04

Code splitting e lazy loading

Transcrição

Como vimos, pode ser interessante dividirmos (split) o código da nossa aplicação em pedaços que são carregados a medida que o usuário vai interagindo com nossa aplicação.

Em nossa aplicação, vamos fazer com que o componente `Cadastro` seja carregado apenas quando usuário acessar o link `Cadastro` ou editar uma foto. Dizemos que estamos adotando uma postura "preguiçosa" de carregamento, no inglês, `lazy loading` . Lazy loading e code splitting caminham juntos para melhorar a experiência do usuário no primeiro carregamento da página.

A boa notícia é que para habilitarmos o code splitting e o Lazy Loading basta realizarmos uma pequena alteração em nosso arquivo `routes.js` .

```
/ alurapic/src/routes.js

import Home from './components/home/Home.vue';

const Cadastro = () => System.import('./components/cadastro/Cadastro.vue');


export const routes = [

    { path: '', name: 'home', component: Home, titulo: 'Home', menu: true },
    { path: '/cadastro', name:'cadastro', component: Cadastro, titulo: 'Cadastro', menu: true },
    { path: '/cadastro/:id', name:'altera', component: Cadastro, titulo: 'Cadastro', menu: false
    { path: '*', component: Home, menu: false }

];
```

Veja que ao invés de importamos o componente que desejamos criar um bundle em separado e carregá-lo preguiçosamente, usamos a sintaxe:

```
const Cadastro = () => System.import('./components/cadastro/Cadastro.vue');
```

Todo restante do nosso arquivo continua o mesmo.

Se rodarmos nossa aplicação com `npm run dev` tudo continua funcionando. Agora, quando gerarmos o build do projeto com `npm run build` teremos dois arquivos `build` gerados, ou seja, um para a nossa aplicação inteira e outro para o componente cadastro. Podemos fazer um teste e copiar os novos arquivo para o nosso server.