02

Criando a classe MensagemView

Transcrição

Criamos o modelo de negociação, agora, criaremos o `this._mensagem` no `NegociacaoController.js` :

```
class NegociacaoController {

    constructor() {

        let $ = document.querySelector.bind(document);
        this._inputData = $('#data');
        this._inputQuantidade = $('#quantidade');
        this._inputValor = $('#valor');
        this._listaNegociacoes = new ListaNegociacoes();

        this._negociacoesView = new NegociacoesView($('#negociacoesView'));
        this._negociacoesView.update(this._listaNegociacoes);
        this._mensagem = new Mensagem();

    }

//...
```

Sabemos que quando for feita uma adição, queremos dizer que o `this._mensagem.texto` :

```
adiciona(event) {

    event.preventDefault();
    this._listaNegociacoes.adiciona(this._criaNegociacao());
    this._mensagem.texto = 'Negociacao adicionada com sucesso';
    this._negociacoesView.update(this._listaNegociacoes);
    this._limpaFormulario();
}
```

Se preenchermos o formulário, os dados serão inseridos na tabela, mas a mensagem não, porque ainda não foi criada a View da mesma. Faremos isto a seguir.

Na pasta `views` , criaremos o arquivo `MensagemView.js` :

```
class MensagemView {

  constructor(elemento) {
        this._elemento = elemento;
  }

  _template(model) {

    return `<p class="alert alert-info">${model.texto}</p>`;
```

```
    }
}
```

Usaremos o `alert alert-info` do bootstrap, seguido pela expressão `${model.texto}` .

Logo abaixo, adicionaremos o método `update()` que receberá o `model` .

```
update(model) {

    this._elemento.innerHTML = this._template(model);
}
```

Vamos agora, importar a View no `index.html` :

```
<script src="js/app/models/Negociacao.js"></script>
<script src="js/app/controllers/NegociacaoController.js"></script>
<script src="js/app/helpers/DateHelper.js"></script>
<script src="js/app/models/ListaNegociacoes.js"></script>
<script src="js/app/views/NegociacoesView.js"></script>
<script src="js/app/models/Mensagem.js"></script>
<script src="js/app/views/MensagemView.js"></script>
<script>
    let negociacaoController = new NegociacaoController();
</script>
```

No `NegociacoesController.js` , colocaremos a View assim que a página for recarregada:

```
class NegociacaoController {

    constructor() {

        let $ = document.querySelector.bind(document);
        this._inputData = $('#data');
        this._inputQuantidade = $('#quantidade');
        this._inputValor = $('#valor');
        this._listaNegociacoes = new ListaNegociacoes();

        this._negociacoesView = new NegociacoesView($('#negociacoesView'));
        this._negociacoesView.update(this._listaNegociacoes);

        this._mensagem = new Mensagem();
        this._mensagemView = new MensagemView();

    }

//...
```

O `MensagemView` recebeu onde queremos incluir a mensagem no HTML. De volta ao `index.html` , vamos colocar a mensagem antes da tag `<form>` :

```
<body class="container">

    <h1 class="text-center">Negociações</h1>

    <div id="mensagemView"></div>

    <form class="form" onsubmit="negociacaoController.adiciona(event)">

<!-- ... -->
```

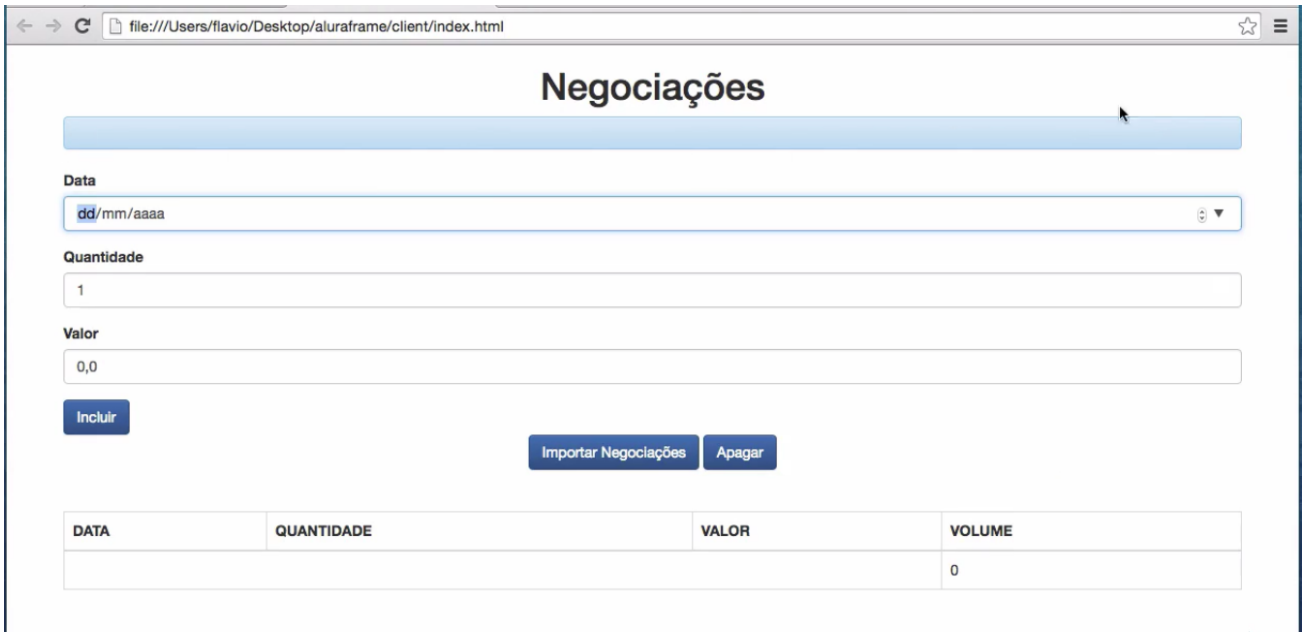
Depois, precisaremos pegar o elemento do DOM no `NegociacaoController.js` , adicionando o `$` .

```
this._mensagem = new Mensagem();
this._mensagemView = new MensagemView($('#mensagemView'));
this._mensagemView.update(this._mensagem);
```

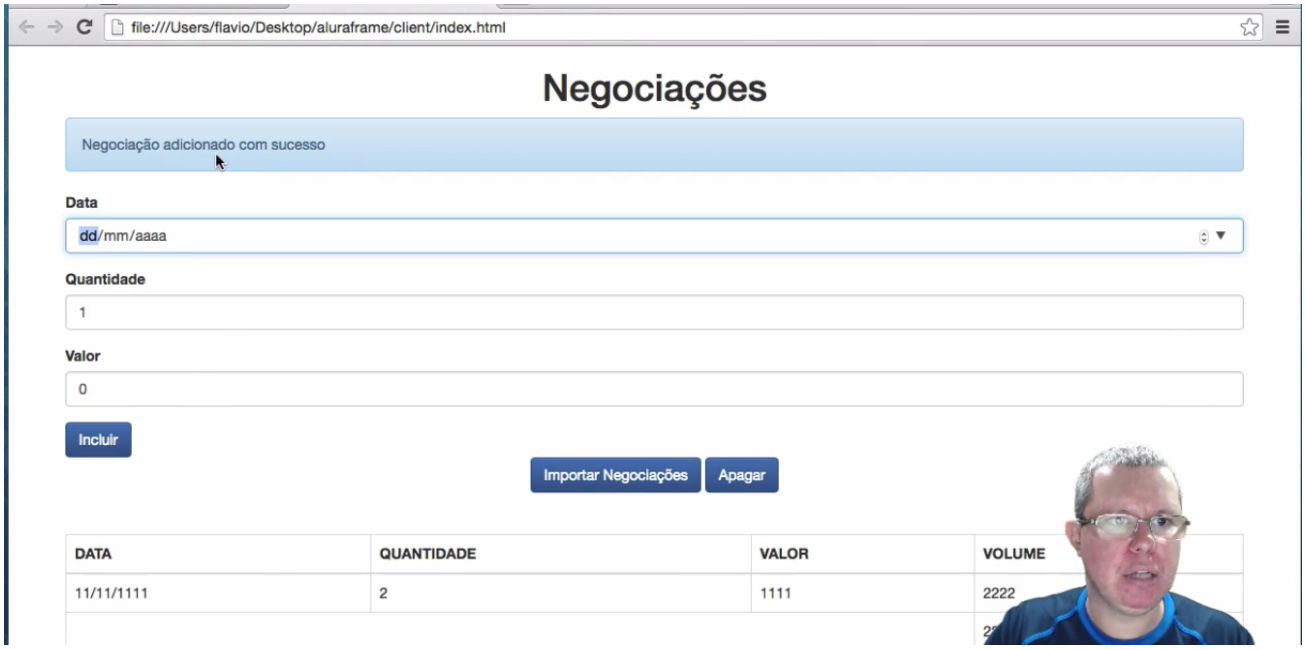
Usamos o `update` e dentro passamos o `this._mensagem` . Vamos incluir o `this._negociacoesView` também no método `adiciona()` :

```
adiciona(event) {

    event.preventDefault();
    this._listaNegociacoes.adiciona(this._criaNegociacao());
    this._negociacoesView.update(this._listaNegociacoes);

    this._mensagem.texto = 'Negociacao adicionada com sucesso';
    this._mensagemView.update(this._mensagem);

    this._limpaFormulario();
}
```

Vamos ver se algo já é exibido no navegador.



Agora aparece um barra com um fundo azul, isto é uma mensagem do bootstrap vazia. A mensagem não deveria estar sendo exibida, considerando que a nossa string está em branco. Vamos testar cadastrar uma nova negociação no formulário.



Conseguimos adicionar os dados a tabela e a mensagem de sucesso apareceu corretamente. Veja que conseguimos usar o mesmo mecanismo de criação da View para lidar com as mensagens do sistema. As ações de importar e apagar negociações podem ser associadas com a atualização de mensagem. Quando chamarmos o `update` na View, passando o `model`, este atualizará a tela. Mas queremos retirar o parágrafo com o fundo azul que aparece acima do formulário. Resolveremos isso em `MensagemView.js`.

Na classe `_template`, faremos um `if` ternário:

```
_template(model) {

    return model.texto ? `<p class="alert alert-info">${model.texto}</p>` : '<p></p>';
}
```

Nós vamos retornar um parágrafo sem a classe. Em JavaScript, uma `string` sem conteúdo é avaliada como falso. Podemos testar se o `modelo.texto` é uma string em branco, `0` ou `null`, nesses casos, a resposta é falso. Mas se tiver texto, vai dar verdadeiro e o retorno será o template. Caso contrário, o retorno será um parágrafo sem a classe `alert-info` e, consequentemente, sem a tarja azul. Se inspecionamos o elemento do DOM no Console, vemos que o paragrafo está vazio:

```
<div id="mensagemView">
    <p></p>
</div>
```

Não aparece a classe do bootstrap. Mas se cadastramos a negociação no formulário, a mensagem aparecerá corretamente. Conseguimos resolver a parte das mensagens para o usuário. Mas será que conseguimos melhorar ainda mais o código?