01

Ops! Não podemos importar negociações duplicadas.

Transcrição

> Começando deste ponto? Você pode fazer o DOWNLOAD (https://github.com/alura-cursos/javascript-avancado-iii/archive/aula3.zip) completo do projeto do capítulo anterior e continuar seus estudos a partir deste capítulo.

Atualmente, precisamos clicar no botão **Importar Negociações** para obtermos as negociações das semanas. Contudo, em nenhum momento verificamos se as negociações trazidas do servidor já existem em nossa lista. Com certeza não queremos importar as mesmas negociações mais de uma vez, inclusive aquelas incluídas através da página `post.html` que criamos no curso anterior.

Vamos relembrar o método `importaNegociacoes()` de `NegociacaoController`:

```
// aluraframe/client/js/app/controllers/NegociacaoController.js

// código anterior omitido

importaNegociacoes() {


    let service = new NegociacaoService();
    service
        .obterNegociacoes()
        .then(negociacoes => negociacoes.forEach(negociacao => {
            this._listaNegociacoes.adiciona(negociacao);
            this._mensagem.texto = 'Negociações do período importadas'
        }))
        .catch(erro => this._mensagem.texto = erro);
}

//...
```

O método `service.obterNegociacoes()` retorna uma *promise*, cujo o método `then` nos retorna as negociações da semana, semana anterior e semana retrasada. Iteramos nas negociações adicionando cada uma delas em nosso modelo. Devido ao mecanismo caseiro de *data binding* que criamos, toda vez que a lista for modificada, automaticamente a view que a apresenta será atualizada automaticamente, sem termos que manipular o DOM diretamente.

A pegadinha do Array.indexOf e a poderosa função filter

O primeiro passo é garantirmos que a lista de negociações iterada tenha apenas aquelas que ainda não existem em `this._listaNegociacoes.negociacoes`. A ideia é filtrarmos essa lista usando a função `filter`, que já vimos nos cursos anteriores.

Lembre-se que a função `filter` precisa receber um critério que será aplicado para cada elemento individualmente. Cada elemento só fará parte da nova lista filtrada se o critério aplicado sobre ele retornar `true`.

```
// aluraframe/client/js/app/controllers/NegociacaoController.js

// código anterior omitido

importaNegociacoes() {


    let service = new NegociacaoService();
    service
        .obterNegociacoes()
        .then(negociacoes =>
            negociacoes.filter(negociacao => true)
         )
        .then(negociacoes => negociacoes.forEach(negociacao => {
            this._listaNegociacoes.adiciona(negociacao);
            this._mensagem.texto = 'Negociações do período importadas'
        }))
        .catch(erro => this._mensagem.texto = erro);
}
//...
```

E se usarmos o método `indexOf()` que todo array possui para sabermos se o elemento existe ou não? Só queremos considerar aqueles que não existem, ou seja, cujo valor de `indexOf` é -1.

```
// aluraframe/client/js/app/controllers/NegociacaoController.js

// código anterior omitido

importaNegociacoes() {


    let service = new NegociacaoService();
    service
        .obterNegociacoes()
        .then(negociacoes =>
            negociacoes.filter(negociacao =>
                      this._listaNegociacoes.negociacoes.indexOf(negociacao) == -1)
       )
        .then(negociacoes => negociacoes.forEach(negociacao => {
            this._listaNegociacoes.adiciona(negociacao);
            this._mensagem.texto = 'Negociações do período importadas'
        }))
        .catch(erro => this._mensagem.texto = erro);
}
// ...
```

Veja que usamos a forma simplificada da *arrow function*, aquela que não usa bloco. Quando usamos essa forma, não precisamos usar a instrução `return`, pois ela fica implícita.

Vamos testar, clicando mais de uma vez no botão **Importar Negociações**? O que acontece? Um problema! Veja que nosso código não funcionou como esperado, pois importamos mais de uma vez as mesmas negociações. O problema está no entendimento do `indexOf`, que funciona de um jeito um pouco diferente do que imaginamos. Ele não resolverá o nosso problema de procurar um objeto dentro do array.