04

Faça como eu fiz: Preparando os Mocks

Os mocks nos permitirão prosseguir com os testes do nosso projeto. Por meio de um mock, conseguiremos testar uma função que interage com o banco de dados, mas sem necessariamente usá-lo, pois não queremos alterar o comportamento dela.

Para gerar um mock, precisamos criar uma pasta chamada *"**mocks**"*. Dentro dela, é necessário criar um arquivo com as mesmas funções que se comunicam com o banco de dados, ou seja, todo o `*exports = {...}*` . Porém, as funções estarão vazias.

No nosso caso, faremos isso para a `TabelaFornecedores.js` , criando o arquivo dentro da pasta *"**mocks**"*, copiando todas funções de `module.exports` e apagando os seus conteúdos.

Devemos fazer as funções retornarem valores fixos. A função de listar precisará retornar uma lista. Poderemos retornar uma lista vazia por meio do `return []` .

A próxima função será `inserir()` . Para que ela funcione, será necessário que retorne um ID, uma data de criação, uma de atualização e uma versão. Retornaremos qualquer valor para esses campos, então teremos, por exemplo, `return {id: 500, dataCriação: '10/12/3420', dataAtualizacao: '10/12/3420', versao:90 }` .

A função `pergarPorId()` terá o mesmo retorno da função `inserir()` . Logo, poderemos retornar os mesmos dados do que ela.

As funções de atualizar e remover não terão retorno. Dessa forma, usaremos o `async` na frente delas em nosso código e os testes não vão esperar por uma resposta inexistente.

Pronto! Agora temos o nosso primeiro mock e conseguiremos escrever nossos testes como já fizemos anteriormente, sem nos preocupar com alterações no banco de dados.