

MVC na prática

A partir do capítulo 2 vamos reescrever o código com a finalidade de torná-lo mais profissional. Para isso, vamos criar estruturas separadas como **Model View Controller**.

Estrutura de pastas

Ao final do capítulo 6 teremos a seguinte estrutura de arquivos:

- client
 - app
 - controllers
 - NegociacaoController.js
 - domain
 - Negociacao
 - Negociacao.js
 - Negociacoes.js
 - ui
 - converters
 - DateConverter.js
 - views
 - NegociacoesView.js
 - app.js
 - css
 - index.html

A organização das pastas segue o padrão **MVC (Model-View-Controller)**, que é utilizado para separar responsabilidades no código, tornando-o mais modular, legível e fácil de manter. Aqui está o motivo para cada pasta e arquivo:

- **app**: Contém a lógica principal da aplicação.
- **controllers**: centraliza a lógica de controle.
- **domain**: Define as entidades principais da aplicação.
- **ui**: Gerencia a conversão do usuário.
 - **converters**: Classes utilitárias para conversão de dados.
 - **DateConverter.js**: Converte datas entre formatos diferentes.
 - **views**: Classes responsáveis pela apresentação dos dados ao usuário.
 - **NegociacoesView.js**: Renderiza a lista de negociações na interface.

Quem é quem no MVC?

- **Model**:
 - **domain/Negociacao/Negociacao.js**:

- Representa uma negociação individual, encapsulando informações como data, quantidade e valor.
- Fornece métodos para acessar e manipular os dados de uma negociação.
- `domain/Negociacao/Negociacoes.js`:
 - Gerencia uma coleção de negociações.
 - Oferece métodos para adicionar negociações e realizar operações sobre a lista, como calcular totais.
- **View:**
 - `ui/views/NegociacoesView.js`
- **Controller:**
 - `controllers/NegociacaoController.js`
- Não se encaixam em MVC
 - **DateConverter:**
 - É uma classe utilitária usada para conversão de dados, como datas, auxiliando na interação entre camadas.
 - **index.html:**
 - Serve como ponto de entrada da aplicação, carregando os scripts, estilos e definindo a estrutura básica da interface.
 - **css:**
 - Contém os arquivos de estilo responsáveis pela aparência da aplicação, como layout, cores e tipografia.

Fluxo da aplicação

Início

Ao abrir a página, as seguintes operações ocorrem:

1. index.html

- Exibe o html, arquivos .js são carregados na sequência em que estão listados

2. Nada mais acontece

Submissão do formulário

- Após o preenchimento dos dados do formulário, o botão "Incluir" é clicado.

1. "Incluir" Clicado

- Isso invoca o `addEventListener` do tipo `submit` no arquivo `app.js`
- O método `adicionar` do `NegociacaoController` é chamado.

2. O método `adicionar` do controller faz:

- A ação `controller.adicionar.bind(controller)` é chamada:
 1. Cria a negociação:
 - - Puxa os dados do formulário e estrutura eles no modelo `Negociacao`.
 - Usa a conversão de dados em `DateConverter`.
 - Adicionar a negociação à lista de negociações (gerenciada pela classe `Negociacoes`)
 2. Atualiza o HTML:
 - Cria um novo objeto de negociação, para não apagar o anterior.
 - Usa a conversão de dados em `DateConverter`.
 - Estrutura as Negociações dentro da estrutura de visualização determinada no `NegociacaoView`:
 - Renderiza a lista de negociações no DOM.
 - Utiliza o método `update` para atualizar a tabela de negociações exibida na interface.
 3. Limpa o formulário.