05

# AJAX com Axios

## Transcrição

[00:00] Nosso objetivo agora é não colocarmos esse JSON *hard coded* na nossa aplicação, mas consumirmos aquela API que criamos - que é `/api/pedidos/aguardando`. Vamos utilizar uma biblioteca chamada Axios. Essa biblioteca é um *client http*, então vamos conseguir consumir os dados da nossa API. Como fazemos isso?

[00:27] Dentro da própria documentação do Vue.js, em "Learn", estamos em "Guide", tem essa parte de "Cookbook", onde tem vários tutoriais. Um desses tutoriais é "Using Axios to Consume APIs", ou seja, vamos consumir a API com Axios.

[00:43] Essa parte de criar API em si nós já fizemos, só que tem esse `mounted` aqui, que depois do `data`, ele colocou uma vírgula e escreveu `mounted`.

[00:54] Esse `mounted` é uma função do ciclo de vida do Vue.js, que é chamado automaticamente. Logo depois que o Vue renderiza o HTML, ou seja, depois que ele renderiza isso daqui, no nosso caso, na página de ofertas, ele chama automaticamente aquela função `mounted` que estava mostrando na documentação. Vou mostrar como isso acontece.

[01:19] Logo depois do `data` eu ponho uma vírgula, coloco o `mounted()` e coloco o `console.log('mounted');`. Vou salvar e vou abrir a tela, vou pressionar "F12" para vermos o console e "Enter". Está vendo? `mounted`. Esse `console.log` é chamado automaticamente, ou seja, essa função é executada automaticamente.

[01:45] O que queremos fazer? Depois que ele renderizou, vamos utilizar o Axios para fazermos a requisição para esse *endpoint*.

[01:58] Para fazermos essa requisição com Axios é só copiarmos esse código, `.get('https://api.coindesk.com/v1/bpi/currentprice.json')` , é um *get*, é exatamente isso mesmo que queremos fazer. E aqui, `.then(response => (this.info = response))` , ele está adicionando a resposta do `this.info` - que é quem? Esse aqui que está no `data` , que no nosso caso é `pedidos` .

[02:17] Só que para utilizarmos o Axios, precisamos baixar - porque o Axios não é uma biblioteca que já está disponível na nossa aplicação com o Vue.js.

[02:26] Eu abri aqui o GitHub dele, acessível neste link (https://github.com/axios/axios), você consegue ver código-fonte tudo mais, e aqui embaixo tem como você vai instalar usando o `npm` , `bower` , `yarn` ou adicionando o *script* lá. Eu vou copiar esse `<script src="https://unpkg.com/axios/dist/axios.min.js"></script>` , que na verdade eu já copiei e colei aqui embaixo do *script* do Vue.js. E pronto, agora o Axios está disponível para utilizarmos.

[02:52] Eu vou copiar esse código do Axios, `.get('https://api.coindesk.com/v1/bpi/currentprice.json')` , `.then(response => (this.info = response))` . É `axios.get` , ou seja, é uma requisição do tipo GET para essa url, que não é a url que queremos, queremos a `http://localhost:8080/api/pedidos/aguardando` .

[03:17] E quando ele nos responder, `then` , nós recebemos uma resposta. Vamos o quê? Vamos adicionar em `pedidos` , então é `this.pedidos.response` . O `this` automaticamente nos permite acessar o `data` .

[03:31] Será que funcionou? Será que isso é o suficiente para substituirmos esse conteúdo e automaticamente renderizarmos aqui? Vamos ver. Vou pressionar "Enter" no endereço do navegador. Beleza, continuou!

[03:49] Só que agora ele trouxe vários pedidos aqui e no aguardando só vem dois pedidos. O que aconteceu para ter tantos pedidos assim? E cadê os dados desses pedidos? Ou seja, tem alguma coisa errada.

[04:07] É que na hora em que você pega o `response`, você tem que pegar os dados que vieram no `response`, que aqui ele não está mostrando. Talvez ele mostre aqui embaixo, `response.data`. Na verdade, ele vai mostrar. Aqui ele só coloca isso, `response`, ou seja, vem um monte de dados em si; e tem o `data`, onde tem o conteúdo mesmo da requisição.

[04:34] Eu vou colocar `response.data`. Será que ele é mais claro aqui embaixo? Aqui ele põe `data.bpi`, que já é um conteúdo interno mesmo do negócio. Ele não usou o `.data` ali, mas tudo bem, o `.data` é exatamente os dados de resposta. Vou salvar, vou atualizar e agora os conteúdos estão vindo do banco de dados.

[04:57] Para ter certeza que está vindo do banco de dados, eu vou alterar a descrição desse "Wireless Earbuds TaoTronics" e colocar "descrição atualizada Wireless Earbuds TaoTronics". Ele deve estar cacheado aqui no "OfertaController.java". Deixe-me apertar as teclas "Enter", salvar e ver se atualiza. Eu vou atualizar essa página oferta, vamos ver se aparece descrição atualizada para esse. Pronto, "descrição atualizada Wireless Earbuds TaoTronics"!

[05:29] Então realmente está vindo do banco de dados, vocês podem confiar em mim. E aqui ainda está "descrição Wireless" normal, "descrição atualizada". O JSON mudou!

[05:39] Agora estamos consumido o JSON da nossa aplicação, com o Vue.js e com o Axios, estamos fazendo uma consulta em nossa API REST. São novas siglas para lidarmos no nosso dia a dia.

[05:58] Até o próximo vídeo!