KRUZER

IMPROVING YOUR DIGITAL EXPERIENCE

TESTE DEV FULLSTACK MEAN

Sumário

Proposta

Crie uma aplicação backend com NodeJS (v12) que irá export uma API RESTFULL de cadastro, gestão e visualização de usuário (CRUD) e crie uma aplicação frontend com Angular (v9) que irá possibilitar, a criação, edição, exclusão e listagem de usuários através da API.

Todos os endpoints devem aceitar e somente enviar JSONs. O servidor deverá retornar um JSON padrão caso o endpoint não tenha sido encontrado ou não exista e status code correspondente ao protocolo HTTP.

A aplicação backend deve persisitir os dados usando banco não relacional.

Todas as mensagens e estruturas de dados devem ser em inglês.

Todas as respostas de erro devem retornar um objeto na seguinte estrutura:

```
{
        “message”: “error message”
}
```

O frontend deve ter uma tela de listagem e uma tela de criação/edição. O botão de exclusão deve estar presente na listagem de usuários.

Modelo de usuário

Todo usuário deve ter as seguintes propriedades:

{

_id: “123456789”,

name: “Kruzer”

lastName: “Inc”,

email: “kruzer@kruzer.io”,

password: “secret”,

createdAt: “2020-05-27 21:20:30.848Z”,

updatedAt: “2020-05-27 21:20:30.848Z”,

birthday: “2020-05-27 21:20:30.848Z”

}

- _id e email são campos unique.
- email, password, name, lastName e birthday são required.

API – Listar Usuários

- Este endpoint deverá retornar todos os usuários e deverá aceitar filtros de acordo com o modelo de Usuário.
- Deverá retornar status code seguindo padrão de protocolo HTTP.

Front – Listar Usuários

- Deve-se apresentar uma listagem de usuários que não estejam deletados.
- Os campos apresentados na listagem devem ser _id, name, email, data de criação, botões de editar e excluir.

API – Criar Usuários

- Este endpoint deverá receber um usuário no corpo da requisição com os campos definidos no modelo.
- Deverá retornar status code de acordo com protocolo HTTP.
- Id: Pode ser o id gerado pelo banco
- name: Nome do usuário
- lastName: Sobrenome do usuário
- email: Email do usuário
- password: Senha do usuário
- createdAt: Data de criação. (Imutável)
- updatedAt: Data de criação.
- birthday: Data de aniversário do Usuário. (Pode ser salvo no padrão de data do Banco)

API – Atualizar Usuários

- Este endpoint deverá receber em sua rota o Id do usuário e no corpo da requisição todos os campos do Usuário.
- Deverá retornar status code de acordo com protocolo HTTP.
- updatedAt: Atualizar a data sempre que o usuário sofrer atualização.

Front – Criar/Atualizar Usuários

- Deve-se apresentar um formulário que permita a criação ou edição de um usuário.
- Os campos de _id, data de criação não devem ser editáveis.
- O campo birthday deve ser um datepicker.
- Deve haver validação de preenchimento de campos obrigatórios.

API – Deletar Usuário

- Este endpoint deve receber o _id od usuário a ser deletado e o mesmo deve ser apagado do banco de dados.
- Deverá retornar status code de acordo com protocolo HTTP.

Front – Deletar Usuário

- Deve ser um botão apresentado ao lado das informações dos Usuários na tela de listagem.

Requisitos Obrigatórios

- Persistência de dados.
- Gestão de dependências via gerenciador de pacotes. (npm/yarn)
- API: Express, Hapi, Koa ou similares.
- Utilização de banco NoSQL.
- Frontend Angular.

Requisitos Desejáveis

- Criptografia não reversível (hash) na senha.
- Na tela de listagem podem haver filtros por nome, data, email.
- Na criação/edição de usuário fazer validação da mascara de email, para que seja inserido sempre email válido.
- Mongo DB.
- Após pressionar botão de deletar apresentar modal de confirmação de exclusão.
- Collection no postman com todas as chamadas da API. (Exportar URL)

Submissão

- O desafio deve ser entregue via GitHub em dois projetos públicos e separados. Um para a API e outro para o Front.
- O Help de ambos os projetos deve estar preenchidos com instruções para clonarmos e iniciarmos ambos os projetos.
- Após terminar o desafio você deverá enviar um email em resposta ao solicitante do desafio com as URLs do GitHub e demais observações.