05

Implementando a lógica de alteração

Transcrição

Agora que em `Cadastro` já sabemos como obter o `id` da foto passado na URL já podemos implementar a lógica de alteração. No caso, vamos alterar o método `cadastra` de `FotoService` . A solução para saber se estamos incluindo ou alterado é simples. Se o objeto foto passado como parâmetro tiver a propriedade `_id` é porque estamos alterando, se não tiver, é porque ele nunca foi gravado no banco e precisamos incluí-lo através da nossa API.

Aprendemos que o método `save` do nosso resource é aquele que realiza um POST por debaixo dos panos, para alteração, usaremos o método `update` que recebe dois parâmetros. O primeiro é o id da foto que desejamos alterar em nossa API, porque noss API depende dessa informação para encontrar o recurso. O segundo, é o objeto foto em si.

```
// alurapic/src/service/FotoService.js

export default class FotoService {

    constructor(resource) {

        this._resource = resource('v1/fotos{/id}');
    }

    cadastra(foto) {

        if(foto._id) {
```

```
            return this._resource.update({ id: foto._id },
foto);

        } else {

            return this._resource.save(foto);
        }

    }

  // código posterior omitido
}
```

COPIAR CÓDIGO

Será que funciona? Vamos primeiro cadastrar uma novo foto para ver se nada quebrou. Em seguida, vamos escolher uma foto da lista e alterar seu nome. Quando clicamos em `GRAVAR` o formulário é limpo. Mas será que realmente gravou? Para isso, vamos voltar para `Home` e constatar a mudança, excelente! Mas pode ficar ainda melhor.

Quando cadastramos uma novo foto, faz sentido limparmos o formulário e continuarmos em `Cadastro` para que o usuário possa cadastrar mais fotos. Já a alteração, que tal assim que alterarmos com sucesso a foto voltarmos para `Home` ? Precisamos aprender a gerar uma navegação através do nosso componente.

O `VueRouter` disponibiliza no global view instance o objeto `$router` que possui o método `push` . Podemos passar para esse método um objeto JavasScript que representa nossa rota nomeada, assim como fizemos em `router-link` :

```
// alurapic/src/components/cadastro/Cadastro.vue

  methods: {
```

```
    grava() {

      this.service
        .cadastra(this.foto)
        .then(() => {
          if(this.id) this.$router.push({ name: 'home'});
          this.foto = new Foto()
        },
        err => console.log(err));

    }
  },
```

COPIAR CÓDIGO

Reparem que a navegação só será realizada caso o ID da foto esteja preenchido, ou seja, em uma alteração.

A diretiva v-if e v-else

Para ficar ainda melhor, podemos exibir o título "Inclusão" ou o título "Alteração" no formulário de acordo com seu estado. Para isso, vamos alterar o título `h1` já existente para `Alteração` e adicionar outro título `h1` com o texto `Inclusão` em nosso formulário:

```
<!-- alurapic/src/components/cadastro/Cadastro.vue -->

<template>

  <div>

    <h1 class="centralizado">Alteração</h1>
    <h1 class="centralizado">Inclusão</h1>
<!-- código posterior omitido -->
```

COPIAR CÓDIGO

Não precisamos refletir muito sobre essa alteração para saber que ela exibirá os dois títulos, independente se estamos alterando ou incluindo. É ai que entra a diretiva `v-if` e `v-else` . Alterando:

```
<!-- alurapic/src/components/cadastro/Cadastro.vue -->

<template>

  <div>

    <h1 v-if="foto._id" class="centralizado">Alteração</h1>
    <h1 v-else class="centralizado">Inclusão</h1>
    <!-- código posterior omitido -->
```

COPIAR CÓDIGO

Se o ID da foto for definido, apenas o h1 com o texto "Alteração" será exibido, caso contrário será o com o texto "Inclusão".

A diretiva `v-if` e `v-else` são parecidas com a diretiva `v-show` , mas enquanto o `v-show` apenas esconde o elemento do DOM através da propriedade "display: none" por debaixo dos panos, o uso do `v-if e v-else` literalmente removem e adicionam o elemento no DOM.