



Back-end Pleno

Você está recebendo um desafio voltado para o desenvolvimento Backend. Leia abaixo os requisitos e tecnologias para que você possa sair bem neste teste:

Requisitos

- Boas práticas de código (clean code)
- Node.js
- Banco de dados SQL - pgsql/mysql/mariadb
- Typescript
- REST API
- GIT - Versionamento
- RabbitMQ/MQTT

Diferenciais

- Vue.js
- PHP - Laravel
- Linux - Conhecimentos básicos
- Apache
- Testes unitários
- Ecossistema AWS (EC2, S3, IAM)

Entrega

- O código fonte deve ser desenvolvido e disponibilizado no github.
- Crie um Readme.md com instruções de como executar sua aplicação.
- Você tem um prazo de 5 dias para entregar o máximo que conseguir.

O Desafio

Este teste consiste em 2 partes: um projeto desenvolvido em Node.js e algumas questões referentes a alterações de sistema.

Projeto Node.js

Utilizando-se de Node.js (Express, Nest, Fastify, etc), com um banco relacional, desenvolva um sistema de cadastramento de cursos, no formato de uma API REST. O sistema deverá conter, pelo menos, 3 módulos sendo eles:

1. Usuários, contendo:
 - a. Primeiro Nome, Último Nome, Avatar, Tipo (Admin ou Aluno).
2. Cursos
 - a. Nome, Setor, Duração.
3. Atividade
 - a. Nome, Peso (Nota máxima da atividade).

Regras de Negócio:

Cada usuário poderá ser cadastrado em apenas um único curso.

Cada curso poderá ter várias atividades.

Cada atividade terá a possibilidade de anexar diversos documentos, somente, em formato pdf.

Deverá ter um sistema para download de todos os anexos de uma mesma atividade de uma só vez.

O cadastro de usuários e cursos poderá, apenas, ser feito por um administrador autenticado.

Ao cadastrar uma nova atividade é necessário que seja enviado para uma fila do RabbitMQ para ser consumida pela aplicação frontend (Não é necessário ter um consumidor, apenas que o serviço de mensageria esteja funcionando). Além de uma notificação por e-mail para os alunos cadastrados no respectivo curso.

Rotas Obrigatórias:

- GET:/users

- Irá retornar todos os usuários cadastrados no banco.
- GET:/users/:id
 - Irá retornar os dados de um usuário específico e o curso em qual está matriculado.
- GET:/user/:id
 - Puxa os dados de um usuário da api <https://reqres.in/api/users> em JSON.
- GET:/user/avatar/:id
 - Na primeira vez que for feita a requisição irá pegar o avatar cadastrado na api <https://reqres.in/api/users> do usuários de respectivo id e irá salvar localmente.
 - Nas requisições subsequentes irá pegar a imagem salva na máquina.

Questões Teóricas

- 1) Uma empresa de vagas de emprego possui um sistema para gerenciar os usuários que foram empregados usando a sua plataforma. O sistema foi desenvolvido com uma regra estrita para cada usuário poder estar ativo em apenas uma empresa por vez. Porém, após uma reestruturação, a empresa começou a trabalhar também com vagas PJ, permitindo que um usuário estivesse ativo em mais de uma empresa ao mesmo tempo. Como você conduziria as alterações desse sistema?
- 2) Em um sistema de cadastramento de campanhas, depois de 3 meses em produção, notou-se que o desempenho do site estava sendo comprometido pelo tamanho das imagens cadastradas pelos usuários. Dessa forma é necessário que uma ação seja tomada para reduzir o tamanho destes arquivos, sem perder os arquivos já cadastrados. É possível resgatar o caminho dessas imagens a partir das campanhas a qual elas estão atreladas. Como faria o tratamento das imagens já existentes no projeto?