



CRIANDO MEU PRIMEIRO APLICATIVO

HÍBRIDO E MULTI-PLATAFORMA

por Fábio Rogério SJ

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, certamente não existe apenas uma resposta para esta pergunta, 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 desenvolvimento de aplicativos móveis utilizando tecnologias híbridas como HTML/CSS/Javascript e gerando o mesmo aplicativo para diversas plataformas como Android, iOS e Windows Phone. 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, exceto os cursos gerados a partir deste material devido a custos com gravação, edição, correções de exercícios, divulgação entre outros fatores que não consigo custear sozinho :(

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 o desenvolvimento de aplicativos móveis 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 como pesquisador e desenvolvedor em uma equipe de P&D na TecnoSpeed e ministra treinamentos em desenvolvimento de aplicações web e mobile. 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

Introdução	3
Apache Cordova	4
MobileUI	4
Instalando o ambiente	5
NodeJS	5
Sublime Text	5
Apache Cordova e MobileUI	6
Criando o primeiro app	7
Estrutura dos projetos Cordova	9
config.xml	9
www	10
Removendo códigos de exemplo inicial	10
Componentes	11
Layout básico de um app	11
Menu de navegação Springboard	12
Ícones	16
Navegando entre telas	19
Header transparente	23
Aplicativo nativo	26
Rodando o aplicativo no Android	26
Rodando o aplicativo no iOS	29
Conclusão	31

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.

Sabemos que o desenvolvimento de aplicativos móveis já não é novidade e está bem consolidado na comunidade de desenvolvedores. Entretanto, algumas discussões sempre ocorrem no momento de escolher qual plataforma desenvolver, e sem dúvidas essa resposta não tem apenas uma resposta.

Na escolha da plataforma de desenvolvimento levamos em consideração as nativas como: Android, iOS, Windows Phone entre outras. Quando escolhido este caminho o desenvolvedor precisa dominar as tecnologias de cada plataforma como Java para Android, Swift para iOS e C# para Windows Phone, sem dúvida este caminho é árduo e custoso. Outra escolha que muitos devs estão seguindo é o desenvolvimento híbrido onde é desenvolvido o aplicativo em HTML/CSS/JavaScript e o mesmo código pode ser gerado para Android, iOS, Windows Phone entre outros.

O desenvolvimento híbrido é uma boa abordagem para quando você precisa lançar rapidamente aplicativos móveis ao mercado e também ótimo para o ambiente corporativo como aplicativos que fazem parte de um sistema gerencial, CRM, ERP, etc. Outro ponto de análise é a manutenção, pois você irá precisar manter apenas uma arquitetura de código e o mesmo será gerado para diferentes plataforma utilizando o framework Apache Cordova.

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

E os componentes de interface precisa criar tudo do zero o HTML e CSS? Não, para isso temos ferramentas que já entregam os componentes prontos, como o MobileUI. O MobileUI é um repositório de componentes para aplicativos móveis híbridos, basta escolher os componentes como lista, botões, gráfico de barras, menu, tabs, e fazer a instalação no seu app.

O requisito mínimo para desenvolver aplicativos híbridos é ter um conhecimento básico em programação. Se tiver conhecimento em HTML, CSS e JavaScript sua evolução será mais rápida.

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, como descrito na introdução, também é *open source* e é utilizado para acessar as funcionalidades nativas dos aparelhos móveis como acelerômetro, câmera e geolocalização. 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>.

MobileUI

Quando falamos de aplicações híbridas estamos dizendo que podemos criar apps utilizando todo o poder da web como HTML/CSS/Javascript e poder utilizar também recursos nativos das plataformas. Entretanto, os componentes de interface como botões, listas, campos de entrada de dados, menus entre outros são implementados puramente com as tecnologias web e fazer isso do zero não é produtivo, para isso podemos usar o MobileUI.

MobileUI é um repositório de componentes de interface instaláveis a qualquer aplicação híbrida, ou seja, se você precisa usar menu, aba, lista, botões e campos basta você instalar na pasta do seu projeto e toda a estrutura de CSS já está pronta para ser utilizada. O MobileUI foi criado pensando em tornar sua aplicação híbrida mais rápida e menor, uma vez que você apenas instala o que você realmente vai usar para os elementos de interface.

Para quem já utiliza um framework que dispõe de elementos de interface, como Onsen UI, Ionic Framework entre outros é possível utilizar o MobileUI para os componentes específicos como gráficos, uma vez que o MobileUI instala em sua aplicação apenas as dependências necessária para o componente instalado.

MobileUI também é perfeito para criar aplicativos progressivos, conhecido como PWA (*Progressive Web Apps*) que é uma aposta para a nova geração de aplicativos móveis.

Instalando o ambiente

Todos os comandos do Apache Cordova, como criar um novo projeto, testar e distribuir, são feitos direto no terminal através de um CLI (*command-line interface*). O Apache Cordova utiliza o NodeJS para executar tarefas. O MobileUI também depende do NodeJS instalado e disponibiliza um CLI onde você irá instalar os componentes em seu projeto .

Tanto o Apache Cordova quanto o MobileUI podem ser instalados em Windows, Linux ou Mac, os passos seguintes são os mesmos para os três sistemas operacionais.

Você basicamente irá precisar o Apache Cordova, MobileUI e um editor de HTML.

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.

Sublime Text

Sublime Text é um editor de código simples, rápido e eficiente, porém 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>.

Apache Cordova e MobileUI

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

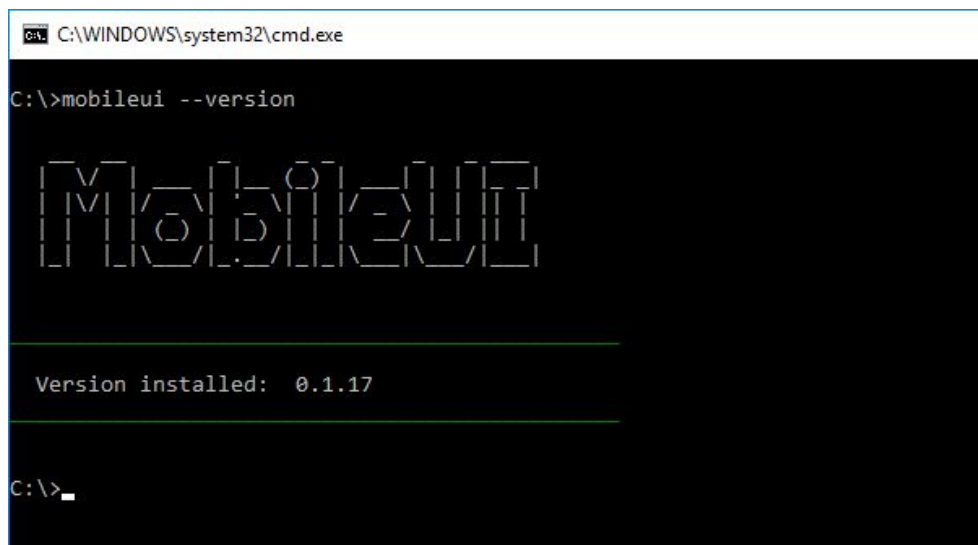
Abra um terminal, certifique-se que você tenha permissão de administrador, e digite o comando abaixo. Os exemplos deste material são apresentados em ambiente Windows, sendo assim para acessar o terminal abra o programa CMD.

```
npm install -g cordova mobileui
```

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:

```
mobileui --version
```

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



```
C:\WINDOWS\system32\cmd.exe
C:\>mobileui --version

MOBILEUI

Version installed: 0.1.17

C:\>
```

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/2017/08/10/desenvolvendo-aplicativos-mobile-hibrido-post-1>

Criando o primeiro app

Nosso primeiro app será um aplicativo com recursos visuais ricos, ou seja, não será apenas um app com textos de “olá mundo” e com exemplos de botões simples e listagem simples, vamos de fato implementar algo que poderia ser utilizado por você e outras pessoas. O app é um catálogo de lugares nas montanhas para escalar, acampar, correr, fazer trilha de bike e fazer rapel. Veja nosso resultado final:



Após você concluir a criação deste seu primeiro aplicativo, entre no site do MobileUI (<https://mobileui.github.io>) e implemente todos os componentes para praticar e melhorar seu conhecimento sobre o desenvolvimento híbrido.

Então vamos ao trabalho, primeiro precisamos criar a estrutura base do projeto com os arquivos mínimos para implementar nosso aplicativo, essa tarefa é fácil pois o Cordova faz isso apenas com um comando. No terminal digite o comando:

```
cordova create PrimeiroApp
```

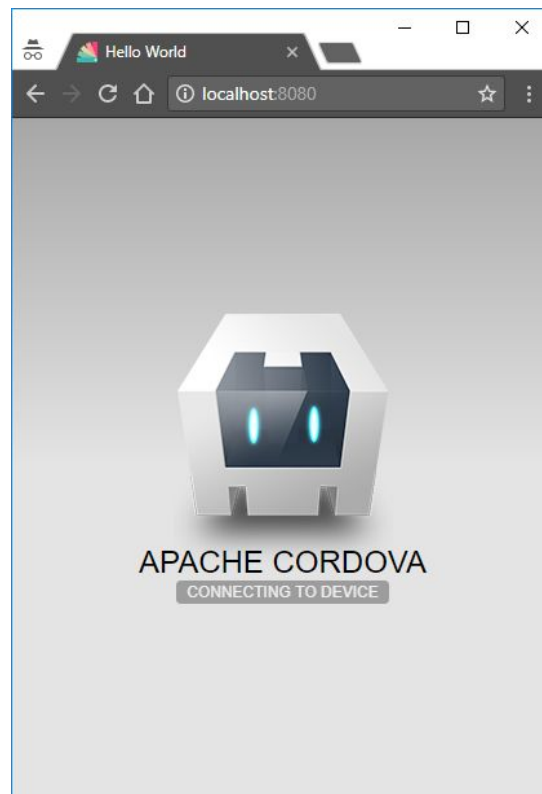

Para testar nosso primeiro aplicativo entre na pasta que foi criada, ao executar o comando acima, com o nome do seu aplicativo, via linha de comando:

```
cd PrimeiroApp
```

Em seguida, utilizando o MobileUI, digite o comando para rodar um servidor do seu aplicativo local com o objetivo de poder testar o app direto pelo *browser*, sem a necessidade de compilar e rodar em um smartphone físico, isso irá deixar nosso trabalho mais produtivo para testes:

```
mobileui preview
```

Ao executar este comando o MobileUI irá exibir uma mensagem informando que o aplicativo está rodando e disponível na porta 8080, sendo assim abra seu navegador, recomendo utilizar o Google Chrome, e digite a url <http://localhost:8080>, você deverá ver a seguinte tela:

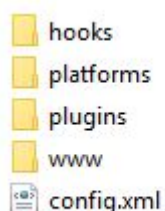


Você pode interromper o servidor a qualquer momento pressionando as teclas Ctrl+C no terminal.

Estrutura dos projetos Cordova

Quando criamos um novo aplicativo, o Cordova cria a estrutura base do app contendo várias pastas e arquivos que irá ser utilizado para gerar o aplicativo para diferentes plataformas como Android, iOS e Windows Phone. Vamos entender um pouco sobre cada pasta e arquivo.

Abra a pasta PrimeiroApp e veja os seguintes arquivos:

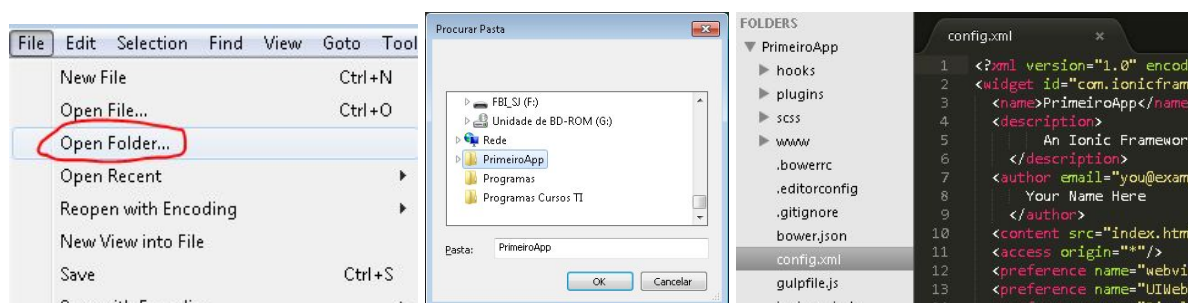


Vamos discutir cada arquivo no seu devido momento, por enquanto precisamos apenas conhecer o config.xml e a pasta www.

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.

```
<?xml version='1.0' encoding='utf-8'?>
<widget id="io.cordova.hellocordova" version="1.0.0"
  xmlns="http://www.w3.org/ns/widgets"
  xmlns:cdv="http://cordova.apache.org/ns/1.0">

  <name>HelloCordova</name>

  <description>
    A sample Apache Cordova application that responds to the deviceready
    event.
  </description>
  <author email="dev@cordova.apache.org" href="http://cordova.io">
    Apache Cordova Team
  </author>
  <content src="index.html" />
```

A tag *name* define o nome do aplicativo que irá aparecer nos atalhos do *device*. As demais linhas são configurações de inicialização, ativação 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. No arquivo *css/index.css* iremos codificar os estilos personalizados dos elementos quando necessário. Na pasta *img* iremos colocar todas nossas imagens. O arquivo *js/index.js* é o principal arquivo JavaScript do aplicativo.

Removendo códigos de exemplo inicial

Como podemos ver no aplicativo que está rodando no *browser*, ao criar um novo app pelo Cordova um exemplo é apresentado com uma logo um texto "APACHE CORDOVA" e algumas linhas de CSS de exemplo, para deixarmos o aplicativo limpo sem esses exemplo vamos instalar um template do cordova limpo, que tem como objetivo apenas limpar tudo que não é necessário em nossa aplicação, para isso digite no terminal dentro da pasta do seu aplicativo o comando abaixo, lembre se que você precisa parar o *preview* caso ele esteja rodando, para isso digite no terminal as teclas Ctrl+C:

```
mobileui install template cordova-blank
```

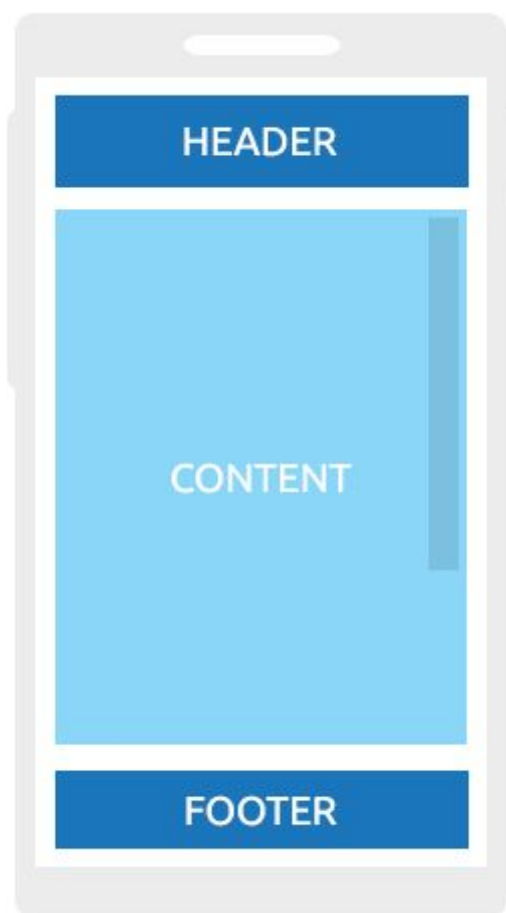
Após instalado, execute o comando *mobileui preview* novamente para subir o servidor e testar no *browser*. Perceba que o aplicativo no *browser* ficou todo branco, ou seja, sem os códigos de exemplo.

Componentes

Os aplicativos nada mais são que uma coleção de componentes utilizados de forma harmônica e coerente em combinações de tamanho, tipos de fonte, posicionamento, cores e interações com o usuário. Para realizar tal tarefa vamos entender a estrutura visual dos componentes.

Layout básico de um app

Um aplicativo móvel geralmente se divide em até três áreas dependendo do seu objetivo. Na área de header definimos o nome da aplicação e alguns botões de navegação como acesso ao menu, botão de pesquisa entre outros. Esta área tem uma característica de ser fixada ao topo da aplicação.



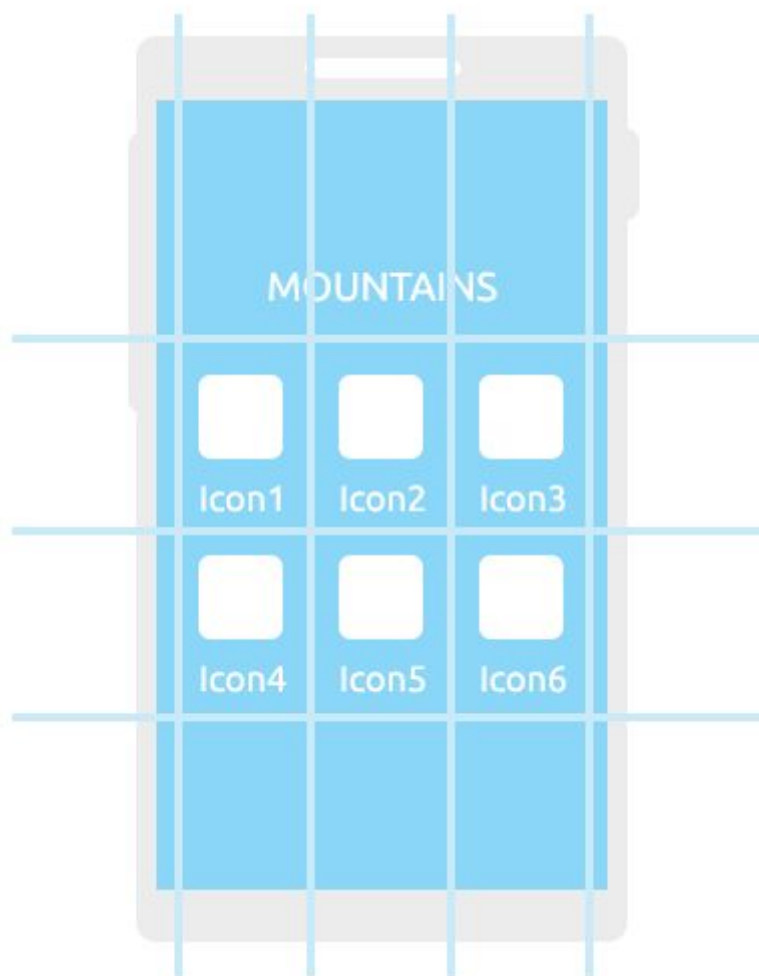
O footer tem o mesmo objetivo que o header porém é fixado ao rodapé da aplicação. Já a área content é a responsável por ter todo nosso conteúdo e apenas esta poderá sofrer

barra de rolagem se necessário para ajustar o conteúdo exibido, nela também poderá conter listas, botões, formulários, gráficos, etc.

Este layout é o mais comum de se identificar mas não é obrigatório, podemos ver aplicativos bem conhecidos que seguiu um layout diferente como o Uber, Snapchat entre outros. Você pode customizar os componentes do MobileUI do jeito que achar melhor mas tome cuidado para não prejudicar a usabilidade e experiência do usuário, pois os componentes do MobileUI já foram feitos pensando na performance e usabilidade fluida com o usuário.

Menu de navegação Springboard

A navegação Springboard é utilizada para exibir, em um formato de grid, ícones com títulos representando um atalho a uma ação, este padrão de navegação é bastante utilizado em aplicativos que precisam exibir atalhos rápidos a determinadas ações e eventos.



Para nossa primeira tela vamos utilizar, inicialmente, os componentes grid, que é responsável por organizar linhas e colunas alinhadas em nosso aplicativo, e o componente base, que tem por objetivo disponibilizar recursos de cores, tamanho de fonte,

posicionamentos entre outros recursos básicos. Para isso instale os dois componentes executando o comando abaixo.

```
mobileui install base grid
```

Depois de instalado você verá que uma linha nova foi adicionada automaticamente no seu arquivo index.html da pasta www.

```
<script type="text/javascript" src="cordova.js"></script>
<script type="text/javascript" src="mobileui/mobileui.js"></script>
<script type="text/javascript" src="js/index.js"></script>
</body>
</html>
```

Isso significa que a instalação ocorreu com sucesso e você já pode usar estes dois componentes instalados.

Vamos então implementar nossas primeiras linhas de código na tela inicial de navegação, para isso edite o arquivo index.html e adicione um elemento div contendo três classes do CSS, que são nossos componentes.

```
<body>
<div class="content row-center blue">
</div>
<script type="text/javascript" src="cordova.js"></script>
<script type="text/javascript" src="mobileui/mobileui.js"></script>
<script type="text/javascript" src="js/index.js"></script>
</body>
```

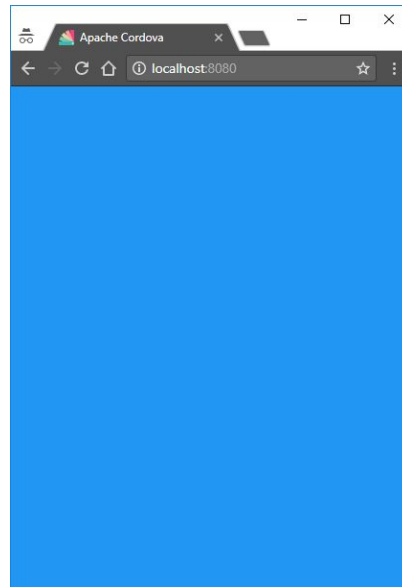
A classe content é responsável pelo componente da área de conteúdo, a row-center é a definição que iremos criar uma linha de grid alinhado ao centro na vertical e a classe blue é a definição de cor do fundo, utilizando o tema de cores disponível no MobileUI.

red	pink	purple	deep-purple
red-50	pink-50	purple-50	deep-purple-50
red-100	pink-100	purple-100	deep-purple-100
red-200	pink-200	purple-200	deep-purple-200
red-300	pink-300	purple-300	deep-purple-300
red-400	pink-400	purple-400	deep-purple-400
red-500	pink-500	purple-500	deep-purple-500
red-600	pink-600	purple-600	deep-purple-600
red-700	pink-700	purple-700	deep-purple-700
red-800	pink-800	purple-800	deep-purple-800
red-900	pink-900	purple-900	deep-purple-900

blue	light-blue	cyan	teal
blue-50	light-blue-50	cyan-50	teal-50
blue-100	light-blue-100	cyan-100	teal-100
blue-200	light-blue-200	cyan-200	teal-200
blue-300	light-blue-300	cyan-300	teal-300
blue-400	light-blue-400	cyan-400	teal-400
blue-500	light-blue-500	cyan-500	teal-500
blue-600	light-blue-600	cyan-600	teal-600
blue-700	light-blue-700	cyan-700	teal-700
blue-800	light-blue-800	cyan-800	teal-800
blue-900	light-blue-900	cyan-900	teal-900

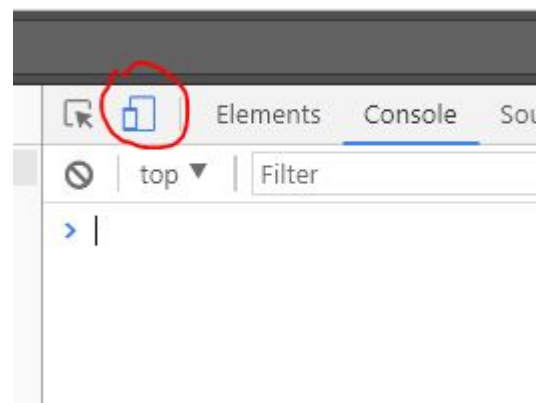
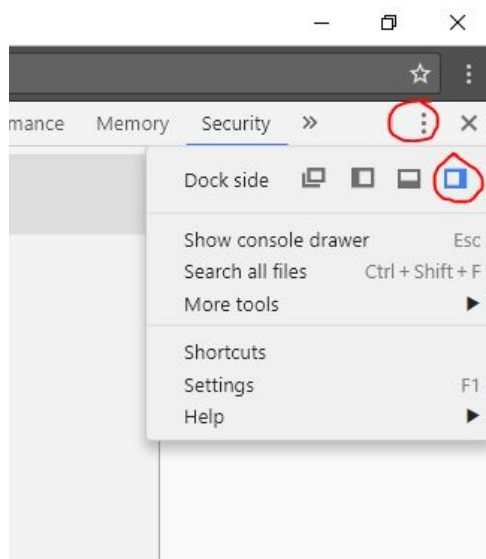
Veja a lista completa das possíveis cores e o nome de cada uma no link: <https://mobileui.github.io/#colors>

Perceba que quando você faz alguma alteração no seu projeto e salva, o aplicativo no *browser* é automaticamente recarregado.

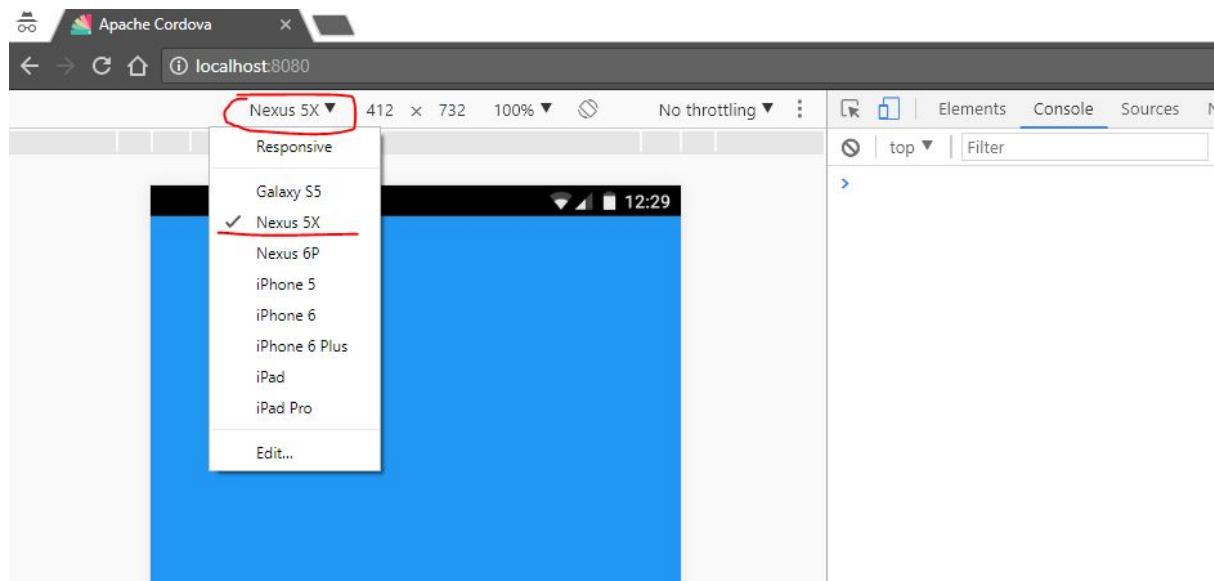


Esta visualização não é muito útil pois não simula o tamanho de tela do *device*. Você pode redimensionar seu *browser* para que fique menor na largura ou utilizar o módulo visualização *mobile* do Chrome. Para isso pressione F12 no Windows/Linux ou option+command+J no Mac. Ao lado direito do terminal que foi apresentado clique nos três pontinhos e escolha a visualização *Dock to right*, para que seu terminal fique ao lado direito da página 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*.



Agora você tem uma visão de como realmente será exibido em diferentes *device* com a opção de mudar o simulador.



Vamos adicionar uma coluna que fique alinhada ao centro verticalmente e horizontalmente, para isso precisamos adicionar outra classe ao componente content para definir todos os elementos ao centro horizontalmente e adicionar um elemento div contendo a classe col dentro de content.

```
<div class="content row-center blue align-center">
  <div class="col">
    coluna1
  </div>
</div>
```

Dentro da coluna1 vamos implementar nosso menu Springboard.

```
<div class="content row-center blue align-center">
  <div class="col">
    <div class="row">
      <div class="col">linha 1 coluna 1</div>
    </div>
    <div class="row">
      <div class="col">linha 2 coluna 1</div>
      <div class="col">linha 2 coluna 2</div>
      <div class="col">linha 2 coluna 3</div>
    </div>
    <div class="row">
      <div class="col">linha 3 coluna 1</div>
      <div class="col">linha 3 coluna 2</div>
      <div class="col">linha 3 coluna 3</div>
    </div>
  </div>
</div>
```


Na primeira linha iremos adicionar o título do nosso aplicativo e as classes responsáveis por deixar o texto grande, que pode ser entre `text-small`, `text-big` e `text-huge`, e a classe para deixar o texto em negrito.

```
<div class="content row-center blue align-center">
  <div class="col">
    <div class="row">
      <div class="col">
        <h1 class="text-huge text-strong">MOUNTAINS</h1>
      </div>
    </div>
  </div>
</div>
```

Ícones

Quando instalamos o componente base, o MobileUI instalou também um pacote 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>`.

No site ionicons ao clicar sobre um determinado ícone você verá a classe que deve ser adicionado no elemento.



Sabendo disso vamos adicionar os ícones e os títulos dos nossos atalhos, veja como irá ficar nossa primeira linha de ícones.

```

<div class="row">
  <div class="col">
    <h1 class="text-huge text-strong">MOUNTAINS</h1>
  </div>
</div>
<div class="row">
  <div class="col">
    <i class="icon ion-ios-pulse icon-big"></i>
    <h2>CORRER</h2>
  </div>
  <div class="col">
    <i class="icon ion-ios-home-outline icon-big"></i>
    <h2>ACAMPAR</h2>
  </div>
  <div class="col">
    <i class="icon ion-ios-paperplane-outline icon-big"></i>
    <h2>ESCALAR</h2>
  </div>
</div>

```

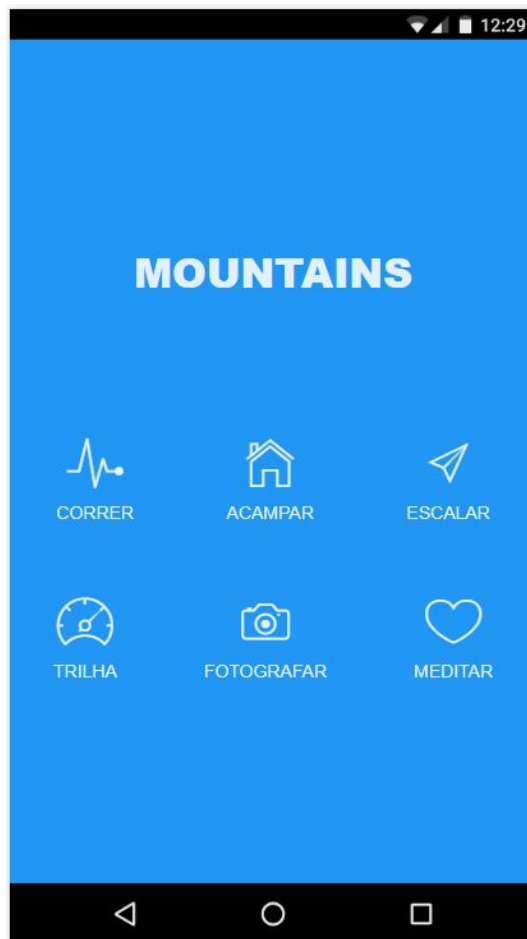
Perceba que o título do aplicativo ficou muito próximo dos ícones, e esteticamente isso não é legal, para resolver esse problema vamos adicionar um espaço grande entre as duas linhas.

```

<div class="row">
  <div class="col">
    <h1 class="text-huge text-strong">MOUNTAINS</h1>
  </div>
</div>
<div class="space-huge"></div>
<div class="row">
  <div class="col">
    <i class="icon ion-ios-pulse icon-big"></i>
    <h2>CORRER</h2>
  </div>
  <div class="col">
    <i class="icon ion-ios-home-outline icon-big"></i>
    <h2>ACAMPAR</h2>
  </div>
  <div class="col">
    <i class="icon ion-ios-paperplane-outline icon-big"></i>
    <h2>ESCALAR</h2>
  </div>
</div>

```

Agora é com você, faça a outra linha de atalhos e adicione um espaço entre as linhas dos ícones (space-big), o resultado esperado até agora será:



Para melhorar esteticamente ainda mais nosso app vamos usar o componente *cover*, que tem como objetivo mesclar uma imagem de fundo com uma cor do tema. Antes de usar precisamos instalar o componente executando o comando abaixo.

```
mobileui install cover
```

Agora vamos ajustar nosso componente content adicionando a classe *cover* e o modo de mesclagem que será *blend-soft-light*, e por fim adicionamos uma imagem de fundo.

Para desenvolver este app você pode baixar todas as imagens utilizadas nos códigos pelo link (<https://github.com/fabiorogerosj/app-montains/tree/master/www/img>) e salvar dentro da pasta *img* do seu projeto.

```
<div class="content row-center blue align-center cover blend-soft-light"
  style="background-image: url(img/fundo.jpg);">
  <div class="col">
    <div class="row">
```



Você pode visualizar o código completo da tela inicial no link:
<https://github.com/fabiorogerosj/app-montains/blob/master/www/index.html>

Navegando entre telas

Precisamos neste momento criar as telas que serão abertas ao clicar em cada atalho, como correr, acampar, etc. Para isso vamos utilizar o componente Page, responsável por carregar novas páginas em nosso aplicativo, cada página é uma nova tela. Primeiro vamos instalar o componente executando o comando abaixo.

```
mobileui install page
```

A princípio você precisa saber que cada tela é um arquivo .html diferente e o nome da tela será o nome do arquivo html. Sendo assim vamos criar um novo arquivo na pasta www com o nome correr.html contendo o seguinte código:

```

<div class="page">

  <div>
    <h1>LUGARES PARA CORRER</h1>
  </div>

</div>

```

Perceba que precisamos adicionar a classe page no primeiro elemento div, depois criamos uma div sem classe, por enquanto, contendo um h1 com um título. Agora vamos adicionar um evento de clique na coluna de correr, para quando o usuário clicar nela a página de correr.html seja aberta.

No index.html adicione a propriedade onclick na coluna de correr chamando a função openPage(), como parâmetro iremos passar o nome da página que é correr.html, veja que no código abaixo não é necessário passar o .html.

```

<div class="content row-center blue align-center cover blend-soft-light">
  <div class="col">
    <div class="row">
      <div class="col">
        <h1 class="text-huge text-strong">MOUNTAINS</h1>
      </div>
    </div>
    <div class="space-huge"></div>
    <div class="row">
      <div class="col" onclick="openPage('correr')">
        <i class="icon ion-ios-pulse icon-big"></i>
        <h2>CORRER</h2>
      </div>
      <div class="col">
        <i class="icon ion-ios-home-outline icon-big"></i>
        <h2>ACAMPAR</h2>
      </div>
    </div>
  </div>
</div>

```

Para testar basta clicar no atalho correr, com isso a nova tela irá abrir apenas com o texto que adicionamos. Agora vamos criar o header da nova tela, para isso faça a instalação do componente header.

mobileui install header

Perceba no seu terminal que o MobileUI também instalou o componente button, isso porque o header depende do button para funcionar. Após instalado você precisa adicionar a classe header no elemento div que irá ser nosso topo, também vamos adicionar uma cor de fundo.

```
<div class="page">

  <div class="header blue">
    <h1>LUGARES PARA CORRER</h1>
  </div>

</div>
```

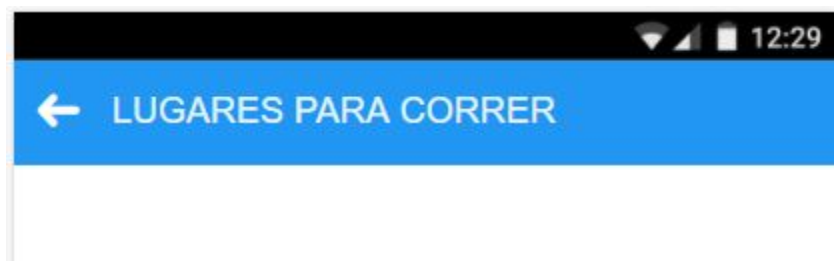
Com isso você já irá visualizar o componente header após abrir a tela de correr. Vamos adicionar um botão para voltar à tela inicial. Neste botão iremos adicionar a classe `left` para deixar posicionado nosso botão a esquerda e iremos informar um ícone, e também adicionar um evento ao clicar no botão para executar a função `backPage()`, que tem como objetivo fechar a tela atual e voltar a anterior.

```
<div class="page">

  <div class="header blue">
    <button class="left icon ion-arrow-left-c" onclick="backPage()"></button>
    <h1>LUGARES PARA CORRER</h1>
  </div>

</div>
```

Nosso header está completo e podemos ver o resultado abaixo.



Vamos implementar a listagem de lugares, para isso iremos utilizar o componente `list`, sendo assim faça a instalação executando o comando abaixo.

```
mobileui install list
```

Em seguida vamos criar uma lista contendo vários itens, e cada item contém uma imagem, um título e um parágrafo. Perceba que no código abaixo a lista está dentro do componente `content`, pois ele poderá sofrer barra de rolagem.

Você pode baixar as imagens utilizadas ebook pelo link:

<https://github.com/fabiorogerosj/app-montains/tree/master/www/img>

```

<div class="page">

  <div class="header blue">
    <button class="left icon ion-arrow-left-c" onclick="backPage()"></button>
    <h1>LUGARES PARA CORRER</h1>
  </div>

  <div class="content">
    <div class="list">
      <div class="item">
        
        <h2>Matterhorn</h2>
        <p>O Matterhorn ou Monte Cervino é talvez a montanha mais conhecida dos Alpes.</p>
      </div>
    </div>
  </div>

</div>

```

Provavelmente você identificou um problema, pois sua lista está ficando por baixo do header, isso ocorre pois nem sempre o header existirá, quando ele existe precisamos informar ao content que tem uma header adicionando a classe has-header no content.

```

<div class="content has-header">
  <div class="list">
    <div class="item">

```

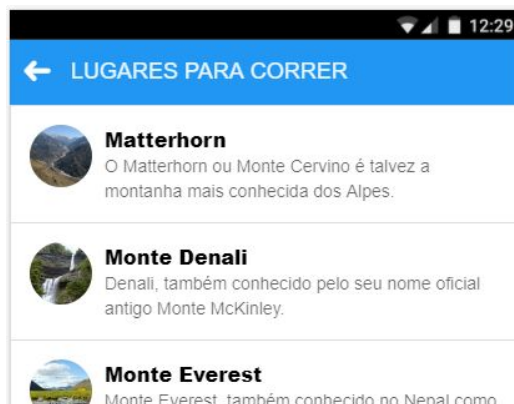
Vamos deixar o título do item em negrito, o parágrafo com opacidade e a imagem redonda e com um tamanho padrão.

```

<div class="content has-header">
  <div class="list">
    <div class="item">
      
      <h2 class="text-strong">Matterhorn</h2>
      <p class="opacity-60">O Matterhorn ou Monte Cervino é tal
    </div>
  </div>
</div>

```

Agora é com você, crie os demais itens para obter o resultado abaixo.



Header transparente

A estrutura da nossa próxima página é bem semelhante a listagem que criamos anteriormente, porém, agora, iremos criar um header com fundo transparente para focar na imagem do topo. Sendo assim no primeiro item da nossa listagem em correr.html vamos adicionar a função `openPage()` no evento de cliente, para quando for clicado abrir a página nova que iremos criar chamada `correr1.html`.

```
<div class="content has-header">
  <div class="list">
    <div class="item" onclick="openPage('correr1')">
      
      <h2 class="text-strong">Matterhorn</h2>
      <p class="opacity-60">O Matterhorn ou Monte Cervino é t
    </div>
```

Em nossa nova página vamos criar a estrutura básica de header e content, porém com algumas classes diferentes que deixa o header preto com opacidade e criamos um cover de altura de 300px.

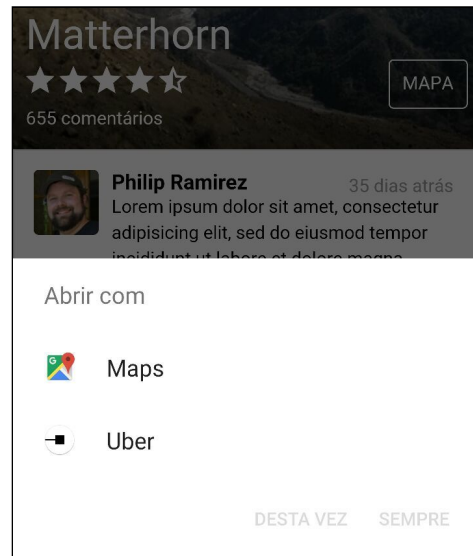
```
<div class="page">
  <div class="header black-opacity-30">
    <button class="left icon ion-arrow-left-c" onclick="backPage()"></button>
  </div>

  <div class="content">
    <div class="cover cover-300" style="background-image: url(img/correr1.jpg);">
    </div>
  </div>
</div>
```

Vamos adicionar algumas informações no rodapé do cover contento o nome do local e a qualificação do mesmo. Por fim adicionamos um botão para abrir o mapa da localização deste local. Perceba que estamos colocando no href o geo, isso fará com que o dispositivo abra o Google Maps ou outro app de navegação.

```
<div class="cover cover-300" style="background-image: url(img/correr1.jpg);">
  <div class="padding gradient bottom text-white">
    <h1 class="text-huge">Matterhorn</h1>
    <i class="icon ion-android-star"></i>
    <i class="icon ion-android-star"></i>
    <i class="icon ion-android-star"></i>
    <i class="icon ion-android-star"></i>
    <i class="icon ion-android-star-half"></i>
    <a href="geo:38.897096,-77.036545" class="border-white text-white padding right radius">
      MAPA
    </a>
    <p>655 comentários</p>
  </div>
</div>
```


Veja como fica no Android ao clicar no link MAPA:



Por fim vamos criar uma outra lista contendo algumas informações de comentários sobre o local que está sendo exibido. Perceba que utilizamos dois novos elementos em nossa lista, uma classe para deixar um texto a direita e a classe block para fazer com que o texto respeite o espaço da imagem.

```
<div class="cover cover-300" style="background-image: url(img/corner1.jpg);">
  <div class="padding gradient bottom text-white">
    <h1 class="text-huge">Matterhorn</h1>
    <i class="icon ion-android-star"></i>
    <i class="icon ion-android-star"></i>
    <i class="icon ion-android-star"></i>
    <i class="icon ion-android-star"></i>
    <i class="icon ion-android-star-half"></i>
    <a href="geo:38.897096,-77.036545" class="border-white text-white padd
      MAPA
    </a>
    <p>655 comentários</p>
  </div>
</div>
<div class="list">
  <div class="item">
    <div class="right text-small text-grey align-top padding-top">
      35 dias atrás
    </div>
    
    <h2 class="text-strong">Philip Ramirez</h2>
    <p class="block">
      Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do e
    </p>
  </div>
</div>
```

No final teremos este resultado para uma das opções de correr, agora é com você, crie as demais opções para praticar. Lembre-se que a prática leva à perfeição.



Você pode visualizar o projeto completo no meu github pelo link:
<https://github.com/fabiorogerosj/app-montains>

Aplicativo nativo

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.

Rodando o aplicativo no 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

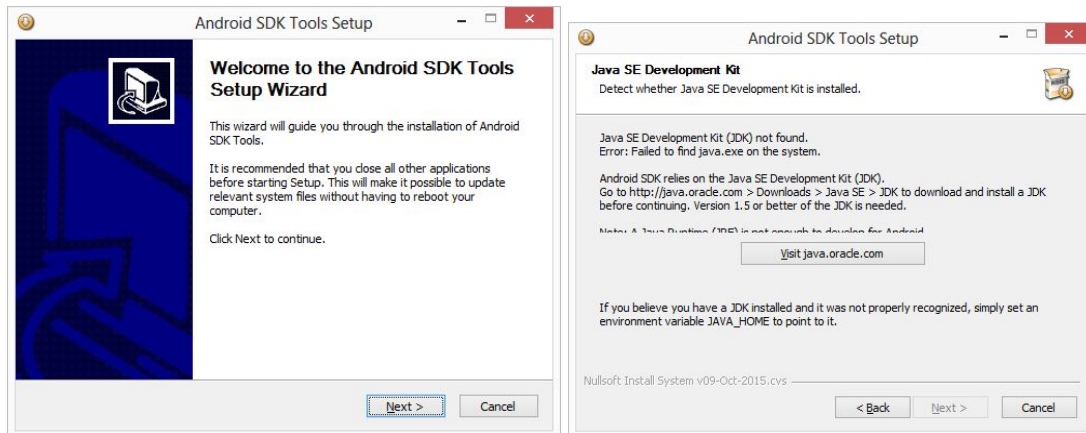
If you do not need Android Studio, you can download the basic Android command line tools below.

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

Para usuários Linux e Mac 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.

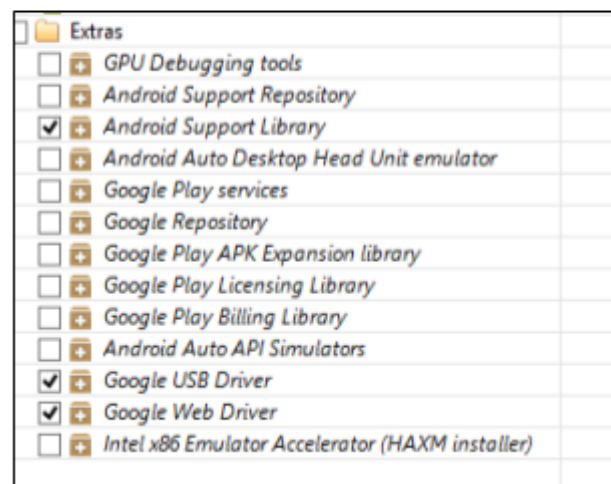
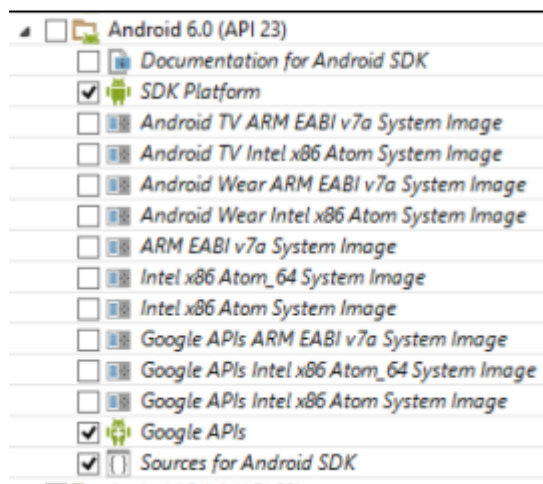
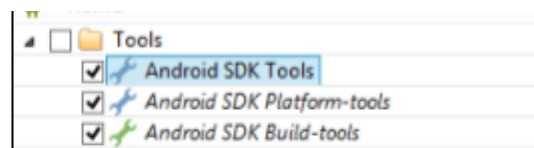
Para usuários do Windows podemos usar o instalador, onde já é verificado se existe o Java JDK instalado, e instalar o SDK em uma determinada pasta.

Execute o instalador e siga os passos abaixo:

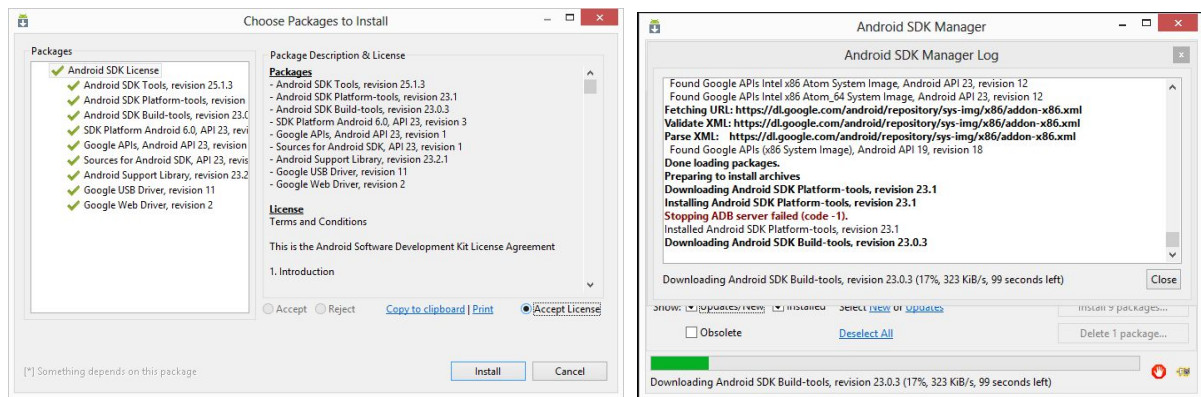


Se você tiver o Java JDK instalado o instalador irá seguir para a instalação do Android SDK, caso contrário será exibida esta tela solicitando a instalação do Java JDK:

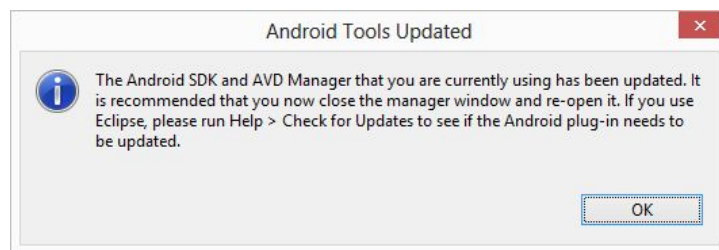
O Cordova precisa de algumas dependências do Android para fazer *build*, ou seja, gerar o aplicativo final, para isso vamos fazer alguns *Update* no Manager Android SDK. No Manager desmarque tudo que já vem marcado e marque apenas as opções listadas abaixo. Em seguida clique em **Install 9 packages**:



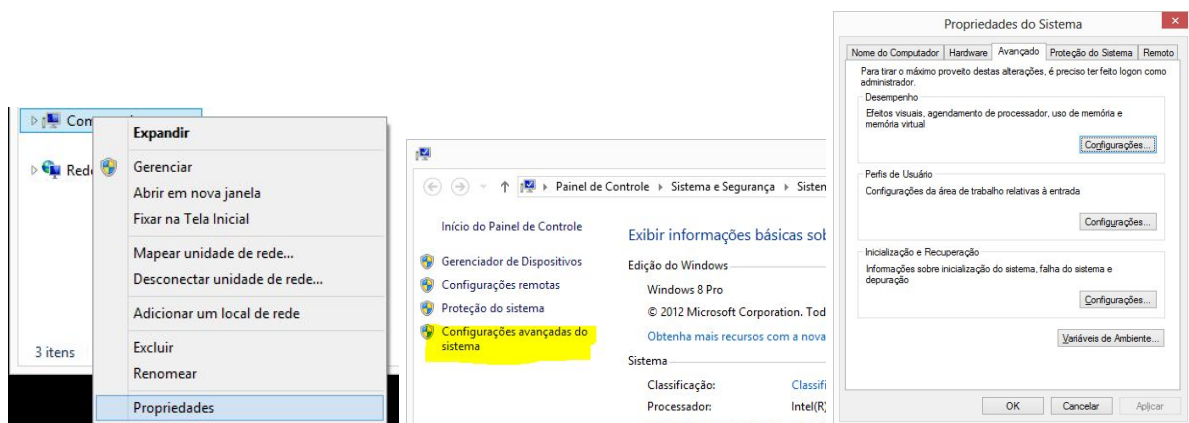
Em seguida clique em **Accept License** e **Install** para o download começar. Este processo pode demorar alguns minutos.



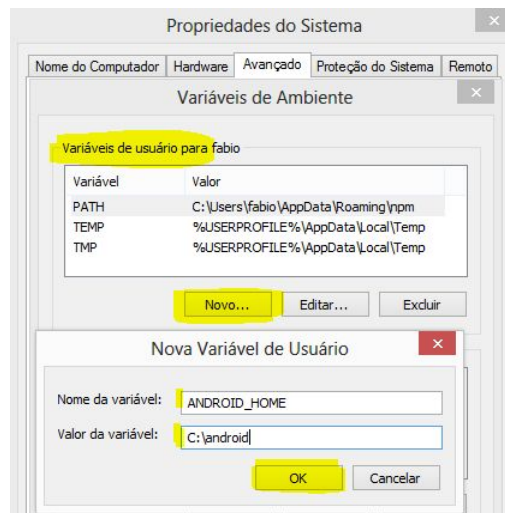
Ao concluir a tela abaixo será apresentada:



Vamos agora adicionar algumas variáveis de ambiente para que o Cordova 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:

```
cordova platform add android
cordova 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 última 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:
<http://fabiorogerosj.com.br/desenvolvendo-aplicativos-mobile-ionic-post-1.html>.

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:
<http://fabiorogerosj.com.br/desenvolvendo-aplicativos-mobile-ionic-post-2.html>

Rodando o aplicativo no iOS

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

Existem algumas plataformas como Adobe Phonegap e Ionic.io, que oferecem algumas soluções em nuvem para aplicativos implementados em Cordova, para compilar o projeto para iOS sem precisar de uma máquina Mac, porém este serviço é *free* com limitações. Iremos falar sobre build em nuvem 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:

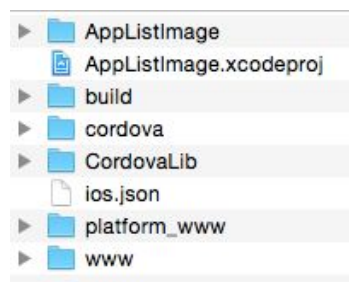
```
npm install -g ios-sim
```

Em seguida, dentro da pasta do projeto, digite:

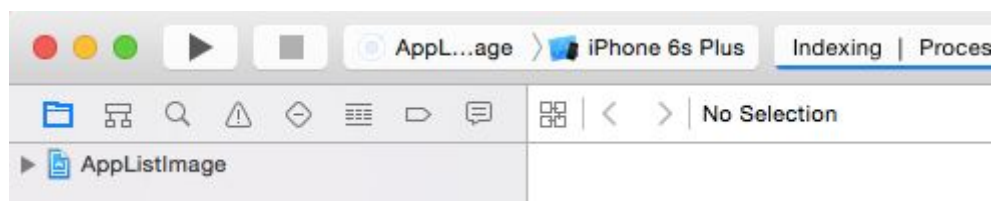
```
cordova build ios
```

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.

Conclusão

Primeira missão cumprida, aqui terminamos nossa aventura no fantástico mundo do desenvolvimento de aplicativos móveis híbridos e multi-plataformas, entretanto não é o final, e sim apenas o primeiro passo onde entendemos a importância de desenvolver aplicações híbridas, quando a necessidade é lançar aplicativos de forma rápida ao mercado levando em consideração as muitas plataformas existentes e o grau de complexidade que enfrentaremos se fossemos criar aplicativos nativos.

Discutimos que o MobileUI fornece inúmeros componentes, não só os utilizamos neste ebook, mas tenha em mente que se quisermos criar aplicativos do zero utilizando apenas o nosso HTML, CSS e JavaScript, isso é possível, porém, utilizar os componentes do MobileUI certamente traz um ganho de produtividade enorme em velocidade de desenvolvimento.

Perceba que nosso primeiro aplicativo não é apenas um “olá mundo” como estamos acostumados a ver em outros materiais. O objetivo foi mostrar como é simples, rápido e fácil criar aplicativos com interfaces ricas em elementos visuais sem a necessidade de saber sobre HTML e CSS avançado.

Em nosso próximo passo iremos abordar a utilização mais complexa dos componentes do MobileUI e o acesso nativo dos recursos de câmera, GPS e outros sensores.

Fique ligado na página que você baixou esse ebook free e lembre-se, pratique o máximo que conseguir e implemente os outros componentes existentes na documentação oficial do MobileUI, certamente isso irá melhorar seu conhecimento e abrir sua mente para criar seu segundo, terceiro, quarta e demais aplicativos sozinho.

Você pode dar seu feedback deixando seu depoimento se gostou deste material e no que ele te ajudou, para isso preencha o formulário de depoimento contido no site que você baixou este ebook, isso é o combustível que me anima a criar mais material free acessível para todos :)