07

Isolando a complexidade do importaNegociacoes em NegociacaoService

Transcrição

Para finalizar, faremos alterações no `importaNegociacoes()`. O método atualmente está assim:

```
importaNegociacoes() {

    this._service
        .obterNegociacoes()
        .then(negociacoes any
            negociacoes.filter(negociacao =>
                !this._litsaNegociacoes.negociacoes.some(negociacaoExistente =>
                    JSON.stringify(negociacao) == JSON.stringify(negociacaoExistente)))
        )
        .then(negociacoes => negociacoes.forEach(negociacao => {
            this._listaNegociacoes.adiciona(negociacao);
            this._mensagem.texto = 'Negociações do período importadas'
          }))
          .catch(erro => this._mensagem.texto = erro);
}
```

Observe que o trecho possui muito detalhes para conseguirmos importar uma negociação. Temos que aplicar um filtro... Para solucionar isto, em `NegociacaoService.js`, criaremos o método `importa()`. Com ele, queremos obter as negociações, por isso, usaremos o `this.obterNegociacoes()`:

```
importa(listaAtual) {

       return this.obterNegociacoes()
           .then(negociacoes =>
               negociacoes.filter(negociacao =>
                   !listaAtual.some(negociacaoExistente =>
                       JSON.stringify(negociacao) == JSON.stringify(negociacaoExistente)))
           )
   }
```

No `obterNegociacoes`, recebemos a lista de negociações, mas devemos ter a lista atual que é exibida para o usuário. Quando chamamos o `importa()` precisamos receber a `listaAtual`, que devolverá somente as negociações ainda não importadas. No fim, teremos uma lista de negociação que será importada. Agora adicionaremos o `catch` para os casos de erro.

```
importa(listaAtual) {

       return this.obterNegociacoes()
           .then(negociacoes =>
               negociacoes.filter(negociacao =>
                   !listaAtual.some(negociacaoExistente =>
                       JSON.stringify(negociacao) == JSON.stringify(negociacaoExistente)))
           )
           .catch(erro => {
```

```
                console.log(erro);
                throw new Error("Não foi possível importar as negociações");
            });
    }
```

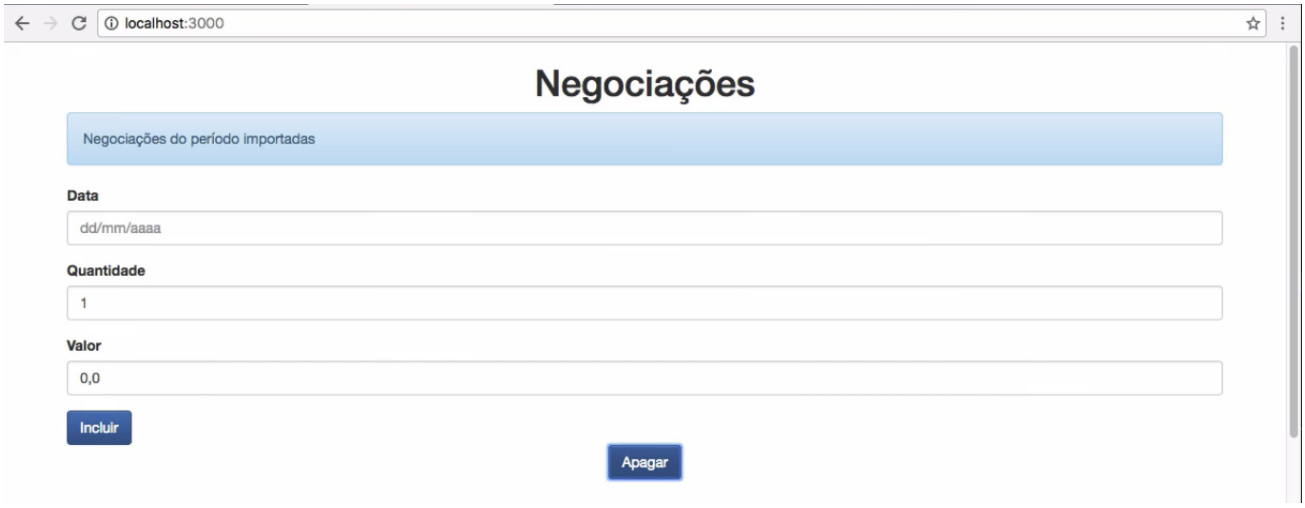
Simplificaremos o método `importaNegociacoes()` do arquivo `NegociacaoController.js` .

```
importaNegociacoes() {

    this._service
        .importa(this._listaNegociacoes.negociacoes)
        .then(negociacoes => negociacoes.forEach(negociacao => {
            this._listaNegociacoes.adiciona(negociacao);
            this._mensagem.texto = 'Negociações do período importadas'
          }))
        .catch(erro => this._mensagem.texto = erro);
}
```

Porém, o parâmetro que passamos para o `importa()` é a lista atual. Internamente, ele irá comparar a lista trazida do Back-end com a atual para decidir quais serão as negociações que vão passar para o `then()` , ou seja, as que ainda não foram importadas.

Adicionaremos o `return` no método `apaga()` . Sem o retorno, o método não funcionará corretamente. Com a alteração, o trecho do código ficará assim:

```
apaga() {

    return ConnectionFactory
        .getConnection()
        .then(connection => new NegociacaoController(connection))
        .then(dao => dao.apagaTodos())
        .then(() => 'Negociações apagadas com sucesso')
        .catch(erro => {
            console.log(erro);
            throw new Error('Não foi possível apagar as negociações')
          })
}
```



Agora, as importações funcionam corretamente. Nós refatoramos o nosso código e com isto, facilitamos a leitura e o entendimento.