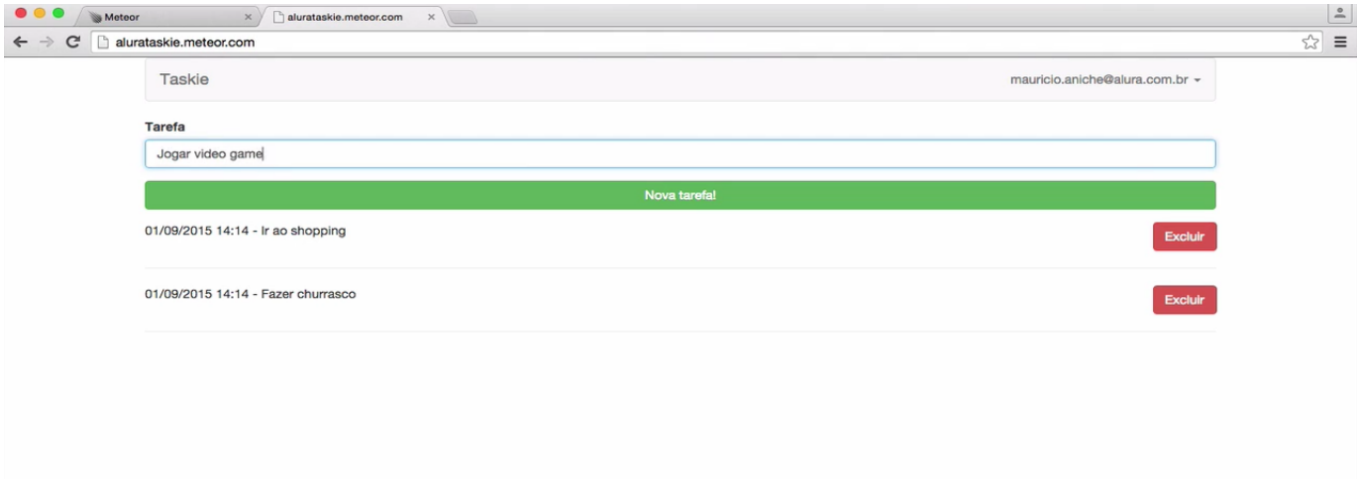
02

Sistema de Tarefas e o Meteor

O curso de *Meteor* JS serve para criar uma aplicação da *web*. Ele ajuda muito a criar uma *single page aplication*.

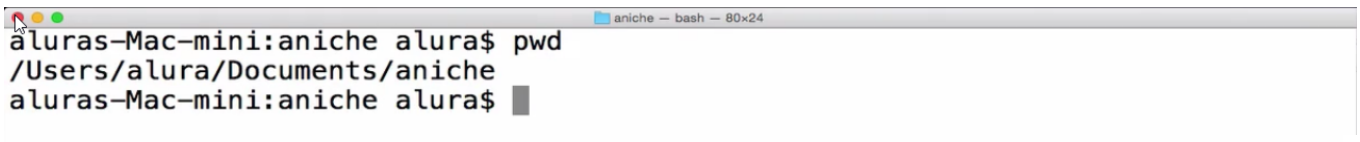
Na imagem de baixo temos uma tela de uma aplicação que é a que criaremos ao longo desse curso. Batizei ela de "Taskie", é uma simples lista de tarefas. Repare que estamos logados com o próprio usuário e podemos adicionar novas tarefas no campo "Tarefas". E excluir essas tarefas clicando em "Excluir".



Quando uma aplicação inteira acontece em uma única página, chamamos de *sigle page application*. E essa aplicação é bem mais simples do que outras, como, o *Java*.

Para iniciar vamos no site [www.meteor.com (http://www.meteor.com)](http://www.meteor.com) e precisamos instalar essa aplicação. Para isso basta clicar no botão vermelho da página inicial "Install Meteor".

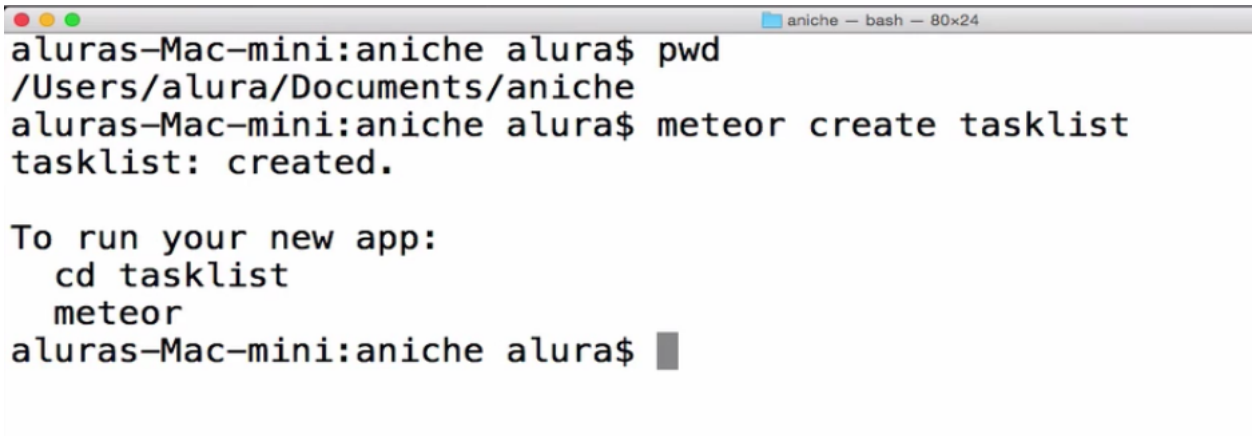
Seremos redirecionados para uma página que dá dicas de instalação para cada plataforma, Linux, Mac ou Windows . No Linux ou no Mac basta executar o comando e no Windows basta seguir os caminhos propostos. Após instalado o *Meteor* vamos abrir o terminal:



Estamos em um terminal qualquer. Para criar uma aplicação basta digitar `meteor create` e o nome da aplicação, `tasklist`. Ele vai criar a estrutura de diretórios necessária para começar:

Vamos ver isso no *Sublime*, que é um editor de texto padrão, normalmente utilizado. Os três arquivos que foram criados estão na pasta ".meteor", nela encontraremos os documentos "tasklist.css", o "tasklist.html" e o "tasklist.js".

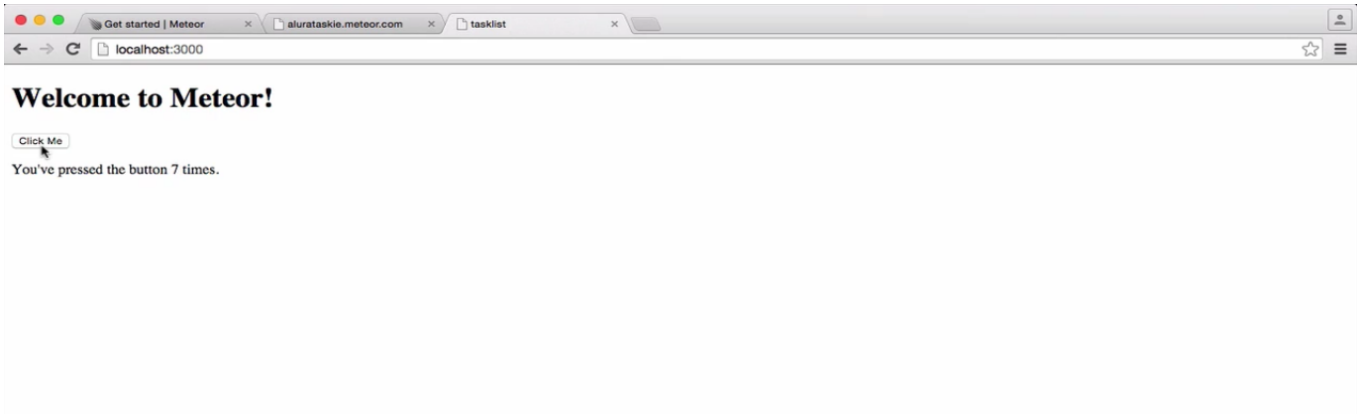
Vamos voltar para a "tasklist". Se viermos no diretório e digitarmos *meteor* e darmos um "Enter" ele já vai subir o servidor. Agora, estamos com o servidor embarcado, sem nenhum segredo!



Repare que ao final de tudo ele fornece uma porta o "http://localhost:3000 (http://localhost:3000)". Agora, vamos buscar esse servidor na Internet, digitaremos para isso "localhost:3000".

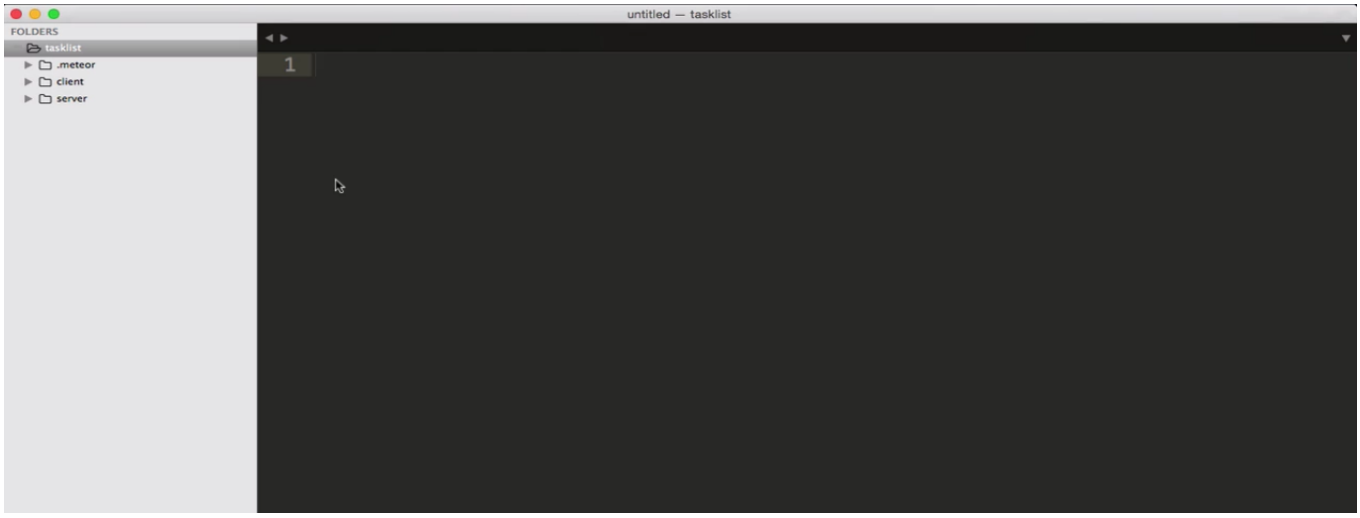
Essa já é uma diferença do *Meteor*, o servidor já está embarcado, diferente de outras aplicações que talvez você conheça.

Vai abrir uma janela, com uma aplicação simples. Teremos um botão de "Click Me" e toda vez que clicarmos ele vai fazer uma contagem. Ele vai atualizar sem dar *refresh*.



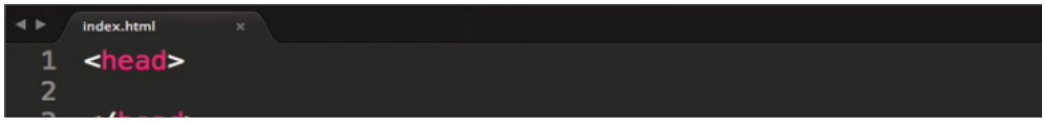
Vamos voltar e jogar fora o esqueleto que está no *Sublime Text*, apagaremos aqueles três arquivos que foram criados anteriormente.

Temos que começar a a desenvolver essa *Task*. Primeiro, criamos duas pastas. Uma que será "client" e a outra que será "server". Fazemos isso para separar o código do cliente e do servidor. O cliente é o que vai para o *browser* e o do servidor é o que contêm alguma regra nossa.

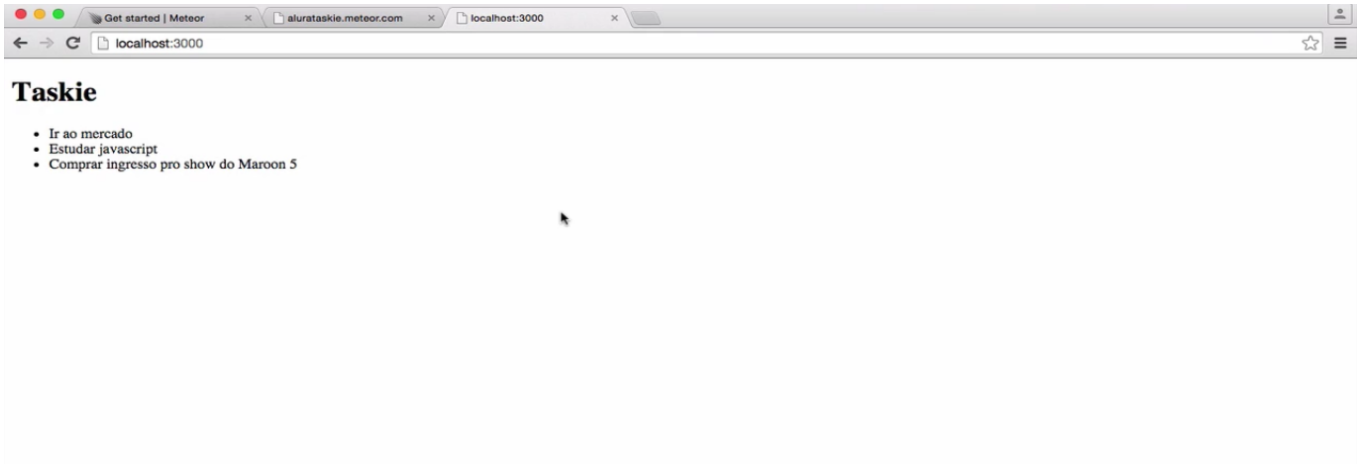


É obrigatório que os nomes sejam "client" e "server", pois é uma convenção do *meteor*.

No "client" a primeira coisa é criar um arquivo `html`. Chamaremos de `index.html`. Abriremos e fecharemos nesse arquivo um `head` e um `body`. Repare que não colocaremos o `html` pois isso é um problema do *Meteor*. Dentro do `body` colocaremos um `h1` que chamaremos de `Taskie` e colocaremos um `ul` qualquer, por exemplo, em cada linha uma tarefa da nossa lista "Ir ao mercado", "Estudar javascript" e "Comprar ingresso pro show do Maroon5". Chamamos esse arquivo de `index.html` porque a hora que a aplicação do *Meteor* subir, ele vai entrar na pasta "client" e procurar o arquivo principal, que se chama `index.html`.



Se dermos um *refrash* na nossa página da internet veremos que os itens da nossa lista de tarefas estarão lá.



Nosso próximo passo é voltar no *Sublime* e puxar o `ul`. Ele está fixo, meio *hard cod* e queremos trazer ele do banco.

Quando pensamos em *single page aplication*, pensamos em componentes. Vamos pensar no nosso resultado final. Tem um componente que adiciona novas tarefas, tem um que é a lista, tem um *header* e um *login*. Cada um deles é um componente separado e quando criamos uma *single page aplication* desmembramos os componentes para facilitar manutenção, reuso...

Vamos criar esses componente!

Voltamos na *sublime* e na pasta "client" clicamos com o botão direito e "New File". Criaremos o primeiro componente, que nomearemos de "lista". Agora temos um diretório "lista" e dentro dele criaremos outro arquivo chamado "lista.html". O código dele será o nome do template, `template name="lista"` e fecharemos isso.

O `template name` é algo próprio do *Meteor*. Ele vai pedir para salvar isso, nós salvaremos como `template.name`. Tudo o que tiver aqui dentro é o componente.

Vamos mover o `ul` do "index.html" para a `lista.html` para dentro do componente "lista". Teremos:

Nó código principal temos que colocar o componente, faremos isso abaixo de `h1` e digitaremos entre chaves duplas, um `>`, e o nome do componente, `lista`. Ficaremos com `{{> lista}}`.

Durante tudo o que fizemos o servidor não parou, pois, no *Meteor* não precisamos parar.

Vamos criar subpastas no "client" para cada um dos demais componentes e poderemos nomear da maneira que quisermos. Dentro do lista, temos um `html` e dentro deste temos um código, o do componente que precisamos colocar dentro do `template name`. Os aqruivos que colocamos no no `template name` continuam em *hard coded*. Para que eles não permaneçam dessa maneira criaremos outro arquivo, o "lista.js" que armazenaremos na pasta "lista".

Na "lista.js" escreveremos `template` e o nome dele, no caso, `lista` e `helpers`. Teremos `Template.lista.helpers` e abriremos as chaves. Abrimos chaves pois essa função recebe um mapa, no `helper` vamos adicionar o código que vai buscar no banco de dados e imprimir na `index.html` os `li` para cada elemento.

Podemos denominar o mapa de qualquer coisa, chamaremos aqui de `tarefas`, ele virará algo no `html` mais adiante. "Tarefas" é um `function`, teremos `tarefas : function` e acrescentaremos na próxima linha um `return` e passaremos uma lista através da *Array*, escreveremos, `{nome:"Ir ao mercado"}` e acrescentaremos nas próximas linhas `{nome: "Comprar Sorvete"}` e `{nome: "estudar"}`. A palavra "tarefas" devolve uma lista de tarefas.

Teremos:

```
Template.lista.helpers({

    tarefas : function() {
        return [
            {nome: "Ir ao mercado"},
            {nome: "Comprar sorvet"},
            {nome: "estudar"}
        ];
    }
});
```

Não foi no banco ainda, mas fizemos um pouco de *javascript* no *template*. As chaves fazem parte do mapa do *javascript* e mais adiante isso virará um banco.

Todos os `helpers` que forem declarados na "lista.js" vão virar alguma coisa no "index.html".

No "lista.html" vamos usar entre as ´ul´ que restaram um `each` e acrescentaremos a palavra `tarefas`. Teremos `{{#each tarefas}}`. A palavra `tarefas` evoca a função do "lista.js". Lembre-se de fechar o `each`. No meio dele digitaremos entre `li` e a palavra "nome" que vai resgatar os itens da nossa lista de tarefas, lembre-se do, `{nome: "Comprar Sorvete"}`.

Teremos:

```
<template name="lista">

    <ul>

        {{#each tarefas}}
        <li>{{nome}}</li>
        {{/each}}
    </ul>


</template>
```