04

Removendo todas as negociações

Transcrição

Antes de implementarmos o `ApagaTodos` do DAO, vamos tratar os casos de erro dentro do `ConnectionFactory`:

```
ConnectionFactory
    .getConnection()
    .then(connection => new Negociacao(connection))
    .then(dao => dao.listaTodos())
    .then(negociacoes =>
        negociacoes.forEach(negociacao =>
            this._listaNegociacoes.adiciona(negociacao)))
    .catch(erro => {
        this._mensagem.texto = erro;
      });
```

Agora, no `NegociacaoDAO.js`, adicionaremos o `apagaTodos()`. Moveremos para ela, o trecho referente ao `cursor` e que pega a store.

```
apagaTodos() {

    return new Promise((resolve, reject) => {

        let cursor = this._connection
            .transaction([this._store], 'readwrite')
            .objectStore(this._store)
            .clear();

      });
  }
```

No entanto, em vez de pegarmos um cursor, finalizaremos com o `clear()`. Ele nos devolverá uma requisição, por isso, substituiremos o `cursor` pelo `request`. Depois, incluiremos o `onsuccess`.

```
apagaTodos() {

    return new Promise((resolve, reject) => {

        let request = this._connection
            .transaction([this._store], 'readwrite')
            .objectStore(this._store)
            .clear();

        request.onsuccess = e => resolve('Negociações removidas com sucesso');

        request.onerror = e => {
          console.log(e.target.error);
          reject('Não foi possível remover as negociações');
        }
```

```
        });
    }
```

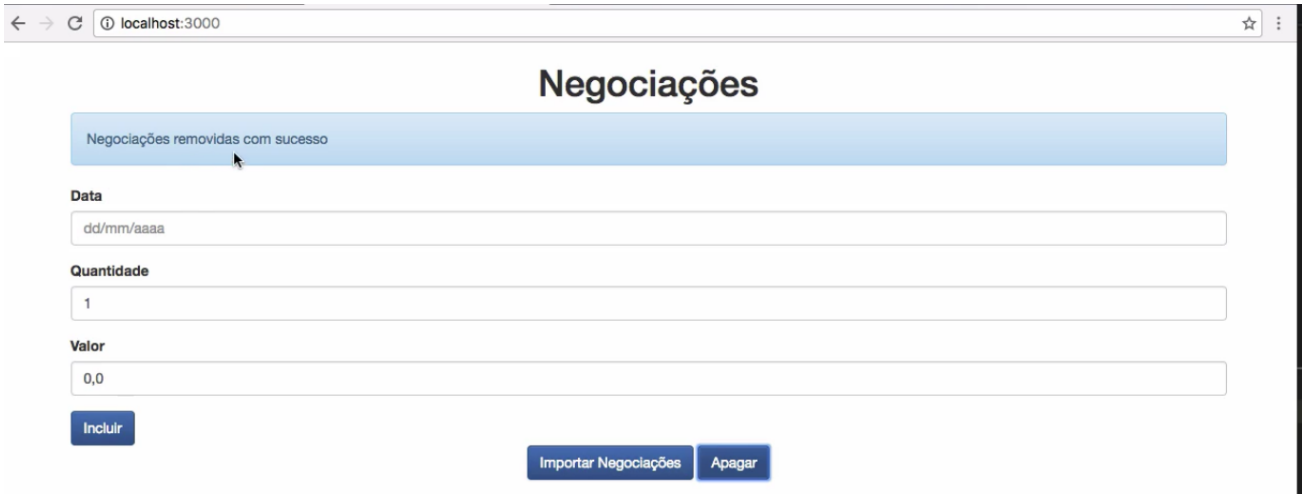
Usamos a mesma lógica do `cursor`, mas apagaremos o banco. No `NegociacaoController`, o método `apaga()` está assim:

```
apaga() {

    this._listaNegociacoes.esvazia();
    this._mensagem.texto = 'Negociações apagadas com sucesso';

}
```

A seguir, vamos incluir o `ConnectionFactory`:

```
apaga() {

ConnectionFactory
    .getConnection()
    .then(connection => new NegociacaoDao(connection))
    .then(dao => dao.apagaTodos())
    .then(mensagem => {
        this._mensagem.texto = mensagem;
        this._listaNegociacoes.esvazia();
    });
}
```

Temos a conexão e dela, criamos o DAO,e se correu tudo bem, o `apagaTodos retornará uma mensagem. Após ela ser exibida, esvaziaremos o modelo. Vamos voltar para o navegador e testar o que fizemos. Ao clicarmos no botão "Apagar", as informações serão apagadas.

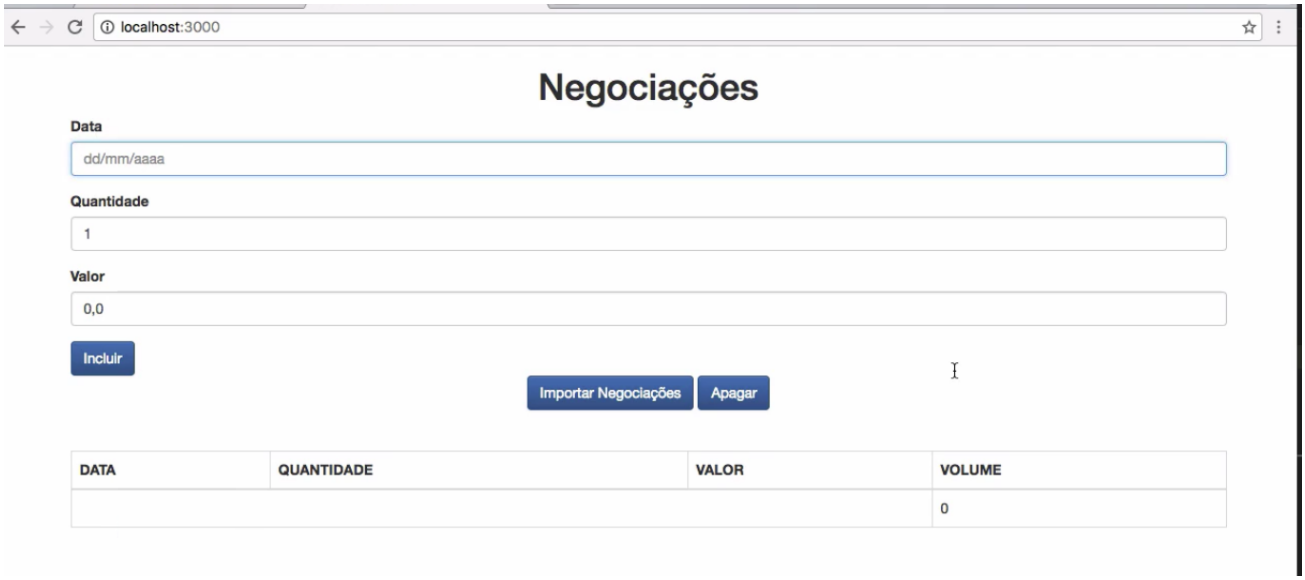


Também recebemos a mensagem de que as negociações foram apagadas. Vamos usar a palavra "apagar" no lugar de "remover" em `NegociacaoDao.js`:

```
apagaTodos() {

    return new Promise((resolve, reject) => {
```

```
        let request = this._connection
            .transaction([this._store], 'readwrite')
            .objectStore(this._store)
            .clear();

        request.onsuccess = e => resolve('Negociações apagadas com sucesso');

        request.onerror = e => {
            console.log(e.target.error);
            reject('Não foi possível apagar as negociações');
        };

    });
```

Agora, quando recarregarmos a página, veremos que ela está completamente vazia e nenhuma negociação é exibida na tabela.



Inclusive, se olharmos na aba "Application" e analisarmos a Object Store, veremos que todas as negociações foram apagadas também. Mas se preenchermos novamente o formulário, ele continuará salvando as negociações.

Com as últimas alterações, realizamos a operação de persistência de inclusão, listagem e deleção da nossa aplicação. Nós usamos uma API para nos auxiliar na manutenção do estado da aplicação, utilizando o banco de dados do navegador.

Pratique nos exercícios todo o conteúdo visto até aqui. Na próxima aula, faremos mais melhorias no código da aplicação.