

TesteQA - Resolução

Luis Eduardo Bet

Email: luiseduardobet22@gmail.com

Github: <https://github.com/luisEduardoBet>

1. PROPOSTA DO TESTE

O teste proposto foi baseado em uma aplicação web de cadastro de gerenciamento de usuários que utiliza a linguagem JavaScript composta pelo ambiente de execução Node.js, o framework Express e a biblioteca React. Para o gerenciamento e construção de banco de dados utilizou-se MySQL. Para a aplicação foram levantados os seguintes requisitos:

- Tela de cadastro com campos de nome, email, senha, cpf, idade e perfil,
- Tela de exibição dos usuários cadastrados.
- Gerenciamento dos usuários cadastrados. Com a opção de ativar,, inativar ou excluir um usuário.

Para o teste, foi disponibilizado o sistema desenvolvido, porém com algumas inconformidades com os requisitos funcionais levantados. Logo o objetivo do desafio proposto é realizar testes, ajustar as inconformidades encontradas de forma a corresponder com os requisitos e melhorar a experiência do usuário.

2. DESENVOLVIMENTO

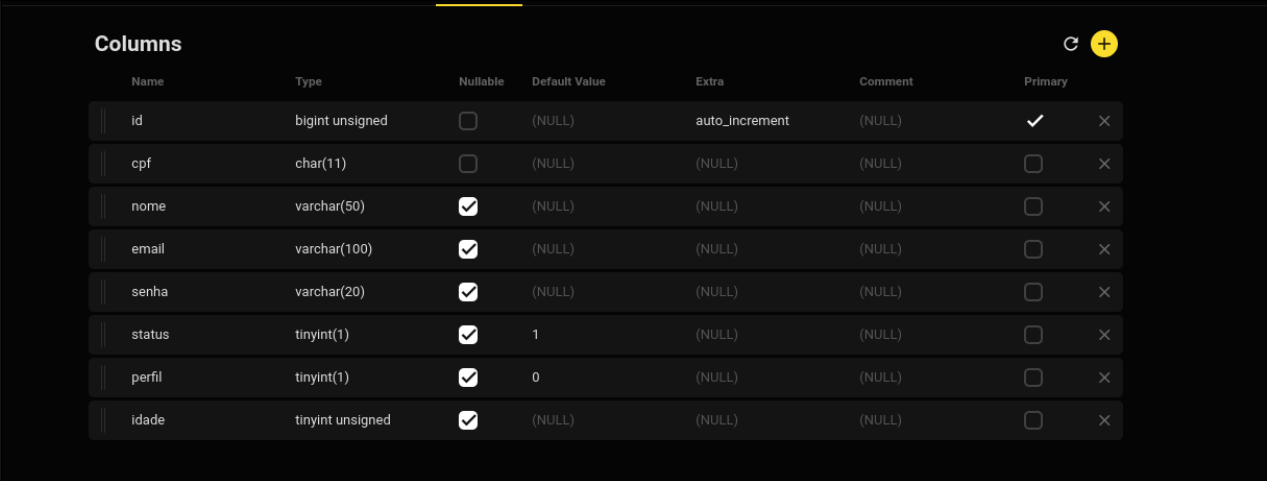
A priori, a abordagem mais adequada seria a construção e aplicação de testes unitários utilizando o framework Jest, com o objetivo de identificar possíveis inconformidades na aplicação. No entanto, devido ao tempo limitado para o estudo do framework e à falta de familiaridade com sua utilização, não foi possível desenvolver nem aplicar esses testes.

Diante disso, optou-se por realizar testes manuais, avaliando o funcionamento do sistema em relação aos requisitos estabelecidos. Além disso, foram realizados testes mais básicos, como o envio de requisições por meio do aplicativo Insomnia, a fim de verificar se as respostas retornadas pelo backend estavam de acordo com o esperado. Por meio destes testes mais “primitivos”, foram encontrados alguns bugs, que foram resolvidos, e algumas melhorias foram feitas para entrar em conformidade com os requisitos.

A primeira melhoria realizada foi na estrutura do banco de dados. As tabelas *usuarios* e *perfis* foram unificadas em uma única tabela denominada *usuario*. Apesar da relação original ser do tipo 1:N, a tabela *perfis* continha apenas dois atributos (identificador e nome) sendo que apenas o identificador era utilizado nas relações, enquanto o nome pode ser tratado diretamente na camada de aplicação. Dessa forma, ao meu ver se torna vantajoso a unificação da tabela como forma de economizar espaço no banco e otimizar consultas futuras. Mas vale ressaltar que

caso houvesse mais colunas/linhas na tabela perfis o ideal era manter a tabela usuários e perfis separadas.

Além disso, foram ajustados alguns tipos e restrições de campos do banco, bem como a adição de campos como o CPF. Na Figura 1 vemos a imagem da estrutura do banco.



Name	Type	Nullable	Default Value	Extra	Comment	Primary
id	bigint unsigned	<input type="checkbox"/>	(NULL)	auto_increment	(NULL)	<input checked="" type="checkbox"/>
cpf	char(11)	<input type="checkbox"/>	(NULL)	(NULL)	(NULL)	<input type="checkbox"/>
nome	varchar(50)	<input checked="" type="checkbox"/>	(NULL)	(NULL)	(NULL)	<input type="checkbox"/>
email	varchar(100)	<input checked="" type="checkbox"/>	(NULL)	(NULL)	(NULL)	<input type="checkbox"/>
senha	varchar(20)	<input checked="" type="checkbox"/>	(NULL)	(NULL)	(NULL)	<input type="checkbox"/>
status	tinyint(1)	<input checked="" type="checkbox"/>	1	(NULL)	(NULL)	<input type="checkbox"/>
perfil	tinyint(1)	<input checked="" type="checkbox"/>	0	(NULL)	(NULL)	<input type="checkbox"/>
idade	tinyint unsigned	<input checked="" type="checkbox"/>	(NULL)	(NULL)	(NULL)	<input type="checkbox"/>

Figura 1: Estrutura do banco demonstrada no app BeeKeeper Studio.

A segunda melhoria foi implementada na estrutura de rotas do React, com o objetivo de aprimorar a experiência do usuário e organizar melhor o código-fonte, que anteriormente estava concentrado inteiramente no arquivo *App.js*. Para isso, foram criadas duas rotas principais: a rota */*, destinada ao gerenciamento e visualização dos usuários, e a rota */SignUp*, voltada para o cadastro de novos usuários.

Além disso, o conteúdo do *App.js* foi refatorado e dividido em dois componentes separados: *Home.js*, responsável pela visualização e gerenciamento de usuários, e *SignUp.js*, responsável pelo formulário de cadastro. Essa separação proporcionou uma melhor organização do código, facilitando a identificação de bugs e a implementação de futuras melhorias.

Posteriormente, foram resolvidos os bugs/melhorias no backend da aplicação, como:

- Corrigido o problema em que, mesmo havendo erro durante uma requisição POST, o sistema ainda retornava um status HTTP 200;
- Corrigida a lógica de atualização de status, que anteriormente gravava o status como ativo (status=1) independentemente do valor enviado.
- Refatorada a funcionalidade de exclusão de usuários, eliminando queries desnecessárias e garantindo que a exclusão seja realizada apenas com base no identificador do usuário (antes utilizava nome e/ou idade).

No frontend, além da separação do conteúdo do App.js em dois arquivos distintos, também foram realizados ajustes na estilização da aplicação, especialmente nos arquivos CSS. As modificações tiveram como principal objetivo tornar o design mais responsivo, garantindo uma melhor adaptação da interface a diferentes tamanhos de tela.

OBS: O código modificado com as melhorias e o dump do banco de dados pode ser visualizado no [Github](#).

3. CONCLUSÃO

Com as melhorias implementadas, todos os requisitos do sistema foram, aparentemente, atendidos. No entanto, ainda é necessário um refinamento mais profundo no código, especialmente no que diz respeito ao feedback de erros para os usuários e ao tratamento adequado dos dados antes de sua inserção no banco de dados.

Idealmente, essas melhorias devem ser acompanhadas pela implementação de testes unitários e de integração, para garantir a confiabilidade da aplicação. No entanto, devido à limitação de tempo e à falta de familiaridade com as bibliotecas e frameworks de testes, essas etapas não puderam ser realizadas neste momento.

Apesar dessas limitações, o desenvolvimento do sistema contribuiu para o aprimoramento dos conhecimentos práticos, especialmente nas tecnologias que possui pouca familiaridade, como React, Node.js e Express.