



Tutorial CRUD em Node.js com driver nativo do MongoDB – Parte 3

Atualizado em 19/07/2020!

O tutorial de hoje é uma continuação de uma série bastante acessada aqui no blog, onde ensino como fazer um sistema de cadastro bem simples em Node.js, usando o web framework ExpressJS e o banco de dados não-relacional MongoDB. Ou seja, o famoso CRUD.

Nesta terceira e última parte, introduzo um conceito importante de construção de interfaces que é a modularização da página através do recurso de partial views do EJS (view-engine que estamos usando com ExpressJS) e ensino como dar um “tapa” no visual usando o framework front-end Bootstrap, muito utilizado mundialmente para estruturar aplicações web responsivas e elegantes.

Para conseguir acompanhar todos os códigos, é importante que você tenha realizado a parte anterior ou pelo menos baixe os códigos-fonte que se encontram no formulário ao final do tutorial anterior (parte 2). Caso prefira assistir um vídeo, a primeira parte dessa série pode ser resumida com este vídeo abaixo que é uma aula gratuita do meu curso de Node.js e MongoDB.

Aproveite!

Node.js e MongoDB #22 - Aulas Grátis





O conteúdo do post é:

- O que são Partial Views?
- Partial views com EJS
- Adicionando o Bootstrap
- Customizando a aparência

Vamos lá!

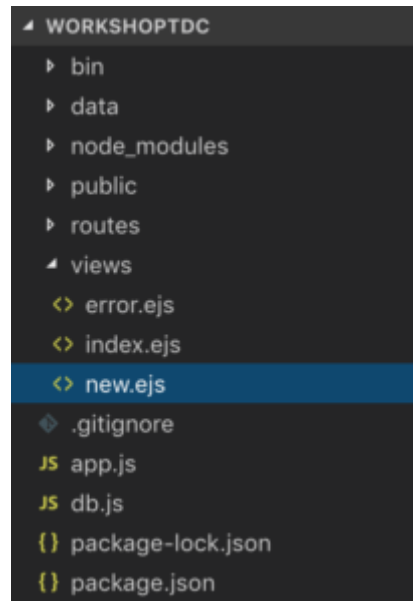
Atenção, uma versão ainda mais completa deste tutorial está disponível em videoaula em meu curso de Node.js e MongoDB.

O que são Partial Views?

É muito comum quando estamos construindo aplicações web que certas porções de nossas views (telas) acabem se repetindo. Por causa disso, todas as linguagens de programação para web possuem recursos de partial views, ou seja, de modularização da interface a partir de “partes” da tela que são construídas separadas e depois são montadas em conjunto para que o usuário veja apenas o resultado final, completo.

Fazendo uma analogia com o que fazemos no código backend do Node.js, uma partial view é como se fosse um módulo NPM, que depois usamos o require para carregá-lo em nosso código. Em outras plataformas isso é chamado de import, include, using, etc.

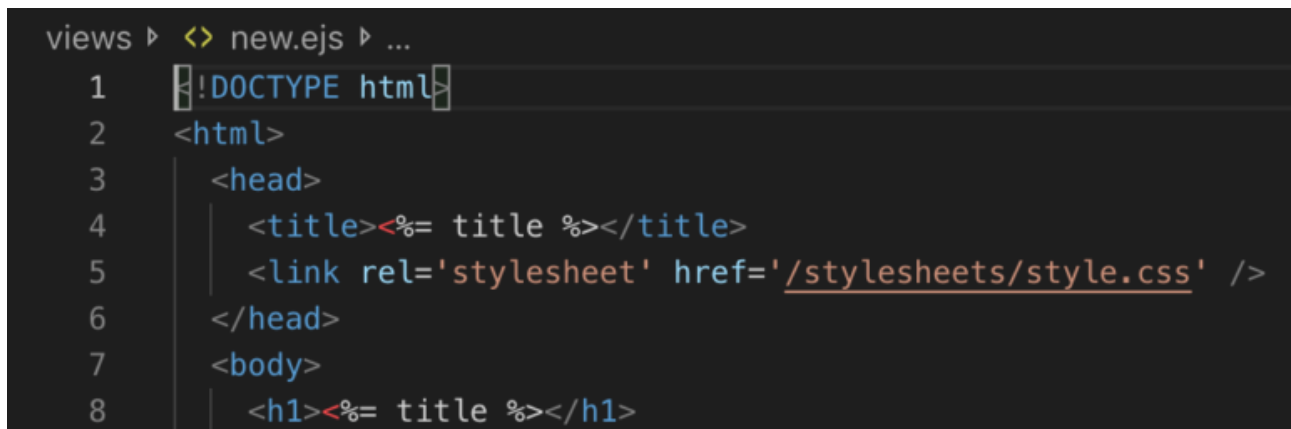
Nossa aplicação é bem simples, possui apenas três views: index, new e error, conforme você vê na imagem abaixo.



Estrutura do projeto

Essas views por sua vez também são mega simples, logo, não há uma reeeaaal necessidade de usar partial views aqui, usaremos mais a título de conhecimento. No entanto, esse conceito é muito poderoso e além de tornar seu código de front-end mais elegante e fácil de ler, ele torna a programação de novas páginas muito mais produtiva, pois sempre aproveitaremos elementos genéricos da interface que serão criados apenas uma vez.

Se olharmos agora para as views index e new, você deve notar que existem muitas tags em comum no início e no fim delas. É por aí que vamos começar a melhorar.

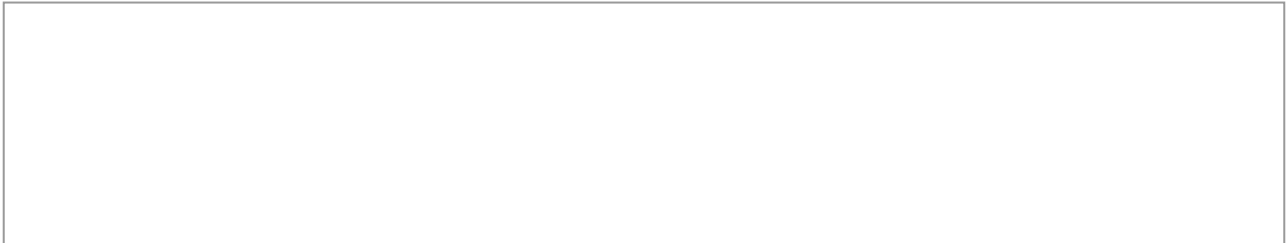


Estrutura que se repete

Partial views com EJS

Não é de hoje que eu uso EJS (Embedded JavaScript) em meus tutoriais e neste não será diferente. Apesar de não ser a view-engine (motor de visualização) padrão do ExpressJS (esse título é do PUG/Jade), ele é meu favorito pois a curva de aprendizado é ridiculamente baixa, considerando que usa HTML + JS, coisa que todo dev web está careca de saber.

Como não poderia deixar de ser, o EJS possui a funcionalidade de criar partial views (lembro dos meus tempos de ASP.NET em que chamávamos isso de MasterPages...). Isso é bem fácil de fazer, comece criando um arquivo top.ejs (top=topo) na sua pasta views e nele vamos colocar toda a parte comum que teremos no topo de todas páginas do nosso sistema, que é apenas o HTML abaixo por enquanto:



Note que nesta top.ejs estamos usando algumas variáveis JS que devem vir do backend, através do model passado na função render do Node.js, lembra?

Agora vamos criar mais um arquivo novo, o bottom.ejs (bottom=rodapé), também na pasta views e nele vamos colocar toda a parte comum que teremos no rodapé de todas páginas do nosso sistema, que é apenas o HTML abaixo por enquanto:



E agora, como fazemos para as demais views do projeto usarem essas partial views para compor a sua estrutura?

Bem simples, vamos começar pela index.ejs, remova a parte do topo do arquivo que é repetida ao arquivo top.ejs e no lugar desse pedaço, deixe como abaixo, usando o comando ‘include’ para “incluir” a partial view na primeira linha do arquivo:



Note que já incluí também o `include` para o `bottom.ejs` no final do arquivo. Na hora que o arquivo HTML vai ser construído para ser enviado ao browser para renderização, o EJS vai ler essa linha e entender que parte desse HTML está em outro arquivo. Ele vai ir nesse outro arquivo, copiar o conteúdo de lá e colar bem nesse local.

Assim, com esses dois `includes`, nós mantemos nossa aplicação modularizada em três pedaços: topo, centro e rodapé, facilitando manutenção e aumentando o reuso de código.

Sim, reuso de código, pois agora nas páginas `new.ejs` e `error.ejs` nós podemos fazer os mesmos `includes` para não ter de repetir aqueles blocos iniciais e finais de HTML. Os ganhos agora são pequenos, mas no futuro, são enormes pois facilitam muito a evolução e manutenção do sistema web.

Eu não vou colocar aqui o código dos outros dois arquivos `ejs` com os `includes`, embora você possa baixar o projeto completo no final desse artigo deixando seu e-mail.

Depois de fazer estas alterações, coloque sua aplicação Node para rodar (não esqueça de subir o banco MongoDB também) e você vai ver que nada mudou para o usuário, que a tela continua renderizando do mesmo jeito e que as alterações só servem para ajudar o programador a trabalhar melhor mesmo.

Um dos grandes ganhos desta abordagem vem agora: manutenção. Vamos adicionar o framework front-end Bootstrap nessa aplicação pra dar um “tapa” no visual dela e vamos fazer isso de maneira muito fácil porque já deixamos a aplicação preparada para este tipo de alteração.

Adicionando o Bootstrap

Eu não vou me prolongar muito na teoria do Bootstrap por aqui pois falo bastante disso no meu livro [Programação Web com Node.js](#). Inclusive aquele “B” na capa é o símbolo do Bootstrap (os outros símbolos são o JQuery, o HTML5 e o MongoDB).



Livro Node

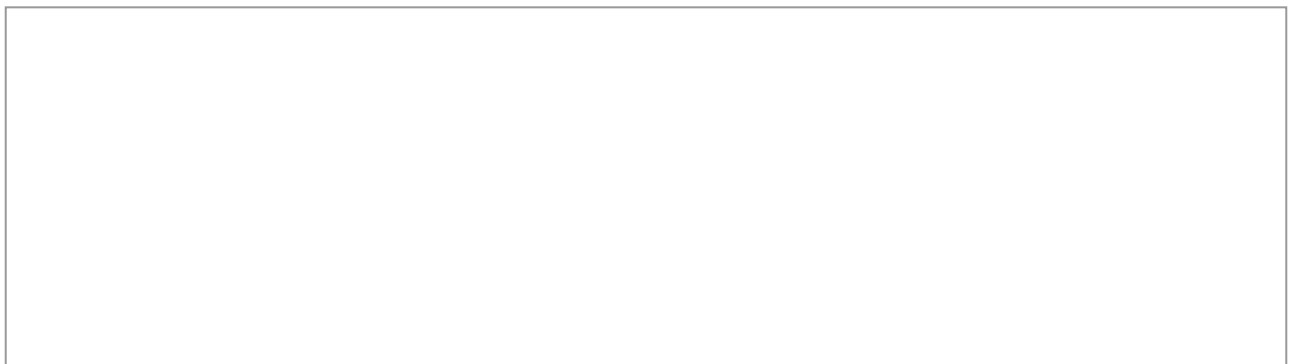
Basicamente o Bootstrap é um framework front-end muito popular no mundo inteiro, fornecendo uma estrutura e comportamentos padronizados que muitos devs web sabem como mexer, acelerando o desenvolvimento e manutenção de qualquer sistema que use Bootstrap como padrão para front-end.

É como se os sites tivessem a mesma estrutura básica, entende?

Apesar da estrutura básica ser a mesma, a aparência pode variar enormemente através da customização de arquivos de estilo (CSS).

A aplicação do Bootstrap em um projeto web se dá através de seus arquivos de CSS e de seus arquivos de JavaScript.

Vamos começar pelo arquivo CSS, que é obtido no site oficial e deve ser adicionado no topo do nosso arquivo `top.ejs`, logo antes da chamada original de CSS que foi colocada pelo `express-generator`:



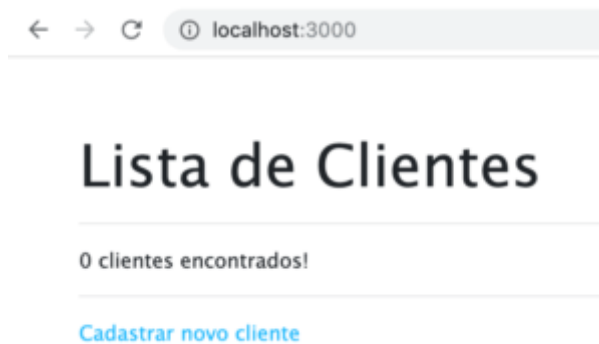
Como adicionamos esta linha no `top.ejs`, TODAS as páginas que incluem essa `partial-view` vão estar agora com o CSS do Bootstrap. Isso é produtividade na programação!

Note que também adicionei duas meta-tags, conforme sugerido na página do Bootstrap, para garantir máxima compatibilidade com os recursos do framework, como a responsividade em dispositivos móveis.

Agora, vamos adicionar os scripts JS necessários pro Bootstrap funcionar corretamente no arquivo `bottom.ejs`. Eles devem ser adicionados logo antes da tag de `fecha-body`:

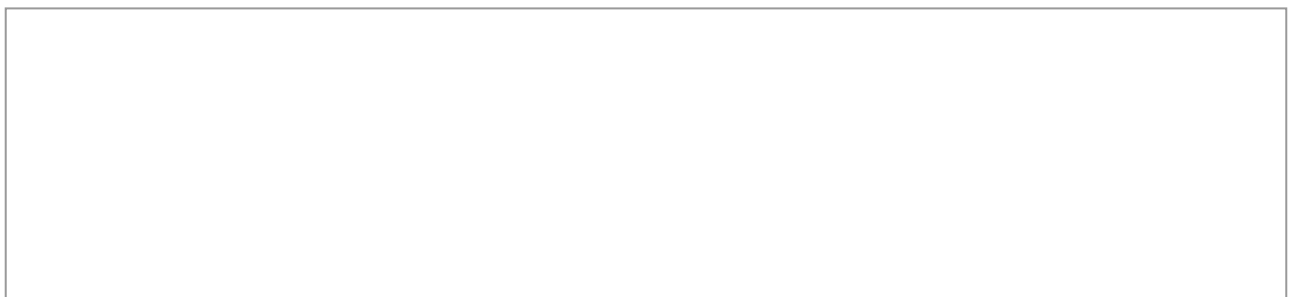


Só o fato de termos feito essas adições já vão gerar alguns efeitos visuais na nossa aplicação, como pode ver na imagem abaixo:



Mas para despertar todo o “poder embelezador” do Bootstrap em nossa aplicação nós temos de fazer alguns ajustes na nossa estrutura, bem como usar algumas classes CSS definidas pelo Bootstrap.

Assim, vamos começar definindo uma “`div class container`” que vai englobar todo o conteúdo do `body` das páginas. Para fazer isso, basta adicionar o conjunto de tags `div` abaixo, com a `abre-div` no `top.ejs` e a `fecha-div` no `bottom.ejs`, de modo que ambas fiquem dentro do `body`.



Note que só mudou uma linha, logo acima do H1, bem no final do top.ejs. Já no bottom.ejs, também só vai mudar uma linha, que fica bem no início do arquivo:

Após essa adição, teremos mais um ajuste perceptível nas páginas já existentes, que ficarão com uma margem e mais centralizadas. A div container faz a sua página funcionar em um grid-system de 12 colunas virtuais, ou seja, o Bootstrap permite que você organize o conteúdo das suas páginas em 12 blocos de tamanhos iguais, como em um Lego. Desde que você siga esta regra cuidadosamente, a sua aplicação web será responsiva, alterando a sua largura conforme a tela do aparelho que estiver acessando a mesma.

E isso é só o início!

Customizando a aparência

Eu poderia ficar aqui durante muitas e muitas horas escrevendo sobre Bootstrap, então vou ser breve. Vamos apenas dar uma embelezada na página para te mostrar o poder do framework e depois vamos encerrar o tutorial. Fica a seu critério depois se aprofundar, seja em [meu livro](#) ou na [documentação oficial](#).

Uma vez adicionado a div container, que é o principal bloco da sua página, vamos personalizar os botões, ou os elementos que queremos que se pareçam com botões. Para isso, basta usarmos duas classes em cada elemento “btn btn-primary” para os botões principais e apenas “btn” para os secundários.

Cada âncora ou submit/button deve ficar parecida com o HTML abaixo:

```
1 | <a href="" class="btn btn-primary">Teste</a>
```

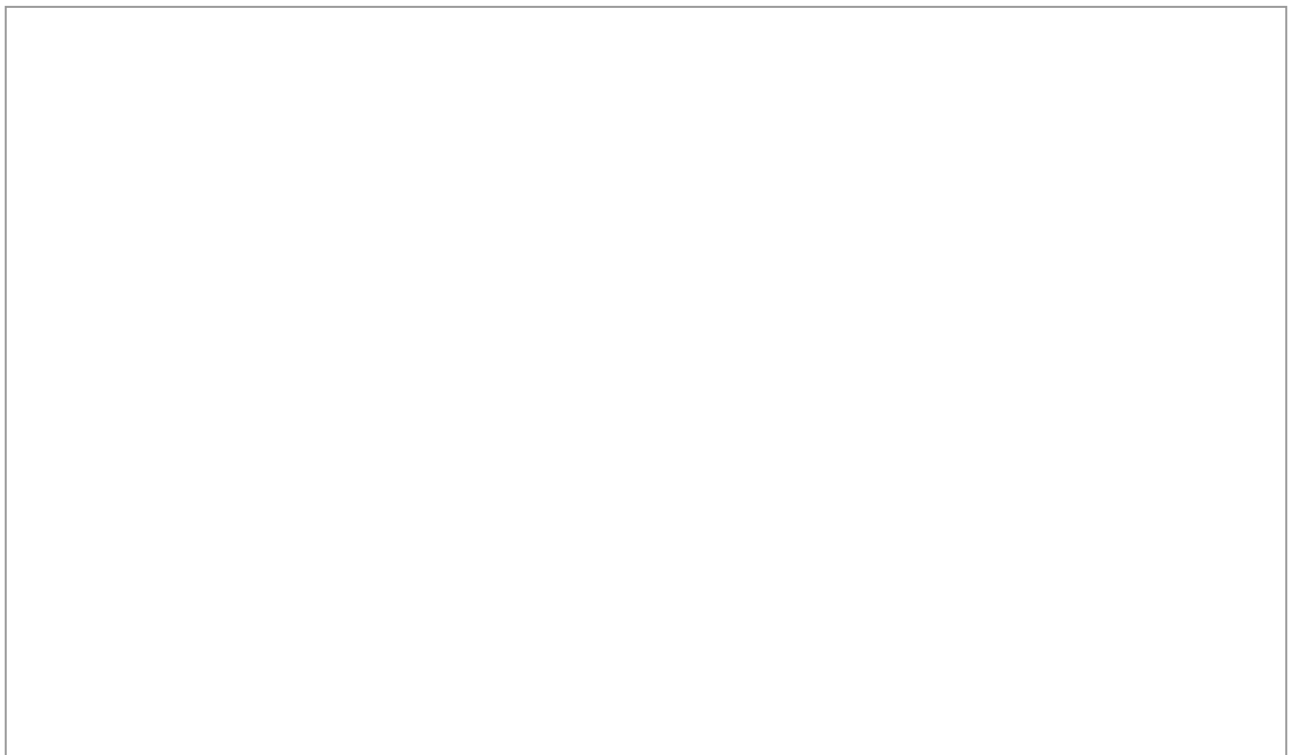
O botão primário é azul no Bootstrap, mas isso obviamente você pode alterar sobrescrevendo ou editando o CSS original do Bootstrap, ou ainda através de templates

que você pega na Internet. As classes CSS de botão também já adicionam um efeito de sombra ao passar o mouse por cima deles, o chamado “hover”.

Para “fechar” a tela inicial, vamos customizar a tabela além do botão, para ela ficar bem bacanuda. Para isso, além de usar o padrão HTML5 para tabelas (com thead, tbody, etc), vamos adicionar uma série de classes a saber:

- **table**: classe padrão Bootstrap para tabelas;
- **table-bordered**: com linhas nas bordas;
- **table-striped**: com linhas “zebradas”;
- **table-hover**: com sombra quando passa o mouse;
- **thead-dark** (no cabeçalho): pro cabeçalho ficar escuro;

O código final que lista os clientes cadastrados pode ser visto abaixo, e substitui a lista antiga:



Fechando de vez esta tela, atualize o código de paginação para ele ficar estilizado conforme recomendado pelo Bootstrap:

```
1 <nav aria-label="Paginação">
2   <ul class="pagination">
3     <%
4       for(var i=1; i <= qtdPaginas; i++) {
5         %>
6         <li class="page-item"><a class="page-link" href="/<%= i %>"><%= i %></a><
7       %> }%>
8   </ul>
```

9 | `</nav>`

Com estas alterações, nossa tela inicial ficará muito melhor do que anteriormente, como mostra a imagem abaixo (coloquei btn-danger ao invés de btn-primary nos botões de exclusão):

Lista de Clientes

Nome	Excluir
Luiz	<input type="button" value="X"/>
Fernando	<input type="button" value="X"/>
Teste	<input type="button" value="X"/>
Outro Teste	<input type="button" value="X"/>
Pedro	<input type="button" value="X"/>

11 clientes encontrados!

1 2 3

Cadastrar novo cliente

Agora a segunda tela, de cadastro de cliente (new.ejs). Além de ajustarmos os botões com as classes que mencionei antes, vamos ajustar o form HTML conforme instruções do próprio Bootstrap:

O resultado, você confere na imagem abaixo (os tamanhos são ajustáveis, mas por padrão ele é responsivo):

Novo Cadastro

Nome

Idade

E com isso encerramos o tutorial de hoje e acredito que esta série sobre o CRUD com driver nativo do MongoDB. Qualquer dúvida que você tiver, deixe aí nos comentários que terei o maior prazer em responder.

Interessado em mais conteúdos sobre Node.js e MongoDB, clique no banner abaixo e conheça o meu curso em videoaulas sobre o assunto!



Curso Node.js e MongoDB

📅 26/07/2020 👤 Luiz Duarte 📁 Desenvolvimento, Node.js 🔗 mongodb, nodejs, web

0 Comentários LuizTools  Política de Privacidade  Entrar ▾

 Recomendar  Tweet  Compartilhar Ordenar por Mais votados ▾





Iniciar a discussão...

FAZER LOGIN COM

OU REGISTRE-SE NO DISQUS 

Nome

Seja o primeiro a comentar.

 Inscreva-se  Adicione o Disqus no seu siteAdicionar DisqusAdicionar

▲ Do Not Sell My Data

Orgulhosamente mantido com WordPress

Não foi possível conectar-se ao serviço reCAPTCHA. Verifique sua conexão com a Internet e atualize a página para ver um desafio reCAPTCHA.