

# Recife DEVDAY 2017

*Hands On - Apps híbridos com Ionic Framework*



ionic  
ionic

André Porto

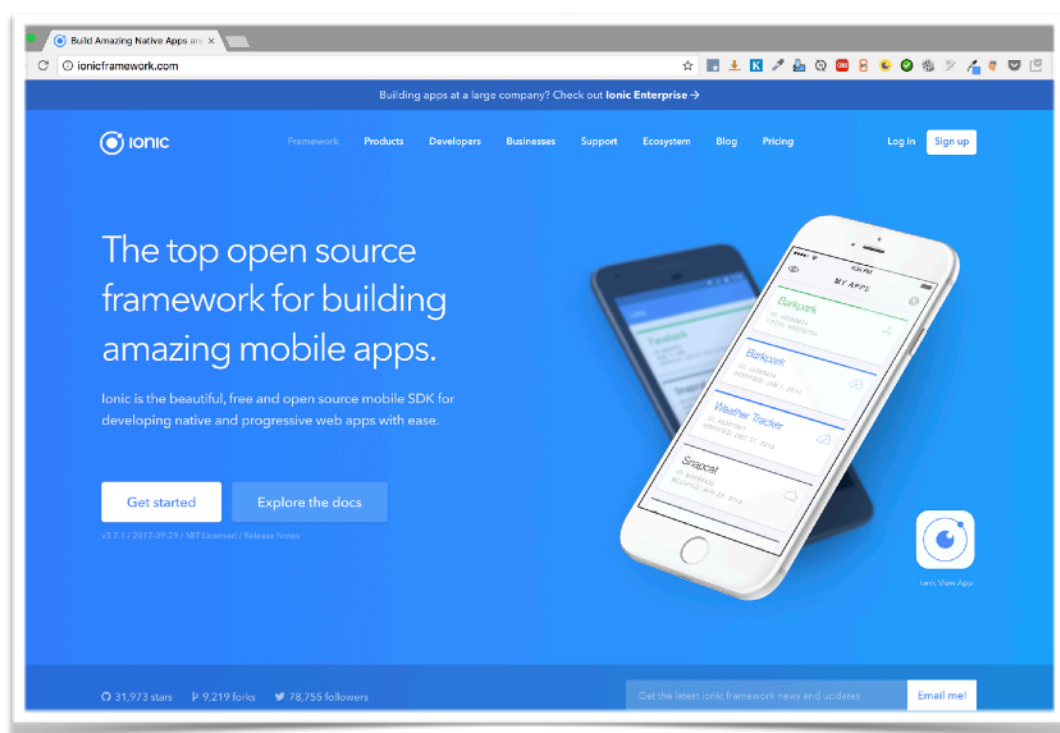
Novembro 2017

# Ionic Framework

*O que é e como se estrutura?*

## Conceituação

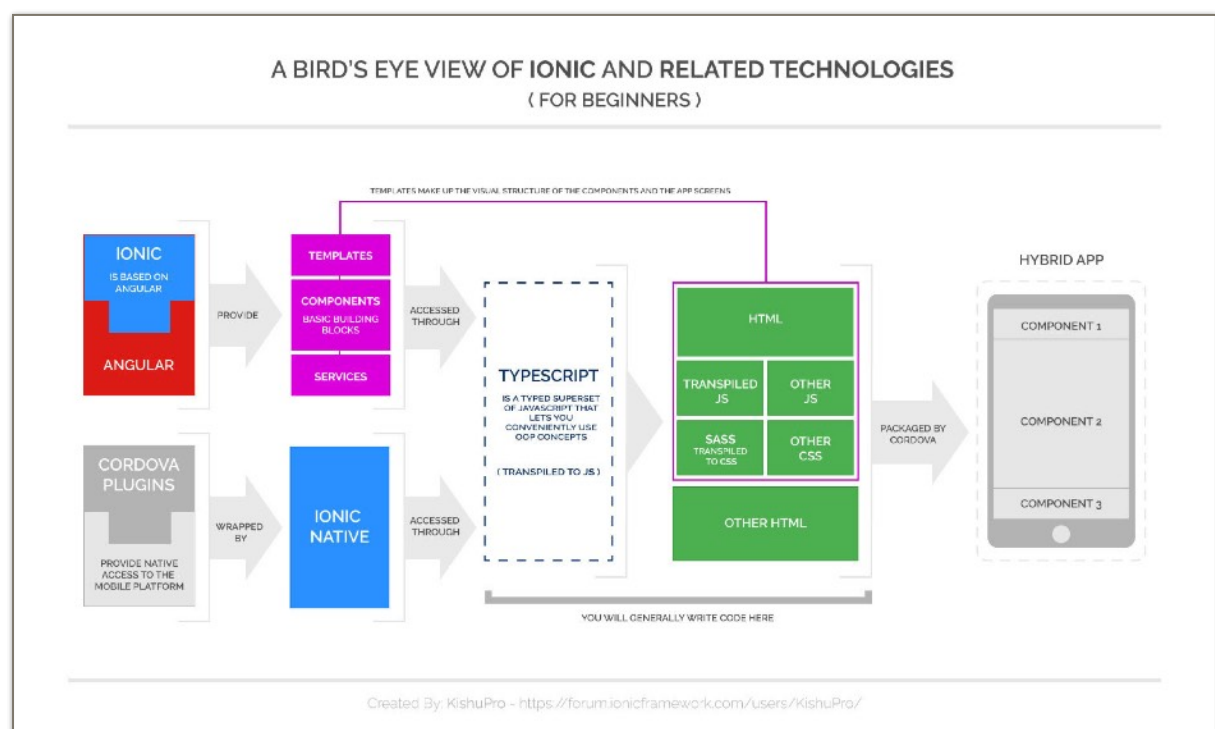
Ionic é um “framework” (composição de várias bibliotecas e utilitários) para desenvolver Apps para dispositivos mobile de forma híbrida, ou seja, misturando itens nativos e não nativos em sua construção.



Como podemos encontrar na página inicial do site oficial, Ionic pode ser entendido como um Software Development Kit open source, grátis e belo para desenvolver com facilidade "Apps nativos" e Progressive Web App.

Na realidade, os apps criados com Ionic e que rodam de "forma nativa" nos dispositivos móveis, na realidade são “Web Apps” encapsulados em um Chrome Browser Engine (V8) de forma transparente e que fornecem uma experiência do usuário muito semelhante a dos apps desenvolvidos com linguagem nativa.

Um ponto a se considerar é performance. Para apps que necessitam do uso intensivo dos recursos de hardware dos dispositivos, talvez não seja a melhor opção para o desenvolvimento, mas para a maioria dos apps comerciais que usam eventualmente os recursos nativos dos dispositivos e que seu foco seja o tratamento de dados ( CRUD like ) o Ionic é uma ótima opção para dar agilidade ao time de desenvolvimento, pois com a mesma tecnologia Web de construção de sites comuns se pode desenvolver uma aplicação para dispositivos móveis e disponibiliza-la com pouquíssima ou nenhuma adaptação para diversas plataformas ( Android, iOS e Windows Phone).



# Ambiente Ionic

*Que ferramental se deve ter para desenvolver usando Ionic?*

## Ionic Pro

O “ecossistema” Ionic oferece a possibilidade de se usar um IDE específico, o Ionic Creator, mas para utilizar esta ferramenta plenamente o desenvolvedor terá que assinar um serviço pago mensal. Este serviço tem suas vantagens, pois tem recursos adicionais como o Package que permite a compilação em nuvem do projeto, o que pode ser uma saída para os que querem desenvolver para iOS e não desejam adquirir uma máquina Apple somente para isso.

Incluso no pacote também se tem a possibilidade de usar serviços como deployment online, que permite enviar atualizações sem passar pelas lojas de Apps tradicionais novamente. Outro produto interessante é o ViewApp, usado para fazer mockups online, onde um cliente cadastrado ou o testador beta poderia ter acesso de imediato a uma versão da aplicação antes mesmo de se enviar para uma App Store.

## Ionic CLI

Para este treinamento, não usaremos os serviços pagos e/ou assinados do Ionic Pro. Usaremos a ferramenta de linha de comando para nos auxiliar na geração da estrutura dos componentes necessários a nossa aplicação de exemplo. Essa ferramenta é a Ionic CLI.

O Ionic CLI oferece a funcionalidade de se criar toda a estrutura e configuração de um projeto Ionic, bem como fornece a possibilidade de criação dos diversos tipos de componentes necessários no desenvolvimento de um App Híbrido, gerando a estrutura de cada um deles.

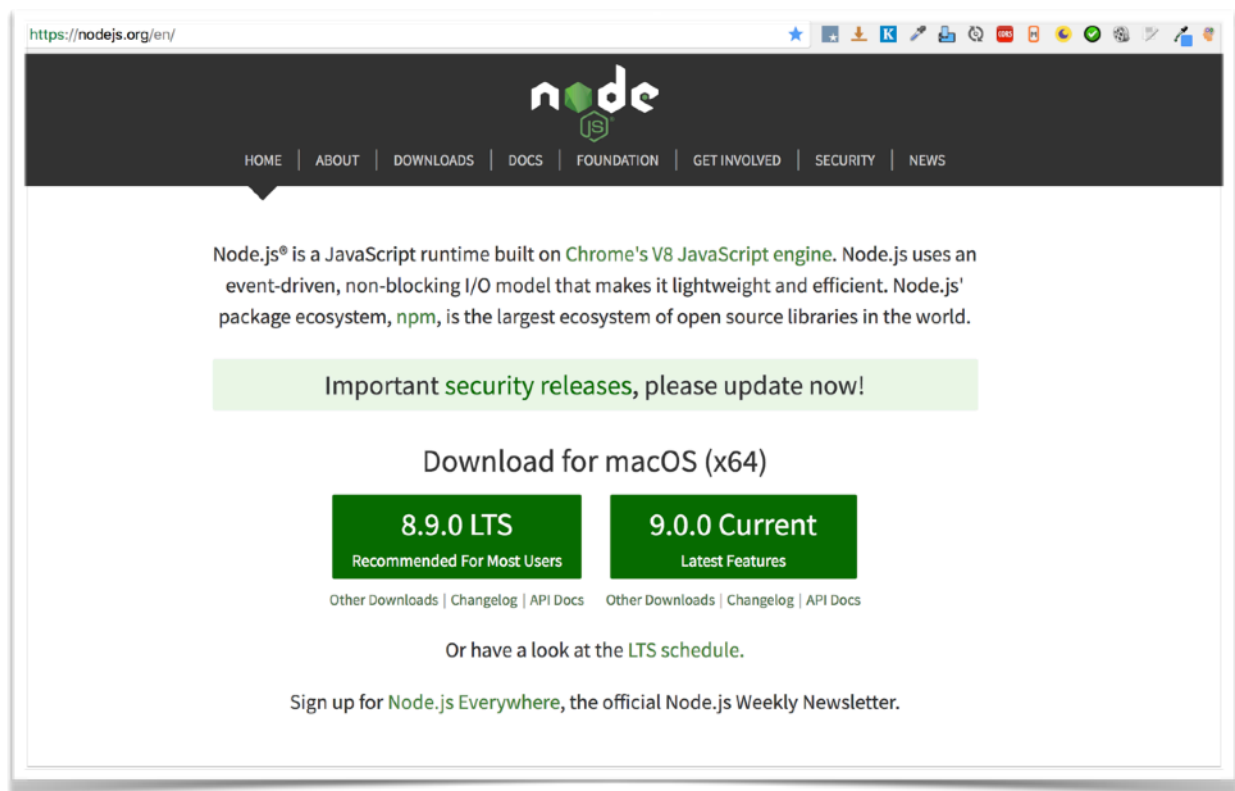
## Ferramentas

Para montarmos o ambiente de desenvolvimento, precisaremos das seguintes ferramentas:

- NodeJS + npm;
- VisualStudio Code;
- Cordova;
- Ionic CLI.

## NodeJS + npm

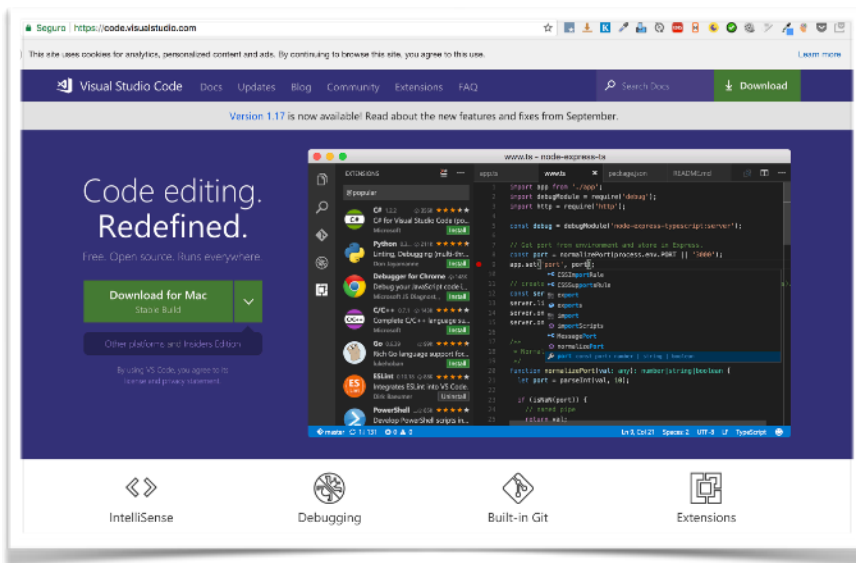
Para instalarmos o NodeJS e seu gerenciador de pacotes npm, podemos ir no site oficial <http://nodejs.org> e seguir as orientações específicas para nosso sistema operacional. Ao instalar o NodeJS seu gerenciador de pacotes também já será instalado automaticamente.



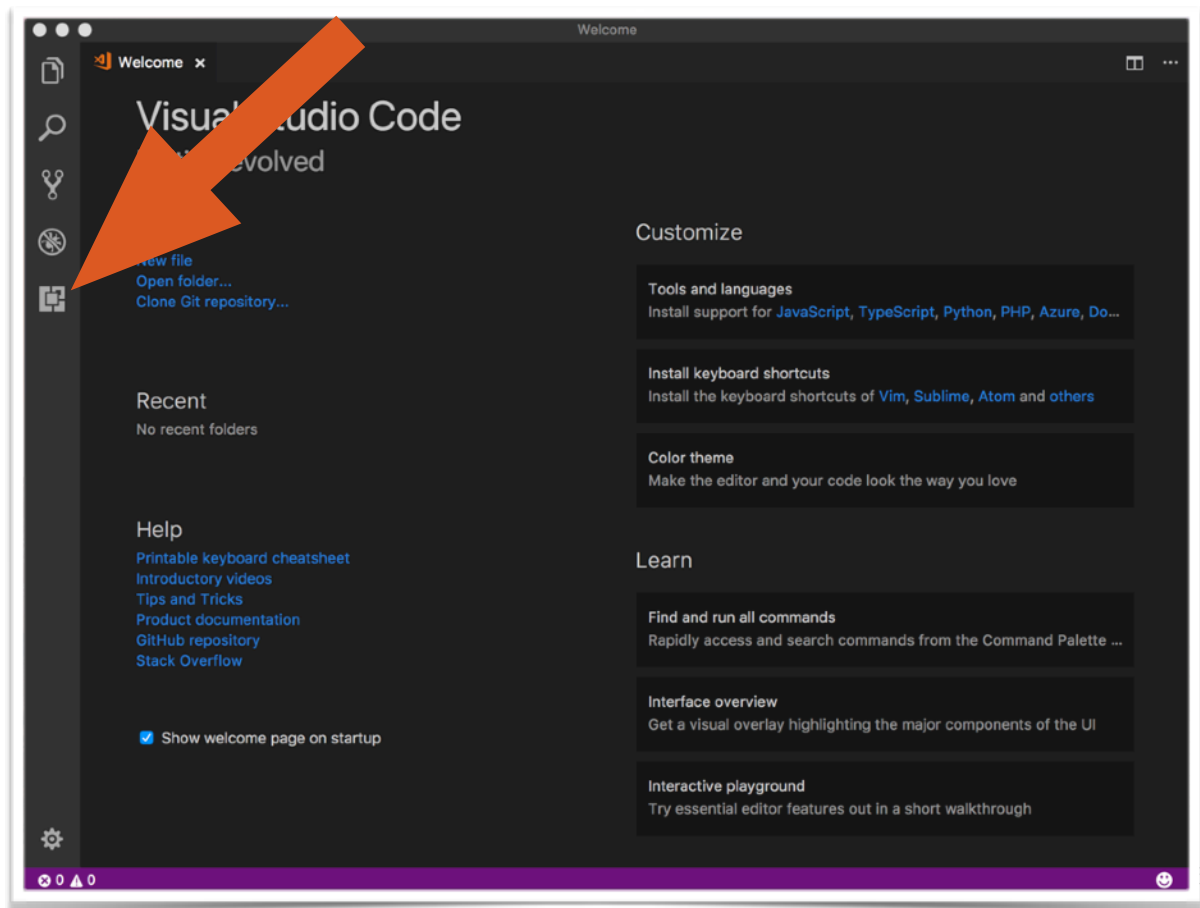
# VisualStudio Code

Em nosso desenvolvimento precisaremos de um editor de texto para escrevermos nosso código. Escolhemos o VisualStudio Code por ser uma ferramenta gratuita e com vários recursos que facilitarão nosso treinamento, mas é possível utilizar o editor de texto que mais for familiar a você.

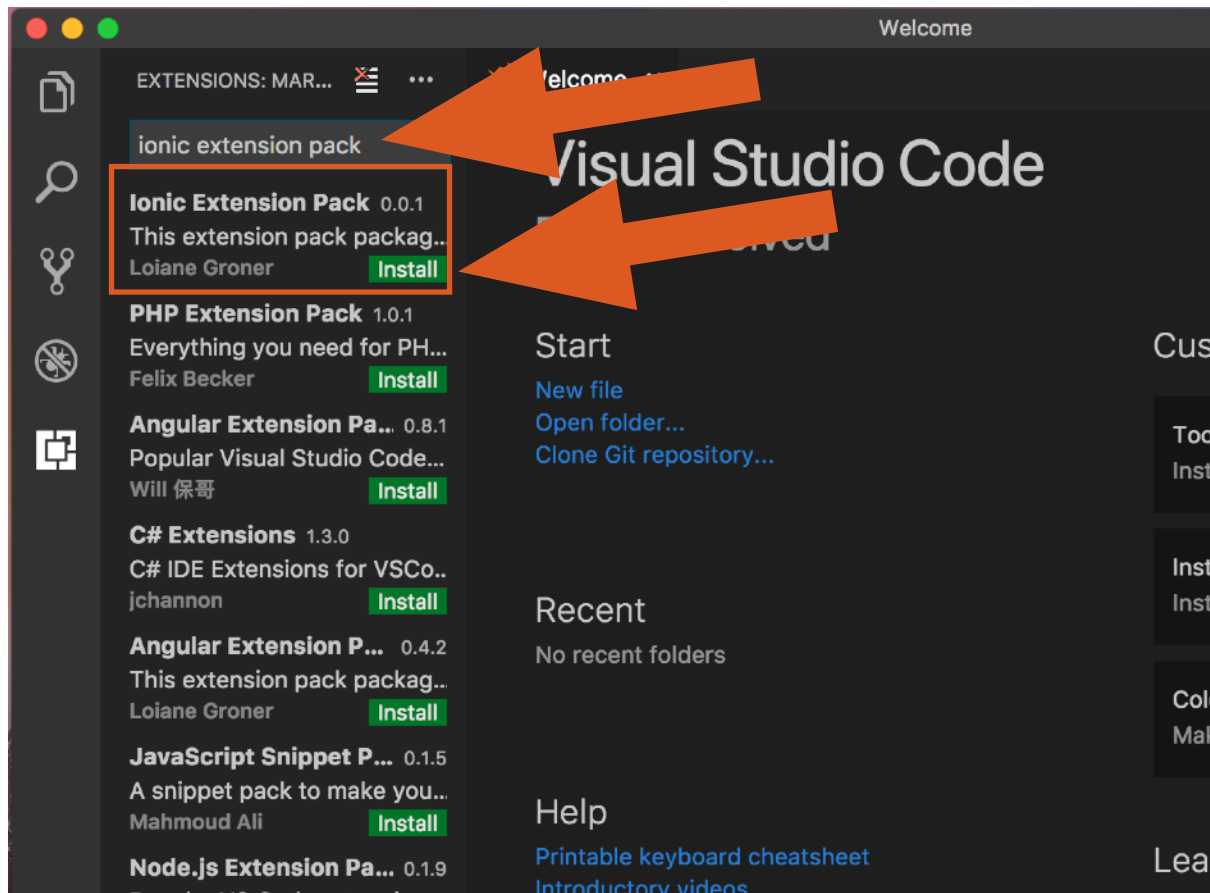
O VSCode pode ser encontrado sem nenhum custo no endereço <http://code.visualstudio.com>. Basta seguir a orientação de instalação para a sua plataforma (Linux, Windows ou Mac OS X).



Após a instalação do VSCode, iremos executá-lo e iremos adicionar nele alguns plugins que facilitarão ainda mais nosso processo de desenvolvimento. Para isso iremos acionar o botão de plugins



Iremos informar no campo de busca o termo “ionic extension pack” e depois clicar em “Install” no Pack de plugins indicado na figura abaixo.



Será necessário reiniciar o VSCode após a instalação do pacote de plugins.

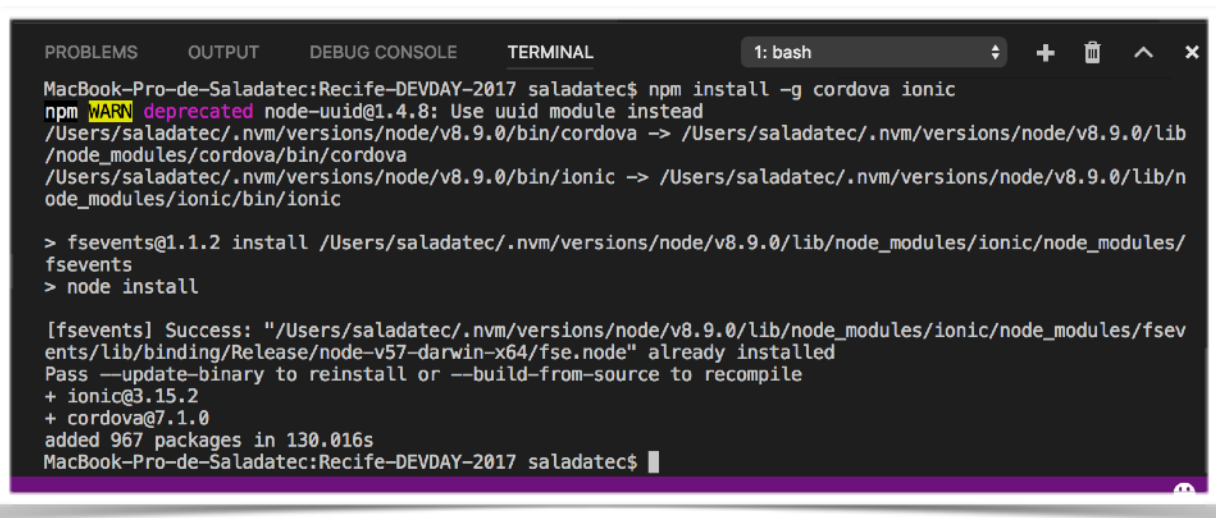
## Instalando Ionic CLI e Cordova

Para instalarmos o Ionic CLI e o Cordova iremos utilizar o npm. Para tanto, iremos abrir um terminal de linha de comando, o que pode ser feito diretamente no VSCode digitando-se as teclas <CTRL> e <`> ou acionando o menu “View > Integrated Terminal”.

Iremos então usar o seguinte comando:

```
npm install -g cordova ionic
```

Isto irá instalar globalmente todas as dependências do Ionic CLI e do Cordova e nos possibilitará a criação de nosso projeto e seus componentes.



```
MacBook-Pro-de-Saladatec:Recife-DEVDAY-2017 saladatec$ npm install -g cordova ionic
npm WARN deprecated node-uuid@1.4.8: Use uuid module instead
/Users/saladatec/.nvm/versions/node/v8.9.0/bin/cordova -> /Users/saladatec/.nvm/versions/node/v8.9.0/lib/
node_modules/cordova/bin/cordova
/Users/saladatec/.nvm/versions/node/v8.9.0/bin/ionic -> /Users/saladatec/.nvm/versions/node/v8.9.0/lib/n
ode_modules/ionic/bin/ionic

> fsevents@1.1.2 install /Users/saladatec/.nvm/versions/node/v8.9.0/lib/node_modules/ionic/node_modules/
fsevents
> node install

[fsevents] Success: "/Users/saladatec/.nvm/versions/node/v8.9.0/lib/node_modules/ionic/node_modules/fsev
ents/lib/binding/Release/node-v57-darwin-x64/fse.node" already installed
Pass --update-binary to reinstall or --build-from-source to recompile
+ ionic@3.15.2
+ cordova@7.1.0
added 967 packages in 130.016s
MacBook-Pro-de-Saladatec:Recife-DEVDAY-2017 saladatec$
```



# Projeto Lista de Tarefas

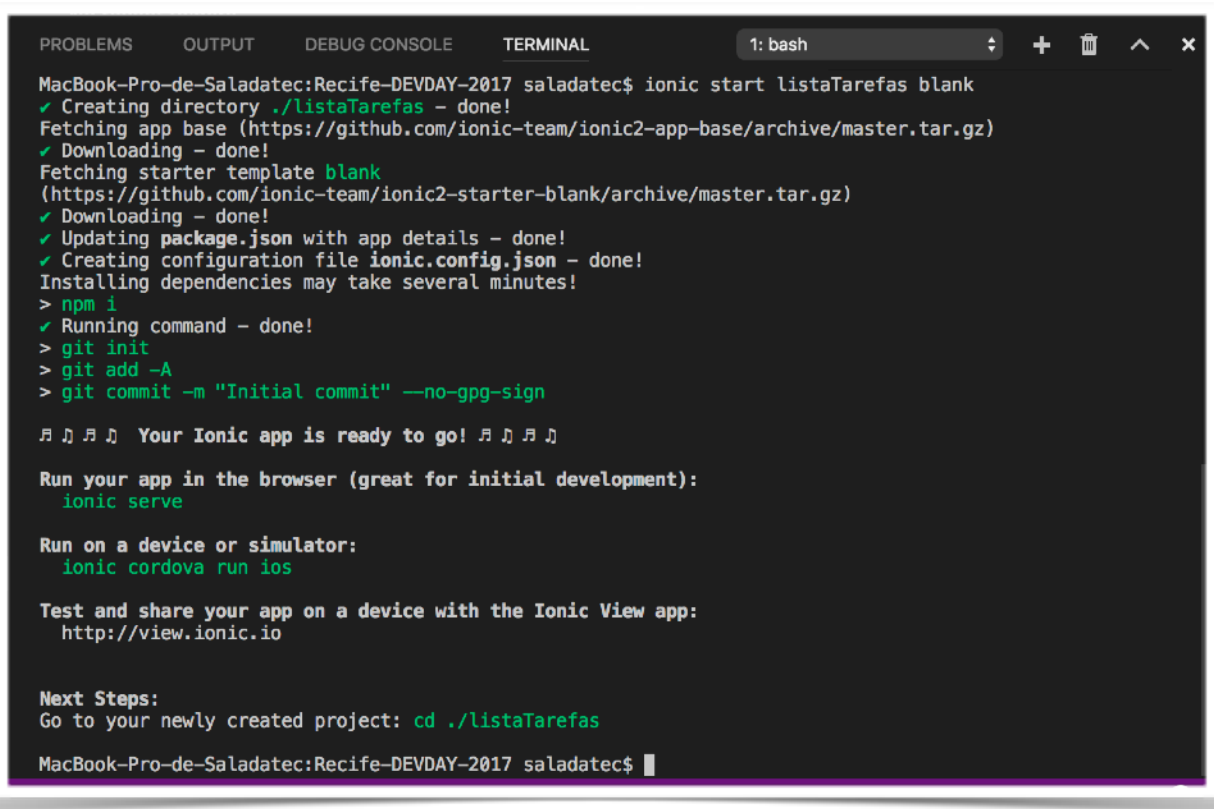
*Como criar e executar um projeto mobile?*

## Criando um projeto novo

Para criar nosso projeto, iremos usar o parâmetro “start” do Ionic CLI da seguinte forma:

```
ionic start listaTarefas blank
```

Com isso o nosso projeto será criado a partir do template *blank* do ionic.

A screenshot of a terminal window on a Mac. The window has tabs for PROBLEMS, OUTPUT, DEBUG CONSOLE, and TERMINAL. The TERMINAL tab is active, showing the command 'ionic start listaTarefas blank' and its output. The output includes success messages for creating the directory, fetching the app base and starter template, downloading them, updating package.json, creating ionic.config.json, and installing dependencies. It also shows the execution of 'npm i', 'git init', 'git add -A', and 'git commit -m "Initial commit" --no-gpg-sign'. A celebratory message 'Your Ionic app is ready to go!' is displayed with musical notes. Instructions for running the app in a browser, on a device, and testing it are provided. Finally, it shows the 'Next Steps' to navigate to the project directory.

```
MacBook-Pro-de-Saladatec:Recife-DEVDAY-2017 saladatec$ ionic start listaTarefas blank
✓ Creating directory ./listaTarefas - done!
Fetching app base (https://github.com/ionic-team/ionic2-app-base/archive/master.tar.gz)
✓ Downloading - done!
Fetching starter template blank
(https://github.com/ionic-team/ionic2-starter-blank/archive/master.tar.gz)
✓ Downloading - done!
✓ Updating package.json with app details - done!
✓ Creating configuration file ionic.config.json - done!
Installing dependencies may take several minutes!
> npm i
✓ Running command - done!
> git init
> git add -A
> git commit -m "Initial commit" --no-gpg-sign

🎵 🎵 🎵 🎵 Your Ionic app is ready to go! 🎵 🎵 🎵 🎵

Run your app in the browser (great for initial development):
  ionic serve

Run on a device or simulator:
  ionic cordova run ios

Test and share your app on a device with the Ionic View app:
http://view.ionic.io

Next Steps:
Go to your newly created project: cd ./listaTarefas

MacBook-Pro-de-Saladatec:Recife-DEVDAY-2017 saladatec$
```

## Executando/simulando um projeto no browser

Durante o desenvolvimento, pode-se usar um browser para simular a execução de nosso projeto nas fases iniciais, enquanto não se usa recursos de hardware de dispositivo móvel.

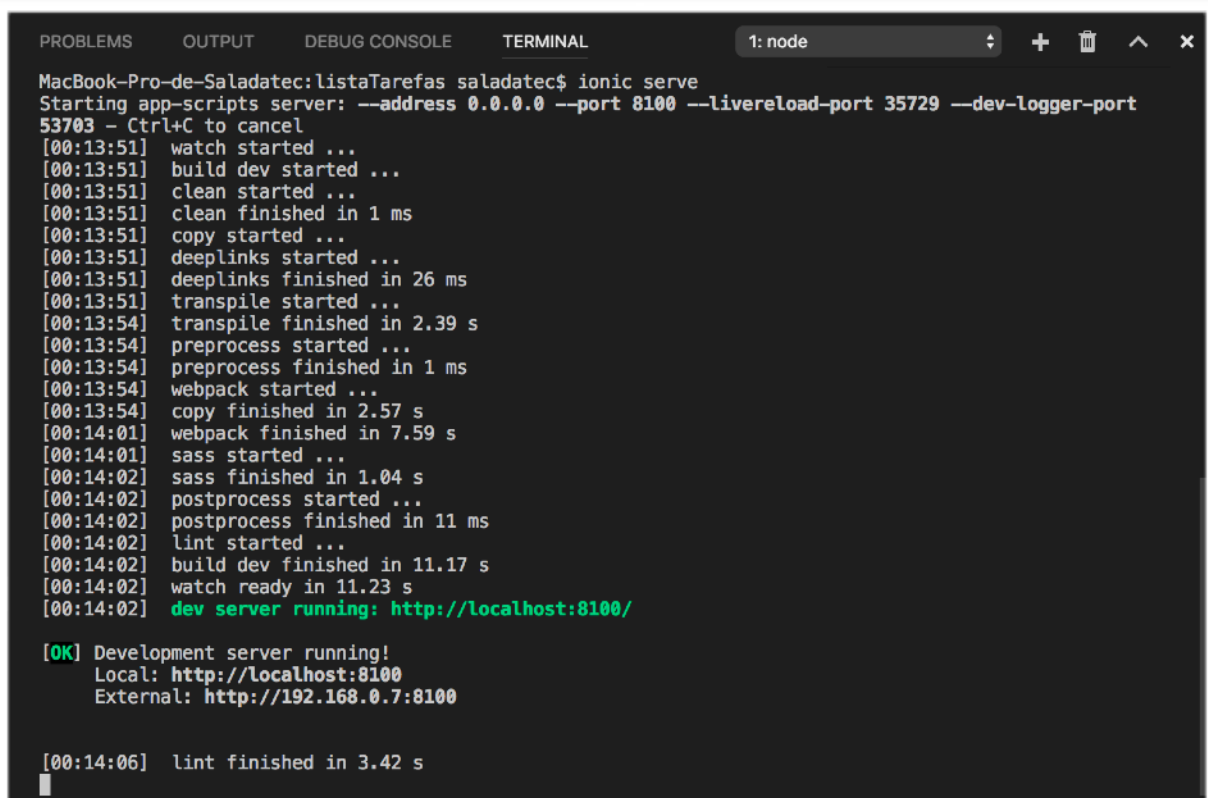
Para executar um projeto usando o Google Chrome por exemplo, iremos:

1) entrar no diretório que foi gerado para nosso projeto

```
cd listaTarefas
```

2) Executar um micro servidor http

```
ionic serve
```



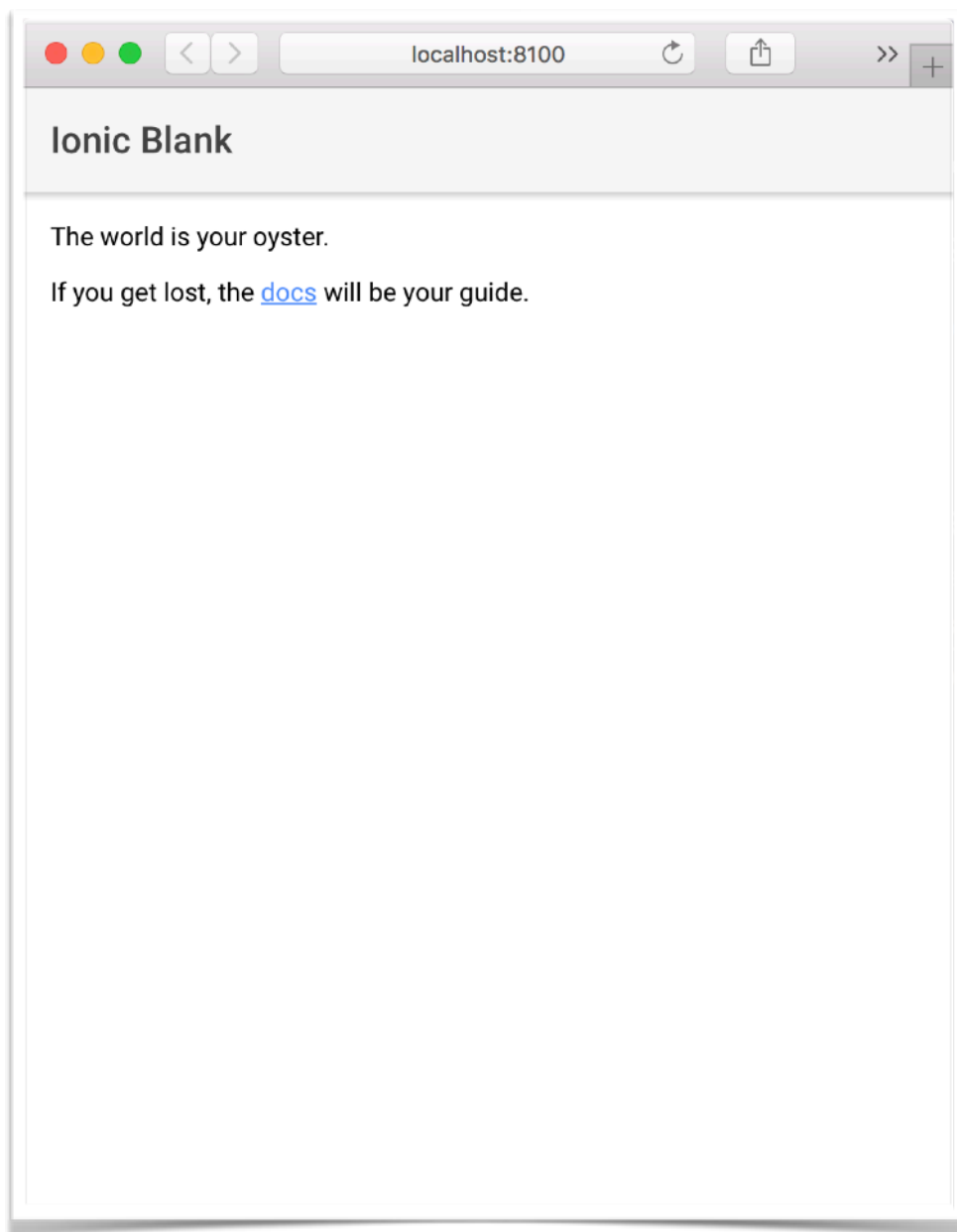
```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL 1: node
MacBook-Pro-de-Saladatec:listaTarefas saladatec$ ionic serve
Starting app-scripts server: --address 0.0.0.0 --port 8100 --livereload-port 35729 --dev-logger-port
53703 - Ctrl+C to cancel
[00:13:51] watch started ...
[00:13:51] build dev started ...
[00:13:51] clean started ...
[00:13:51] clean finished in 1 ms
[00:13:51] copy started ...
[00:13:51] deeplinks started ...
[00:13:51] deeplinks finished in 26 ms
[00:13:51] transpile started ...
[00:13:54] transpile finished in 2.39 s
[00:13:54] preprocess started ...
[00:13:54] preprocess finished in 1 ms
[00:13:54] webpack started ...
[00:13:54] copy finished in 2.57 s
[00:14:01] webpack finished in 7.59 s
[00:14:01] sass started ...
[00:14:02] sass finished in 1.04 s
[00:14:02] postprocess started ...
[00:14:02] postprocess finished in 11 ms
[00:14:02] lint started ...
[00:14:02] build dev finished in 11.17 s
[00:14:02] watch ready in 11.23 s
[00:14:02] dev server running: http://localhost:8100/

[OK] Development server running!
Local: http://localhost:8100
External: http://192.168.0.7:8100

[00:14:06] lint finished in 3.42 s
```

Note que o terminal fica preso e não libera o prompt, pois o processo fica executando e monitorando por mudanças no código. Caso haja alguma alteração de implementação, o código é automaticamente recompilado e poderemos ver o resultado das alterações diretamente no browser, pois o “refresh” será automático também.

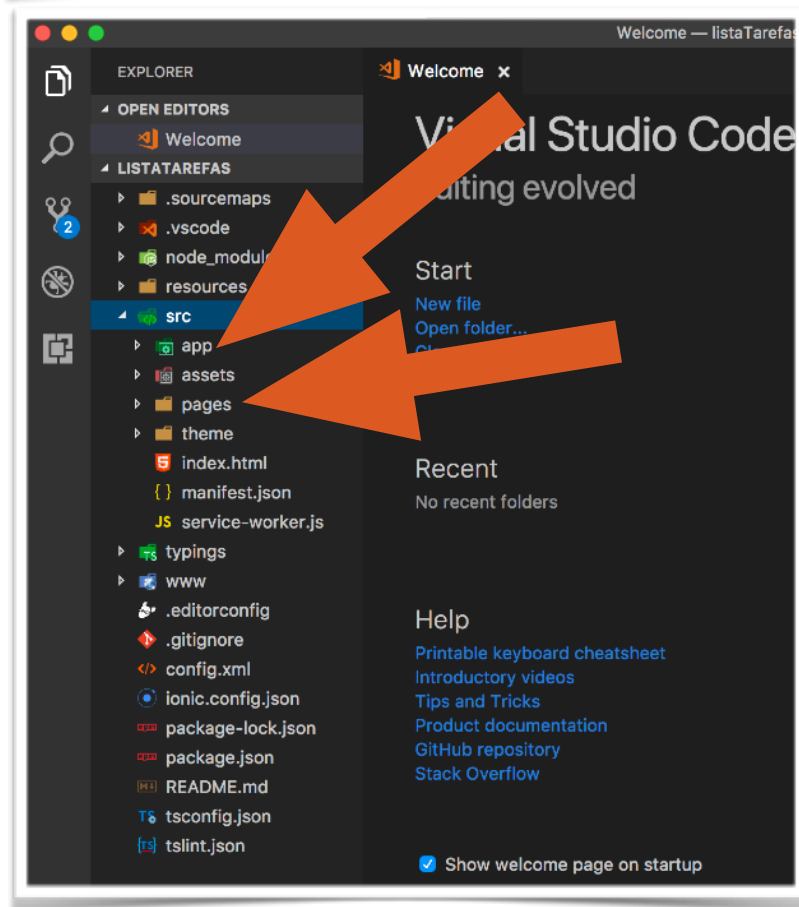
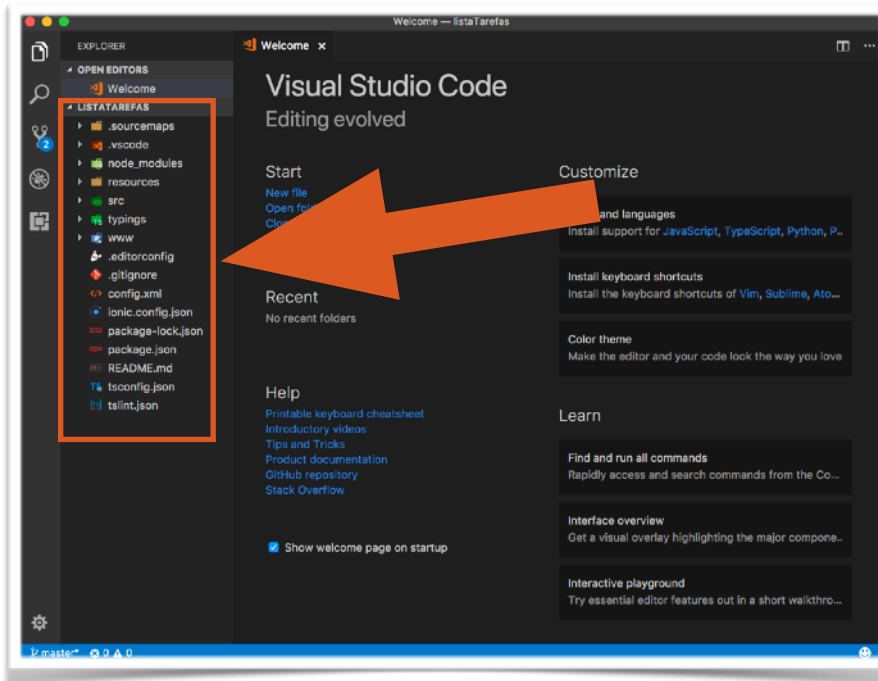
Veja que mesmo que tenhamos escolhido o template **BLANK**, já é colocada uma barra de título e um pequeno texto no corpo da tela do App.



## Alterando o projeto inicial

Vamos agora iniciar a implementação de nosso projeto exemplo. Vamos iniciar o VSCode e abrir a pasta do projeto criado anteriormente.

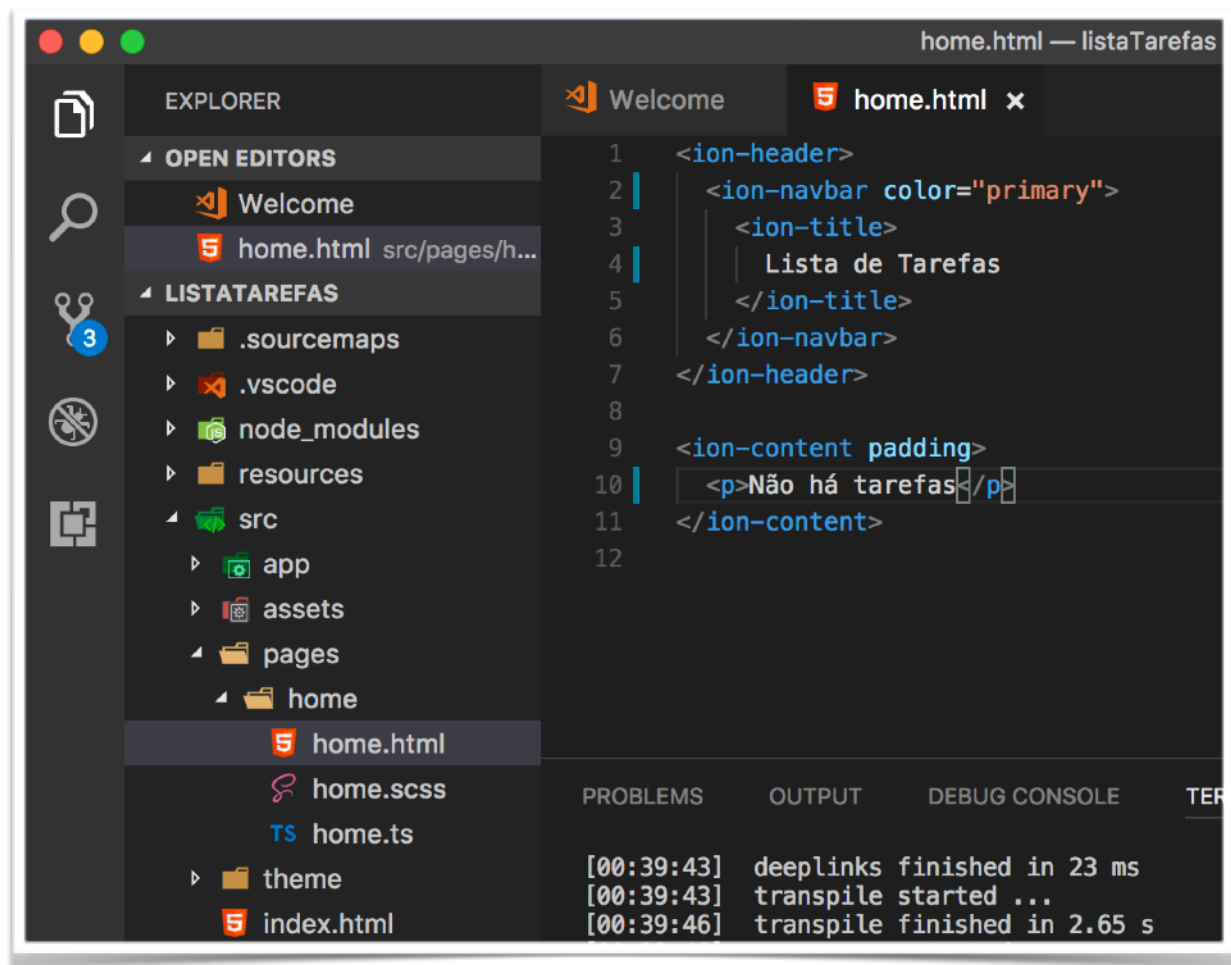
Poderemos ver que o Ionic CLI criou uma série de pastas e arquivos que formam a estrutura de um projeto Ionic. Neste treinamento iremos nos ater basicamente à pasta “src”, onde iremos efetivamente implementar nosso código TypeScript/ Angular.



Na pasta src/app iremos configurar a inicialização da aplicação e na pasta src/pages iremos construir as páginas do App

Nossa primeira alteração será na barra de título.

Iremos alterar o arquivo `src/pages/home/home.html` para o seguinte código:



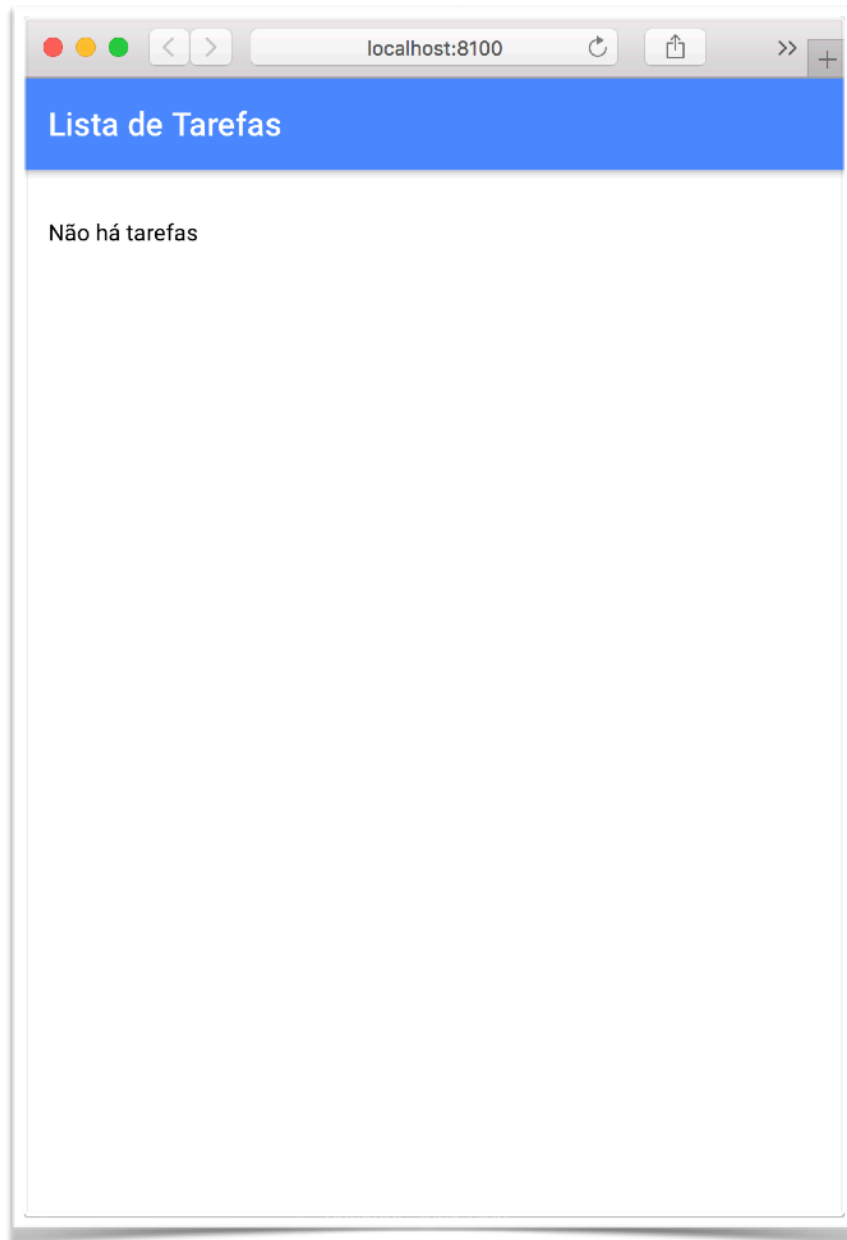
Observe que na tag `ion-navbar` colocamos um atributo `color="primary"`. Isto diz para o Ionic que se deve mudar a cor da barra de navegação/título para o que estiver definido na variável `primary` do tema do projeto.

Como padrão o Ionic define as seguintes cores:



Nossa aplicação então está com esta aparência.

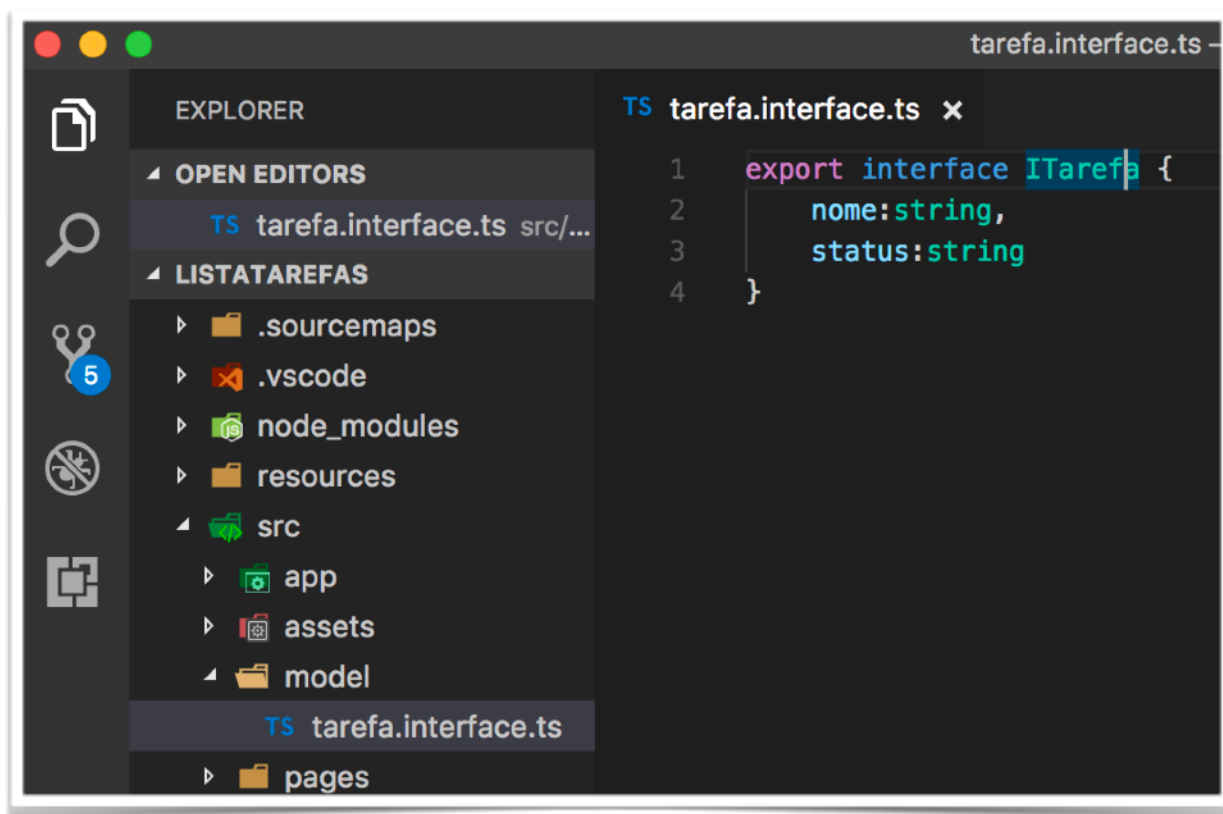
Uma barra azul (primary) de título e uma tag `<p>` com um pequeno texto no lugar onde iremos construir a listagem de tarefas.



## Definindo e apresentando uma listagem de tarefas

Vamos agora iniciar a implementação da listagem com as tarefas. Faremos por enquanto de forma estática, mas evoluiremos isso mais tarde.

Iremos implementar uma interface para representar uma tarefa qualquer. Para isso iremos criar o arquivo *src/model/tarefa.interface.ts* com o seguinte conteúdo:



Iremos usar esta interface para definirmos uma tipagem para nossa lista.

Esta é uma boa prática para usar os recursos do TypeScript a fim de termos um código mais robusto e dirimirmos possíveis erros em tempo de "compilação" .

Agora faremos alteração na implementação do arquivo *src/home/home.ts* para definirmos uma lista estática de tarefas no intuito de listarmos na tela de nosso App.

O código ficará assim:

```
TS home.ts x
1  import { ITarefa } from './../../model/tarefa.interface';
2  import { Component } from '@angular/core';
3  import { NavController } from 'ionic-angular';
4
5  @Component({
6    selector: 'page-home',
7    templateUrl: 'home.html'
8  })
9  export class HomePage {
10
11    tarefas: ITarefa[] = [
12      {
13        nome: "Tarefa 1",
14        status: "pendente"
15      },
16      {
17        nome: "Tarefa 2",
18        status: "pendente"
19      }
20    ]
21
22    constructor(public navCtrl: NavController) {
23
24    }
25
26  }
27
```

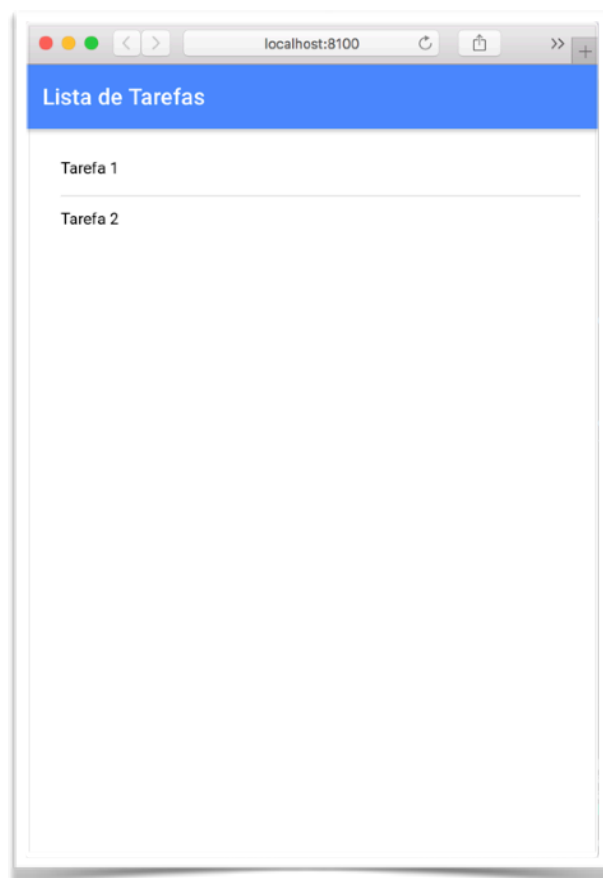
Como podemos ver, criamos um objeto tarefas que é composto de duas tarefas estáticas, Tarefa 1 e Tarefa 2.



```
home.html x
1 <ion-header>
2   <ion-navbar color="primary">
3     <ion-title>
4       Lista de Tarefas
5     </ion-title>
6   </ion-navbar>
7 </ion-header>
8
9 <ion-content padding>
10  <div *ngIf="tarefas.length == 0">
11    <p>Não há tarefas</p>
12  </div>
13  <ion-list>
14    <ion-item *ngFor="let tarefa of tarefas">
15      <h3>{{tarefa.nome}}</h3>
16    </ion-item>
17  </ion-list>
18 </ion-content>
19
```

Alterando o  
arquivo `src/home/  
home.html`, temos:

E o resultado do App agora é:



## Excluindo tarefas existentes

Vamos agora implementar a funcionalidade de excluir tarefas. Para isso, teremos que ter um botão ou outro objeto para ser acionado quando desejarmos excluir uma tarefa previamente cadastrada.

Para implementar a rotina de exclusão, iremos alterar a implementação do arquivo `src/pages/home/home.ts` para:

```
TS home.ts x
1  import { ITarefa } from './../../model/tarefa.interface';
2  import { Component } from '@angular/core';
3  import { NavController } from 'ionic-angular';
4
5  @Component({
6    selector: 'page-home',
7    templateUrl: 'home.html'
8  })
9  export class HomePage {
10
11    tarefas: ITarefa[] = [
12      {
13        nome: "Tarefa 1",
14        status: "pendente"
15      },
16      {
17        nome: "Tarefa 2",
18        status: "pendente"
19      }
20    ]
21
22    constructor(public navCtrl: NavController) {
23
24    }
25
26    onExcluir(t: ITarefa) {
27      this.tarefas.splice(this.tarefas.lastIndexOf(t),1);
28    }
29
30  }
31
```

Criaremos então, um botão slide em cada item da listagem. Então, incrementando a implementação de nossa listagem, teremos o uso do `<ion-item-sliding>` da seguinte forma no arquivo `src/pages/home/home.html`:

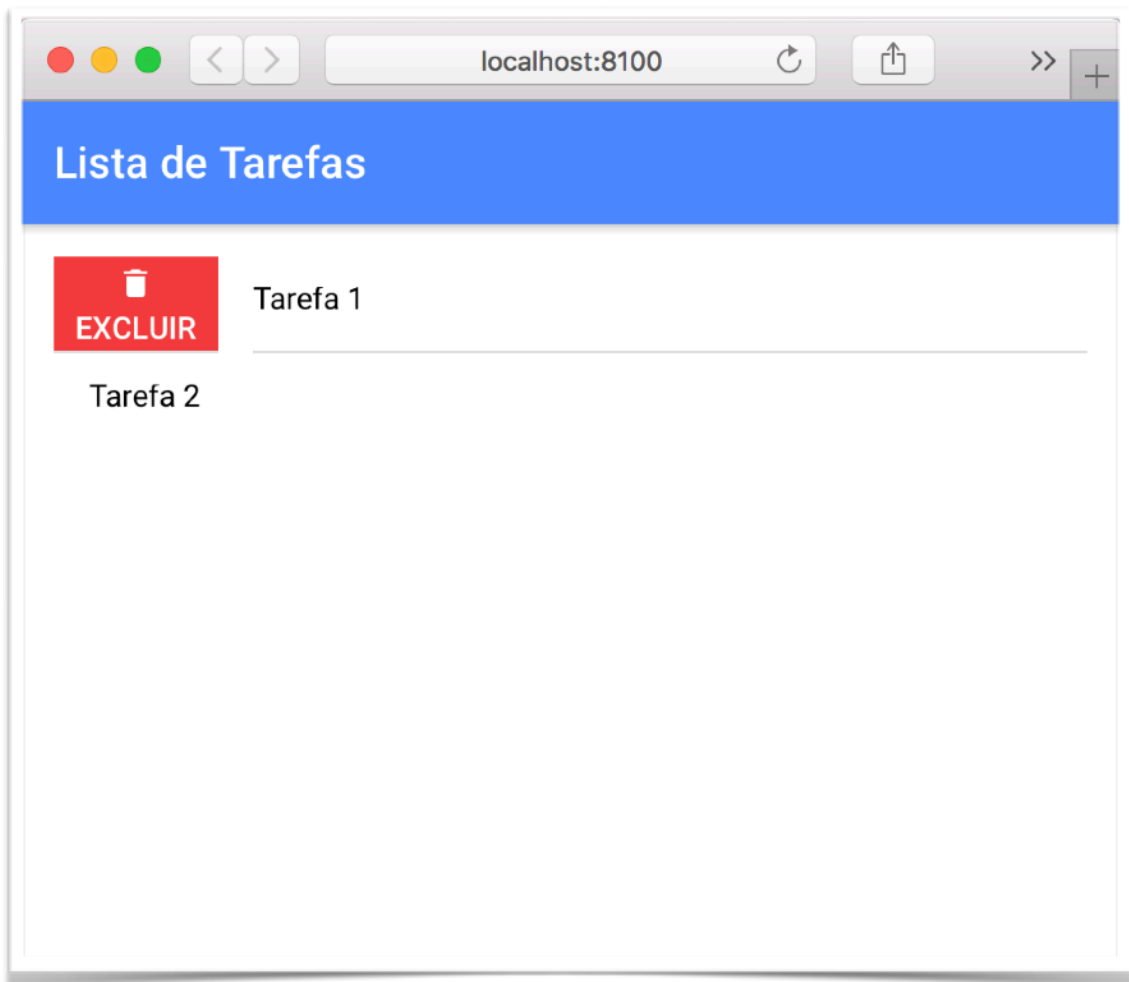


```
1 <ion-header>
2 |   <ion-navbar color="primary">
3 |     <ion-title>
4 |       Lista de Tarefas
5 |     </ion-title>
6 |   </ion-navbar>
7 </ion-header>
8
9 <ion-content padding>
10 |   <div *ngIf="tarefas.length == 0">
11 |     <p>Não há tarefas</p>
12 |   </div>
13 |   <ion-list>
14 |     <ion-item-sliding *ngFor="let tarefa of tarefas">
15 |       <ion-item>
16 |         <h3>{{tarefa.nome}}</h3>
17 |       </ion-item>
18 |       <ion-item-options side="left">
19 |         <button ion-button color="danger" (click)="onExcluir(tarefa)">
20 |           <ion-icon name="trash"></ion-icon>
21 |           Excluir
22 |         </button>
23 |       </ion-item-options>
24 |     </ion-item-sliding>
25 |   </ion-list>
26 </ion-content>
27
28
```

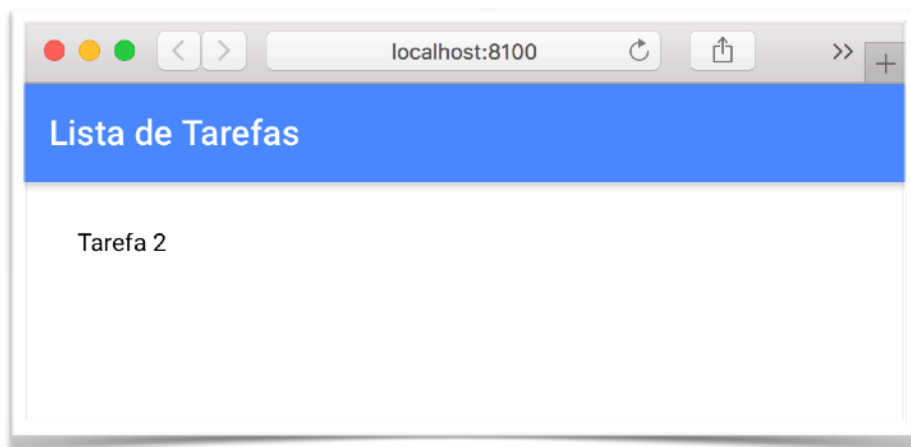
Nossa aplicação agora já tem duas funcionalidades implementadas, listagem e exclusão .

Se ao executarmos nosso App e arrastarmos um item da listagem da esquerda para a direita, poderemos visualizar o botão de exclusão que implementamos

usando a tag `<ion-item-sliding>` em conjunto com `<ion-item-options>`, vejamos a aparência do referido botão:



E ao acionar o botão EXCLUIR, o item será excluído de nossa listagem.



## Adicionando nova tarefa

Precisamos agora de uma forma de alimentar nossa lista, ou seja, uma funcionalidade que nos permita adicionar itens a nossa lista de tarefas.

Iremos implementar o método `onAdicionar` no arquivo `src/pages/home/home.ts` da seguinte forma:

```
28   onAdicionar() {
29     let prompt = this.alertCtrl.create({
30       title: 'Nova Tarefa',
31       message: "Informe o nome da nova tarefa",
32       inputs: [
33         {
34           name: 'nome',
35           placeholder: 'Nome da Tarefa'
36         },
37       ],
38       buttons: [
39         {
40           text: 'Cancelar',
41           handler: data => {
42             console.log('Foi acionado o botão cancelar');
43           }
44         },
45         {
46           text: 'Salvar',
47           handler: data => {
48             let t: ITarefa = {
49               nome: data.nome,
50               status: 'pendente'
51             };
52             this.tarefas.push(t);
53           }
54         }
55       ]
56     });
57     prompt.present();
58   }
59 }
60
```

Já na nossa página, iremos adicionar um botão no lado direito da barra de título e termos então que nosso arquivo `src/pages/home/home.html` ficará com a seguinte implementação:

```
home.html x
1  <ion-header>
2  |   <ion-navbar color="primary">
3  |     <ion-title>
4  |       Lista de Tarefas
5  |     </ion-title>
6  |     <ion-buttons end>
7  |       <button ion-button icon-only (click)="onAdicionar()">
8  |         <ion-icon name="add"></ion-icon>
9  |       </button>
10 |     </ion-buttons>
11 |   </ion-navbar>
12 </ion-header>
13
14 <ion-content padding>
15 |   <div *ngIf="tarefas.length == 0">
16 |     <p>Não há tarefas</p>
17 |   </div>
18 |   <ion-list>
19 |     <ion-item-sliding *ngFor="let tarefa of tarefas">
20 |       <ion-item>
21 |         <h3>{{tarefa.nome}}</h3>
22 |       </ion-item>
23 |       <ion-item-options side="left">
24 |         <button ion-button color="danger" (click)="onExcluir(tarefa)">
25 |           <ion-icon name="trash"></ion-icon>
26 |           Excluir
27 |         </button>
28 |       </ion-item-options>
29 |     </ion-item-sliding>
30 |   </ion-list>
31 </ion-content>
32
```

## Alterando uma tarefa existente

Falta agora permitir a alteração de uma tarefa. Para isso, iremos implementar o método `onAlterar(t: ITarefa, linha: ItemSliding) {}` que ficará da seguinte forma:

```
60 onAlterar(t: ITarefa, linha: ItemSliding) {
61   let prompt = this.alertCtrl.create({
62     title: 'Alteração de Tarefa',
63     message: `Informe o novo nome da tarefa "${t.nome}"`,
64     inputs: [
65       {
66         name: 'nome',
67         placeholder: 'Novo nome'
68       },
69     ],
70     buttons: [
71       {
72         text: 'Cancelar',
73         handler: data => {
74           linha.close();
75           console.log('Foi acionado o botão cancelar');
76         }
77       },
78       {
79         text: 'Salvar',
80         handler: data => {
81           let index = this.tarefas.indexOf(t);
82           t.nome = data.nome;
83           this.tarefas[index] = t;
84           linha.close();
85         }
86       }
87     ]
88   });
89   prompt.present();
90 }
91
```

E a versão final de nosso template do arquivo `src/pages/home/home.html`, ficará com a seguinte implementação:

```
home.html x
1  <ion-header>
2  |   <ion-navbar color="primary">
3  |     <ion-title>
4  |       Lista de Tarefas
5  |     </ion-title>
6  |     <ion-buttons end>
7  |       <button ion-button icon-only (click)="onAdicionar()">
8  |         <ion-icon name="add"></ion-icon>
9  |       </button>
10 |     </ion-buttons>
11 |   </ion-navbar>
12 </ion-header>
13
14 <ion-content padding>
15 |   <div *ngIf="tarefas.length == 0">
16 |     <p>Não há tarefas</p>
17 |   </div>
18 |   <ion-list>
19 |     <ion-item-sliding *ngFor="let tarefa of tarefas; let i = index" #i>
20 |       <ion-item>
21 |         <h3>{{tarefa.nome}}</h3>
22 |       </ion-item>
23 |       <ion-item-options side="left">
24 |         <button ion-button color="danger" (click)="onExcluir(tarefa)">
25 |           <ion-icon name="trash"></ion-icon>
26 |           Excluir
27 |         </button>
28 |       </ion-item-options>
29 |       <ion-item-options side="right">
30 |         <button ion-button color="primary" (click)="onAlterar(tarefa, i)">
31 |           <ion-icon name="create"></ion-icon>
32 |           Alterar
33 |         </button>
34 |       </ion-item-options>
35 |     </ion-item-sliding>
36 |   </ion-list>
37 </ion-content>
38
```



## Aplicação final

