

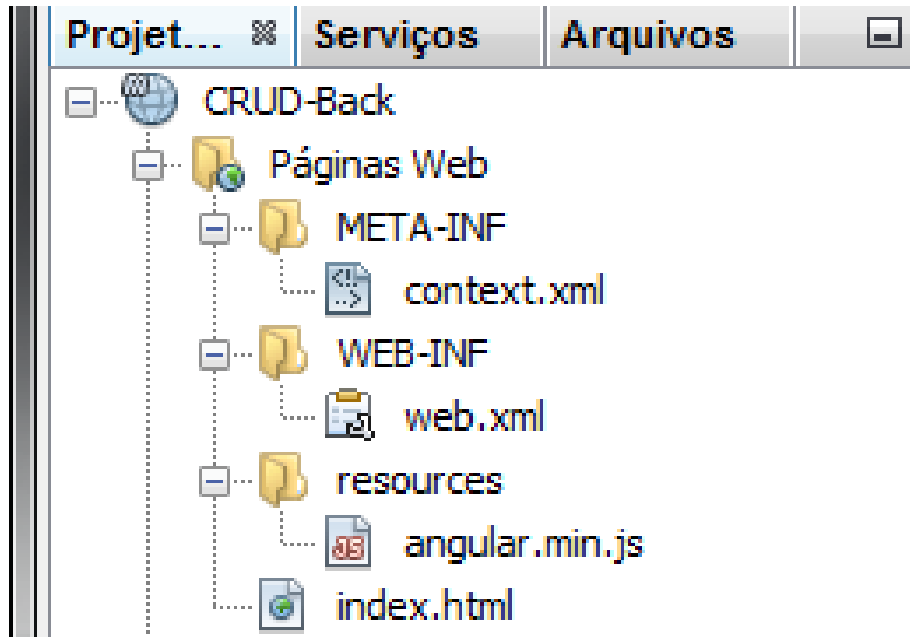
# Introdução ao AngularJS

Professor Anderson Henrique



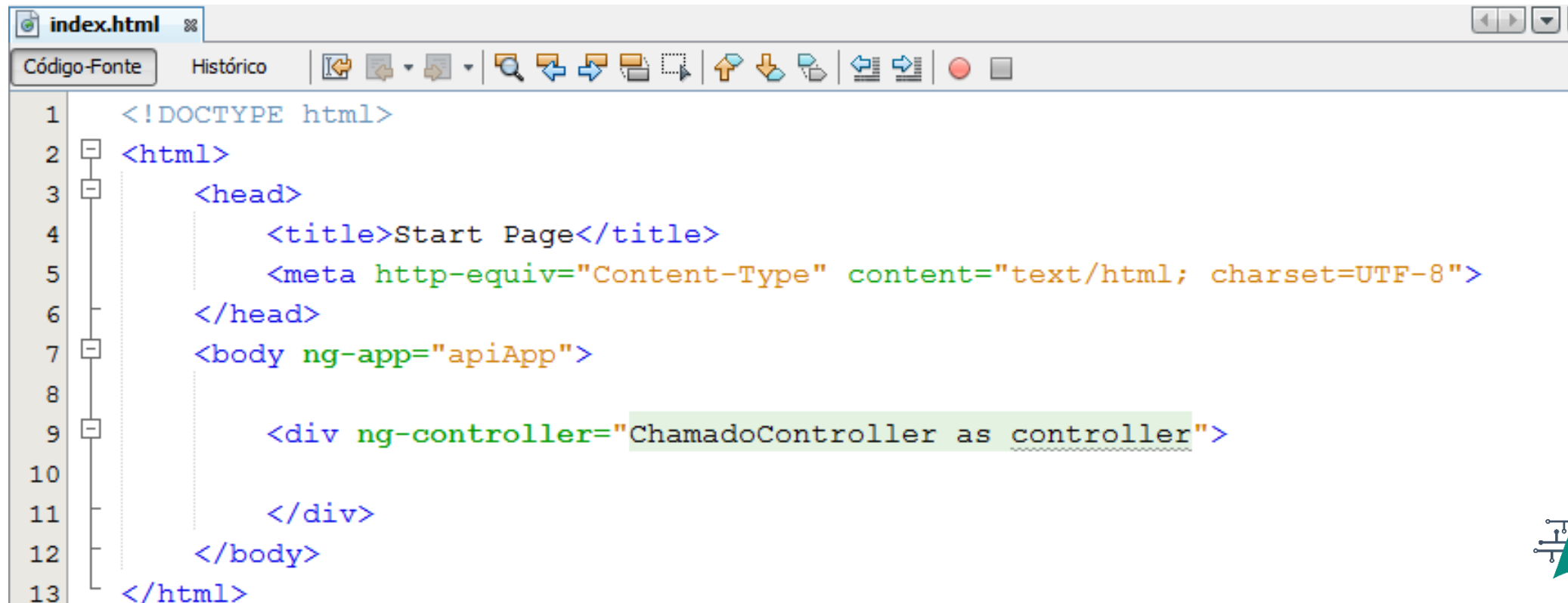
# Front-End (Angularjs)

Crie um diretório dentro da pasta Páginas Web chamado **resources**, salve o arquivo do angular.js dentro dele



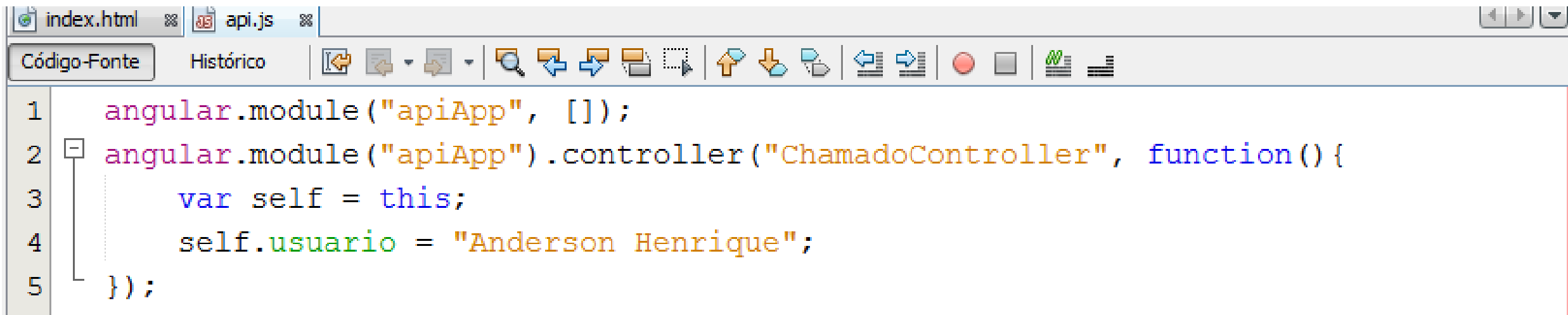
Abra o arquivo **index.html**, dentro da TAG **body** crie a diretiva angular **ng-app** com o valor **apiApp**

Crie uma TAG **div** dentro da TAG **body** com a diretiva **ng-controller** com o valor **ChamadoController** passando um alias **controller**



```
1  <!DOCTYPE html>
2  <html>
3  <head>
4      <title>Start Page</title>
5      <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
6  </head>
7  <body ng-app="apiApp">
8
9      <div ng-controller="ChamadoController as controller">
10
11      </div>
12  </body>
13 </html>
```

Na pasta resources, crie um arquivo **js** chamado **api**, nesse arquivo vamos criar o módulo da nossa api e nosso controller com a função anexada, dentro da função vamos criar uma variável chamada **self** que recebe (**this**) e um atributo **usuario** que recebe um nome

A screenshot of a code editor window. The top bar shows two tabs: 'index.html' and 'api.js'. Below the tabs is a menu bar with 'Código-Fonte' and 'Histórico'. A toolbar with various icons is visible. The main area shows the following JavaScript code:

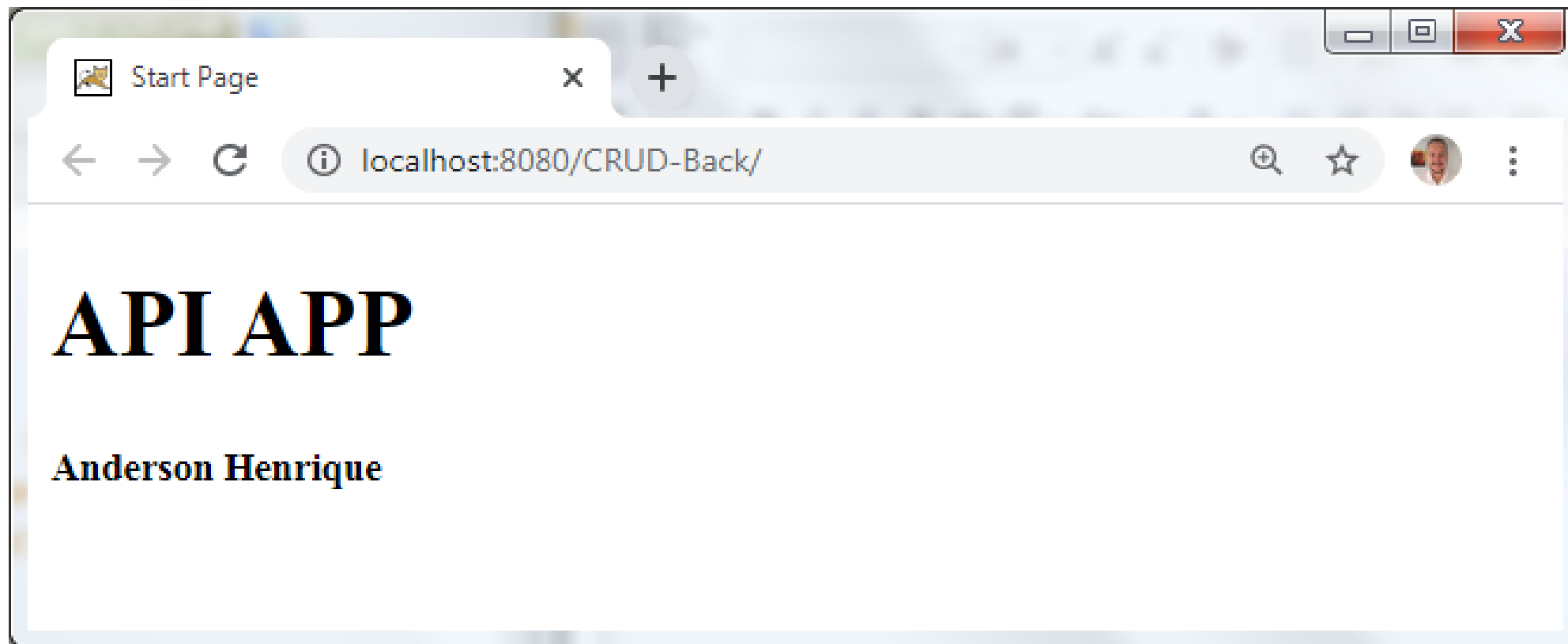
```
1 angular.module("apiApp", []);
2 angular.module("apiApp").controller("ChamadoController", function() {
3     var self = this;
4     self.usuario = "Anderson Henrique";
5 });
```

No arquivo **index**, vamos criar a TAG **h1** com o nosso da nossa aplicação e dentro da TAG **div** que tem o controller, vamos chamar o atributo **usuario** para ser impresso [antes, temos que chamar os documentos javascript dentro do html]



```
1 <!DOCTYPE html>
2 <html>
3   <head>
4     <title>Start Page</title>
5     <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
6     <script src="resources/angular.min.js" type="text/javascript"></script>
7     <script src="resources/api.js" type="text/javascript"></script>
8   </head>
9   <body ng-app="apiApp">
10    <h1>API APP</h1>
11    <div ng-controller="ChamadoController as controller">
12      <h5>{{controller.usuario}}</h5>
13    </div>
14  </body>
15 </html>
```

Assim que você depurar (debug) o projeto, vai abrir a página no servidor local com essas informações:



Vamos criar um formulário para inserir novo chamado no arquivo **index**

```
11 <div ng-controller="ChamadoController as controller">
12   <h5>{{controller.usuario}}</h5>
13   <div style="border: 1px solid black"
14     ng-show="controller.chamado">
15     <label for="chamadoId">ID:</label>
16     <input type="number" id="chamadoId"
17       ng-model="controller.chamado.id" disabled> <br/>
18     <label for="assunto">Assunto:</label>
19     <input type="text" id="assunto"
20       ng-model="controller.chamado.assunto"> <br/>
21     <label for="mensagem">Mensagem:</label>
22     <textarea rows="4" id="mensagem"
23       ng-model="controller.chamado.mensagem"></textarea> <br/>
24
25     <button type="button" ng-click="controller.salvar()">Salvar</button>
26     <button type="button" ng-click="controller.chamado = undefined"></button>
27   </div>
```

Vamos criar uma tabela para exibir os chamados, com os botões de opções:

```
29 <div style="border: 1px solid black"
30     ng-show="!controller.chamado">
31     <button type="button" ng-click="controller.novo()">Novo Chamado</button>
32     <br/>
33     <table border="1">
34         <tr>
35             <th>ID</th>
36             <th>Assunto</th>
37             <th>Mensagem</th>
38             <th>Opções</th>
39         </tr>
40         <tr ng-repeat="chamado in controller.chamados">
41             <td>{{chamado.id}}</td>
42             <td>{{chamado.assunto}}</td>
43             <td>{{chamado.mensagem}}</td>
44             <td>
45                 <button type="button" ng-click="controller.alterar(chamado)">Alterar</button>
46                 <button type="button" ng-click="controller.deletar(chamado)">Excluir</button>
47                 <button type="button" ng-click="controller.concluir(chamado)">Concluir</button>
48             </td>
49         </tr>
50     </table>
51 </div>
```



Agora vamos implementar o nosso controller no arquivo **api.js** passo-a-passo:

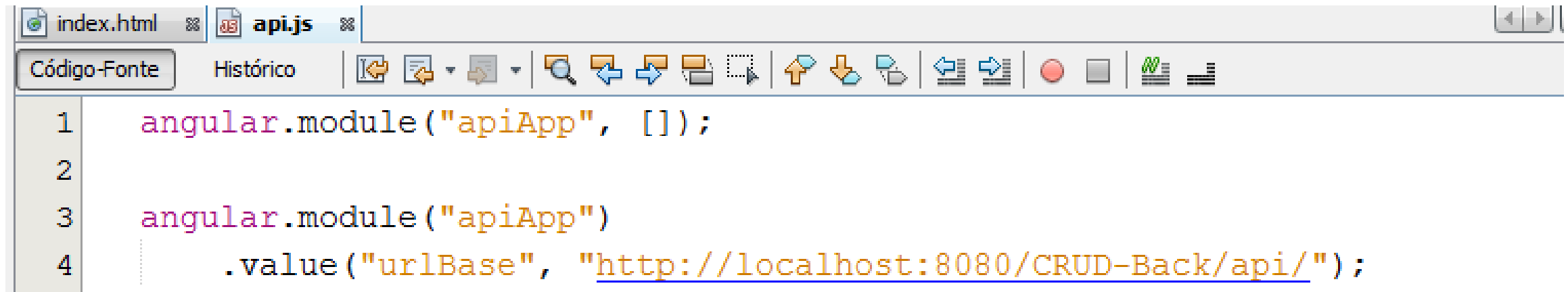
Primeiro criamos as variáveis: chamados [array], chamado recebe o valor **undefined**

```
2 angular.module("apiApp").controller("ChamadoController", function() {  
3     var self = this;  
4     self.usuario = "Anderson Henrique";  
5     self.chamados = [];  
6     self.chamado = undefined;
```

Vamos declarar todos os métodos que vamos implementar no nosso controller:

```
9  self.novo = function() {  
10  
11  };  
12  self.salvar = function() {  
13  
14  };  
15  self.alterar = function() {  
16  
17  };  
18  self.deletar = function() {  
19  
20  };  
21  self.concluir = function() {  
22  
23  };
```

No nosso controller, vamos criar um **value** que será a **urlBase** utilizada no service \$http do nosso controller



```
1  angular.module("apiApp", []);
2
3  angular.module("apiApp")
4    .value("urlBase", "http://localhost:8080/CRUD-Back/api/");
```

The image shows a code editor window with two tabs: 'index.html' and 'api.js'. The 'api.js' tab is active. The editor has a menu bar with 'Código-Fonte' and 'Histórico', and a toolbar with various icons. The code in the editor is as follows:

Também vamos utilizar o protocolo (service) \$http no nosso método salvar, por isso, precisamos injetar o service no nosso controller

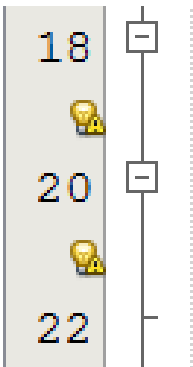
No nosso controller, vamos criar um **value** que será a **urlBase** utilizada no service **\$http** do nosso controller, injetamos **\$http** e **urlBase** no controller

```
6 angular.module("apiApp")
7   .controller("ChamadoController", function($http, urlBase){
```

No método **novo** vamos apenas mudar o valor de **undefined** passando um novo array vazio

```
9   self.novo = function(){
10       self.chamado = {};
11   };
```

No método **salvar** primeiramente criamos uma variável método que por padrão receberá o valor **POST**, vai verificar se o objeto chamado já possui **id** (existe), se **SIM** a variável método receberá o valor **PUT**



```
self.salvar = function() {  
    var metodo = POST;  
    if(self.chamado.id) {  
        metodo = PUT;  
    }  
}
```

O nosso service `$http` precisa receber (**method**, **url**, **data**), consiste no método que vem da variável, a **urlBase** concatenada com '**chamados/**' e os dados que serão trabalhados no método é o próprio **objeto chamado**, caso salve os dados, retorna no callback de sucesso atualização da tabela, caso ocorra um erro retornará no callback um método `ocorreuErro()`

```
24  $http({
25      method: metodo,
26      url: urlBase + 'chamados/',
27      data: self.chamado
28  }).then(function successCallback(response) {
29      self.atualizarTabela();
30  }, function errorCallback(response) {
31      self.ocorreuErro();
32  });
```

Precisamos criar o método **ocorreuErro()**, e será simples apenas retornará uma mensagem de alerta

```
47 self.ocorreuErro = function() {  
48     alert("Ocorreu um erro inesperado!");  
49 };
```

O método **alterar** vamos apenas mudar o valor de undefined passando um novo chamado

```
45 self.alterar = function(chamado) {  
46     this.chamado = chamado;  
47 };
```

O método **deletar** também faz uma chamada ao service \$http, primeiro criamos o objeto chamado

```
39 self.deletar = function(chamado) {  
40     self.chamado = chamado;  
}
```



No service \$http teremos alteração apenas no (**method**, **url**), onde o nosso método é o DELETE e na url temos: urlBase concatenada com '/chamados' concatenada com o 'id' do objeto e concatenada com uma '/'

```
42 $http({
43     method: 'DELETE',
44     url: urlBase + 'chamados/' + self.chamado.id + '/'
45 }).then(function successCallback(response) {
46     self.atualizarTabela();
47 }, function errorCallback(response) {
48     self.ocorreuErro();
49 });
```

Por último, precisamos criar o método **atualizarTabela()** foi invocado tanto no método **salvar** quanto no método **deletar**, antes vamos alterar na classe java **ChamadoController** no método **listChamados** na lista de chamados adicionando o método **setId** para os objetos (c1.setId(1), c2.setId(2), c3.SetId(3))

```
25 Chamado c1 = new Chamado();
26 c1.setId(1);
27 c1.setAssunto("Assunto1");
28 c1.setMensagem("Mensagem1");
29 c1.setStatus(Status.NOVO);
```

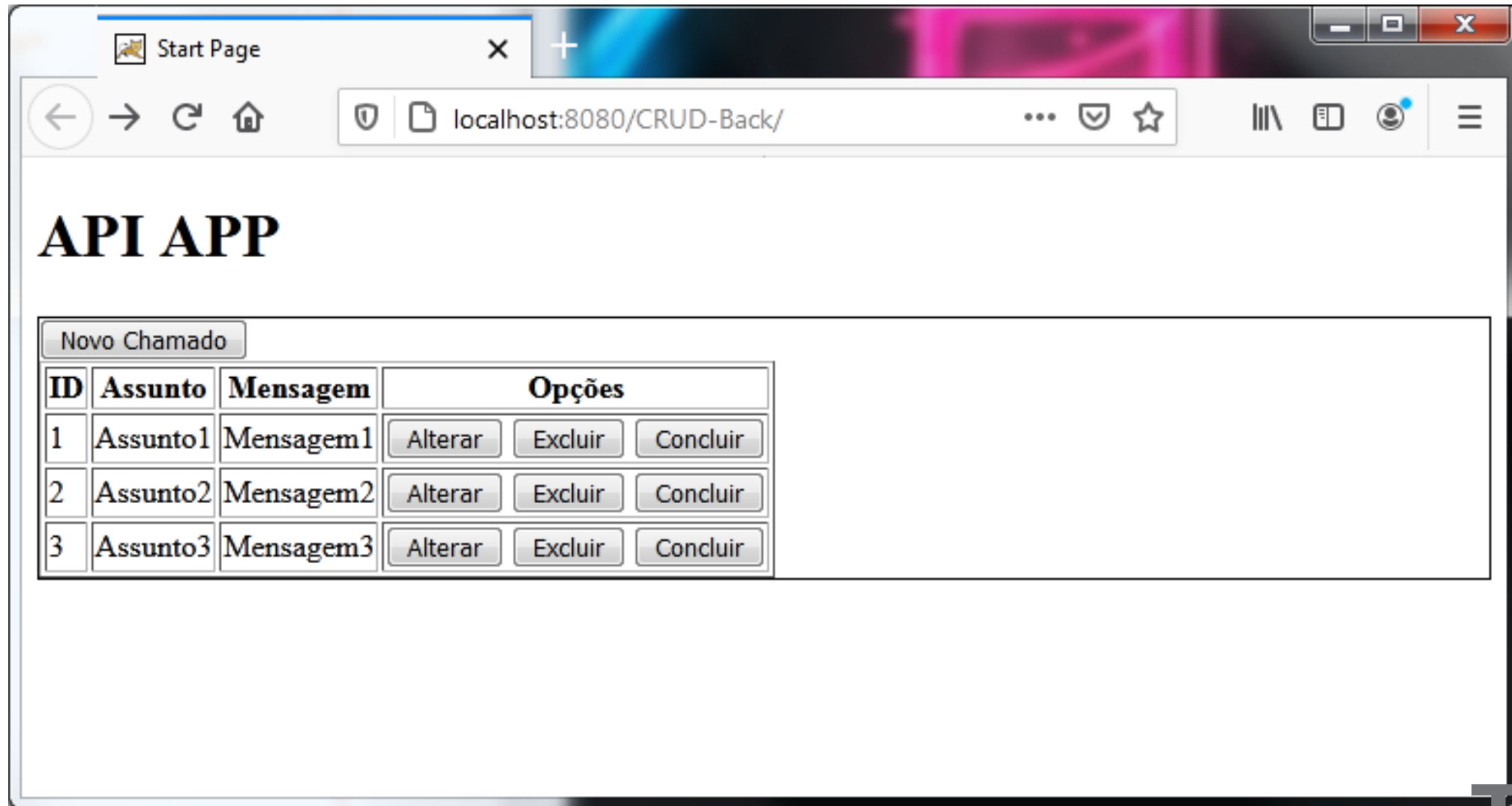
O método **atualizarTabela()** também receberá o service \$http (**method**, **url**), onde o método utilizado é o **GET**, a url consiste na **urlBase** concatenada com '**chamados/**' se ocorrer com sucesso o objeto chamados receberá os dados do back-end do java, e o objeto chamado volta a ser **undefined** (não existirá)

```
60 self.atualizarTabela = function() {
61     $http({
62         method: 'GET',
63         url: urlBase + 'chamados/'
64     }).then(function successCallback(response) {
65         self.chamados = response.data;
66         self.chamado = undefined;
67     }, function errorCallback(response) {
68         self.ocorreuErro();
69     });
70 };
```

É interessante iniciar o controller do angular com a tabela preenchida, não quero retornar a tabela vazia, pois o usuário deve utilizar as opções atualizar, deletar e concluir, por padrão utilizamos a função para atualizarTabela dentro de uma função de startup

```
72  self.activate = function() {  
73      self.atualizarTabela();  
74  };  
75  self.activate();  
76  });
```

Pronto! Vamos testar nossa aplicação executando o projeto no Netbeans



# Debug no Mozilla com a tecla F12

The screenshot shows a Mozilla Firefox browser window with a single tab titled "Start Page". The address bar displays "localhost:8080/CRUD-Back/". The main content area shows a web application titled "API APP". Below the title, there is a button labeled "Novo Chamado" and a table with three rows of data. Each row has columns for "ID", "Assunto", "Mensagem", and "Opções". The "Opções" column contains three buttons: "Alterar", "Excluir", and "Concluir".

ID	Assunto	Mensagem	Opções
1	Assunto1	Mensagem1	Alterar Excluir Concluir
2	Assunto2	Mensagem2	Alterar Excluir Concluir
3	Assunto3	Mensagem3	Alterar Excluir Concluir

Below the table, the browser's developer tools are open, showing the "Rede" (Network) tab. The list of requests shows a GET request to "localhost:8080 /CRUD-Back/api/chamados/" with a status of 200. The response is a JSON object. The "Response" tab is selected, showing the JSON payload:

```
{
  "0": {
    "assunto": "Assunto1",
    "id": 1,
    "mensagem": "Mensagem1",
    "status": "NOVO"
  },
  "1": {
    "assunto": "Assunto2",
    "id": 2,
    "mensagem": "Mensagem2",
    "status": "NOVO"
  }
}
```

The bottom status bar of the developer tools shows "5 requests", "23,86 KB / 366 B transferred", "Finish: 315 ms", "DOMContentLoaded: 92 ms", and "load: 158 ms".



Agora que nossa aplicação Front-End está se comunicando com o nosso Back-End vamos partir para a implementação do nosso Banco de Dados

Até a próxima!

**Dúvidas?**

**Professor Anderson Henrique**





## Para a próxima aula

01 – Acesso ao Banco de Dados (JDBC)

02 – Aplicação Completa com JEE 6 – Utilizando Maven – JAX-RS

Professor Anderson Henrique