01

Plugins no Cordova

Transcrição

Ainda não exploramos uma funcionalidade do Cordova muito importante, o uso dos **plugins**. O Cordova conta com um conjunto de funcionalidades que permitem rodar o nosso HTML, CSS e JS, no formato de app na *WebView* de um sistema operacional. No entanto, algumas funcionalidades mais avançadas não fazem parte do *Cordova core*. Existem centenas de *plugins* que estendem as funcionalidades do Cordova - em geral, relacionadas ao acesso a elementos nativos do sistema.

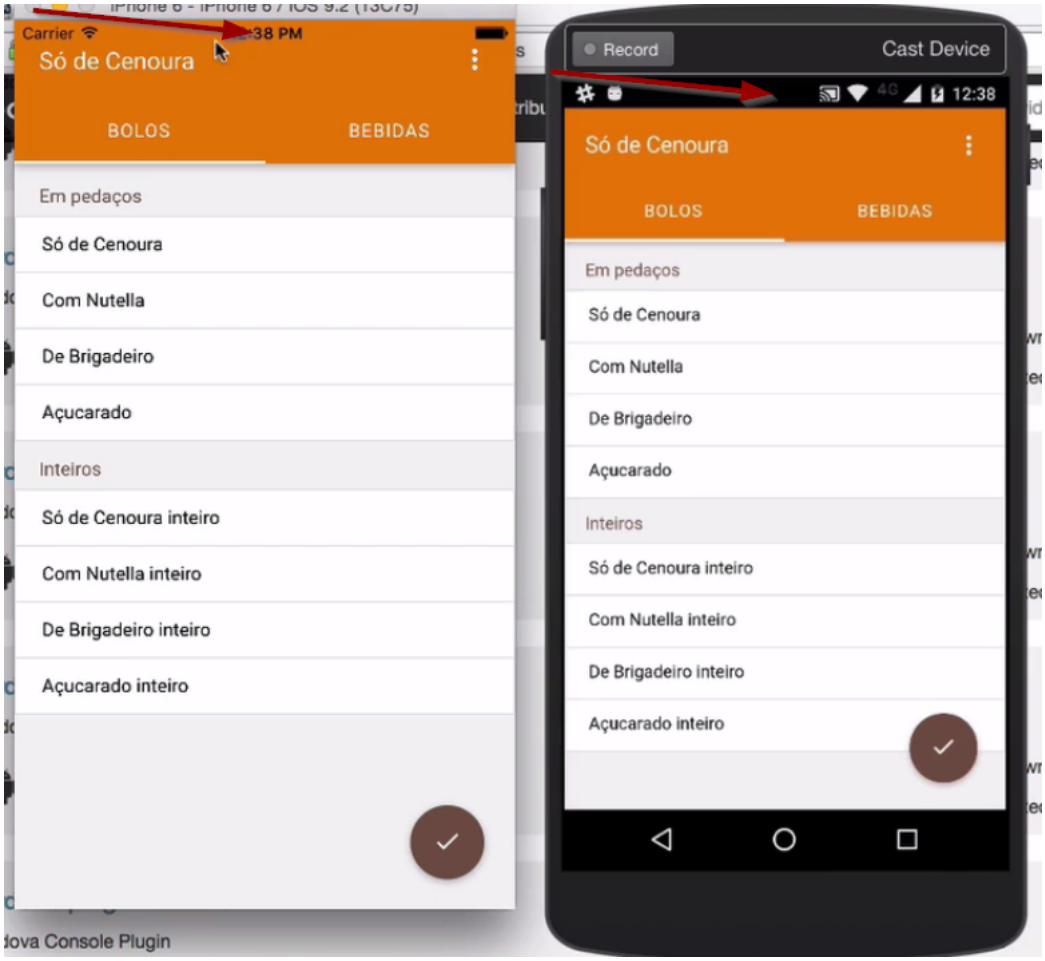
Por exemplo, se você quiser acessar o hardware do usuário, como a câmera. Com um plugin, poderá acessá-la. Na página de plugins (https://cordova.apache.org/plugins/) do Cordova, encontraremos as opções existentes. Alguns são próprios da ferramenta, outros serão do Adobe PhoneGap, além de diversos outros registrados.

Atualmente, podemos encontrar disponível mais de mil plugins. Pelos nomes, podemos deduzir a funcionalidade de alguns. O `corodova-pluging-splashscreen`, permite fazer configurações da *Splash screen*, como animá-la, por exemplo. Com o `cordova-plugin-inappbrowser`, o aplicativo irá disponibilizar um *Browser* para o usuário. O `cordova-plugin-file` permite acessar um sistema de arquivo do sistema operacional. Podemos adivinhar com facilidade a utilidade do `cordova-plugin-camera`, que permite acessar a câmera do aparelho.

Existem vários outros, com diversas funcionalidades: `cordova-plugin-geolocation` (ativa geolocalização), `cordova-plugin-push` (notificações), `cordova-plugin-vibration` (vibração), `cordova-plugin-media` (toca mídia), além de outras.

Na Garçonapp, não iremos utilizar tantos plugins, mas gostaria de explicar o conceito de **plugin** e como ele é aplicado na prática. Iremos usar um deles, que resolverá um problema que acontece no nosso app. Com o `cordova-plugin-statusbar` iremos manipular a **barra de status** do sistema, este é um dos plugins mais usados pelos usuários.

O que é a barra de status? É a barra do topo do dispositivo, que exibe diversas informações como, por exemplo, sinal de rede e horas.



Quando rodamos o aplicativo, observe que a barra está levemente sobre o topo. No iOS, o título fica um pouco desalinhado com a barra de status. No Android, a barra não sobrepõe o app, porém, roda com a cor preta. Nós costumamos usar uma barra com cores do mesmo tom em apps Android, este é o padrão do Material Design. Como configuramos as cores e o comportamento? Com o `cordova-plugin-statusbar`.

Na linha de comando, iremos digitar `cordova plugin add` juntamente com o nome do plugin selecionado.

```
garconapp $ cordova plugin add cordova-plugin-statusbar
```

Ele irá baixar o plugin, que já será instalado nas plataformas Android e iOS.

Agora dentro do projeto irá aparecer a pasta `plugins`, onde irão aparecer os plugins instalados, entre eles o `cordova-plugin-whitelist` e o `cordova-plugin-statusbar`.

Se precisarmos passar o nosso projeto para outro desenvolvedor, precisaremos utilizar todos os plugins utilizados. Para não precisarmos apagar a pasta com todos eles e pedir para o desenvolvedor baixá-los novamente - o que será bastante trabalhoso, se trabalharmos com muitos plugins - podemos usar outro recurso. Juntamente com o mesmo comando utilizado antes, `cordova plugin add` adicionaremos `--save`.

```
garconapp $ cordova plugin add cordova-plugin-statusbar --save
```

Após fazer a instalação, irá salvar no fim do `xml` a *tag* `plugin name`.

```
<plugin name="cordova-plugin-statusbar" spec="~2.0.0" />
```

Esta é uma boa prática, porque ele irá baixar automaticamente o plugin, conforme está declarado no `xml` . Se usarmos `spec="*"` , ele sempre fará o download da versão mais recente.

Agora, iremos configurar a barra de status. Para isto, adicionaremos novas preferências, que são habilitadas apenas quando o plugin está rodando. Primeiramente, mudaremos o *background* do *statusbar*, para um laranja escuro.

```
<preference name="StatusBarBackgroundColor" value="#E86C13" />
```

Iremos mandar o Terminal rodar o projeto, e ele irá gerá-lo novamente. Após o processo, ele já aparecerá com a barrinha laranja. Usando o Material Design, é padrão usarmos a barra de status em um tom mais escuro do que a cor de fundo do sistema. No entanto, não usamos este tipo de comportamento no iOS. Por isso, iremos especificar no `config.xml` que a configuração será válida apenas para Android.

```
<platform name="android">
    <preference name="StatusBarBackgroundColor" value="#E86C13"
    />
</platform>
```

Desta forma, não teremos a barra com a cor escura no iOS. Mas iremos criar outras configurações para ele.Por padrão, a barra de status fica sobreposta ao aplicativo. Nós iremos alterar isto:

```
<platform name="ios">
    <preference name="StatusBarOverlayWebView" value="false"
    />
</platform>
```

Vamos incluir mais alguns ajustes. Com a alteração anterior, a barra de status passou a ficar com uma cor branca, nós iremos modificá-la para o mesmo laranja usado como fundo no projeto.

```
<preference name="StatusBarBackgroundColor" value="#F57F17"
    />
```

Ao rodarmos no Terminal com o comando `run ios` , ele irá buildar o projeto. Quando abri-lo, veremos o menu do topo posicionado mais abaixo, enquanto a barra de status está localizada acima.

Através do `xml` , podemos fazer inúmeras configurações com plugins. Se consultarmos a documentação do *statusbar*, veremos que podemos incluir outras preferências. Por enquanto, manipular a cor é o suficiente.