Criar uma pasta como o nome do projeto

Dentro do VS code instalar os seguintes pacotes:

1 – Vai criar o pacote Json do projeto

npm init –yes

2- Express para criar as rotas

npm install express

3 – Mysql

npm install mysql

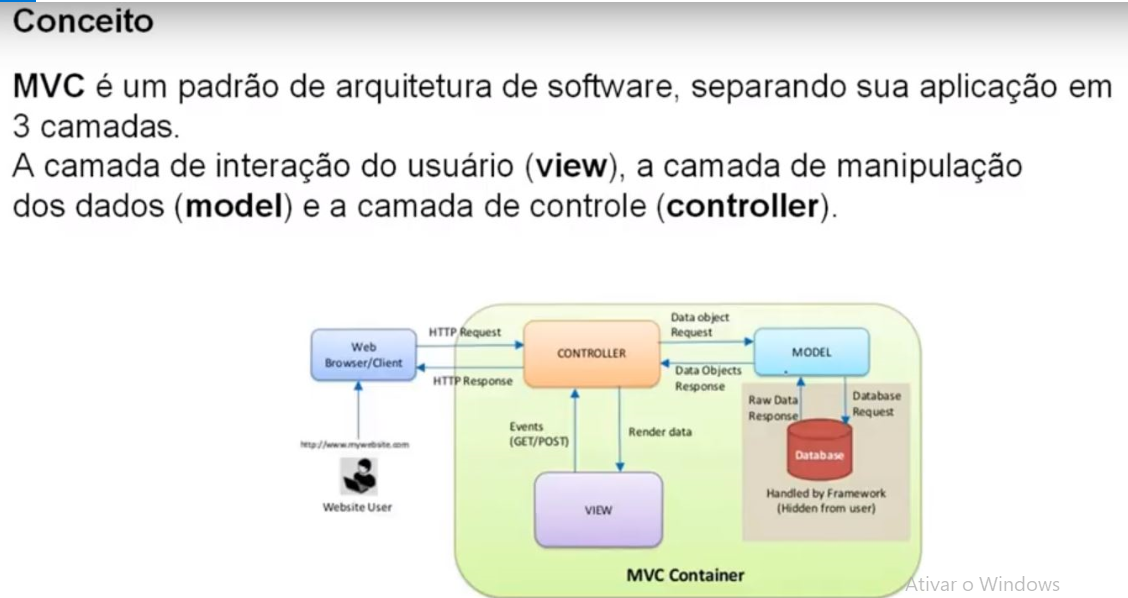
4 – Para realizar o meio de campo entre o node e o mysql

npm install express-myconnection

5- Para ter acesso as requisições por meio do console.log( não é obrigatório)

Npm install Morgan

6 – Modulo que será a engine para visualização de dados

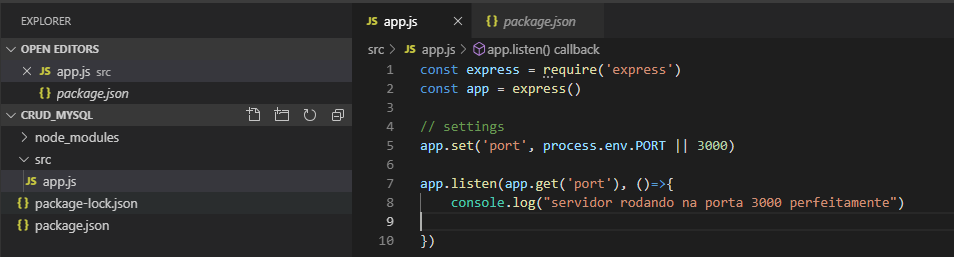


**Começando o Projeto:**

Vamos montar a seguinte estrutura:

Criamos a pasta src dentro da raiz do projeto e o arquivo app.js. esse será o arquivo responsável pelo gerenciamento do servidor.

Construindo o servidor com express:



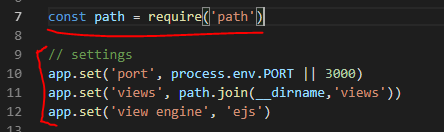


**Configurando a exibição dos arquivos:**

Criamos uma pasta views, tem que ser com esse nome devido ao motor ejs.

Linha 8 – Configuramos que seja utilizado o método ejs dentro da pasta views

Chamamos o modulo path(que já vem dentro do express), para realizar localização dos arquivos.



**Configurando Morgan:**

Criamos a constante Morgan que vai receber o módulo e aplicamos no middleware. Após a aplicação se é atualizado o programa e alguém entra na porta recebemos alguns dados por console.log

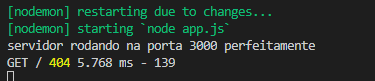
Dentro do app.js

Chamar uma variável para receber o arquivo:



Dentro do middleware atribuir a ele a função abaixo:





**Configurando Mysql:**

Criamos duas constantes que vão receber os módulos mysql e express-myconnection. Posteriormente entramos com os parâmetros de configuração da conexão ln 15 a 22. Esta referencia de single se refere ao express-myconnection

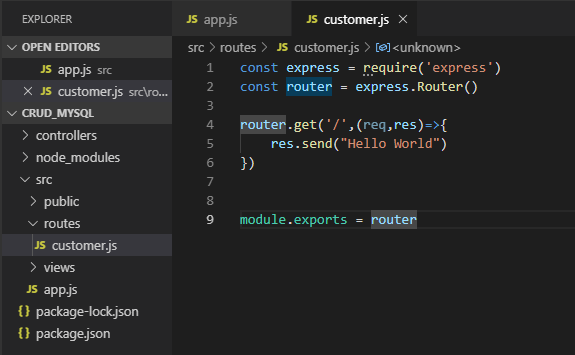


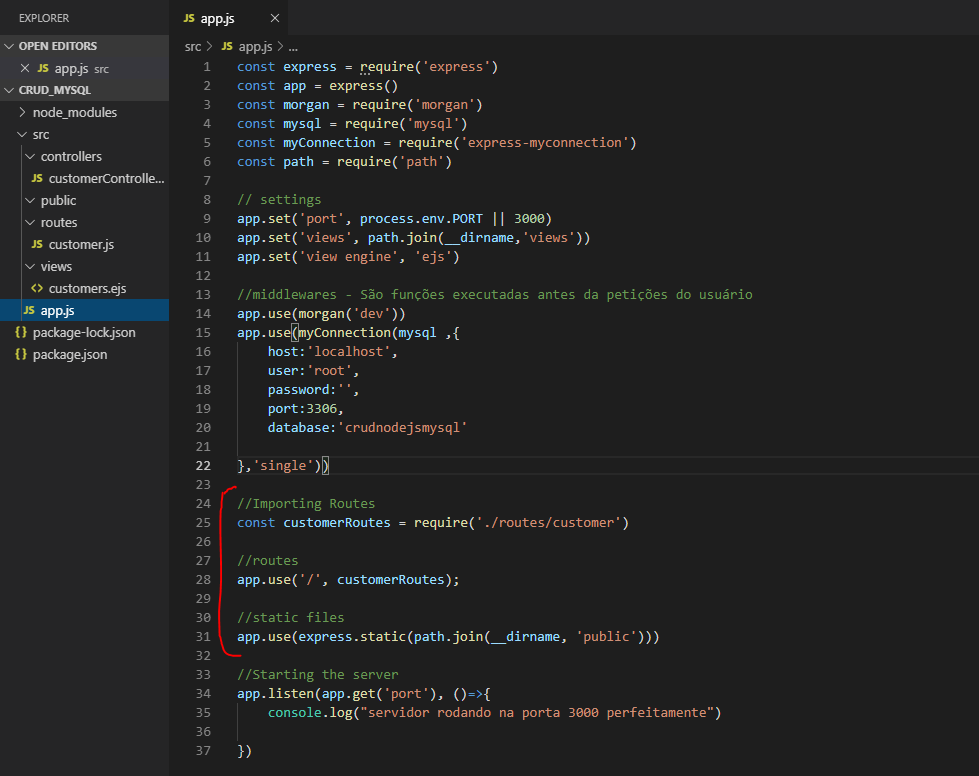
**Configurando Rotas:**

Começamos criando uma pasta dentro do src chamada routes, public e outra chamada controllers.

Dentro de routes criamos o arquivo customer.js

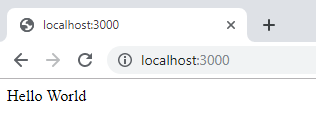
A constante router recebe o método express.Router() para que possamos reutiliza-lo em outras rotas a mesma sintaxe.





Linha 28 A constante criada no Importing Routes(linha 25) vai ser utilizada dentro das rotas.As rotas serão todas criadas dentro do customer.js conforme veremos a seguir.

Linha 32 serve para carregar arquivos estáticos como fotos,css, etc.

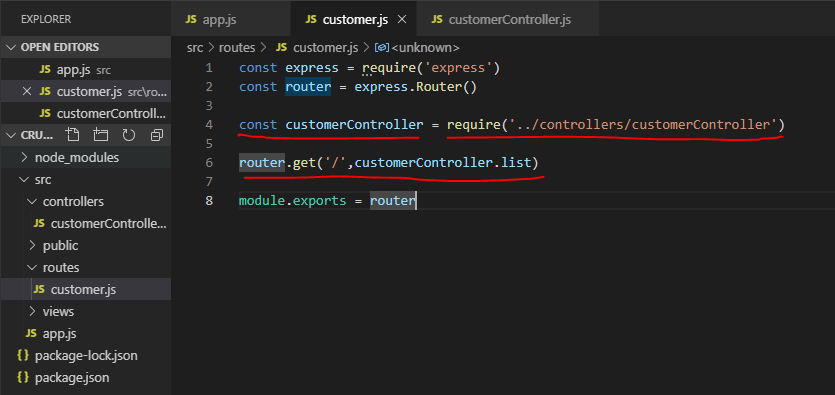


**Usando Controladores:**

Começamos criando um arquivo dentro do src/controllers.



Adicionamos as linhas em vermelho sendo a primeira a atribuição a uma variável o caminho e a segunda referente ao nome da vairavel que acabamos de criar e o método desejado que se encontra dentro do customerController. O método list é um método escolhido para o mysql

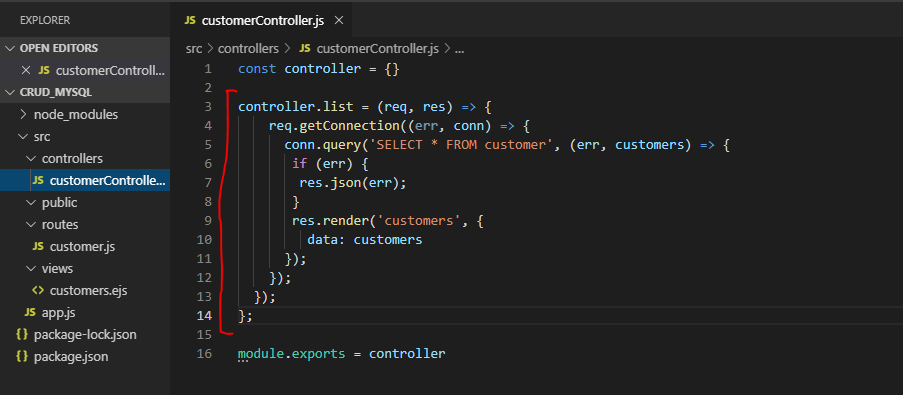


**Configurando o MYSQL:**

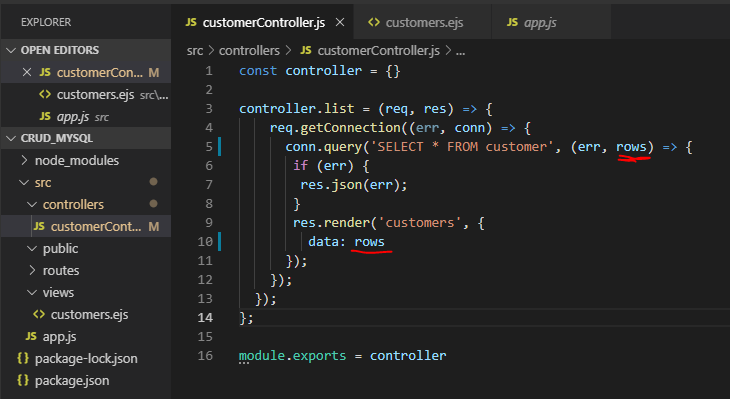
**Visualização**

**Dentro do arquivo customerController :**

1. **Criamos uma variável para receber o vetor**
2. **Essa variável vai receber diversos métodos para a aplicação. Vamos utilizar o método list para que possamos visualizar os dados inicialmente**

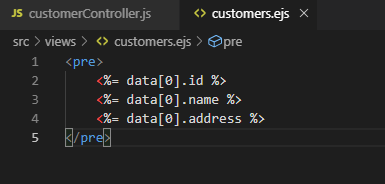


A linha 05 pode ser passada com outra variavel no parênteses, muitas vezes é chamada de row. No exemplo foi criado com o mesmo nome do BD mas pode ser alterado. Segue um exemplo com a variável com outro nome, isso é bom para que possa saber oq é o BD e o que é a variável criada.

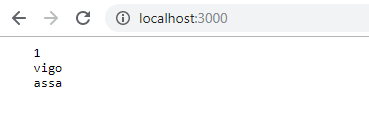


Proximo passo é criar a tela de exibição. Inicialmente colocamos em um arquivo ejs criado dentro de views(customers.ejs)

Para testar vamos utilizar conforme abaixo:



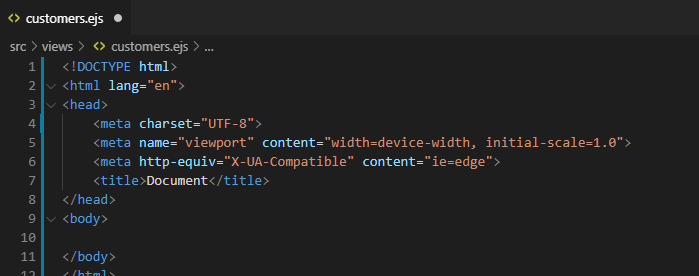
Na tela de saída deve aparecer assim:



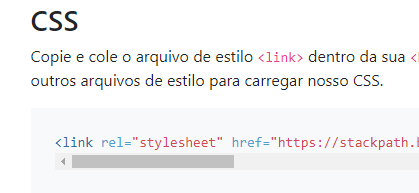
Repare que ele já está exibindo os dados

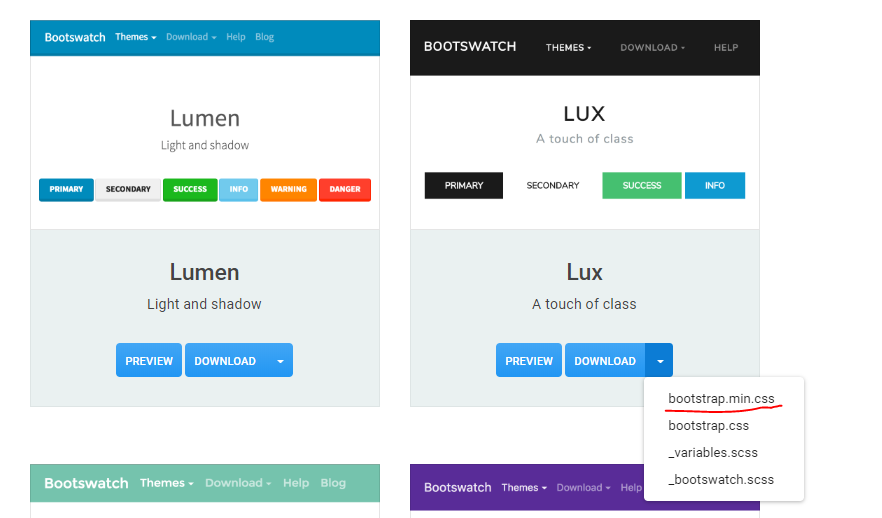
**Construindo a página HTML/Bootstrap para a visualização de dados**

1. Abrimos o arquivo customers.ejs e limpamos os dados anteriores e colocamos um html5



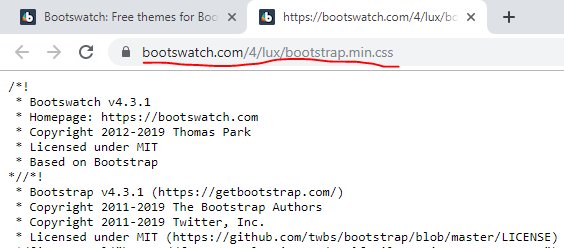
1. Vamos a página do Bootstrap e copiamos o link e depois colamos dentro do head do arquivo. Caso vá utilizar o Bootswatch pular esse passo e passar logo para o passo 3



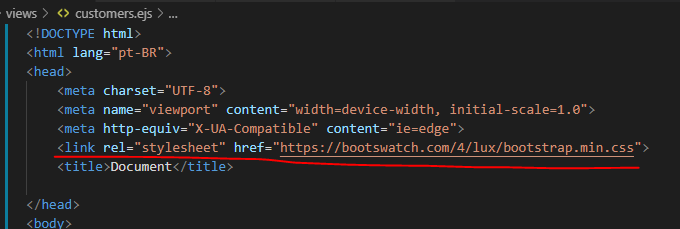
1. Vamos a página bootswatch.com e procuramos o estilo desejado

Clicar com o botão direito do mouse e colocar abrir link em uma nova janela

1. Copiar o link

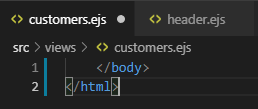


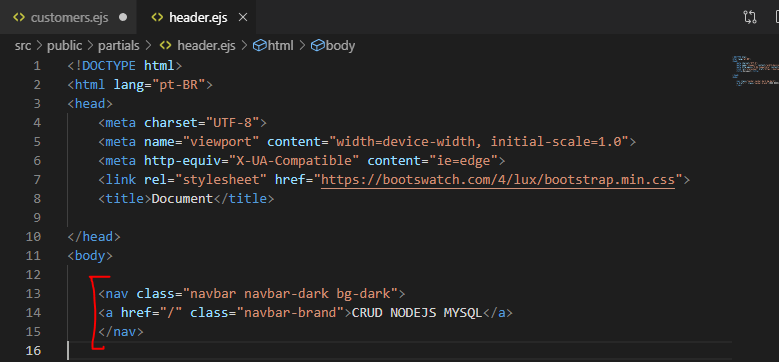
1. Colar onde colocamos o href do bootstrap isso mesmo vamos substitui-lo.



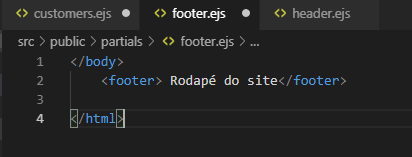
1. Vamos montar o navbar do arquivo em um arquivo separado para que possamos reutiliza-lo em todas as abas do código. Para começar criamos a pasta partials dentro do public : public/partials e dentro da pasta o arquivo header.ejs

Nesse momento nosso arquivo customer.ejs está apenas com o fechamento do body e html e o header.ejs recebeu o restante dos dados acrescentando as linhas do nav(em vermelho)

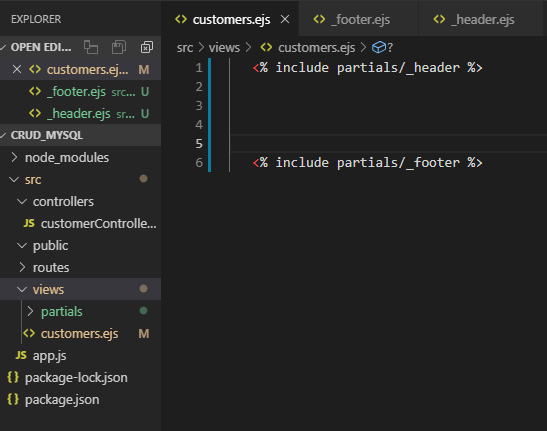




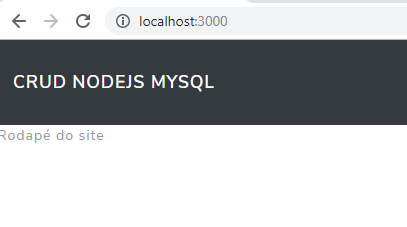
1. Criamos o footer , criando um arquivo dentro do partials/footer.ejs agora pegando o restante do customer.ejs



1. Agora vamos incluir estas partes dentro do customer.ejs



Agora já temos o arquivo com o cabeçalho e o rodapé



1. Criando um formulário para receber os dados dentro de customers.ejs

A formatação está diferente mas a idéia é copiar e colar para facilitar a montagem e trocar os campos de input

<% include partials/\_header %>

<div class="container">

<div class="row mt-5">

<div class="col-md-7">

</div>

<div class="col-md-5">

<div class="card">

<div class="card-body">

<form action="/add" method="post"> Essa linha é correspondente ao endereço que será usado para adicionar o post

<div class="form-group">

<input type="text" name="name" placeholder="Nome" class="form-control">

</div>

<div class="form-group">

<input type="text" name="address" placeholder="Endereço" class="form-control">

</div>

<div class="form-group">

<input type="text" name="phone" placeholder="Telefone" class="form-control">

</div>

<button type="submit" class='btn btn-primary'>

Guardar dados

</button>

</form>

</div>

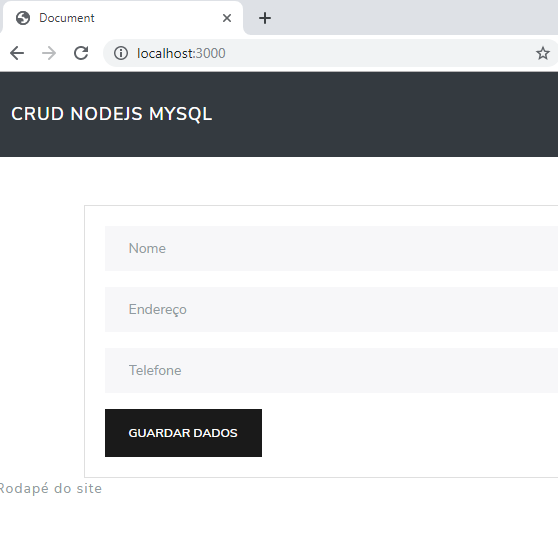
</div>

</div>

</div>

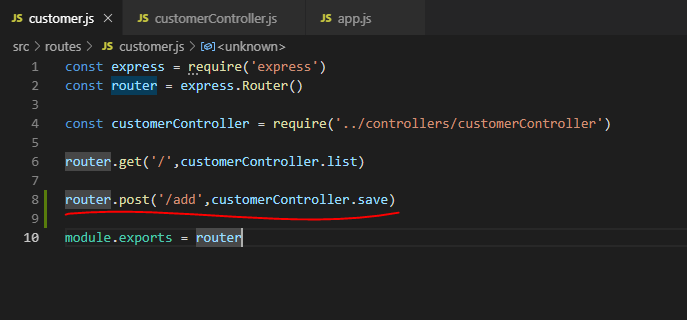
</div>

<% include partials/\_footer %>

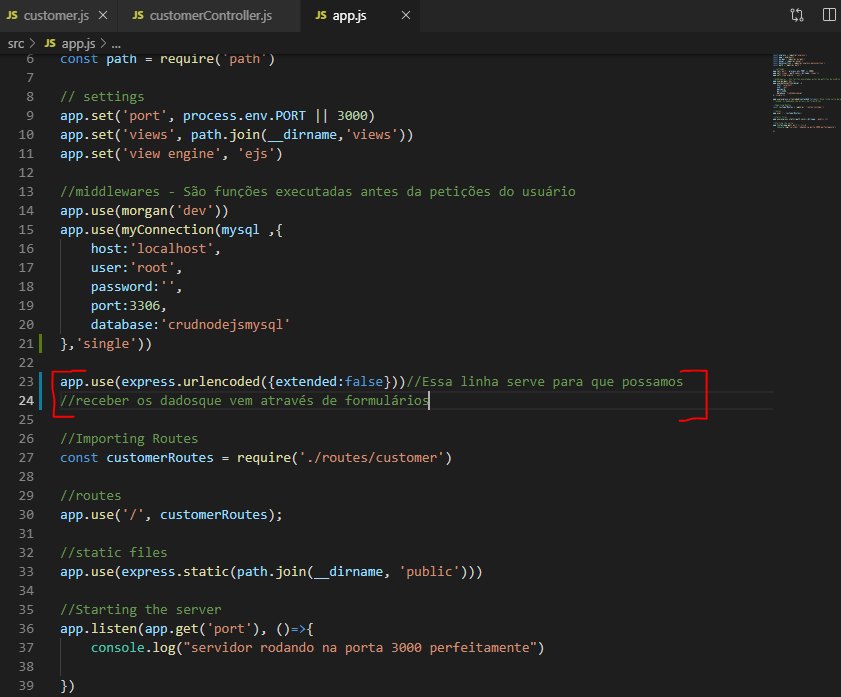


1. Agora vamos tratar de criar a rota em routes/customer.js

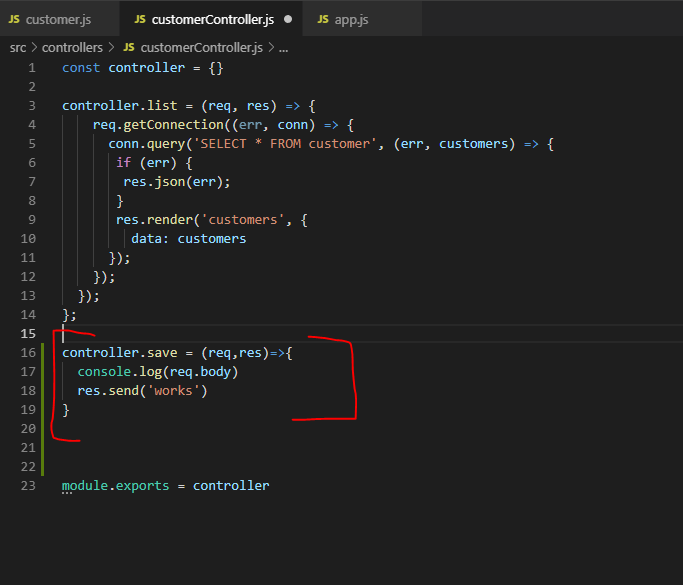
A rota mencionada no formulário com observação em vermelho(acima foi /add) por isso vamos utilizar o mesmo aqui e adicionar a função save



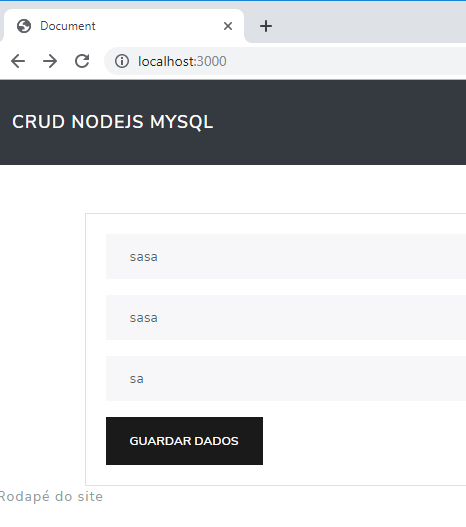
Devemos a linha marcada no app.js para que possamos receber os dados. Ela é referente ao bodyparser porém atualmente é integrada ao express. A extensão false se refere que vamos passar apenas dados.

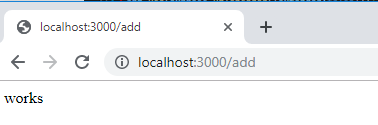


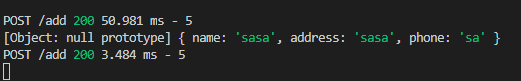
Dentro do customerController.js vamos adicionar o método



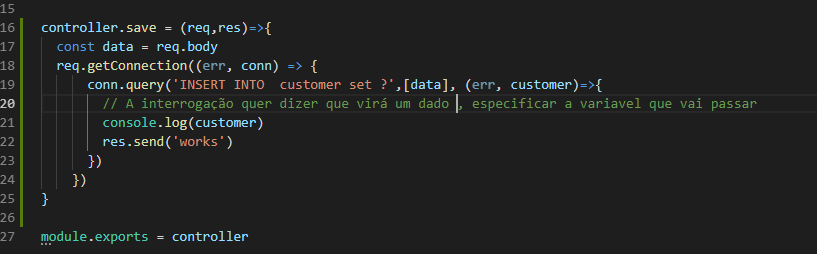
Agora ao imputar algum dado vemos no console log que os dados já estão sendo recebidos embora ainda não estejam sendo imputados no BD.







Agora vamos realizar a conexão com o MYSQL :

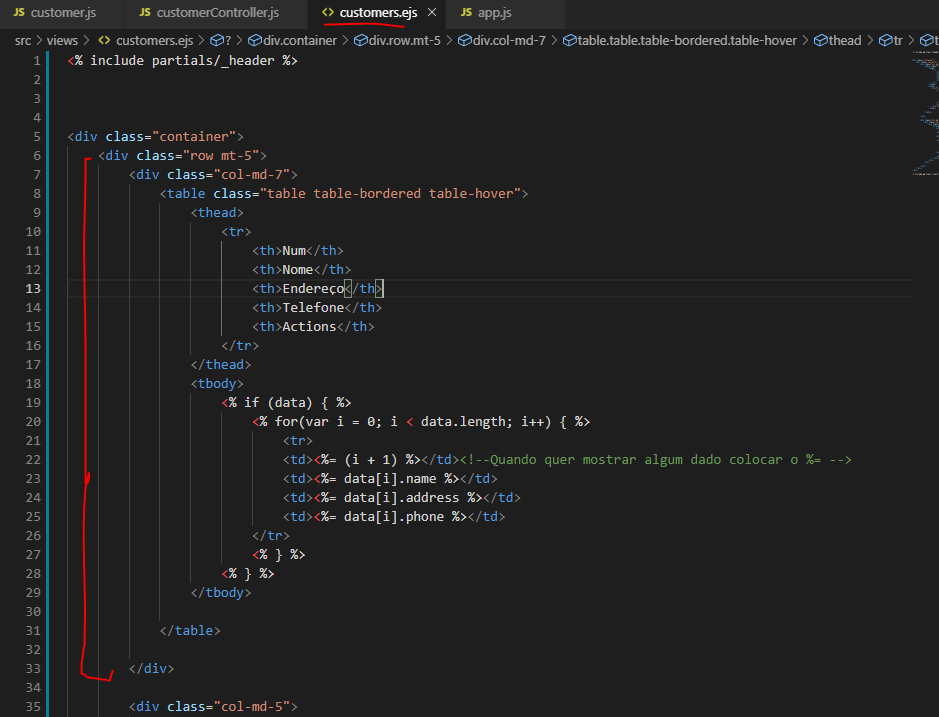


Através de uma const data que vai receber os dados usamos a função conn.query vai inserir os dados no BD.

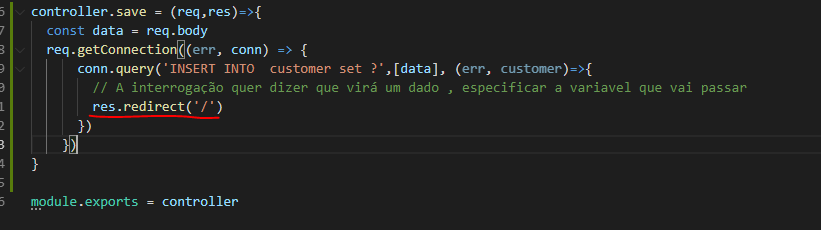
Conseguimos terminar o processo de inserir dados porém ainda se encontra pendente a visualização.

**Visualização de dados**

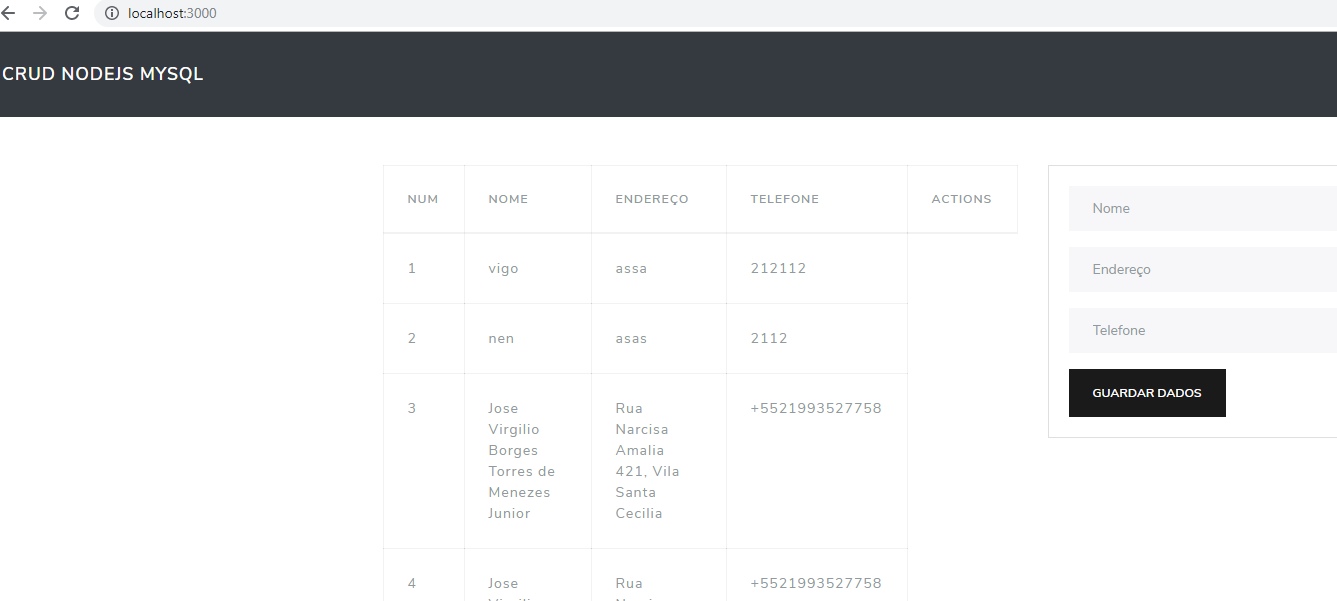
Vamos a customer.ejs e realizar algumas alterações:



Dentro do customerController modificamos o código save para atualizar a página a cada registro.

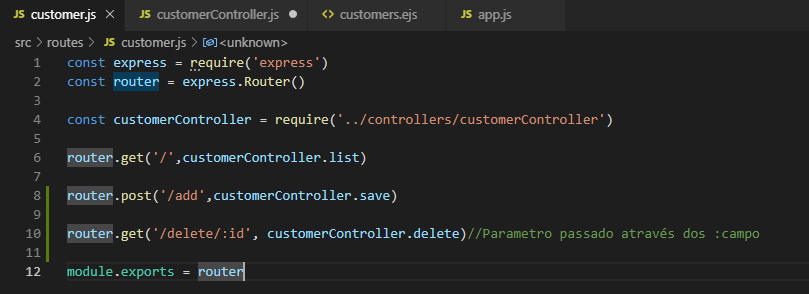


Agora a página já mostra os registros



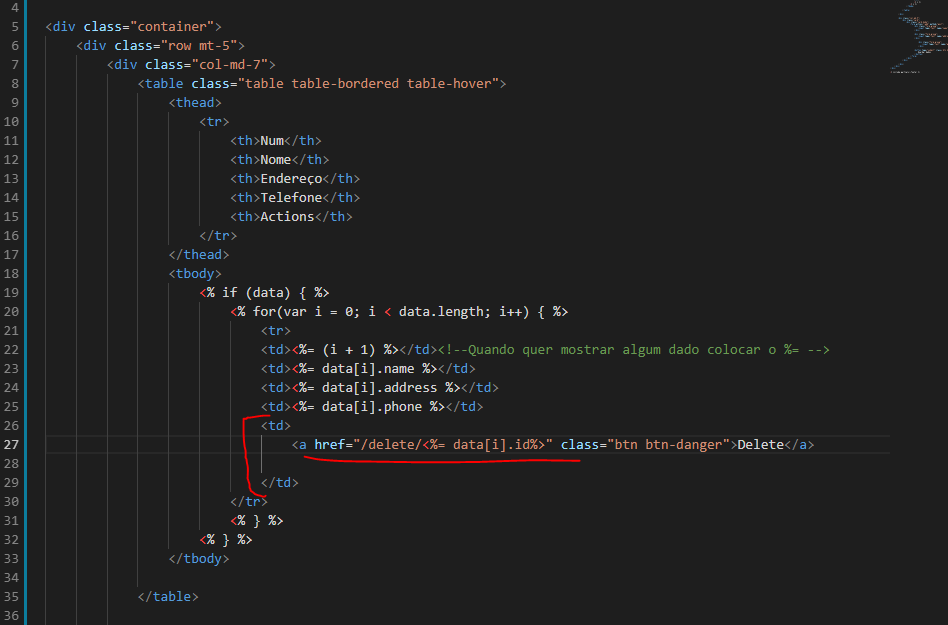
**Deletando os Dados**

Começamos no routes/customer.js criando a rota com passagem de parâmetro id:

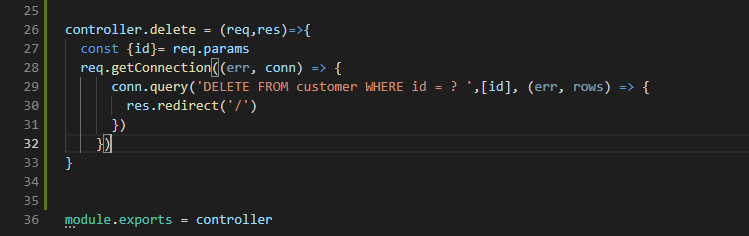


Agora vamos em customer.ejs e acrescentamos o botão delete com passagem de parâmetro. Repare que o href vai enviar a rota delete com o parâmetro do id do bd.

Customer.ejs



Agora vamos configurar o customerController configurar a função delete:



Const {campo desejado do db} = ao parâmetro enviado pelo link

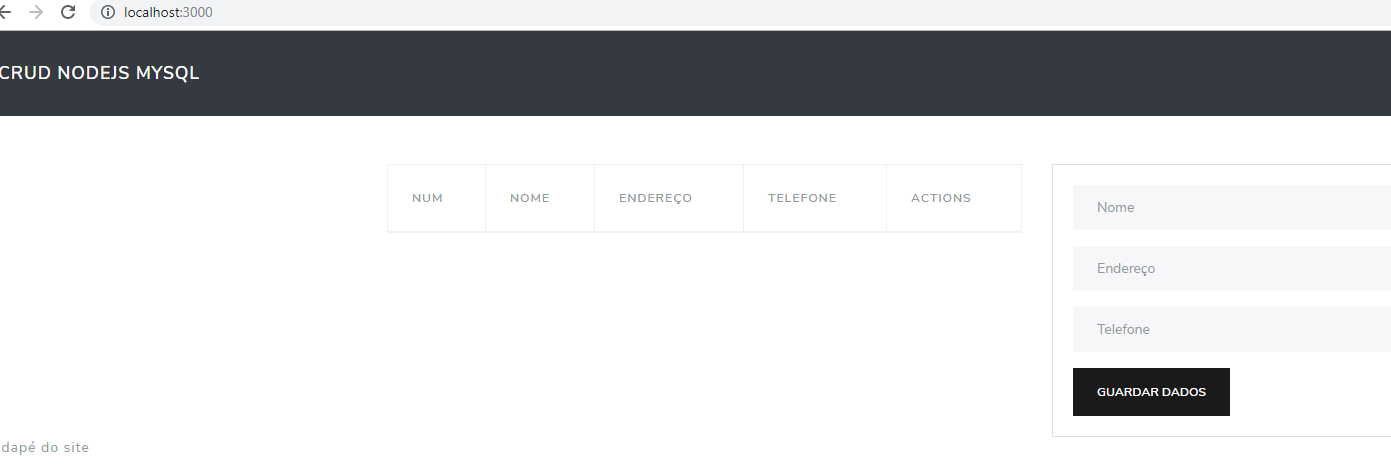
28 – realiza a conexão com o BD

Conn.query ( linha de comando que vai realizar o delete

Res.redirect direciona para a pagina inicial

Salvar e testar pois o delete já está ok.

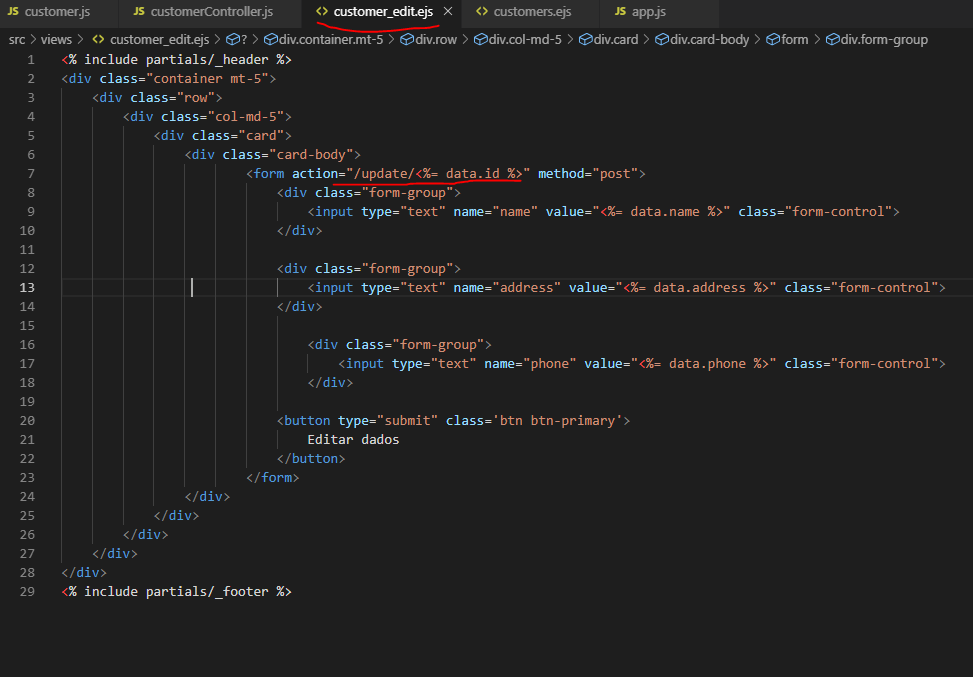
Todos os dados excluídos um a um pelo botão ☺



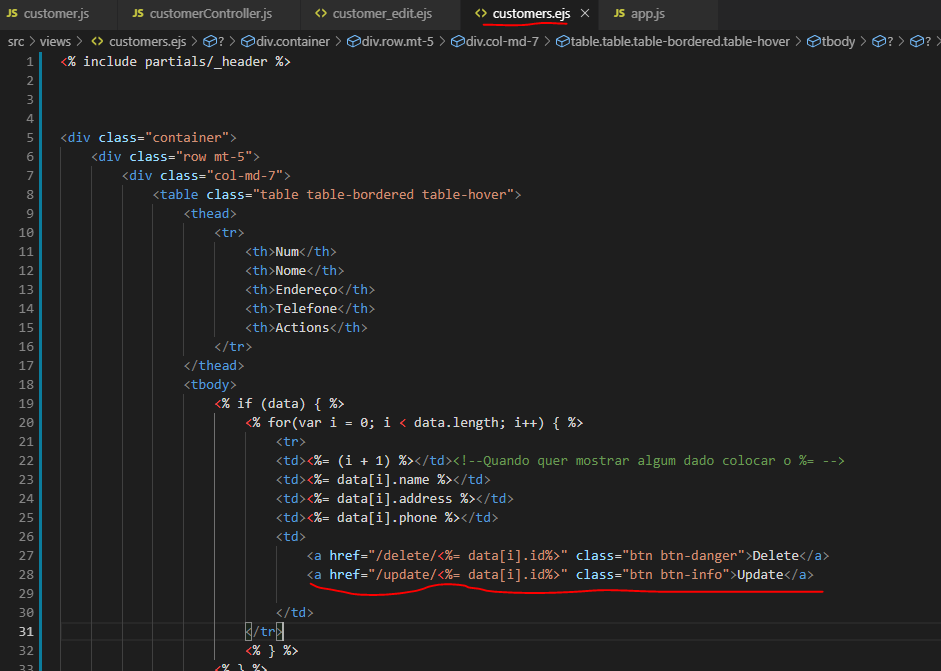
**Editando Dados**

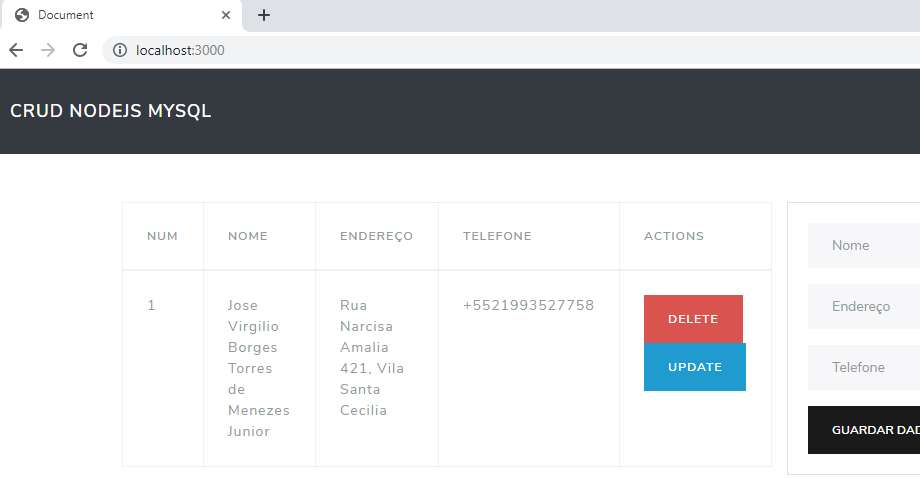
Criamos um arquivo dentro de views chamado customer\_edit.ejs

O Form action vai ter passagem de parâmetro para pegar o id do elemento tb e vai passar os dados pois os campos tem value:

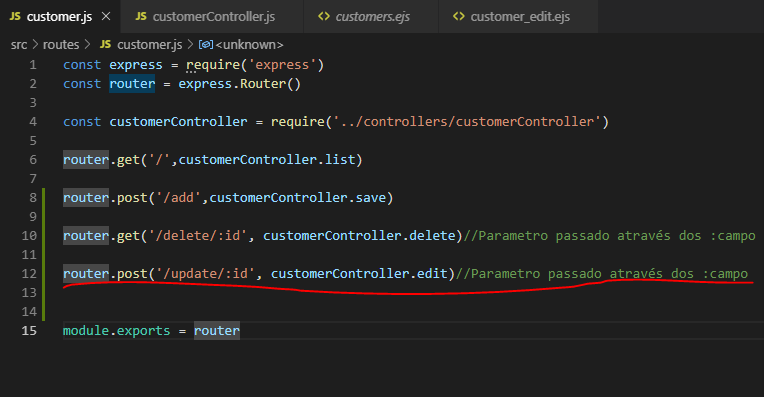


No arquivo customer.ejs vamos acrescentar um botão de update, pois a tela é a que vai dar origem a edição de dados:

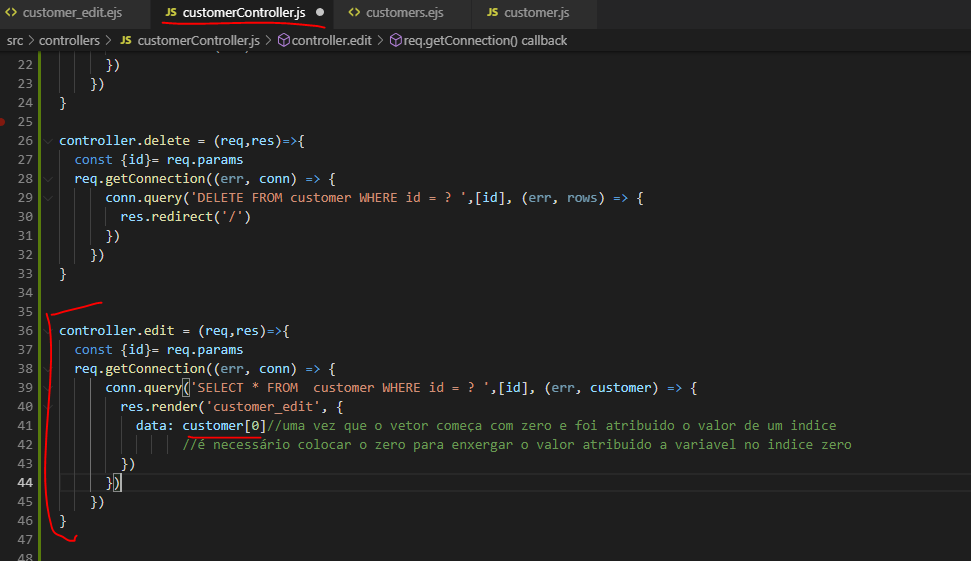




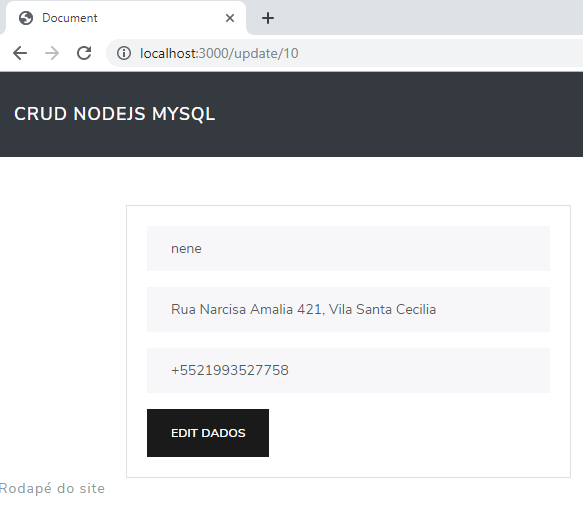
Agora criamos a rota em customer.js



Agora vamos a função de edição de dados:

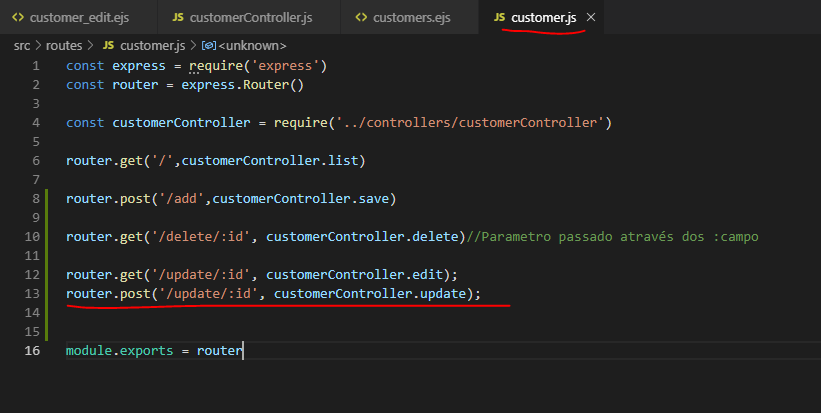


Clicando em Update é passado o formulário com os valores:

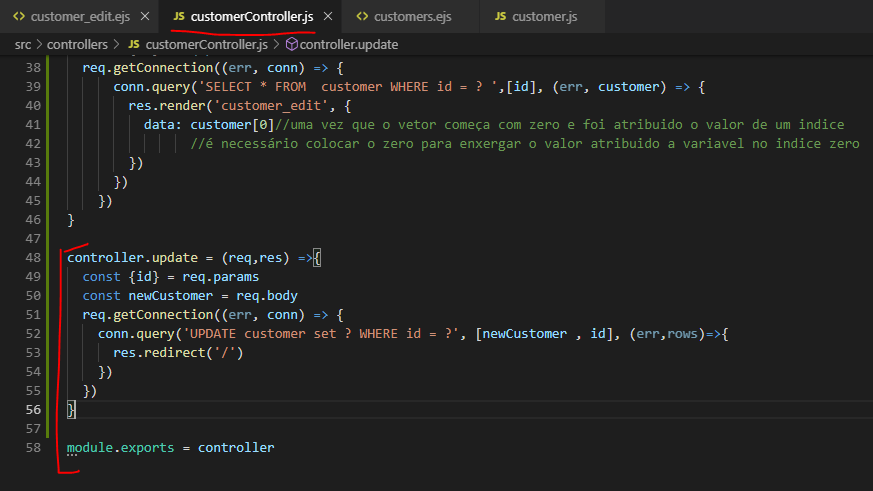


Agora vamos criar a função de edição através do botão edit dados

Vamos criar uma rota igual a anterior porém com o método post e vamos atribuir a uma função chamada .update



Agora Criamos a função:



Projeto Funcionando 100%

Salvo em Node/crud\_mysql

Arquivo do professor / crud\_mysql\_Master

<https://www.youtube.com/watch?v=VxRXlUrV6y0&t=3671s>