

## Organizando recursos em serviços

### Transcrição

Vamos criar uma classe especializada em consumir nossa API. Como o Vue usa ES2015, não temos restrições no que podemos usar essa fantástica linguagem. Até porque, o CLI converte nosso código para ES5 garantindo assim maior compatibilidade entre os navegadores.

Nossa classe se chamará `FotoService` e ficará no mesmo diretório do modelo `Foto` :

```
// alurapic/src/domain/foto/FotoService.js

export default class FotoService {

  constructor(resource) {

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

  cadastra(foto) {

    return this._resource.save(foto);
  }

  lista() {

    return this._resource
      .query()
      .then(res => res.json());
  }

  apaga(id) {

    return this._resource.delete({ id });
  }
}
```

Veja que ela recebe em seu construtor um `resource`, no caso, será o `this.$resource` do componente que utilizará o serviço. Veja que internamente na propriedade `this._resource` guardamos um instância configurada.

Por fim, temos métodos de alto nível que encapsulam os métodos do `resource`. Como os métodos de um recurso do Vue Resource devolvem uma `promise`, podemos retornar a `promise` resultante nesses métodos que criamos. Quem chamar os métodos terá acesso ao resultado encadeando uma chamada a função `then` .

Vamos alterar agora nosso componente `Home` que precisa importar a classe para em seguida, no método `created` , criarmos uma instância dessa classe passando `$resource` como parâmetro. Por fim, não teremos mais a propriedade `resource` adicionada dinamicamente no componente, mas `service` :

```
<!-- alurapic/src/componens/home/Home.js -->
```

```
<template>

<!-- código omitido -->

</template>

<script>

import Painei from '../shared/painei/Painei.vue';
import ImagemResponsiva from '../shared/imagem-responsiva/ImagemResponsiva.vue'
import Botao from '../shared/botao/Botao.vue';

// importando FotoService
import FotoService from '../../domain/foto/FotoService';

export default {

  /* código anterior omitido */

  methods: {

    remove(foto) {

      this.service
        .apaga(foto._id)
        .then(
          () => {
            let indice = this.fotos.indexOf(foto);
            this.fotos.splice(indice, 1);
            this.mensagem = 'Foto removida com sucesso'
          },
          err => {
            this.mensagem = 'Não foi possível remover a foto';
            console.log(err);
          }
        )
    }

  },

  created() {

    // criando uma instância do nosso serviço que depende de $resource
    this.service = new FotoService(this.$resource);

    this.service
      .lista()
      .then(fotos => this.fotos = fotos, err => console.log(err));
  }
}
</script>
<style>

  /* código omitido */
</style>
```

Agora, em Cadastro :

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

<template>

  <!-- código omitido -->

</template>

<script>

import ImagemResponsiva from '../shared/imagem-responsiva/ImagemResponsiva.vue'
import Botao from '../shared/botao/Botao.vue';
import Foto from '../../domain/foto/Foto';
import FotoService from '../../domain/foto/FotoService';

export default {

  // código anterior omitido

  methods: {

    grava() {

      this.service
        .cadastra(this.foto)
        .then(() => this.foto = new Foto(), err => console.log(err));

    }
  },

  created() {

    this.service = new FotoService(this.$resource);
  }

}

</script>
<style scoped>

  /* código omitido */
</style>
```

Agora nossos componentes ficam desacoplados do endereço da API, inclusive dos detalhes de conversão de dados.