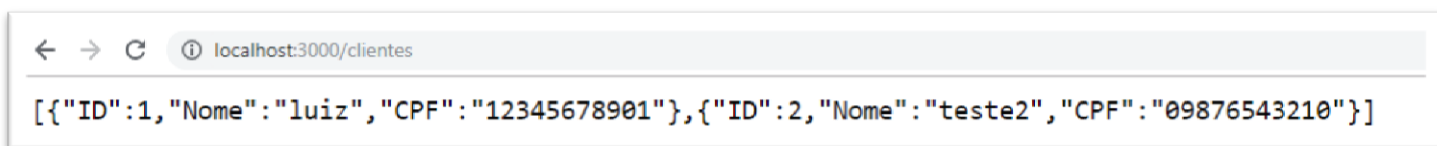


# Criando Páginas Dinâmicas com Node.js

## Aplicação Node.js

Uma vez que temos uma aplicação em Node.js para buscar os dados de um banco de dados, com vimos anteriormente, podemos agora exibi-los em uma página web para o usuário e implementar as demais operações CRUD.

Vamos considerar a aplicação **api.js** que lista os dados da tabela Clientes do banco de dados. Os dados retornados por essa aplicação são exibidos com a seguinte estrutura JSON:



Os dados retornados estão no formato JSON. Agora veremos como exibiremos esses dados em uma página web.

## Exibindo os dados em uma página web

As páginas criadas a seguir são independentes do projeto que possui a API para acessar o banco de dados. Vamos criar um arquivo com a seguinte estrutura HTML:

```
<!DOCTYPE html>
<html>
<head>
  <title>Lista Clientes com Node.js</title>
</head>
<body>
  <h2> Clientes </h2>
  <div id="id01"></div>
</body>
</html>
```

Iremos usar o AJAX para buscar os dados da aplicação acima e exibi-los na página. Vamos inserir o código abaixo para que através do AJAX a página acesse o endereço <http://localhost:3000/clientes> de onde obteremos os dados da tabela Clientes.

O retorno dos dados são direcionados para a função `ExibeDados()` através do comando: **`ExibeDados(this.responseText)`**. Essa função irá montar uma string contendo uma tabela do HTML com os dados obtidos pelo AJAX. Depois disso, essa string será passada para a `<DIV>` com o id "ID01" através do comando **`document.getElementById("id01").innerHTML = out`**. Sendo que **`out`** é a string com a tabela em HTML a ser exibida na página.

```
<script>
var xmlhttp = new XMLHttpRequest();
var url = "http://localhost:3000/clientes";

xmlhttp.onreadystatechange=function() {
  if (this.readyState == 4 && this.status == 200) {
    //quando os dados retornarem da requisição serão enviados para a função ExibeDados()
    ExibeDados(this.responseText);
  }
}
xmlhttp.open("GET", url, true);
xmlhttp.send();

function ExibeDados(response) {
  var arr = JSON.parse(response); //converter JSON em array
  var out = "<table>";

  for(var i = 0; i < arr.length; i++) {
    out += "<tr><td>" + arr[i].ID + "</td><td>" + arr[i].Nome + "</td><td>" + arr[i].CPF + "</td></tr>";
  }
  out += "</table>";
  document.getElementById("id01").innerHTML = out;
}
</script>
```

## Configurando o navegador

Para rodar nossa aplicação Node.js em um servidor web local e poder manipular dados por uma página web, precisamos configurar o navegador Chrome para permitir acessar arquivos locais pelo Javascript. Existem várias maneiras de fazer isso, inclusive instalando extensão no Chrome obtido pelo link

<https://chrome.google.com/webstore/detail/allow-control-allow-origi/nlfbmbojpeacfhkpbjhddihlkkiljbi>.

Outra maneira de permitir que o navegador acesse os arquivos localmente é abrir o navegador com as configurações de segurança desabilitadas. Para fazer isso usamos os comandos abaixo, onde será aberta uma nova janela do navegador com essas alterações de segurança:

```
chrome.exe --user-data-dir="C:\Pasta-onde-está-a-aplicação-Node.js" --disable-web-security
```

Exemplo:

```
chrome.exe --user-data-dir="Z:\ProjetoNode" --disable-web-security
```

Para testar nosso exemplo, acessaremos no navegador que foi aberto sem as configurações de segurança a página web que buscará os dados JSON da nossa aplicação em Node.js.



## Exibindo um registro na página

Para implementar uma pesquisa onde o usuário poderá selecionar um único cliente para ser exibido na página, podemos alterar o exemplo acima adicionando um formulário para que seja inserido o ID do Cliente. Vamos fazer as seguintes alterações:

1. Acrescentar o formulário para a pesquisa pelo ID antes do `</body>`

```
<form>
  ID Cliente: <input type="text" id="idCliente">
  <input type="button" id="btnEnviar" value="Buscar dados" >
</form>
```

2. Vamos colocar o AJAX dentro de uma função chamada `EnviaID()` e acrescentar um `if` para passar um parâmetro de busca, caso esse parâmetro seja passado pelo usuário.

```
<script>
function EnviaID() {

  var xmlhttp = new XMLHttpRequest();
  var url = "http://localhost:3000/clientes";

  xmlhttp.onreadystatechange=function() {
    if (this.readyState == 4 && this.status == 200) {
      //quando os dados retornarem da requisição serão enviados para a função ExibeDados()
      ExibeDados(this.responseText);
    }
  }

  if (document.getElementById("idCliente").length == 0 ) {
    xmlhttp.open("GET", url, true);
  } else {
    var idCliente = document.getElementById("idCliente").value;
```

```

    xmlhttp.open("GET", url+"/"+idCliente, true);
  }

  xmlhttp.send();
} //function EnviaID()

function ExibeDados(response) {
  var arr = JSON.parse(response);
  var out = "<table>";

  for(var i = 0; i < arr.length; i++) {
    out += "<tr><td>" + arr[i].ID + "</td><td>" + arr[i].Nome + "</td><td>" + arr[i].CPF + "</td></tr>";
  }
  out += "</table>";
  document.getElementById("id01").innerHTML = out;
} //function ExibeDados()

</script>

```

No navegador deve aparecer o formulário abaixo:

file:///Z:/ProjetoNode/lista-dados.html

## Clientes

ID Cliente:

Se o usuário clicar no botão sem digitar nada no campo, deverá aparecer todos os registros da tabela.

file:///Z:/ProjetoNode/lista-dados.html

## Clientes

ID Cliente:

1	luiz	12345678901
2	teste2	09876543210

Quando o usuário digitar um número de ID e clicar no botão, deverá aparecer o registro da tabela correspondente ao ID inserido.

file:///Z:/ProjetoNode/lista-dados.html

## Clientes

ID Cliente:

2	teste2	09876543210
---	--------	-------------

## Inserindo dados no BD

Para inserir dados no BD pelo APP vamos criar uma página HTML com o seguinte formulário:

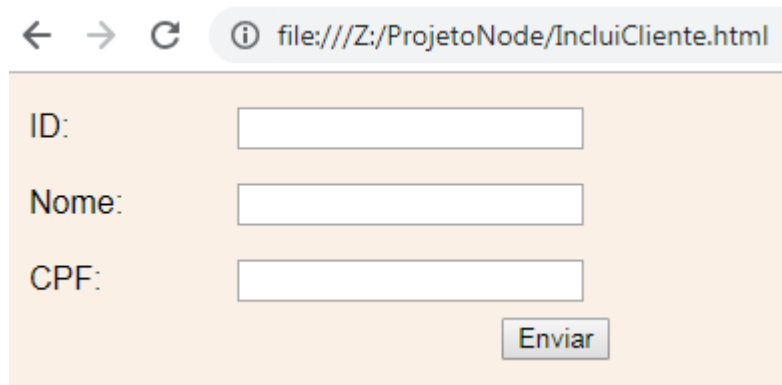
```
<!DOCTYPE html>
<html>
<head>
  <title>Inclusão Clientes - Node.js</title>
  <meta charset="UTF-8">
  <link rel="stylesheet" type="text/css" href="estilo.css">
</head>
<body>

  <form action="http://localhost:3000/clientes" method="post">
    <label>ID: </label>
    <input type="text" name="id" id="idCliente">
    <label>Nome: </label>
    <input type="text" name="nome" id="nomeCliente">
    <label>cpf: </label>
    <input type="text" name="cpf" id="cpfCliente">
    <input type="submit" name="submit" value="Enviar">
  </form>
</body>
</html>
```

A aplicação precisará da nova rota (acrescentar no final do arquivo appSQL.js):

```
rota.post('/clientes', (requisicao, resposta) =>{
  const id = parseInt(requisicao.body.id);
  const nome = requisicao.body.nome.substring(0,150);
  const cpf = requisicao.body.cpf.substring(0,11);
  execSQL(`INSERT INTO Clientes(ID, Nome, CPF) VALUES(${id},'${nome}','${cpf}')`, resposta);
})
```

Sempre que alterarmos a aplicação devemos reiniciá-la (node appSQL.js). Ao abrir o formulário de inclusão no navegador teremos o seguinte:



The screenshot shows a web browser window with the address bar displaying 'file:///Z:/ProjetoNode/IncluiCliente.html'. The browser interface includes back, forward, and refresh buttons. The main content area shows a form with a light orange background. The form contains three labels with corresponding input fields: 'ID:', 'Nome:', and 'CPF:'. Below these fields is a button labeled 'Enviar'.

## Excluindo dados no BD

Para excluir dados no BD pelo APP vamos criar outra página HTML com o seguinte formulário e função com AJAX:

```
<!DOCTYPE html>
<html>
<head>
    <title>Exclusão Clientes - Node.js</title>
    <meta charset="UTF-8">
    <link rel="stylesheet" type="text/css" href="estilo.css">
</head>
<body>

<h1>Exclui Cliente</h1>

<script>
function EnviaID() {
    var xmlhttp = new XMLHttpRequest();
    var url = "http://localhost:3000/clientes";

    xmlhttp.onreadystatechange=function() {
        if (this.readyState == 4 && this.status == 200) {
            document.getElementById('resp').innerHTML = this.responseText;
        }
    }

    if (document.getElementById("idCliente").length == 0 ) {
        xmlhttp.open("GET", url, true);
    } else {
        var idCliente = document.getElementById("idCliente").value;
        alert(url+"/"+idCliente);
        xmlhttp.open("DELETE", url+"/"+idCliente, true);
    }

    xmlhttp.send();
} //function EnviaID()

</script>

<form method="post">
    <label>ID: </label>
    <input type="text" name="id" id="idCliente">

    <input type="button" name="btnEnviar" value="Enviar" onclick="EnviaID();">
</form>

<div id="resp"></div>
</body>
</html>
```

A aplicação precisará da nova rota (acrescentar no final do arquivo appSQL.js):

```
rota.delete('/clientes/:id', (requisicao, resposta) =>{  
    execSQL('DELETE Clientes WHERE ID=' + parseInt(requisicao.params.id), resposta);  
})
```

Lembrar de reiniciar a aplicação (node appSQL.js).