Desafio Desenvolvedor Full Stack

E aí *dev*, beleza? Estamos trabalhando no desenvolvimento de uma ferramenta de gerenciamento de heróis. Nesse desafio você estará responsável tanto pelo frontend como pelo backend de uma aplicação para cadastro de todos os super-heróis.

Aproveite esse desafio para nos mostrar suas habilidades como desenvolvedor full stack.

Proposta

O objetivo desse desafio é a criação de uma aplicação web composta de um frontend em **Vue** e um backend (API) em **.Net Core**, versão 8. O aplicativo deve prover as funcionalidades CRUD (*Create, Read, Update, Delete*) para uma base de dados de **super-heróis**.

Você tem liberdade de desenvolver o frontend, ou seja, a interface gráfica da aplicação, como achar melhor.

É esperado que você seja capaz de encontrar soluções por meio das documentações oficiais, fóruns e artigos.

Leia com atenção todo o documento e “mãos na massa”.

Requisitos

O aplicativo deve prover as seguintes funcionalidades:

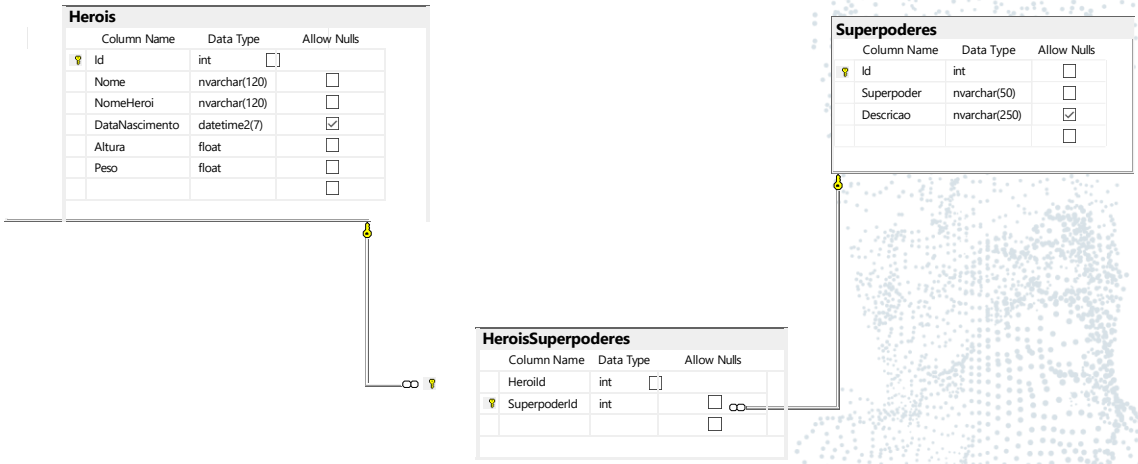
- Cadastro de um novo super-herói
- Listagem dos super-heróis
- Consulta de um super-herói por Id
- Atualização de informações do super-herói por Id
- Exclusão de um super-herói por Id
- Disponibilização da documentação da API utilizando o Swagger
- Entity Framework para acesso ao banco de dados, tanto para leitura como para gravação
- Testes unitários

Cadastro de um super-herói

Na inclusão de um novo super-herói, a API deve receber os dados abaixo para realizar a criação do registro na base de dados. Todos os campos, abaixo, são obrigatórios:

- Nome,
- Nome Herói
- Superpoderes (lista de um ou mais superpoderes)
- Data de Nascimento
- Altura
- Peso

O banco de dados deve ser modelado conforme o diagrama abaixo:



Deve ser gerado automaticamente um ID para o super-herói.

Não deve permitir criar dois super-heróis com o mesmo nome de herói, isso daria uma boa confusão. Imagina dois **Batman**?

A lista superpoderes virá do banco, mas não é necessário a criação de páginas para o CRUD dos superpoderes; apenas no cadastro de heróis você deve ser capaz de listar todos os superpoderes e permitir a seleção de um ou mais superpoderes. Os superpoderes selecionados devem ser associados ao herói através da tabela **HeroisSuperpoderes**.

Listagem de super-heróis

A API deve ser capaz de retornar a lista de todos os super-heróis cadastrados previamente.

Além disso, caso seja chamado o método de listagem, e não existir nenhum super-herói cadastrado, deve-se retornar o *HTTP Code* correto, com mensagem de tratamento.

Consulta de super-herói por Id

A API deve possuir um método para retornar um super-herói, recebendo como parâmetro o código de identificação dele (Id). Novamente, lembre-se de tratar as exceções, como Id inválido por exemplo, que deve retornar uma mensagem adequada para quem estiver utilizando o método.

Atualização de informações do super-herói por Id

A API deve possuir um método para atualização dos dados de um super-herói previamente cadastrado, passando como parâmetro o Id de identificação dele. Lembre-se, Id inválido, deve retornar o tratamento adequado.

Caso seja informado um nome de herói na alteração, que já esteja em uso por outro super-herói cadastrado, a aplicação deve barrar a alteração, e devolver mensagem adequada.

Em caso de sucesso, informar que a alteração foi realizada, e retornar os dados atualizados do super-herói.

Exclusão de um super-herói por Id

A API deve possuir um método de exclusão de um super-herói previamente cadastrado, passando como parâmetro o Id de identificação dele. Lembre-se, Id inválido, deve retornar o tratamento adequado.

Após a exclusão, deve-se retornar apenas o *HTTP Code* de sucesso, com a mensagem de exclusão do super-herói.

Disponibilização da documentação da API utilizando o Swagger

O Swagger é um framework composto por diversas ferramentas que, independentemente da linguagem, auxilia a descrição, consumo e visualização de serviços de uma API REST.

A API REST desenvolvida deve retornar os métodos documentos via Swagger. Lembre-se de usar o método HTTP correto para cada método criado.

Requisitos técnicos

Plataforma: a API deve ser desenvolvida em C# com .NET 8, utilizando o Entity Framework para leitura e gravação no banco de dados.

Banco de dados: pode ser utilizado banco de dados em memória, por exemplo o Entity Framework Core, através do provedor InMemory, listas em memória, bem como pode ser utilizado SQL-Server, MySQL ou qualquer outro banco relacional. Em caso de utilização de banco de dados relacional, disponibilizar os scripts das tabelas no código fonte.

Código fonte: o código fonte deve ser disponibilizado no GitHub pessoal até a data acordada.

Testes Unitários: Crie os testes unitários utilizando a biblioteca XUnit.

Lembre-se de deixar seu repositório público, caso contrário estará desclassificado.

Commits feitos depois do prazo final, serão desconsiderados.

Dicas

- Organize bem o código fonte.
- Revise se todos os requisitos obrigatórios foram atendidos; foque neles primeiro.
- Faça um pequeno arquivo de texto explicando para a gente por que fez a aplicação de tal forma. Um Read.me no GitHub explicando sua API será um diferencial.

Boa sorte!