

# #1 Ionic Framework

## ESSENCIAL

**Por Fábio Rogério da Silva José**

Última atualização em 04/2016

# Sobre este Ebook

O desenvolvimento de aplicativos móveis já não é novidade e está bem consolidado em suas plataformas. Mas algumas discussões sempre ocorrem na hora de definir qual plataforma desenvolver, se você procura está reposta neste ebook não irá encontrar, pois são vários os fatores, e pontos de análise, que definem se você deve desenvolver nativo ou híbrido e vamos deixar essa discussão para um outro momento.

Este ebook é o primeiro de uma série de ebooks que irão abordar o Ionic Framework, neles você irá encontrar, de forma simples e objetiva, explicações, exemplos e muito código sobre o desenvolvimento de aplicativos móveis híbridos. Cada ebook irá abordar um tema específico, exceto este primeiro que é a base de aprendizado para os demais assuntos.

Todos os ebooks elaborados por mim serão gratuitos, pois o objetivo central destes materiais é o compartilhamento de conhecimento e disseminação deste *framework* fantástico.

## Público alvo

Este ebook foi escrito para você que está começando no mundo do desenvolvimento de aplicativos móveis e tem interesse em aprender sobre uma tecnologia híbrida. Não iremos abordar padrões de desenvolvimento e não vamos utilizar lógicas de programação complexas para facilitar o aprendizado e ter um público mais amplo, tendo em vista que iniciantes também no mundo da programação podem estar lendo este ebook.

Este material não deverá ser seu único ponto de referência e guia, pois existem outros ebooks, livros e blogs que falam sobre Ionic Framework e podem ampliar seu conhecimento.

## Sobre o autor

**Fábio Rogério da Silva José**, conhecido como Fábio Rogério SJ nas redes sociais, é desenvolvedor de aplicações web e mobile desde quando o *Internet Explorer 7* era um pesadelo e desenvolver aplicativos híbridos com tecnologias web era um trabalho árduo, ou seja, desde 2007.

Atualmente Fábio Rogério trabalha com consultoria e treinamento em desenvolvimento de aplicações web e mobile utilizando tecnologias híbridas. E também é professor, de curso superior, onde ministra as disciplinas de desenvolvimento *frontend*, desenvolvimento de aplicativos móveis, web design, design de interação e lógica de programação.

# Sumário

<b>Introdução</b>	4
Apache Cordova	4
AngularJS	5
ngCordova	5
Ionic 2.0	5
<b>Instalando ambiente</b>	6
NodeJS	6
Git	6
Sublime Text	7
Cordova e Ionic CLI	7
<b>Criando o primeiro app</b>	8
<b>Estrutura dos projetos Ionic</b>	10
config.xml	10
www	11
<b>Componentes</b>	12
Header	12
Content	14
Footer	15
SubHeader e SubFooter	16
Cores	16
Icons	17
Buttons	17
List	20
Cards	25

Forms	28
Toggle	30
Checkbox	31
Radio Buttons	32
Range	33
Select	34
Tabs	36
Grid	38
Conclusão sobre componentes	41
<b>Empacotando o aplicativo</b>	42
Android	42
iOS	48
<b>Próximo passo</b>	50

# Introdução

Desenvolver aplicativos móveis geralmente é divertido e, em muitos casos, uma tarefa complexa dependendo dos requisitos do projeto. Escolher qual tecnologia utilizar é um ponto altamente discutido nas comunidades e é a etapa principal do ciclo do desenvolvimento, pois se você escolher a tecnologia “errada” o custo para migrar toda a aplicação pode ser alto. Acredito que não existe uma tecnologia “errada”, mas sim pessoas que não dominam tal tecnologia.

O Ionic é um *framework open source* e um dos *frameworks*, para desenvolver aplicativos híbridos, mais utilizado no mundo, tendo mais de 23 mil *stars* e mais de 4 mil *forks* no Github.

Sua premissa é utilizar tecnologias web, como HTML5, CSS e JavaScript, para desenvolver aplicativos móveis híbridos, ou seja, aplicativos que usam tecnologias web e também nativas. Se você sabe criar um site você conseguirá criar um aplicativo sem muito esforço.

O diferencial do Ionic Framework é sua preocupação com a performance e ganho de produtividade no desenvolvimento. Você vai comprovar isto no decorrer desde material enquanto for criando os projetos exemplos.

O Ionic utiliza como base o projeto Apache Cordova e dispõe de uma série de componentes e um CLI (*command-line interface*) completo para criar, testar e publicar aplicativos com apenas alguns comandos.

## Apache Cordova

Em um evento chamado iPhoneDevCamp, ocorrido em São Francisco em 2006, três jovens criaram uma plataforma para criar aplicativos com HTML5/CSS/JS denominado PhoneGap. Em 2010 a Apple confirmou que a plataforma estava de acordo com os parâmetros da licença para desenvolvedores iOS.

Em 2011 a Adobe comprou o PhoneGap e doou o código fonte do projeto para a fundação Apache, porém o nome “PhoneGap” ficou exclusivo para a Adobe e o projeto foi renomeado para Apache Cordova.

O Apache Cordova também é *open source* e é utilizado pelo Ionic para acessar as funcionalidades nativas dos aparelhos móveis como acelerômetro, câmera e geo localização. Ele também tem como requisito gerar o aplicativo para diferentes plataformas como Android, iOS, Windows Phone, blackberry entre outros.

Hoje a maioria das plataformas e *frameworks* para criação de aplicativos móveis híbridos utiliza como base o Apache Cordova, entre eles estão: Adobe PhoneGap, Monaca, Onsen UI, Visual Studio, Taco, Telerik e Ionic. Veja mais detalhes em <http://cordova.apache.org>.

## AngularJS

AngularJS é um *framework* escrito em JavaScript *open source*, mantido pela Google, e tem por objetivo facilitar o desenvolvimento de aplicações web estendendo o HTML tradicional para trabalhar com conteúdo dinâmico, com a ligação direta e bidirecional dos dados (*two-way data-binding*) que permite sincronização automática entre o HTML e o objeto em JavaScript.

Não faz parte do escopo deste material explicar, de forma ampla, o funcionamento do AngularJS, porém você irá aprender com a evolução dos exercícios aplicados.

Atualmente o AngularJS lançou a versão 2.0, que mudou totalmente a forma de implementar aplicações SPA (*single page application*), também não iremos abordar este tema neste material.

O Ionic utiliza o AngularJS para controlar os objetos e configurar as telas, que chamamos de rotas, do aplicativo. Não se preocupe iremos ver isso passo a passo.

## ngCordova

Como já foi dito anteriormente o Ionic utiliza o Apache Cordova para disponibilizar os recursos nativos para a aplicação, mas quando se utiliza o AngularJS seu aplicativo pode não funcionar como deveria, pois o *bind* do objeto pode não ser atualizado na *view* quando não passa pelo *apply* do AngularJS. Se você não conhece o AngularJS essa explicação ficou vaga, não se preocupe pois no futuro você irá entender.

O ngCordova é apenas um atalho para chamar os *plugins* nativos do Apache Cordova, de forma que seja escutado pelo AngularJS.

## Ionic 2.0

No momento em que foi escrito este ebook o Ionic havia lançado recentemente uma nova versão do seu *framework*, com várias melhorias e com a implementação do AngularJS 2.0, algumas lógicas que são feitas no Ionic 1.0 são bem diferentes da versão 2.0, sendo assim irei aguardar a estabilidade desta nova versão e tratar o conteúdo em um novo ebook, focado apenas no Ionic 2.0.

# Instalando ambiente

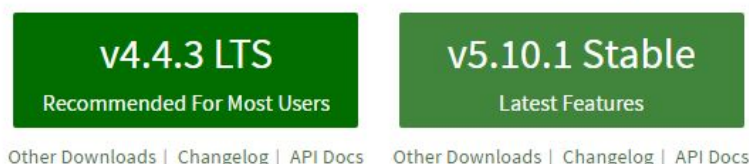
Todos os comandos do Ionic, como criar um novo projeto, testar e distribuir, são feitos direto no terminal através de um CLI (*command-line interface*) chamado Ionic CLI. O Ionic CLI utiliza o NodeJS para executar tarefas e o Git para baixar alguns pacotes.

Para instalar o Ionic Framework você precisa ter instalado o NodeJS, Git e um editor de códigos.

O Ionic pode ser instalado em Windows, Linux ou Mac, os passos seguintes são os mesmos para os três sistemas operacionais.

## NodeJS

NodeJS, ou Node.js, é uma plataforma construída sobre o motor JavaScript do Google Chrome para facilmente construir aplicações rápidas e escaláveis. Para instalar o NodeJS baixe a última versão, no site oficial, escolhendo seu sistema operacional: <https://nodejs.org>.



Após baixado abra o executável e siga os passos de instalação até aparecer a tela de instalação concluída. Juntamente com o NodeJS é instalado o NPM (*Node Package Manager*), que é o gerenciador de pacotes dos módulos escritos em JavaScript utilizando o NodeJS.

## Git

Git é um sistema de controle de versão distribuído e um sistema de gerenciamento de código fonte, com ênfase em velocidade. O Git foi inicialmente projetado e desenvolvido por [Linus Torvalds](#) para o desenvolvimento do kernel Linux, mas foi adotado por muitos outros projetos.

Como outros projetos *open source* o Ionic está hospedado no [GitHub](#), que é um servidor de Git gratuito para projetos abertos, sendo assim precisamos do Git instalado na máquina. Para instalar entre no link: <https://git-scm.com/downloads> e baixe a versão para seu sistema operacional.



Após baixado siga os passos e deixe as configurações padrões de instalação até concluir.

## Sublime Text

Sublime Text é um editor de código simples, rápido e eficiente, porem este passo não é obrigatório caso você já utilize um outro editor. Para baixar o Sublime Text acesse o link: <https://www.sublimetext.com/3>.

## Cordova e Ionic CLI

Com o NodeJS e Git instalado vamos instalar o Cordova e o Ionic, o processo é bem simples e consiste em apenas um comando, pois vamos utilizar o NPM para fazer a instalação.

Abra um terminal, certifique-se que você tenha permissão de administrador, e digite o comando abaixo. Os exemplos deste material são apresentados em ambiente Windows, sendo assim para acessar o terminal abra o programa CMD.

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

Este processo pode demorar alguns minutos dependendo de sua conexão. Ao concluir digite o comando abaixo para verificar se a instalação foi efetuada com sucesso:

```
ionic --version
```

Se estiver instalado corretamente a versão do Ionic Framework deverá ser apresentada no terminal.

Se você teve algum problema desconhecido na instalação acesse a lista de discussão sobre o ambiente de instalação no link abaixo e deixe sua dúvida:

<http://fabiorogerosj.com.br/desenvolvendo-aplicativos-mobile-ionic-post-1.html>



# Criando o primeiro app

O Ionic disponibiliza alguns *templates* de projetos, ou seja, aplicativos exemplos prontos para ser editados, isso facilita para não precisar criar todos os arquivos manuais. Os *templates* disponíveis são:

<b>blank</b>	Cria um projeto em branco apenas com uma tela inicial
<b>sidemenu</b>	Cria um projeto com algumas telas e um menu lateral
<b>tabs</b>	Cria um projeto com algumas abas
<b>maps</b>	Cria um projeto com um mapa exemplo
<b>salesforce</b>	Cria um projeto com Ionic e Salesforce
<b>complex-list</b>	Cria um projeto com uma lista de exemplo

Vamos começar com um projeto em branco, para isso digite o comando *start*, e seus parâmetros, no terminal:

```
ionic start PrimeiroApp blank
```

Neste momento o Ionic começa a fazer download do arquivo de *template* e executar algumas tarefas, dentre elas, se for a primeira vez que você executa este comando, ele pergunta se você deseja criar uma conta na plataforma ionic.io, iremos discutir sobre ionic.io em outro material, e você deve responder sim (Y) ou não (n). Se responder sim ele irá abrir o site do ionic.io e se responder não ele irá apenas continuar com a criação do projeto.

Para este momento responda **n**:

```
Create an ionic.io account to send Push Notifications and use the Ionic View app?
```

```
(Y/n): n
```

Vamos entender o que cada comando faz:

- **start**: Cria um novo aplicativo
- **PrimeiroApp**: Define o nome do aplicativo, uma pasta com este nome será criada no diretório que você está. O nome do aplicativo poderá ser alterado posteriormente caso seja necessário.
- **blank**: Este é o *template* utilizado para criar o app.

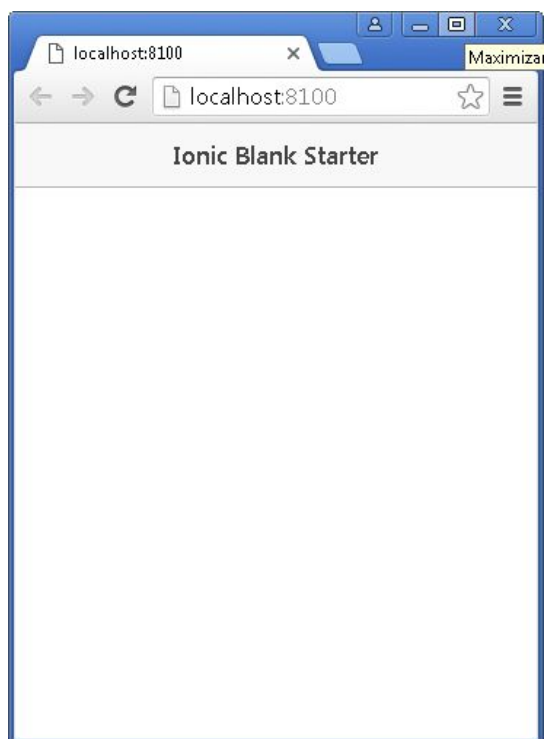
Para testar nosso primeiro aplicativo entre na pasta que foi criada com o nome do seu aplicativo, via linha de comando:

```
cd PrimeiroApp
```

Em seguida digite o comando para rodar o aplicativo no *browser*:

```
ionic serve
```

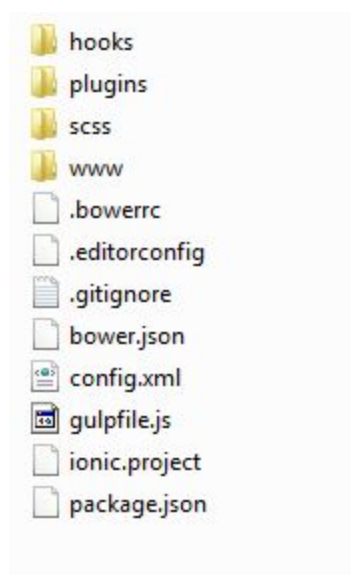
Na primeira vez que você executa este comando o Ionic irá perguntar em qual IP você deseja deixar acessível seu projeto, para este momento escolha localhost. Você verá seu aplicativo executando no navegador.



# Estrutura dos projetos Ionic

Quando criamos um novo aplicativo, o Ionic utiliza o Cordova para criar a estrutura base do aplicativo já com o *template* escolhido, pois como dito anteriormente o Ionic utiliza o Cordova como base para realizar algumas tarefas.

Abra a pasta PrimeiroApp e veja os seguintes arquivos:

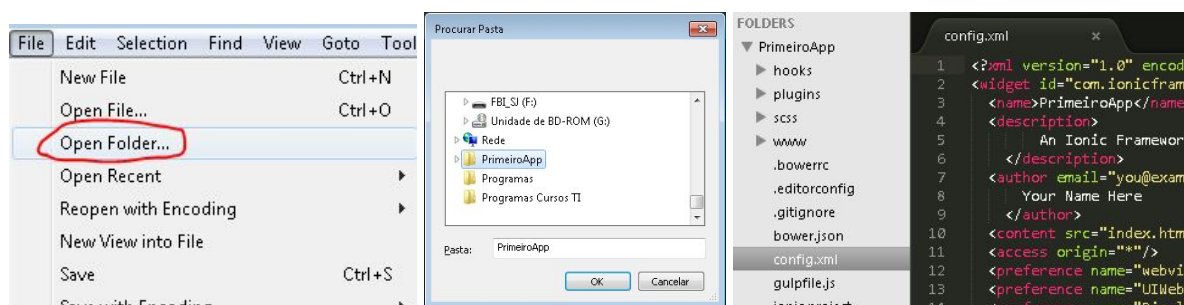


Vamos discutir cada arquivo no seu devido momento, por enquanto precisamos apenas conhecer o config.xml e a pasta www.

## config.xml

Neste arquivo é definido os principais parâmetros de inicialização e permissões que o Cordova precisa para startar e compilar, para diferentes plataformas, seu aplicativo.

Se você estiver utilizando o Sublime Text abra seu projeto e verifique o arquivo config.xml:



A segunda linha define o pacote do projeto e a versão do seu aplicativo, você precisará trocar esta versão toda vez que for disponibilizar uma nova versão do seu aplicativo nos *marketplaces* como Play Store e Apple Store. Veremos mais detalhes de como publicar em outro momento.

```
<widget id="com.ionicframework.primeiroapp117395" version="0.0.1" ...
```

Da linha três até a linha nove podemos ver o nome do aplicativo, que ira aparecer nos atalhos do *device*, a descrição e os dados do autor.

As demais linhas são configurações de inicialização, habilitação de *plugins* e permissões, iremos discutir sobre elas quando chegar o momento.

## WWW

Nesta pasta *www* você irá encontrar os arquivos HTML/CSS e JavaScript do seu aplicativo e todos os arquivos contidos, ou criados por você, nesta pasta poderão ser utilizados pelo seu aplicativo. Vamos entender a estrutura que o Ionic criou dentro desta pasta:

```
www
├── css
│   └── style.css
├── img
│   └── ionic.png
├── js
│   └── app.js
├── lib
│   └── ionic
└── index.html
```

No arquivo *style.css* iremos codificar os estilos personalizados dos elementos quando necessário. Na pasta *img* iremos colocar todas nossas imagens. O arquivo *app.js* é o principal arquivo JavaScript do aplicativo, nele iremos configurar as rotas (telas) e criar os controles. O arquivo *index.html* é nossa primeira tela do aplicativo (*view*).

# Componentes

O Ionic Framework disponibiliza diversos componentes para criar seu aplicativo, estes componentes nada mais são que classes em CSS para renderizar da melhor forma seu elemento em HTML. Você pode adicionar mais atributos em CSS nos componentes ou até mesmo criar seu próprio componente.

Existem duas formas de utilizar alguns dos componentes do Ionic:

- Utilizar elementos nativos do HTML e adicionar as classes CSS

```
<ul class="list">
  <li class="item">
    ...
  </li>
</ul>
```

- Ou utilizar as nomenclaturas dos elementos já customizados do Ionic:

```
<ion-list>
  <ion-item>
    ...
  </ion-item>
</ion-list>
```

O resultado das duas formas são os mesmos. Neste material vamos utilizar, na maioria dos casos, os nomes de elementos do Ionic, para reduzir a digitação de códigos.

## Header

Header é um componente que fica sempre fixo no topo, utilizado para adicionar título e botões de ações sobre a tela aberta.

Lembrando que você pode adicionar novos elementos no CSS para modificar de forma diferente ao que o Ionic disponibiliza ou até mesmo criar um Header totalmente novo com o seu próprio CSS.

Crie um novo projeto chamado AppHeader, não se esqueça de sair da pasta do projeto criado anteriormente, para que não gere confusão nas pastas.

Para voltar uma pasta anterior por linha de comando, digite:

```
cd ..
```

Para criar, entrar na pasta e rodar um novo app digite:

```
ionic start appHeader blank
cd appHeader
ionic serve
```

Quando criamos um novo app com o *template* blank o `index.html` já contém um Header, Vamos editá-lo alterando o arquivo `www/index.html`.

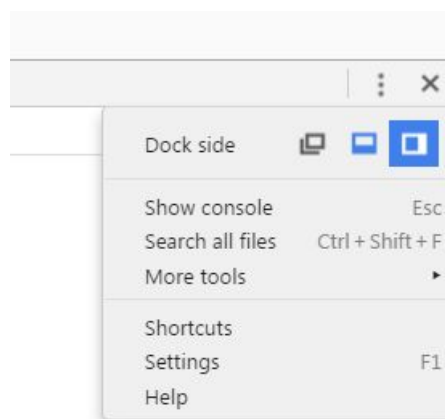
```
<ion-header-bar class="bar-positive">
  <h1 class="title">App Exemplo 1</h1>
</ion-header-bar>
```

Alteramos o estilo do Header e alteramos o seu título. Veremos como adicionar botões na sessão Buttons.

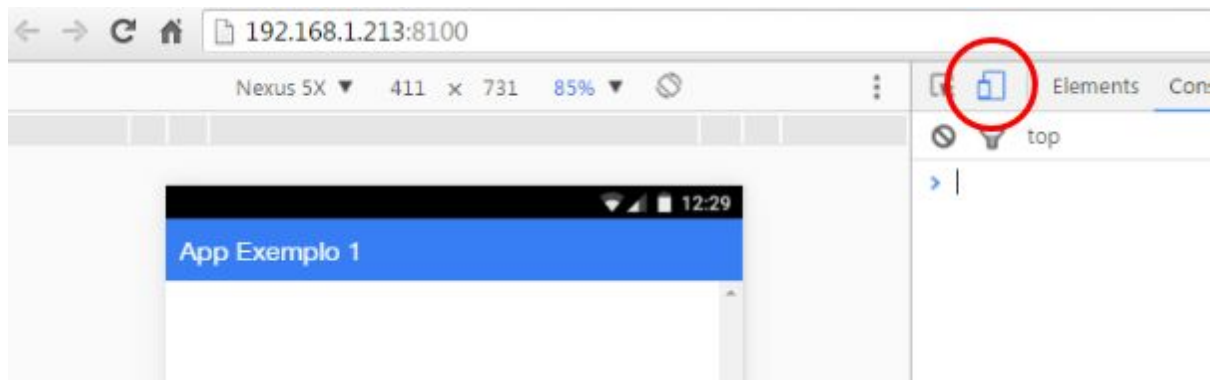
Quando você faz alguma alteração no seu projeto e salva, o aplicativo no *browser* é automaticamente recarregado. Você verá um resultado igual a este:



Esta visualização não é muito útil pois não simula o tamanho de tela do *device*. Você pode redimensionar seu *browser* para que fique menor na largura ou utilizar o módulo visualização *mobile* do Chrome. Para isso pressione F11 no Windows/Linux ou option+command+J no Mac. Ao lado direito do terminal que foi apresentado clique nos três pontinhos e escolha a visualização *Dock to right*, para que seu terminal fique ao lado direito da pagina que é exibida.



Em seguida, ao lado esquerdo do terminal, habilite a visualização para *device*, com isso seu aplicativo será apresentado e configurado nos padrões de um *smartphone*:



Por padrão os títulos em Android ficam ao lado esquerdo, já no iOS ao centro, porém é possível forçar o título ficar no centro em ambos sistemas adicionando o atributo `align-title="center"` no componente header:

```
<ion-header-bar class="bar-positive" align-title="center">
  <h1 class="title">App Exemplo 1</h1>
</ion-header-bar>
```

## Content

Content é a área de conteúdo do Ionic, apenas o que estiver dentro deste elemento irá receber rolagem, ou seja, o Header é fixo no topo e o Content utiliza o restante da página, caso tenha mais elementos do que pode ser exibido na interface ele irá criar uma barra de rolagem.

Ainda no aplicativo AppHeader adicione a classe `padding` no elemento `ion-content`, para adicionar um espaço interno, e adicione os elementos dentro de `ion-content` como mostra o exemplo abaixo:

```
<body ng-app="starter">
  <ion-pane>
    <ion-header-bar class="bar-positive">
      <h1 class="title">App Exemplo 1</h1>
    </ion-header-bar>
    <ion-content class="padding">
      <h1>Exemplo de conteúdo Ionic</h1>
      <h2>Apenas uma demonstração da rolagem</h2>
      <h3>As linhas abaixo são exemplos</h3>
      
      <p>parágrafo 1</p>
      
      <p>parágrafo 2</p>
      
      <p>parágrafo 3</p>
      
    </ion-content>
  </ion-pane>
</body>
```

```

    <p>parágrafo 4</p>
    
    <p>parágrafo 5</p>
    
    <p>parágrafo 6</p>
  </ion-content>
</ion-pane>
</body>

```

Salve seu código e veja no *browser* o resultado com a rolagem.

## Footer

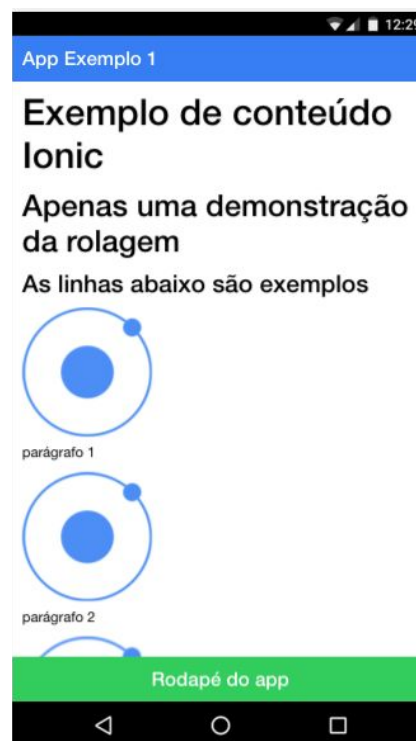
Footer também é um componente fixo porem fica sempre posicionado no rodapé da página. Adicione o código abaixo após o componente `</ion-content>`:

```

<ion-footer-bar class="bar-balanced">
  <div class="title">Rodapé do app</div>
</ion-footer-bar>

```

O resultado será algo parecido com isso:





## SubHeader e SubFooter

Bastante utilizado para combinar botões de ação, o SubHeader e SubFooter também são fixos e ficam como secundários, ou seja, abaixo do Header, para o SubHeader e a cima do Footer, para o SubFooter, altere seu Header e Footer seguindo o código abaixo:

```
...
<ion-pane>
  <ion-header-bar class="bar-positive">
    <h1 class="title">App Exemplo 1</h1>
  </ion-header-bar>
  <ion-header-bar class="bar-subheader bar-dark">
    <h1 class="title">SubHeader</h1>
  </ion-header-bar>

  <ion-content class="padding">
    ...
  </ion-content>

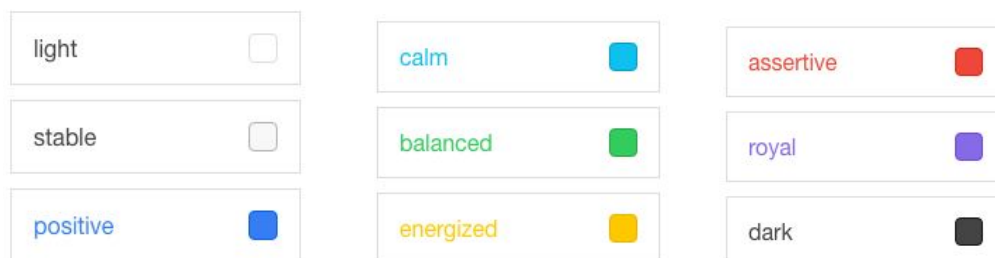
  <ion-footer-bar class="bar bar-balanced">
    <div class="title">Rodapé do app</div>
  </ion-footer-bar>
  <ion-footer-bar class="bar-subfooter bar-assertive">
    <div class="title">Rodapé secundário</div>
  </ion-footer-bar>

</ion-pane>
...
```

Foi adicionado as classes `bar-subheader` e `bar-subfooter` para que o *layout* fosse posicionado da forma correta.

## Cores

O Ionic vem com um conjunto de cores e uma nomenclatura onde podemos ver abaixo:



Podemos alterar as cores conforme o *layout* desejado e é interessante utilizar uma nomenclatura para facilitar o uso dessas classes.

A maioria dos componentes podem ser informados qual tema de cor utilizar, por exemplo o Header e Footer podemos adicionar a classe `bar-NOMETEMA`. Para um elemento título, como H1, H2, etc, podemos adicionar apenas o nome da classe.

Vamos conhecendo as diferentes aplicabilidades dos temas no decorrer deste material.

## Icons

Ionic dispõe de um coleção de ícones baseado em fonte *open source*, com mais de 500 ícones para ser utilizado em seu aplicativo. Para utilizar basta adicionar a classe do ícone desejado, que pode ser encontrado no site <http://ionicons.com>, em um elemento `<i></i>`:

```
<i class="icon ion-star"></i>
```

No site [ionicons](http://ionicons.com) ao clicar sobre um determinado ícone você verá a classe que deve ser adicionado no elemento.



## Buttons

Os botões são elementos essenciais de um aplicativo, no Ionic é possível escolher o tema, estilo e adicionar ícone. Crie um novo projeto chamado appButtons:

```
ionic start appButtons blank
```

Adicione o componente botão utilizando o tema `assertive`:

```
<button class="button button-assertive">Primeiro botão</button>
```

É possível determinar que o componente ocupe 100% da largura da tela adicionando a classe `button-block`, ou se for utilizado em um formulário, por exemplo, e deseja que

seja completo, ou seja, sem bordas arredondadas nas laterais pode utilizar a classe `button-full`:

```
<button class="button button-assertive button-block">
  Primeiro botão
</button>
```

Os Buttons tem classes para tratar botões de diferentes tamanhos como pequenos, normais e grandes:

```
<button class="button button-small button-balanced">
  Botão pequeno
</button>
<button class="button button-dark">
  Botão normal
</button>
<button class="button button-large button-royal">
  Botão grande
</button>
```

Outro estilo bem utilizado é botão com apenas bordas ou apenas textos:

```
<button class="button button-outline button-positive">
  Botão apenas borda
</button>
<button class="button button-clear button-positive">
  Botão apenas texto
</button>
```

Porem sem dúvida o que traz mais elegância para um aplicativo, e uma melhor experiência com o usuário, é um botão com ícone. Podemos customizar de diferente formas os botões com os ícones, basta adicionar o nome da classe do ícone desejado:

```
<button class="button icon-left ion-home">
  Home
</button>
<button class="button icon-left ion-star button-positive">
  Favoritos
</button>
<button class="button icon-right ion-chevron-right button-calm">
  Saiba mais
</button>
<button class="button icon-left ion-chevron-left button-clear button-dark">
  Voltar
</button>
<button class="button icon ion-gear-a"></button>
<button class="button button-icon icon ion-settings"></button>
<button class="button button-outline icon-right ion-navicon button-balanced">
  Ações
```

```
</button>
```

O Header e Footer também aceitam botões e você apenas precisa adicionar antes, ou depois, do elemento `title` para deixar a esquerda ou à direita:

```
<ion-header-bar class="bar-balanced">
  <button class="button icon ion-navicon"></button>
  <h1 class="title">AppButtons exemplos</h1>
  <button class="button">Editar</button>
</ion-header-bar>
```

Ou se preferir sem as bordas:

```
<ion-header-bar class="bar-balanced">
  <button class="button button-icon icon ion-navicon"></button>
  <h1 class="title">AppButtons exemplos</h1>
  <button class="button button-clear">Editar</button>
</ion-header-bar>
```

E por fim, na área de conteúdo, é possível criar uma barra de botões, onde eles ocupam 100% da largura da página e divide o tamanho de cada botão por igual:

```
<div class="button-bar">
  <button class="button button-dark">Botão 1</button>
  <button class="button button-dark">Botão 2</button>
  <button class="button button-dark">Botão 3</button>
</div>
```

Seu projeto deve estar com essa aparência:



## List

O componente List é uma lista de itens e é muito comum sua utilização em aplicativos móveis, e podem conter textos simples, botões, ícones, rótulos e imagens. Crie um novo aplicativo chamado appList:

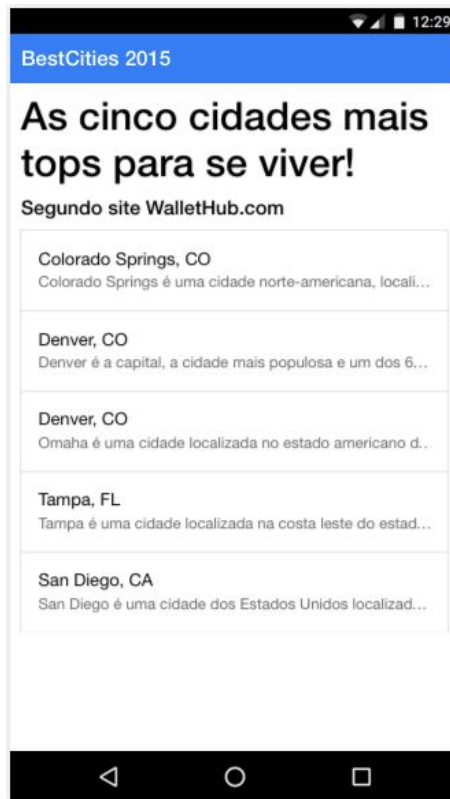
```
ionic start appList blank
```

Na área de conteúdo adicione um título (H1) e um subtítulo (H2), em seguida crie a lista (ion-list) com cinco elementos de item (ion-item). Cada elemento tem um título (H2) e um parágrafo (p). Vamos ao código:

```
<ion-content class="padding">
  <h1>As cinco cidades mais tops para se viver!</h1>
  <h2>Segundo site WalletHub.com</h2>
  <ion-list>
    <ion-item>
      <h3>Colorado Springs, CO</h3>
      <p>Colorado Springs é uma cidade norte-americana, localizada no estado do Colorado, no Condado de El Paso.</p>
    </ion-item>
    <ion-item>
      <h3>Denver, CO</h3>
      <p>Denver é a capital, a cidade mais populosa e um dos 64 condados do estado norte-americano do Colorado.</p>
    </ion-item>
    <ion-item>
      <h3>Denver, CO</h3>
      <p>Omaha é uma cidade localizada no estado americano do Nebraska, no Condado de Douglas.</p>
    </ion-item>
    <ion-item>
      <h3>Tampa, FL</h3>
      <p>Tampa é uma cidade localizada na costa leste do estado norte-americano da Flórida, no condado de Hillsborough, do qual é sede.</p>
    </ion-item>
    <ion-item>
      <h3>San Diego, CA</h3>
      <p>San Diego é uma cidade dos Estados Unidos localizada no sul do estado da Califórnia.</p>
    </ion-item>
  </ion-list>
</ion-content>
```

O Ionic tem algumas tratativas fixas para as listas e, quando necessário, é preciso customizar via CSS demais conteúdos e elementos. Veremos sobre customização de elementos no próximo ebook como informado no final deste material.

O resultado do código acima será este:



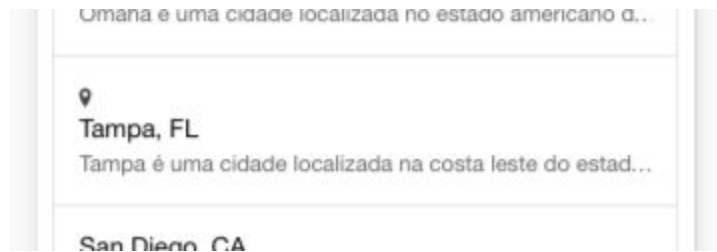
Podemos adicionar um separador entre os itens adicionando a classe `item-divider` no item que será o separador:

```
<ion-item class="item-divider">
  Cidades do Brasil
</ion-item>
<ion-item>
  Rio de Janeiro
</ion-item>
<ion-item>
  Florianópolis
</ion-item>
```

Dentro de cada item pode ser adicionado os principais componentes do Ionic, mas para eles serem exibidos da melhor forma precisamos adicionar as classes que irão tratar sua exibição. Vamos começar pelos ícones, adicionando o elemento `<i>` dentro de um dos itens:

```
<ion-item>
  <i class="icon ion-location"></i>
  <h2>Tampa, FL</h2>
  <p>Tampa é uma cidade localizada na costa ...</p>
</ion-item>
```

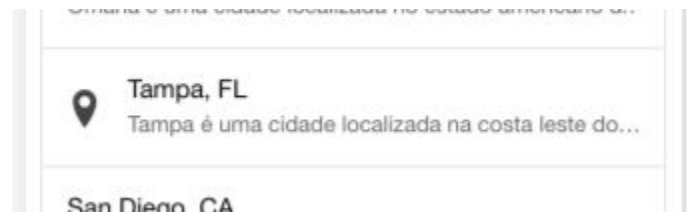
O resultado desta alteração ficou assim:



Podemos observar que o ícone ficou pequeno, e não ficou ao lado direito, como é comum ser utilizado. Agora vamos adicionar a classe `item-icon-left`, que altera o estilo do ícone dentro do item ficando assim:

```
<ion-item class="item-icon-left">
  <i class="icon ion-location"></i>
  <h2>Tampa, FL</h2>
  <p>Tampa é uma cidade localizada na costa ...</p>
</ion-item>
```

Agora podemos visualizar a exibição correta:



Você irá perceber, com o tempo, que a maioria dos elementos do Ionic são customizados desta forma, ou seja, adicionando classes para compor com outros elementos.

Alterando a classe do item para `item-icon-right` o ícone passa a ser exibido na direita:

```
<ion-item class="item-icon-right">
```

Destacar uma determinada área é comum de ver em listas, em Ionic chamamos esse atributo de Badge. Altere outro item adicionando um `<span>` com as classes abaixo:

```
<ion-item>
  <h2>Denver, CO</h2>
  <p>Omaha é uma cidade localizada no estado ...</p>
  <span class="badge badge-assertive">3</span>
</ion-item>
```

Também é possível adicionar um texto informativo de um item, chamada de Note. Altere o item Rio de Janeiro que contem apenas um texto:

```

<ion-item>
  Rio de Janeiro
  <span class="item-note">
    Popular
  </span>
</ion-item>

```

Um item pode ser clicável, basta alterar o componente para `<i>` e adicionar a classe `item`. Vamos alterar um dos itens deixando o código assim:

```

<a class="item">
  <h2>Colorado Springs, CO</h2>
  <p>Colorado Springs é uma cidade norte-americana ...</p>
</a>

```

Perceba que quando clicamos neste item um efeito de fundo cinza é exibido.



Vamos inserir um botão em um dos itens que tem apenas um texto, adicionando a classe `item-button-left` ou `item-button-right`:

```

<ion-item class="item-button-right">
  Florianópolis
  <button class="button button-positive">
    <i class="icon ion-ios-telephone"></i>
  </button>
</ion-item>

```

Nossa lista deve apresentar algo como:

Colorado Springs, CO	
Colorado Springs é uma cidade norte-american..	
Denver, CO	3
Omaha é uma cidade localizada no estado ...	
Denver, CO	
Omaha é uma cidade localizada no estado ame..	
 Tampa, FL	
Tampa é uma cidade localizada na costa ...	
San Diego, CA	
San Diego é uma cidade dos Estados Unidos L...	
Cidades do Brasil	
Rio de Janeiro	Popular
Florianópolis	



As listagem que aprendemos acima é bem utilizada para telas de parâmetros, formulários e ações, vamos agora ver listagem com exibição de imagens. Para isso vamos criar um novo projeto chamado `appListImage`:

```
ionic start appListImage blank
```

Na pasta `img` do seu projeto adicione três imagens diferentes para usarmos como exemplo, você pode usar as imagens utilizadas neste material que estão no [GitHub](#). Salve todas as imagens na pasta `img`.

Na área de conteúdo adicione a lista abaixo e três itens com a classe `item-avatar`, para que a exibição das imagens sejam arredondadas:

```
<ion-content>
  <ion-list>
    <ion-item class="item-avatar">
      
      <h2>San Francisco, CA</h2>
      <p>São Francisco é a quarta cidade mais populosa do estado da Califórnia.</p>
    </ion-item>
    <ion-item class="item-avatar">
      
      <h2>Lexington, KY</h2>
      <p>Lexington é a segunda cidade mais populosa do estado americano do
Kentucky.</p>
    </ion-item>
    <ion-item class="item-avatar">
      
      <h2>Washington, DC</h2>
      <p>Washington, D.C. é a capital dos Estados Unidos.</p>
    </ion-item>
  </ion-list>
</ion-content>
```

O resultado deste componente será este:



Outra opção para lista com imagem é utilizar a classe `item-thumbnail-left`, para adicionar uma imagens maior e sem arredondamento ao lado esquerdo, ou o `item-thumbnail-right` para adicionar ao lado direito. Também podemos adicionar a classe `item-text-wrap` para o texto não ser limitado com reticências:

Adicione dois item em nossa lista como thumbnail:

```
<ion-item class="item-thumbnail-left">
  
  <h2>Victoria, BC</h2>
  <p>Victoria é a capital da província canadense de Colúmbia Britânica.</p>
</ion-item>
<ion-item class="item-thumbnail-right item-text-wrap">
  
  <h2>Sacramento, CA</h2>
  <p>Sacramento é a capital do estado norte-americano da Califórnia e sede do
condado de Sacramento.</p>
</ion-item>
```

O resultado esperado:



## Cards

É comum ver este componente sendo utilizado nas aplicações mais recentes, pois com ele podemos exibir informações com um design lúdico de um cartão. Embora dentro do Card pode ser adicionado qualquer estrutura de elementos, é indicado utilizar itens como utilizamos em listas.

Crie um novo aplicativo com o nome appCard:

```
ionic start appCard blank
```

Adicione o componente Card e um item de texto sem limitação:

```

<ion-content>
  <div class="card">
    <ion-item class="item-text-wrap">
      Este é um simples Card com um texto para visualizar o comportamento do
      componente.
    </ion-item>
  </div>
</ion-content>

```

O resultado:



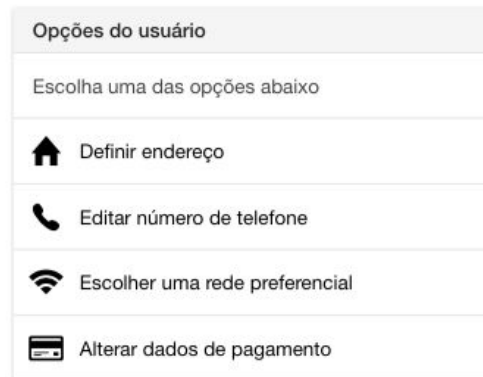
Você pode adicionar a classe `list` em um card para ele se comportar igual a uma lista, desta forma podemos utilizar qualquer componente de lista em nosso card. Veja um exemplo com diferente classes de itens:

```

<div class="list card">
  <div class="item item-divider">
    Opções do usuário
  </div>
  <div class="item item-text-wrap">
    Escolha uma das opções abaixo
  </div>
  <a href="#" class="item item-icon-left">
    <i class="icon ion-home"></i>
    Definir endereço
  </a>
  <a href="#" class="item item-icon-left">
    <i class="icon ion-ios-telephone"></i>
    Editar número de telefone
  </a>
  <a href="#" class="item item-icon-left">
    <i class="icon ion-wifi"></i>
    Escolher uma rede preferencial
  </a>
  <a href="#" class="item item-icon-left">
    <i class="icon ion-card"></i>
    Alterar dados de pagamento
  </a>
</div>

```

O resultado:



Outro exemplo de itens combinados:

```
<div class="list card">
  <div class="item item-avatar">
    
    <h2>Fábio Rogério SJ</h2>
    <p><b>32min</b> - Compartilhou uma foto </p>
  </div>
  <div class="item item-body">
    
    <p> Pensa em um lugar show! Recomendo todos irem visitar a grande Maringá no
Paraná :)</p>
    <p>
      <a href="#" class="subdued">1 Curtida</a>
      <a href="#" class="subdued">5 Comentários</a>
    </p>
  </div>
</div>
```

O resultado:



## Forms

Formulários são componentes essenciais para diferentes aplicativos e também um grande desafio, pois pesquisas revelam que usuários de aplicativos móveis não gostam de preencher formulários grandes em seus *devices*.

Ionic fornece uma estrutura de formulários que são utilizados com as listas e itens, sendo assim cada campo é um item de uma lista composto por um `label`. Como os outros componentes estes também podem ser combinados com os demais componentes do *framework*.

É importante definir corretamente o tipo do campo, para que o teclado virtual seja customizado, dependendo do tipo. Para saber quais tipos são suportados veja na [documentação oficial do Ionic Framework](#) ou outros documentos sobre tipos em HTML5.

Vamos aos códigos, crie um novo aplicativo chamado `appForms`:

```
ionic start appForms blank
```

Na área de conteúdo vamos criar três campos e um botão. Cada campo deve estar dentro de um `label` que recebe a classe `item` e `item-input` para que os campos ocupem 100% do espaço do item da lista e definimos um valor para a propriedade `placeholder`, do HTML5, para quando um valor for digitado no campo o rótulo oculte. O botão está em uma outra `div` com a classe `padding` para não ficar colado nas laterais.

```
<ion-list>
  <label class="item item-input">
    <input type="text" placeholder="Nome completo">
  </label>
  <label class="item item-input">
    <input type="email" placeholder="E-mail">
  </label>
  <label class="item item-input">
    <textarea placeholder="Comentário"></textarea>
  </label>
</ion-list>
<div class="padding">
  <button class="button button-block button-positive">
    Enviar comentário
  </button>
</div>
```

O resultado esperado é:



Podemos adicionar um rótulo fixo ao lado esquerdo adicionando um span com a classe `input-label`:

```
<label class="item item-input">
  <span class="input-label">Username</span>
  <input type="text">
</label>
```

Para deixar um rótulo fixo no topo adicione a classe `item-stacked-label` no item:

```
<label class="item item-input item-stacked-label">
  <span class="input-label">Primeiro nome</span>
  <input type="text" placeholder="Fábio">
</label>
```

Outra opção interessante para os rótulos do topo é utilizar a classe `item-floating-label` para que o rótulo fique flutuando:

```
<label class="item item-input item-floating-label">
  <span class="input-label">Sobrenome</span>
  <input type="text" placeholder="Digite seu sobrenome">
</label>
```

Os campos podem ser agrupados com botões:

```
<div class="item item-input-inset">
  <label class="item-input-wrapper">
    <input type="email" placeholder="Email">
  </label>
  <button class="button button-small button-balanced">
    Enviar
  </button>
</div>
```

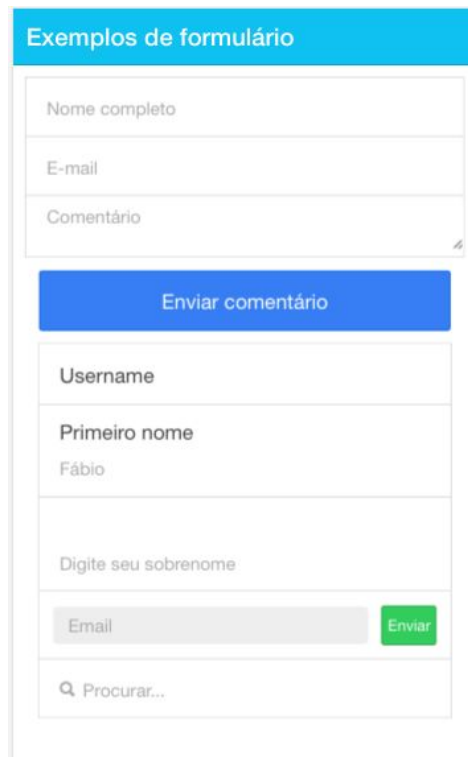
Também é possível adicionar ícones:

```

<label class="item item-input">
  <i class="icon ion-search placeholder-icon"></i>
  <input type="text" placeholder="Procurar...">
</label>

```

Nosso formulário de exemplo deverá está parecido com o resultado abaixo:



Exemplos de formulário

Nome completo

E-mail

Comentário

Enviar comentário

Username

Primeiro nome

Fábio

Digite seu sobrenome

Email

Enviar

Procurar...

## Toggle

Toggle é um componente criado especificamente para aplicativos móveis, hoje também podemos ver em aplicações web. Crie um novo projeto com o nome appToggle:

```
ionic start appToggle blank
```

Vamos criar uma lista de opções sendo cada item um toggle marcado com a classe `item-toggle`, sendo assim o texto fica ao lado esquerdo e o botão toggle ao lado direito. As cores de cada toggle pode ser customizado conforme os temas do Ionic.

Para complementar nosso exemplo vamos adicionar um card como pergunta:

```

<ion-content class="padding">
  <div class="card padding">
    Quais tecnologias você esta aprendendo?
  </div>
</ion-list>

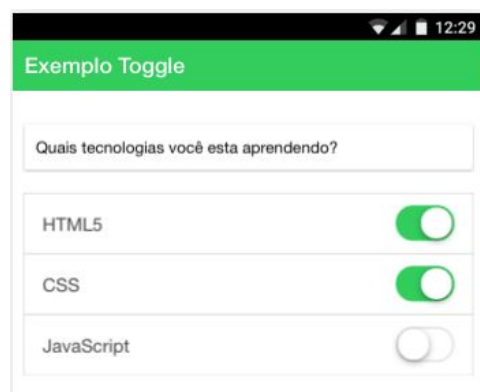
```

```

<ion-item class="item-toggle">
  HTML5
  <label class="toggle toggle-balanced">
    <input type="checkbox">
    <div class="track">
      <div class="handle"></div>
    </div>
  </label>
</ion-item>
<ion-item class="item-toggle">
  CSS
  <label class="toggle toggle-balanced">
    <input type="checkbox">
    <div class="track">
      <div class="handle"></div>
    </div>
  </label>
</ion-item>
<ion-item class="item-toggle">
  JavaScript
  <label class="toggle toggle-balanced">
    <input type="checkbox">
    <div class="track">
      <div class="handle"></div>
    </div>
  </label>
</ion-item>
</ion-list>
</ion-content>

```

O resultado:



## Checkbox

Os checkbox permitem que o usuário selecione um número de itens em um conjunto de opções. Crie um novo projeto chamado appCheckbox:

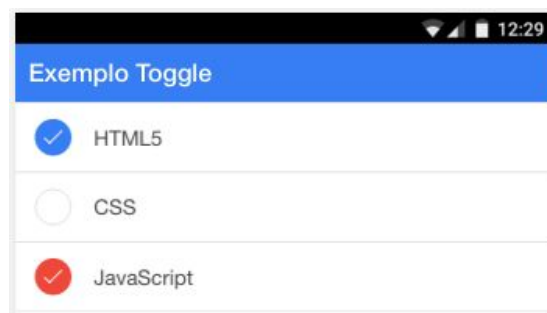


```
ionic start appCheckbox blank
```

Este componente também usa como base o componente lista onde cada item recebe um checkbox. Você pode customizar as cores utilizando os temas, para exemplo vamos deixar o último item com uma cor diferente:

```
<ion-list>
  <ion-item class="item-checkbox">
    <label class="checkbox">
      <input type="checkbox">
    </label>
    HTML5
  </ion-item>
  <ion-item class="item-checkbox">
    <label class="checkbox">
      <input type="checkbox">
    </label>
    CSS
  </ion-item>
  <ion-item class="item-checkbox">
    <label class="checkbox checkbox-assertive">
      <input type="checkbox">
    </label>
    JavaScript
  </ion-item>
</ion-list>
```

O resultado:



## Radio Buttons

Radio Buttons, ao contrario dos checkbox, permite selecionar apenas uma opção dentre a listagem enquanto os checkbox permitem selecionar mais de uma opção. Crie um novo projeto chamado appRadioButtons:

```
ionic start appRadioButtons blank
```

Vamos criar um subHeader e uma lista, na lista vamos adicionar alguns radio buttons. o body da página deverá estar assim:

```
<body ng-app="starter">
  <ion-pane>
    <ion-header-bar class="bar-positive" align-title="center">
      <h1 class="title">Exemplo Radio Buttons</h1>
    </ion-header-bar>
    <ion-header-bar class="bar-subheader bar-calm" align-title="center">
      <h1 class="title">Qual tecnologia você mais domina?</h1>
    </ion-header-bar>
    <ion-content>
      <ion-list>
        <ion-radio>PHP</ion-radio>
        <ion-radio>Java</ion-radio>
        <ion-radio>JavaScript</ion-radio>
        <ion-radio>.NET</ion-radio>
        <ion-radio>Ruby</ion-radio>
        <ion-radio>Python</ion-radio>
        <ion-radio>Go</ion-radio>
      </ion-list>
    </ion-content>
  </ion-pane>
</body>
```

O resultado:



## Range

O componente Range é utilizado para definir um intervalo de valores com o clicar e arrastar do toque. É possível customizar seus ícones e cores.

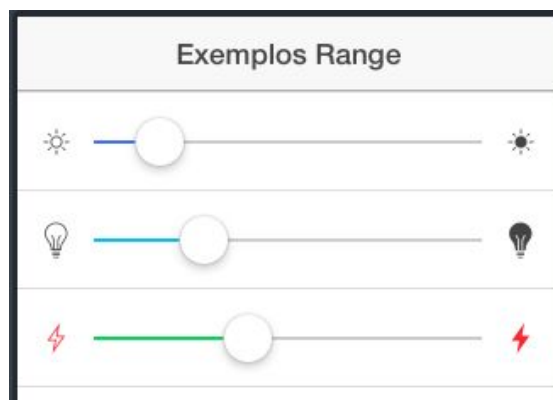
Crie um novo projeto com o nome appRange:

```
ionic start appRange blank
```

Adicione na área de conteúdo algumas opção com cores diferente para exemplificar o componente:

```
<ion-list>
  <ion-item class="range range-positive">
    <i class="icon ion-ios-sunny-outline"></i>
    <input type="range" min="0" max="100" value="12">
    <i class="icon ion-ios-sunny"></i>
  </ion-item>
  <ion-item class="range range-calm">
    <i class="icon ion-ios-lightbulb-outline"></i>
    <input type="range" min="0" max="100" value="25">
    <i class="icon ion-ios-lightbulb"></i>
  </ion-item>
  <ion-item class="range range-balanced">
    <i class="icon ion-ios-bolt-outline assertive"></i>
    <input type="range" min="0" max="100" value="38">
    <i class="icon ion-ios-bolt assertive"></i>
  </ion-item>
</ion-list>
```

O resultado:



## Select

O componente de seleção listada é muito utilizado em sistemas desktop e web, o Ionic não customiza muito este componente pois os próprios *browsers*, de cada sistema operacional móvel, já customizam a visualização com o seu padrão. Neste momento vamos testar apenas no *browser* do desktop.

Crie um novo projeto com o nome appSelect:

```
ionic start appSelect blank
```

Adicione uma lista com um item contendo um input de select:

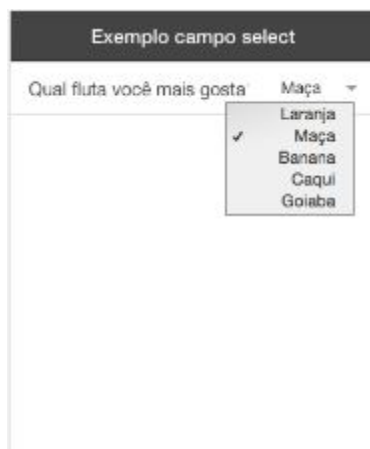
```

<ion-content>
  <ion-list>
    <ion-item class="item-input item-select">
      <div class="input-label">
        Qual fluta você mais gosta?
      </div>
      <select>
        <option>Laranja</option>
        <option selected>Maça</option>
        <option>Banana</option>
        <option>Caqui</option>
        <option>Goiaba</option>
      </select>
    </ion-item>
  </ion-list>
</ion-content>

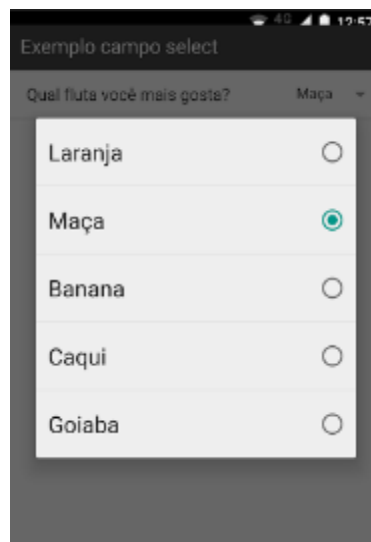
```

O resultado nos diferentes *devices*:

No *browser desktop*:



No Android:



No iOS:



# Tabs

O componente tabs permite uma navegação entre telas simulando uma seleção de abas. Muitos aplicativos em iOS utilizam este padrão de navegação. Crie um novo projeto chamado appTabs:

```
ionic start appTabs blank
```

Este componente é melhor utilizado com as rotas em AngularJS, mas como neste material veremos apenas o essencial dos componentes então vamos criar algo mais simples apenas para conhecer o componente.

As tabs podem ser apenas texto, apenas ícone ou texto e ícone. Também podemos escolher a posição do ícone e as cores conforme temas do ionic.

Cada tab tem sua própria área de conteúdo, então vamos adicionar três tabs com texto e ícone, e vamos definir diferentes ícones para quando a aba estiver selecionada ou não. Vamos ao código:

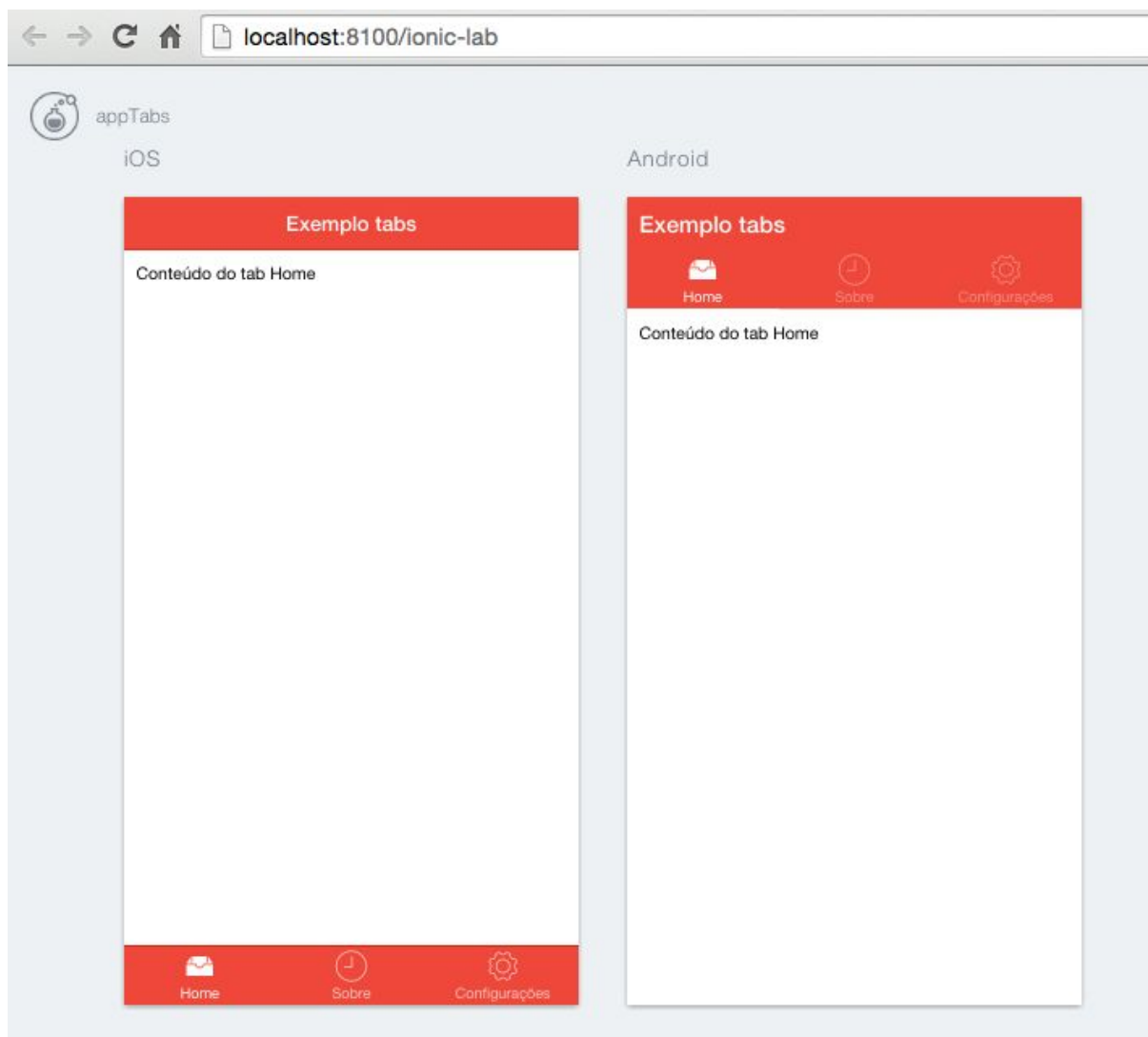
```
<body ng-app="starter">
  <ion-pane>
    <ion-header-bar class="bar-assertive">
      <h1 class="title">Exemplo tabs</h1>
    </ion-header-bar>
    <ion-tabs class="tabs-assertive tabs-icon-top">
      <ion-tab title="Home" icon-on="ion-ios-filing"
icon-off="ion-ios-filing-outline">
        <ion-view>
          <ion-content class="padding">
            Conteúdo do tab Home
          </ion-content>
        </ion-view>
      </ion-tab>
      <ion-tab title="Sobre" icon-on="ion-ios-clock"
icon-off="ion-ios-clock-outline">
        <ion-view>
          <ion-content class="padding">
            Conteúdo do tab Sobre
          </ion-content>
        </ion-view>
      </ion-tab>
      <ion-tab title="Configurações" icon-on="ion-ios-gear"
icon-off="ion-ios-gear-outline">
        <ion-view>
          <ion-content class="padding">
            Conteúdo do tab Configurações
          </ion-content>
        </ion-view>
      </ion-tab>
    </ion-tabs>
  </ion-pane>
</body>
```

```
</ion-tab>
</ion-tabs>
</ion-pane>
</body>
```

Por ser um componente muito utilizada para navegação, o Android e iOS exibem diferente, podemos testar utilizando o Ionic lab, que tem por objetivo mostrar nos dois sistemas a aplicação. Para utilizar o Ionic lab digite `--lab` no final do comando para testar o aplicativo no *browser*:

```
ionic serve --lab
```

Será exibido desta forma no seu *browser*:



## Grid

As grids em Ionic são simples e fáceis de implementar, pois o componente utiliza o recurso [FlexBox do CSS3](#) para desenhar os elementos. Crie um novo projeto chamado appGrid:

```
ionic start appGrid blank
```

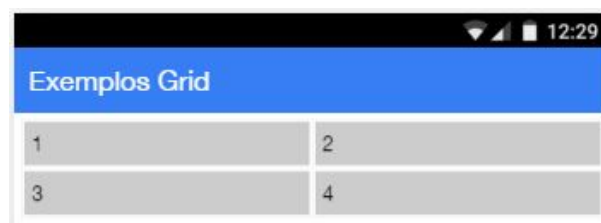
Vamos criar uma grid simples com duas linhas e duas colunas, adicione na área de conteúdo o componente:

```
<ion-content>
  <div class="row">
    <div class="col">1</div>
    <div class="col">2</div>
  </div>
  <div class="row">
    <div class="col">3</div>
    <div class="col">4</div>
  </div>
</ion-content>
```

Para ser possível verificar com mais clareza nossos exemplos vamos alterar a cor de fundo de cada coluna e adicionar uma borda, editando o style.css da pasta css:

```
/* Empty. Add your own CSS if you like */
.col {
  background-color: #CCC;
  border: 2px solid #fff;
}
```

Podemos ver o resultado:

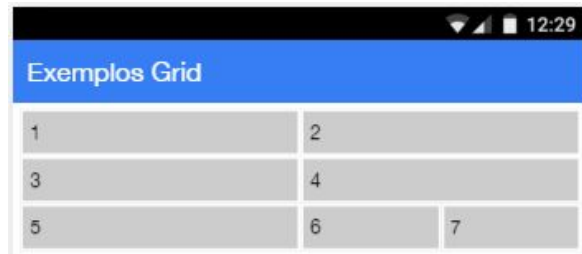


É possível definir um tamanho fixo para uma determinada coluna. Adicione uma nova linha:

```
<div class="row">
  <div class="col col-50">5</div>
  <div class="col">6</div>
  <div class="col">7</div>
```

</div>

Perceba que a primeira coluna está ocupando 50% do espaço da linha, e as demais colunas, que estão sem tamanho fixo, ocupam o espaço que sobrou:



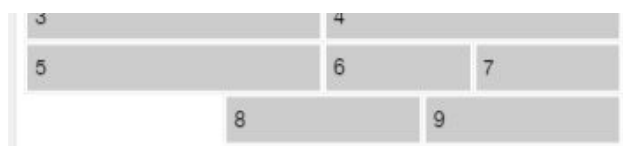
Os tamanhos possível podem ser vistos na tabela abaixo:

Classe	Tamanho ocupado
.col-10	10%
.col-20	20%
.col-25	25%
.col-33	33.3333%
.col-50	50%
.col-67	66.6666%
.col-75	75%
.col-80	80%
.col-90	90%

Outro recurso é deixar uma coluna em branco, ou seja, reservar o espaço porem não exibir nenhum elemento, adicione uma nova linha:

```
<div class="row">  
  <div class="col col-33 col-offset-33">8</div>  
  <div class="col">9</div>  
</div>
```

Resultado:



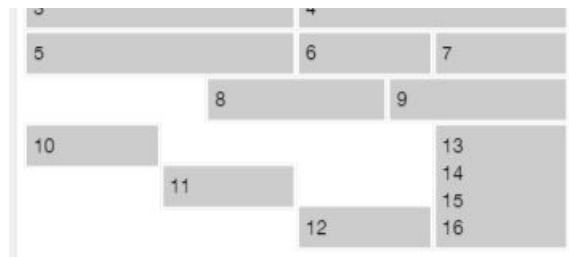


Os tamanhos para os espaços em branco são os mesmos da tabela acima, porém o nome deve ser `col-offset-TAMANHO`.

Quando temos uma coluna com altura variável, podemos definir como será o alinhamento vertical sendo no topo, centro ou no rodapé. Adicione uma nova linha:

```
<div class="row">
  <div class="col col-top">10</div>
  <div class="col col-center">11</div>
  <div class="col col-bottom">12</div>
  <div class="col">13<br>14<br>15<br>16</div>
</div>
```

Veja os alinhamentos:



Caso o alinhamento for para a linha inteira adicione a classe direto no `row` ao invés de adicionar na `col` desta forma:

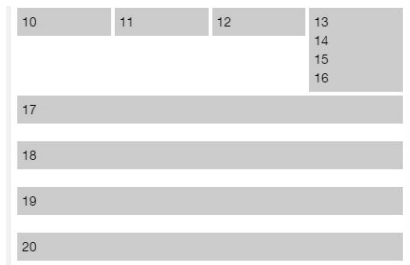
```
<div class="row row-top">
  <div class="col">10</div>
  <div class="col">11</div>
  <div class="col">12</div>
  <div class="col">13<br>14<br>15<br>16</div>
</div>
```

Utilizar números fixos de colunas pode não ser apropriado se você precisa de um layout responsivo, nestes casos você pode utilizar três classes para tamanhos diferentes. Vamos ver isso na prática, adicione uma nova linha com quatro colunas:

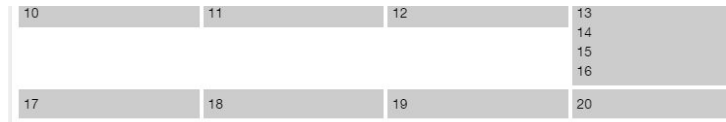
```
<div class="row responsive-sm">
  <div class="col">17</div>
  <div class="col">18</div>
  <div class="col">19</div>
  <div class="col">20</div>
</div>
```

Vejam no resultado que o tamanho da tela é validado e conforme existe mais espaço as colunas são exibidas lado a lado, caso seja menor elas são exibidas uma em baixo da outra.

Tela menor:



Tela maior:



Os tamanhos validados podem ser visto na tabela a baixo juntamente com suas respectivas classes de utilização:

Classe	Quebrar quando
.responsive-sm	For menor que telefone em visualização modo paisagem
.responsive-md	For menor que tablet em visualização modo retrato
.responsive-lg	For menor que tablet em visualização modo paisagem

Estas são classes criadas pelo *framework*, mas nada impede de você criar seus próprios tamanhos e validação de responsividade.

## Conclusão sobre componentes

Aprendemos e testamos todos os componentes, de HTML e CSS, do Ionic framework, agora você tem uma visão ampla do que é possível fazer com os componentes já prontos do Ionic, mas lembre-se, você pode, e deve, criar seus próprios componentes com CSS.

Em outro ebook iremos abordar customização de componentes, bem como a criação de novos estilos.

Você pode consultar a documentação oficial dos componentes do Ionic Framework no link: <http://ionicframework.com/docs/components/>

# Empacotando o aplicativo

O processo de *build*, também conhecido como empacotamento, é utilizado para testar e publicar o aplicativo final.

Neste material vamos abordar apenas os comandos para gerar o aplicativo para testar nos diferentes *devices*. Para publicar nas *Stores* iremos ver em outro ebook.

## Android

Para criar o aplicativo final na plataforma Android precisamos do Android SDK, que é todas as ferramentas para criar o aplicativo final (.apk) e poder ser testado em emuladores e *devices*.

Primeiro passo é baixar o Android SDK no site oficial do Android Developer (<http://developer.android.com/intl/pt-br/sdk/index.html>). Role até o final da página e baixe apenas o SDK para linha de comando, como mostrado na imagem abaixo. Escolha o seu sistema operacional e faça o download:

### Get just the command line tools

If you do not need Android Studio, you can download the basic Android command line tools below.

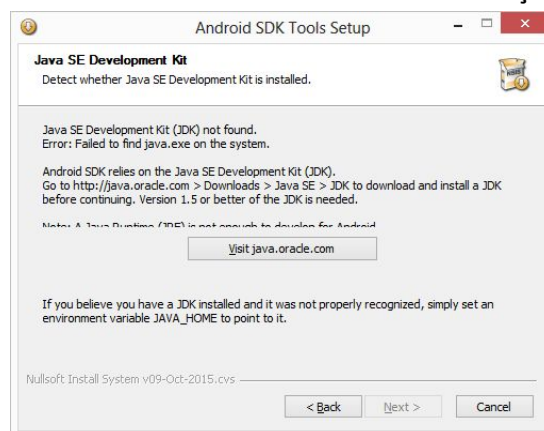
Platform	SDK tools package	Size	SHA-1 checksum
Windows	<a href="#">installer_r24.4.1-windows.exe</a>	144 MB (151659917 bytes)	f9b59d72413649d31e633207e31f456443e7ea0b
	<a href="#">android-sdk_r24.4.1-windows.zip</a> No installer	190 MB (199701062 bytes)	66b6a6433053c152b22bf8cab19c0f3fef4eba49
Mac OS X	<a href="#">android-sdk_r24.4.1-macosx.zip</a>	98 MB (102781947 bytes)	85a9cccb0b1f9e6f1f616335c5f07107553840cd
Linux	<a href="#">android-sdk_r24.4.1-linux.tgz</a>	311 MB (326412652 bytes)	725bb360f0f7d04eacff5a2d57abdd49061326d

Para usuários Linux o processo é bem parecido com algumas diferenças nas definições de variáveis de ambiente. Neste material vamos abordar a instalação do Android SDK em ambiente Windows.

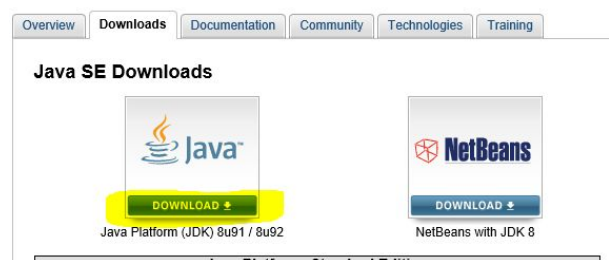
Para usuários do Windows podemos usar o instalador, onde já é verificado se existe o Java JDK instalado, e instalar o SDK em uma determinada pasta. Execute o instalador e siga os passos abaixo:



Se você tiver o Java JDK instalado o instalador irá seguir para a instalação do Android SDK, caso contrario será exibido esta tela solicitando a instalação do Java JDK:

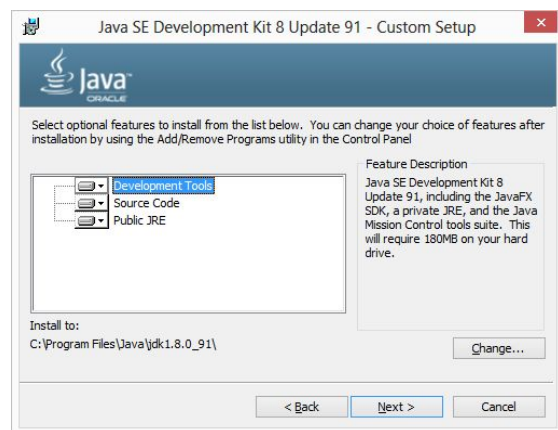


Clique em Visit `java.oracle.com` e siga os passos abaixo:

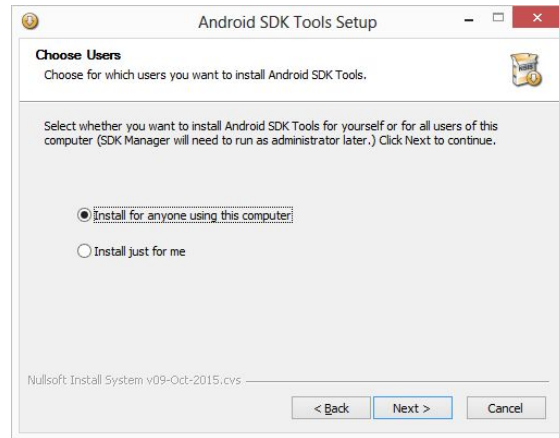
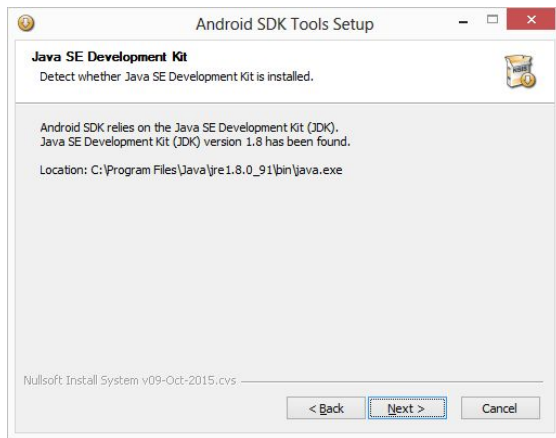


Java SE Development Kit 8u91		
You must accept the <a href="#">Oracle Binary Code License Agreement for Java SE</a> to download this software.		
<input type="radio"/> Accept License Agreement <input checked="" type="radio"/> Decline License Agreement		
Product / File Description	File Size	Download
Linux ARM 32 Hard Float ABI	77.72 MB	<a href="#">jdk-8u91-linux-arm32-vfp-hflt.tar.gz</a>
Linux ARM 64 Hard Float ABI	74.69 MB	<a href="#">jdk-8u91-linux-arm64-vfp-hflt.tar.gz</a>
Linux x86	154.74 MB	<a href="#">jdk-8u91-linux-i586.rpm</a>
Linux x86	174.92 MB	<a href="#">jdk-8u91-linux-i586.tar.gz</a>
Linux x64	152.74 MB	<a href="#">jdk-8u91-linux-x64.rpm</a>
Linux x64	172.97 MB	<a href="#">jdk-8u91-linux-x64.tar.gz</a>
Mac OS X	227.29 MB	<a href="#">jdk-8u91-macosx-x64.dmg</a>
Solaris SPARC 64-bit (SVR4 package)	139.59 MB	<a href="#">jdk-8u91-solaris-sparcv9.tar.Z</a>
Solaris SPARC 64-bit	98.95 MB	<a href="#">jdk-8u91-solaris-sparcv9.tar.gz</a>
Solaris x64 (SVR4 package)	140.29 MB	<a href="#">jdk-8u91-solaris-x64.tar.Z</a>
Solaris x64	96.78 MB	<a href="#">jdk-8u91-solaris-x64.tar.gz</a>
Windows x86	182.11 MB	<a href="#">jdk-8u91-windows-i586.exe</a>
Windows x64	187.41 MB	<a href="#">jdk-8u91-windows-x64.exe</a>

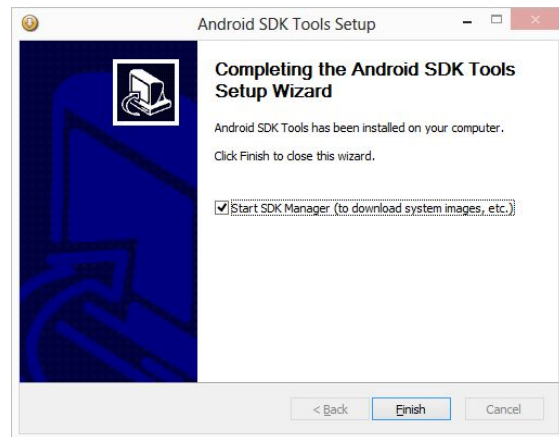
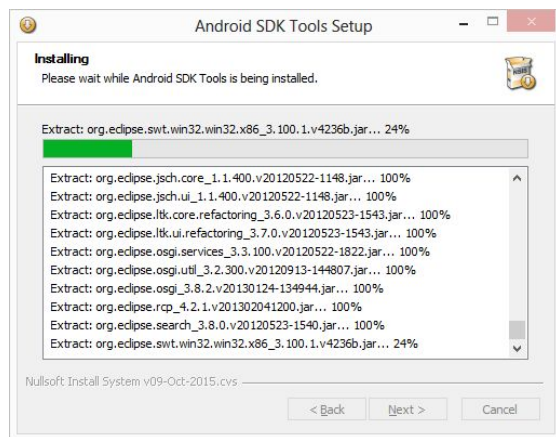
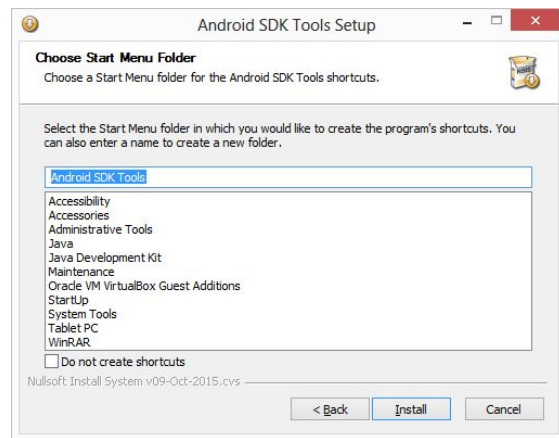
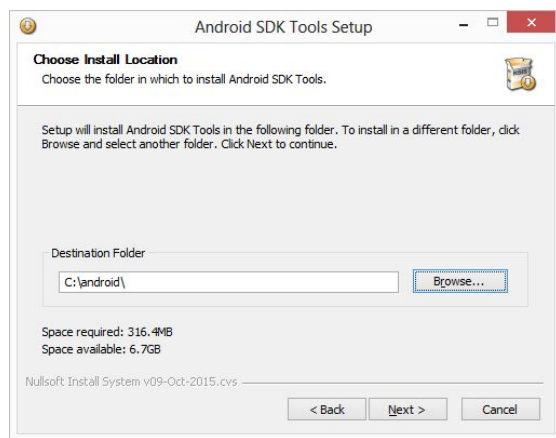
Após o download concluído siga os passos abaixo para instalar o Java JDK e suas dependências:



Ao concluir a instalação do Java JDK, execute novamente o instalador do Android, ou se ainda estiver aberto clique no botão **Back** e depois **Next**, você verá que agora ele irá encontrar a instalação do Java e assim podemos prosseguir com a instalação do Android SDK clicando em **Next**:

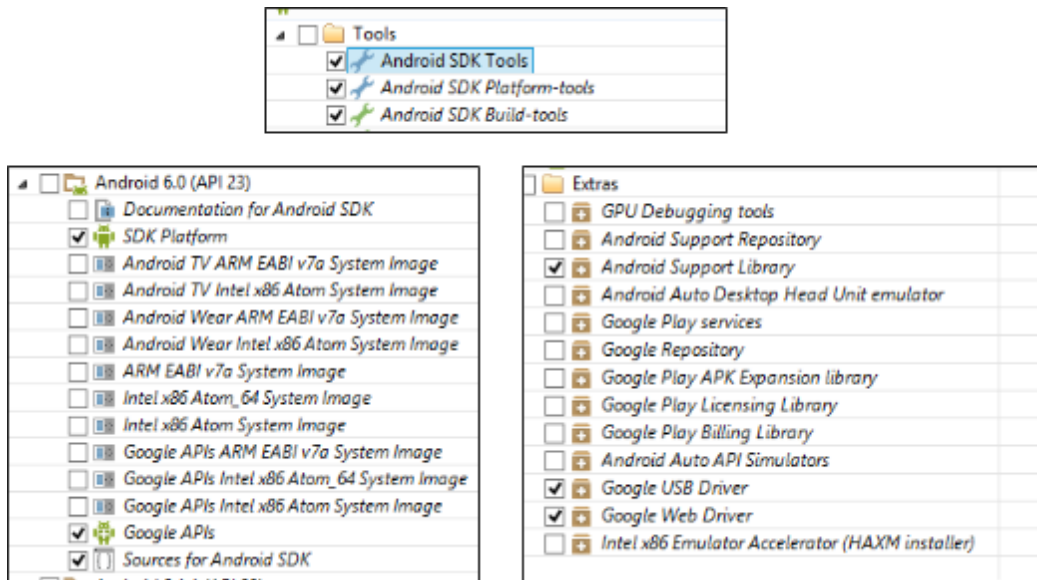


Escolha um caminho de fácil acesso para a instalação, pois em alguns momentos você terá que entrar nesta pasta para executar o *Update* do Android:

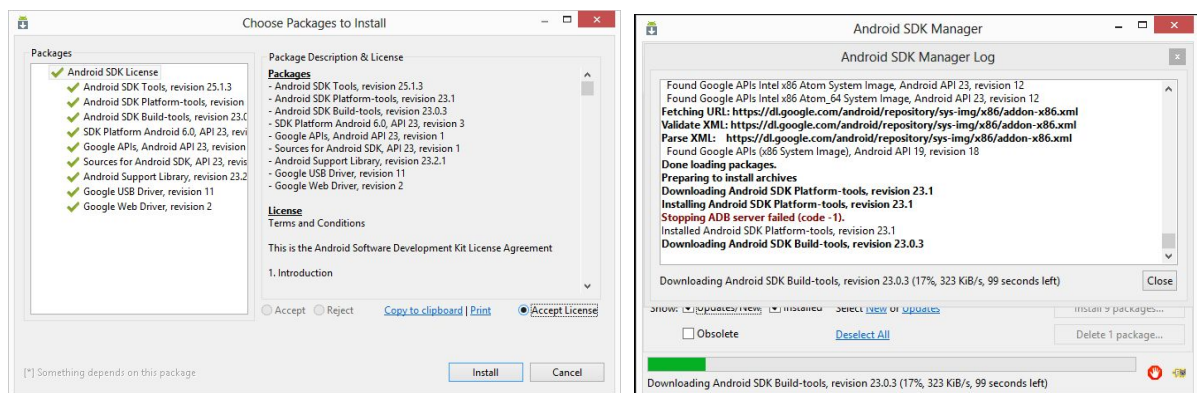


O Ionic precisa de algumas dependências do Android para fazer *build*, ou seja, gerar o aplicativo final, para isso vamos fazer alguns *Update* no Manager Android SDK. No Manager desmarque tudo que já vem marcado e marque apenas as opções listadas abaixo. Em seguida clique em **Install 9 packages**:

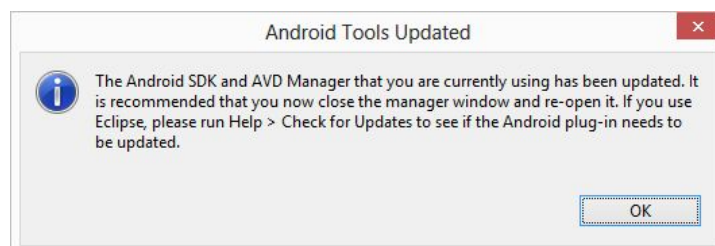




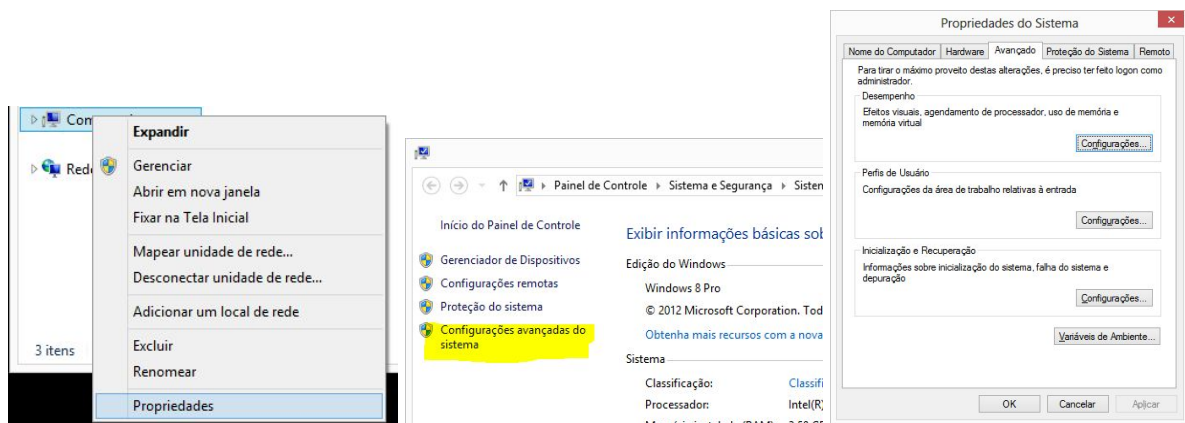
Em seguida clique em **Accept License** e **Install** para o download começar. Este processo pode demorar alguns minutos.



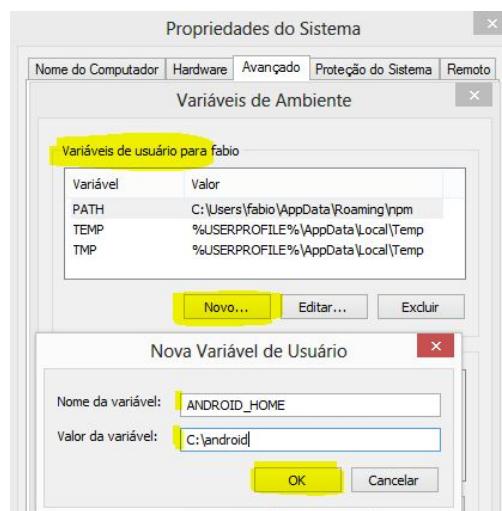
Ao concluir a tela abaixo será apresentada:



Vamos agora adicionar algumas variáveis de ambiente para que o Ionic encontre o local onde está instalado o Android SDK. Entre nas **propriedades** do seu computador e vai em **Configurações avançadas do sistema** e **Variáveis de Ambiente**:



Em variáveis do usuário adicione uma nova chamada ANDROID\_HOME e em valor preencha com o caminho onde foi feito a instalação do SDK:



Agora vamos adicionar a plataforma android em nosso projeto e fazer o *build*. Se seu terminal estiver aberto feche ele e abra novamente para que as variáveis de ambiente sejam carregadas. Em seguida digite os comandos dentro da pasta de algum projeto criado nos exemplos acima:

```
ionic platform add android
ionic build android
```

Se o build ocorrer correto uma mensagem será exibida em seu terminal parecida com essa:

```
BUILD SUCCESSFUL
Total time: 2 mins 57.657 secs
```



Built the following apk(s):

C:/workspace/app1/platforms/android/build/outputs/apk/android-debug.apk

Na ultima linha é apresentado o caminho onde foi gerado o aplicativo final, no caso o android-debug.apk. Para testar em seu *device* basta instalar este arquivo nele.

Se você se deparar com algum erro deixe sua mensagem nesta [lista de discussão](#).

Para testar em emuladores ou até mesmo fazer a instalação direto em seu *device*, sem precisar copiar o arquivo apk, [leia este post](#) em meu site com várias opções e formas.

## iOS

Para gerar o aplicativo final para a plataforma iOS (.ipa), utilizando o Ionic Framework, precisamos estar em um ambiente Mac, ou seja, em uma máquina com o sistema operacional iOS.

O Ionic.io, que é a plataforma de soluções em nuvem para aplicativos implementados em Ionic Framework, fornece uma solução para compilar o projeto para iOS sem precisar de uma máquina Mac, porém este serviço é *free* com limitações. Iremos falar sobre o Ionic.io em outro ebook.

Para compilar nosso projeto vamos precisar do `ios-sim`, um módulo NPM escrito em JavaScript rodando via NodeJS responsável por rodar aplicações via linha de comando direto no emulador iOS.

Para instalar digite:

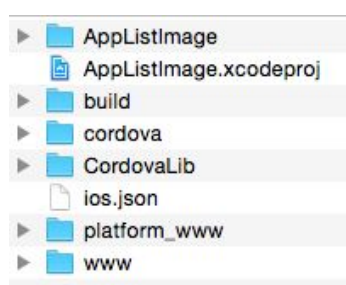
```
npm install -g ios-sim
```

Em seguida, dentro da pasta do projeto, digite:

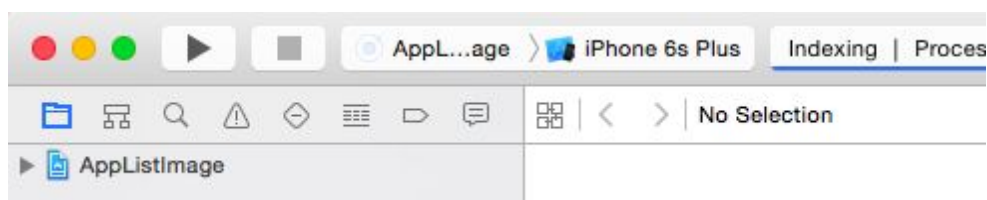
```
ionic build ios
```

Este processo irá compilar nosso aplicativo e agora podemos abri-lo via Xcode para emular ou rodar em um *device* com iOS.

Dentro da pasta `platforms/ios` foi criado um arquivo `AppListImage.xcodeproj`, que é o arquivo inicializador do projeto em Xcode:

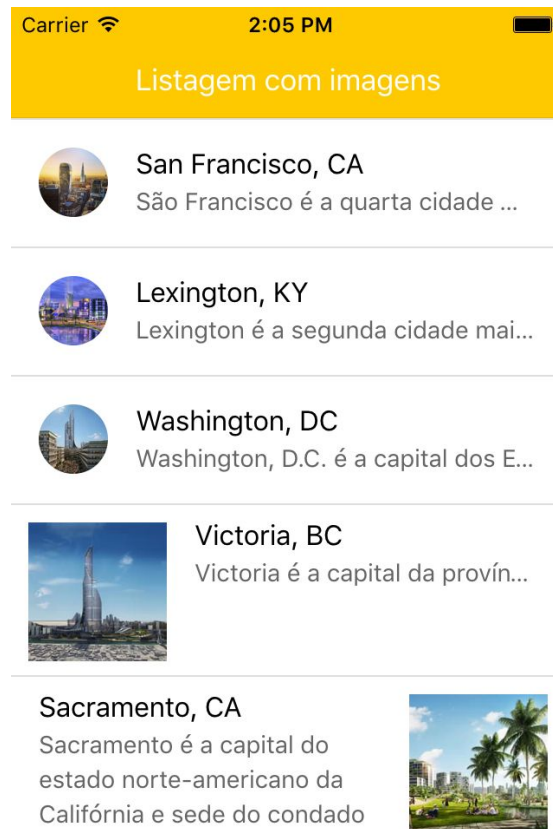


De um duplo clique neste arquivo e o projeto será aberto no Xcode:



Ao abrir basta escolher um emulador, ou um *device* conectado via USB, onde está "iPhone 6s Plus" e clicar no ícone play.

Este processo irá executar o emulador, caso você escolha um dos emuladores, e em seguida irá instalar e rodar o aplicativo:



# Próximo passo

Este ebook abordou os recursos introdutórios e essenciais para se iniciar no Ionic Framework, agora vamos aprender como customizar os componentes para criar um design mais proprietário e melhorar a experiência do usuário.

Veja o [#2 Ionic Framework - Customizando e criando componentes](#):

