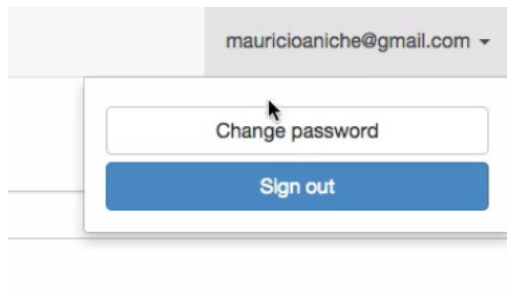


Melhorando a interface com o plugin Bootstrap

O próximo passo é cuidar um pouco da aparência da nossa página, isto é, embelezar ela. Para fazer isso utilizaremos o *Bootstrap*, pois é simples, mas deixa a aplicação com uma cara bonita.

O *Meteor* já faz isso por nós. Então, digitamos no nosso código do terminal o seguinte: `meteor add mrt:bootstrap-3`. Ele vai baixar o pacote e vai configurar isso no *Meteor*. Assim não teremos que ficar inserindo o `csc`. O *Mongo* vai fazer isso por nós. Com o pacote instalado basta usar as classes do `bootstrap`. A imagem de baixo é um *plugin* do `bootstrap` já pronto.



Na nossa tela após a finalização da instalação do *plugin* teremos:

```
tasklist — bash — 80x24
bash      mongo
=> App running at: http://localhost:3000/
=> Client modified -- refreshing
=> Errors prevented startup:

While building the application:
client/lista/lista.html:1: bad formatting in HTML template

=> Your application has errors. Waiting for file change.
=> Modified -- restarting.
=> Meteor server restarted
=> Client modified -- refreshing (x10)
=> Meteor server restarted
=> Meteor server restarted
=> Client modified -- refreshing^C
Alura:tasklist Alura$ meteor add mrt:bootstrap-3

Changes to your project's package version selections:

mrt:bootstrap-3 added, version 0.3.8

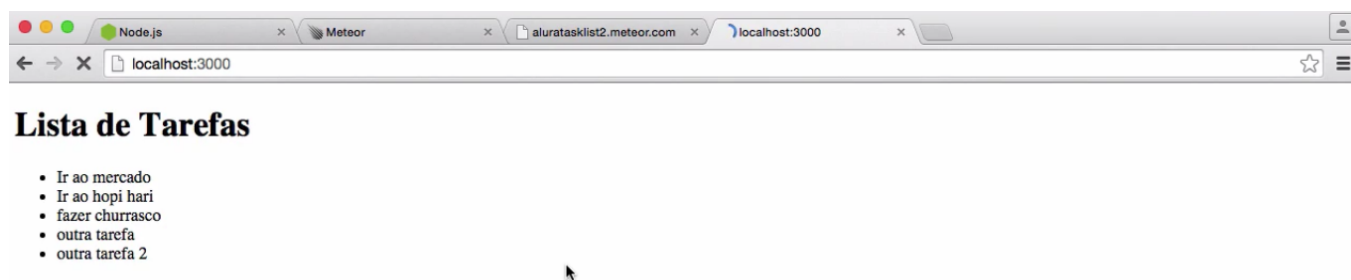
mrt:bootstrap-3: Provides bootstrap 3.
Alura:tasklist Alura$
```

No "index.html" apagaremos o `h1`. Digitaremos dentro do `body`, `header class="container"` para criar um `container`. Na linha de baixo escreveremos `nav class="navbar navbar-default"` para adicionar um menu de navegação e ficaremos com um template *default*. Colocaremos um `div class="navbar-header"` e um `href="#" class="navbar-brand"`, esse `href` por enquanto não vai apontar para lugar nenhum. Chamaremos a aplicação de `Taskie`. Deixaremos também um espaço para o login `div class="navbar-collapse collapse"`. Acrescentaremos um `ul class="nav navbar-nav navbar-right"` para ele ir para a direita. Colocaremos, ainda, dentro da lista `main class="container"`.

Teremos o seguinte:

```
4
5 <body>
6   <header class="container">
7
8     <nav class="navbar navbar-default">
9       <div class="navbar-header">
10         <a href="#" class="navbar-brand">
11           Taskie
12         </a>
13       </div>
14
15       <div class="navbar-collapse collapse">
16         <ul class="nav navbar-nav navbar-right">
17
18         </ul>
19       </div>
20     </nav>
21   </header>
22
23   <main class="container">
24
25     {{> lista}}
26
27   </main>
28
29 </body>
```

Vamos rodar a aplicação digitando "enter" e teremos o seguinte no nosso browser:



Vamos modificar a aparência da nossa lista. Através de uma `div class="row"`, na linha de baixo adicionamos `div class="col xs-12"`, para pegar tudo, e dentro dele colocamos um outro `div class="row"` e um `div class="col-xs-11"`. Na linha de baixo acrescentamos `tarefa`. Adicionaremos `div class="col-xs-1"` e o `button class="btn btn-danger>Excluir"`.

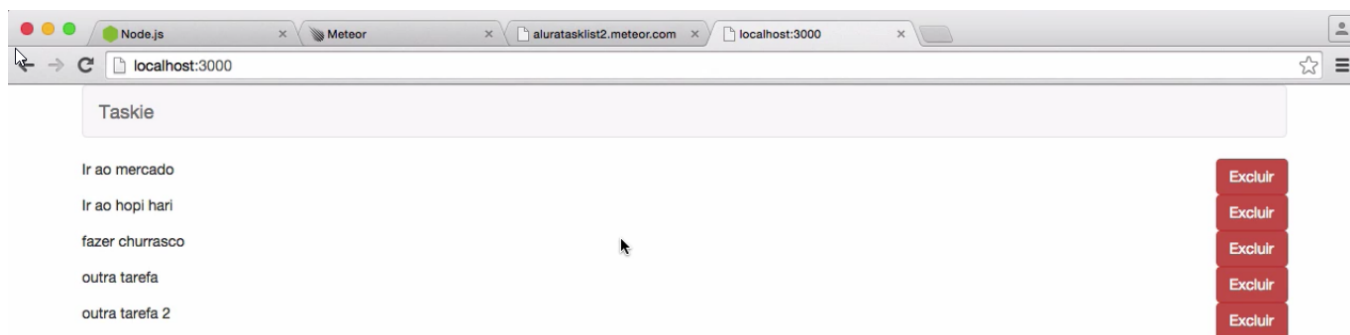
```
index.html x lista.html lista.js tarefas.js x
1 <template name="lista">
2
3
4 <div class="row">
5   <div class="col-xs-12">
6
7     <div class="row">
8
9       <div class="col-xs-11">
10        tarefa
11      </div>
12      <div class="col-xs-1">
13        <button class="btn btn-danger">Excluir</button>
14      </div>
15    </div>
16  </div>
17
18 </div>
19
20
21 </div>
22
23 </template>
```

Esse código todo é o que temos que repetir com cada componente. Para indicar isso acrescentamos `{{#each tarefas}}` e colocaremos no lugar de "tarefa", o `{{nome}}` para trazer nossa lista de tarefas.

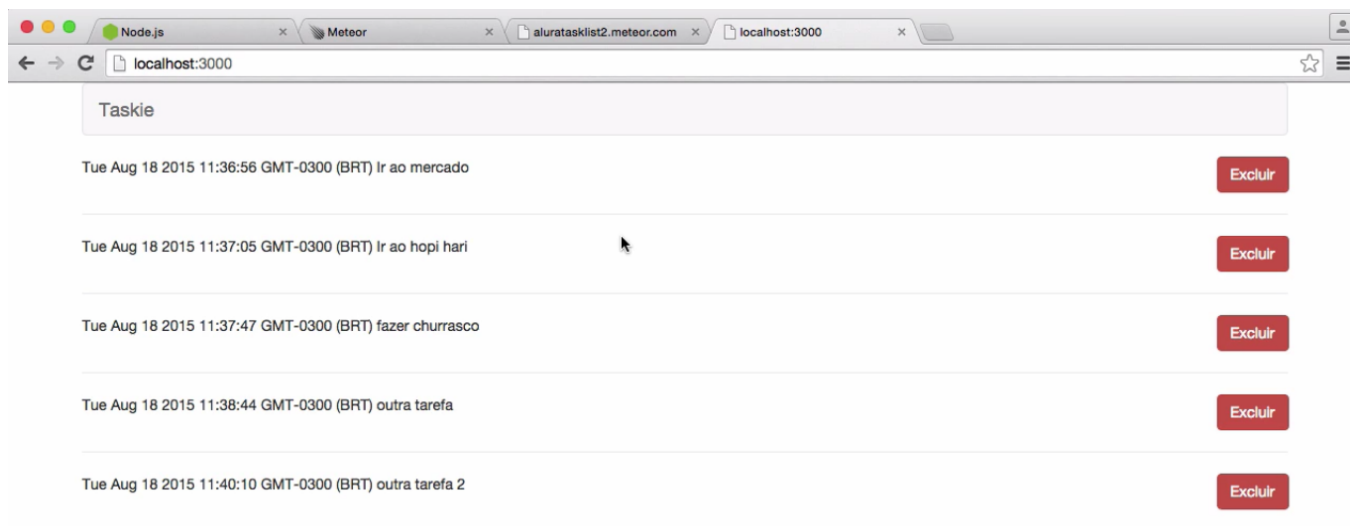
Ficaremos com:

```
index.html x lista.html x lista.js x tarefas.js x
1 <template name="lista">
2
3
4 <div class="row">
5   <div class="col-xs-12">
6
7     {{#each tarefas}}
8     <div class="row">
9
10      <div class="col-xs-11">
11        {{nome}}
12      </div>
13      <div class="col-xs-1">
14        <button class="btn btn-danger">Excluir</button>
15      </div>
16    </div>
17    {{/each}}
18  </div>
19
20 </div>
21
22 </div>
23
24 </template>
```

Nossa browser ficará da seguinte maneira:



Podemos acrescentar um `hr` para dar um espaçamento e podemos colocar ao lado de "nome", a "data" para ser exibida, `{{data}}`. Mas, se colocarmos apenas isso a data vai parecer muito grande em relação ao resto das informações.



Existe uma biblioteca de *JavaScript* que chama-se "Moment Js". Como é uma biblioteca muito comum no mundo do *javascript*, o *Meteor*, já tem uma maneira de fazer isso fácil. Vamos digitar no terminal, `meteor add momentjs:moment`, e ele vai instalar o pacote. Após baixar, acabamos de adquirir a biblioteca de `Moment.js`. Teremos:

```
tasklist — bash — 80x24

bash      mongo      +

mrt:bootstrap-3: Provides bootstrap 3.
Alura:tasklist Alura$
Alura:tasklist Alura$
Alura:tasklist Alura$ meteor
[[[[[ ~/Documents/aniche/tasklist ]]]]]

=> Started proxy.
=> Started MongoDB.
=> Started your app.

=> App running at: http://localhost:3000/
=> Client modified -- refreshing (x3)^C
Alura:tasklist Alura$ meteor add momentjs:moment

Changes to your project's package version selections:
momentjs:moment  added, version 2.10.6

momentjs:moment: Moment.js (official): parse, validate, manipulate, and display
dates - official Meteor packaging
Alura:tasklist Alura$
```

Vamos usar a biblioteca. Tudo o que é *javascript* fica na `lista.js`. então, vamos na aba `lista.js` e adicionamos uma função, `formaData : function` e acrescentamos na próxima linha um `return moment(this.data).format('DD/MM/YYYY HH:mm')`. Lembrando que é necessário separar as várias funções por vírgula. Ficaremos com:

```
1 Template.lista.helpers({
2   tarefas: function() {
3     return Tarefas.find({});
4   },
5
6   formataData : function() {
7     return moment(this.data).format('DD/MM/YYYY HH:mm');
8   }
9 });
```

Voltamos na `lista.html` e alteramos o `"data"`, por `"formataData"`. Podemos acrescentar um hífen entre o `"formataData"` e `"nome"` e teremos `{{formataData}} - {{nome}}`. Assim, a data ficará separada das atividades por um hífen.

```
1 <template name="lista">
2
3
4   <div class="row">
5     <div class="col-xs-12">
6
7       {{#each tarefas}}
8       <div class="row">
9
10        <div class="col-xs-11">
11          {{formataData}} - {{nome}}
12        </div>
13        <div class="col-xs-1">
14          <button class="btn btn-danger">Excluir</button>
15        </div>
16      </div>
17      <hr>
18    </each>
19  </div>
20
21
22
23 </div>
24
25 </template>
```

O *Meteor* sabe que o `this` do `"lista.js"` é cada uma das tarefas. O `this.data` é a data da tarefa e o *Meteor* a formata conforme o que inserimos no código `DD/MM/YYYY HH:mm`.

Vamos subir a aplicação de novo, digitando `meteor` no terminal. Aqui, é um costume, parar a aplicação quando um pacote novo for instalado, para não sujar o código e não bagunçar muito.

Teremos o seguinte: