06

Adicionando o método lista em NegociacaoService

Transcrição

Vamos fazer mais melhorias no `NegociacaoController`, adicionando o novo método ao `_init()`. Atualmente ele está assim:

```
_init() {

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

Em seguida, adicionaremos o método `lista()`:

```
_init() {

    new NegociacaoService()
        .lista()
        .then(negociacoes =>
            negociacoes.forEach(negociacao =>
                this._listaNegociacoes.adiciona(negociacao)))
        .catch(erro => {
            console.log(erro);
            this._mensagem.texto = erro;
        });
//...
```

Pedimos para o `NegociacaoService` dar todas as negociações, depois, varreremos cada uma das negociações e adicionaremos à lista.

Agora, no arquivo `NegociacaoService.js`, abaixo do `cadastra()` adicionaremos o `lista()`.

```
lista() {

  return ConnectionFactory
          .getConnection()
          .then(connection => new NegociacaoDao(connection))
          .then(dao => dao.listaTodos())
}
```

O `dao.listaTodos` listará todos os dados do `indexDb` .

Estamos retornando uma Promise que terá um `DAO` que será o retorno do `dao.listaTodos()` . Se todos os elementos foram listados com sucesso, será exibida uma mensagem. No entanto, se ocorrer um erro, será exibido `Não foi possível obter as negociações` .

```
lista() {

  return ConnectionFactory
          .getConnection()
          .then(connection => new NegociacaoDao(connection))
          .then(dao => dao.listaTodos())
          .catch(erro => {
              console.log(erro);
              throw new Error('Não foi possível obter as negociações')
            })
}
```

Em seguida, faremos alguns ajustes em `NegociacaoController.js` . O erro do `_init` é de alto nível e será exibido diretamente.

```
_init() {

    new NegociacaoService()
        .lista()
        .then(negociacoes =>
            negociacoes.forEach(negociacao =>
                this._listaNegociacoes.adiciona(negociacao)))
        .catch(erro => this._mensagem.texto = erro);


        setInterval(() => {
            this.importaNegociacoes();
        }, 3000);
}
```

Se recarregarmos a página no navegador, todas as negociações serão exibidas na tabela. Iremos melhorar ainda o método `apaga()` , que atualmente está assim:

```
apaga() {

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

Este bloco de código será substituído por outro. Começaremos adicionando o `new NegociacaoService()` .

```
apaga() {

    new NegociacaoService()
        .apaga()
        .then(mensagem => {
            this._mensagem.texto = mensagem;
            this._listaNegociacoes.esvazia();
        })
        .catch(erro => this._mensagem.texto = erro);
}
```

Depois, criaremos o `apaga()` no `NegociacaoService.js`.

```
apaga() {

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

Se tudo funcionar corretamente, a mensagem exibida será `Negociações apagadas com sucesso`. É uma boa prática logar o erro, por isso, faremos o mesmo com o método `cadastra()`.

```
cadastra(negociacao) {

      return ConnectionFactory
         .getConnection()
         .then(connection => new NegociacaoDao(connection))
         .then(dao => dao.adiciona(negociacao))
         .then(() => 'Negociação cadastrada com sucesso')
         .catch(erro => {
            console.log(erro);
            throw new Error('Não foi possível adicionar a negociação')
         });
 }
```

Nós logamos o erro do DAO - que pode ser uma mensagem de baixo nível - e exibiremos para o usuário uma mensagem de alto nível. Vamos voltar para o navegador e testar o que fizemos até agora.

Ao clicarmos no botão "Apagar", os dados serão apagados. Depois, se analisarmos o nosso código, veremos que o `new NegociacaoService` se repete diversas vezes. No entanto, podemos transformá-lo em um atributo de `NegociacaoController.js`, localizado antes `this._init()`.

```
this._service = new NegociacaoService();
```

E em todos os lugares em que `new NegociacaoService` era necessário, iremos substitui-lo por `this._service`. Veremos isto com o método `_init()`:

```
_init() {

      this._service
          .lista()
          .then(negociacoes =>
              negociacoes.forEach(negociacao =>
                  this._listaNegociacoes.adiciona(negociacao)))
          .catch(erro => this._mensagem.texto = erro);

          setInterval(() => {
              this.importaNegociacoes();
          }, 3000);
}
```

No navegador, veremos que as negociações são importadas automaticamente. Se cadastrarmos novas informações no formulário, elas serão incluídas. Só um detalhe: as negociações importadas não estão sendo salvas no `indexDb`. Apenas as que são inclusas pelo formulário. Mas você tem a opção de gravar todas elas.