06

# Edição de livros

### Transcrição

Nessa etapa implementaremos a funcionalidade de edição dos nossos livros. De volta ao arquivo `rotas.js`, criaremos uma nova rota para atender à URL `/livros/form/:id` - ou seja, a URL do nosso formulário de cadastro, com a adição de uma sintaxe para criação de variáveis.

Essa rota receberá uma função *callback* que será executada sempre que o usuário requisitá-la. Com o `id` passado na URL, criaremos uma instância de `livroDao`, buscaremos o livro e, se tudo der certo, devolveremos a página de formulário da nossa aplicação, passando como parâmetro um objeto JavaScript com a propriedade `livro` e recebendo o valor `livro` que acabamos de receber.

Se houver algum erro, vamos exibi-lo no console com `console.log()`:

```
app.get('/livros/form/:id', function(req, resp) {
    const id = req.params.id;
    const livroDao = new LivroDao(db);

    livroDao.buscaPorId(id)
        .then(livro =>
            resp.marko(
                require('../views/livros/form/form.marko'),
                { livro: livro }
            )
        )
        .catch(erro => console.log(erro));

});
```

No `<input type "hidden" />` do arquivo `form.marko`, atribuiremos o valor do `id` do livro que foi passado para nosso template ( `${data.livro.id}` ). Faremos a mesma coisa para o título ( `${data.livro.titulo}` ), para o preço ( `${data.livro.preco}` ) e para a descrição ( `${data.livro.descricao}` ).

```
<html>
    <body>
        <h1>Cadastro de livros</h1>

        <form action="/livros" method="post">

            <input type="hidden" id="id" name="id" value="${data.livro.id}" />

            <div>
                <label for="titulo">Titulo:</label>
                <input type="text" id="titulo" name="titulo" value="${data.livro.titulo}" placel
            </div>
            <div>
                <label for="preco">Preço:</label>
                <input type="text" id="preco" name="preco" value="${data.livro.preco}" placehol
            </div>
```

```
            <div>
                <label for="descricao">Descrição:</label>
                <textarea cols="20" rows="10"  id="descricao" name="descricao"  placeholder="fa
            </div>

            <input type="submit" value="Salvar" />
        </form>
    </body>
</html>
```

Já no link de edição do arquivo `lista.marko`, preencheremos o atributo `href`, que antes estava apenas com `#`, com `/livros/form/${livro.id}`, que é exatamente a rota que acabamos de criar. Com essas alterações salvas, podemos testar a nova funcionalidade no navegador.

Acessando http://localhost:3000/livros/ (http://localhost:3000/livros/), clicaremos em qualquer um dos links "Editar" ao lado dos elementos da lista. Se clicarmos no primeiro ("Node na prática"), por exemplo, seremos redirecionados para http://localhost:3000/livros/form/1 (http://localhost:3000/livros/form/1) - a página do nosso formulário já preenchida com as informações do primeiro livro.

Porém, se tentarmos entrar na página http://localhost:3000/livros/form (http://localhost:3000/livros/form), nada será exibido. No console, teremos uma mensagem informando que os cabeçalhos não foram enviados com sucesso para o cliente.

Isso acontece porque, na pagina `form.marko`, estamos esperando um atributo chamado `livro`. Porém, isso só acontece quando estamos editando um livro em `/livros/form/:id`, passando o `livro` retornado da nossa busca para o *template*.

No entanto, na rota `/livros/form`, nenhum dado `livro` é passado, quebrando nossa aplicação. Para resolvermos esse problema, nessa rota, passaremos para o método `resp.marko()` um objeto JavaScript com a propriedade `livro` vazia:

```
app.get('/livros/form', function(req, resp) {
    resp.marko(require('../views/livros/form/form.marko'), { livro: {} });
});
```

Após salvarmos nosso projeto, a URL http://localhost:3000/livros/form (http://localhost:3000/livros/form) voltará a funcionar normalmente. Voltando para nossa listagem, se tentarmos editar um dos livros na nossa lista e clicarmos em "Salvar", teremos um problema: ao invés das informações serem atualizadas, um novo livro será adicionado à listagem.

Isso acontece pois, no nosso formulário, estamos submetendo as informações com o método POST. Porém, no nosso arquivo `rotas.js`, a rota `/livros` foi criada para adicionar um novo livro na tabela. Consequentemente, o método de adição que é executado.

Resolveremos esse problema a seguir!