamos a qualidade acima da integralidade dos recursos. Um dos objetivos deste desafio é nos ajudar a identificar o que você considera código pronto para produção, ou seja, você deve considerar este código pronto para a revisão final com seu colega.

Os aspectos do seu código que avaliaremos incluem:

- **Arquitetura:** quão clara é a separação entre camadas e responsabilidades?
- **Modelagem:** os modelos de dados são bem definidos e estruturados?

- **Correção**: a aplicação faz o que foi solicitado? Se houver algo faltando, o README explica o motivo?
- **Qualidade do código**: o código é simples, fácil de entender e de fácil manutenção? Há algum code smell ou outros sinais de alerta? O estilo de codificação é consistente em toda a base de código?
- **Escolhas técnicas**: as escolhas de bibliotecas, bancos de dados, arquitetura etc. parecem apropriadas para a aplicação escolhida?
- **Commits:** definem bem a alteração realizada, são bem divididos e demonstram a evolução do sistema?
- **Testes (se houver)**: quão completos são os testes automatizados? Existem alguns testes unitários e alguns de integração?
    - Não estamos buscando uma cobertura completa (dada a restrição de tempo), mas apenas tentando ter uma ideia de suas habilidades em testes.