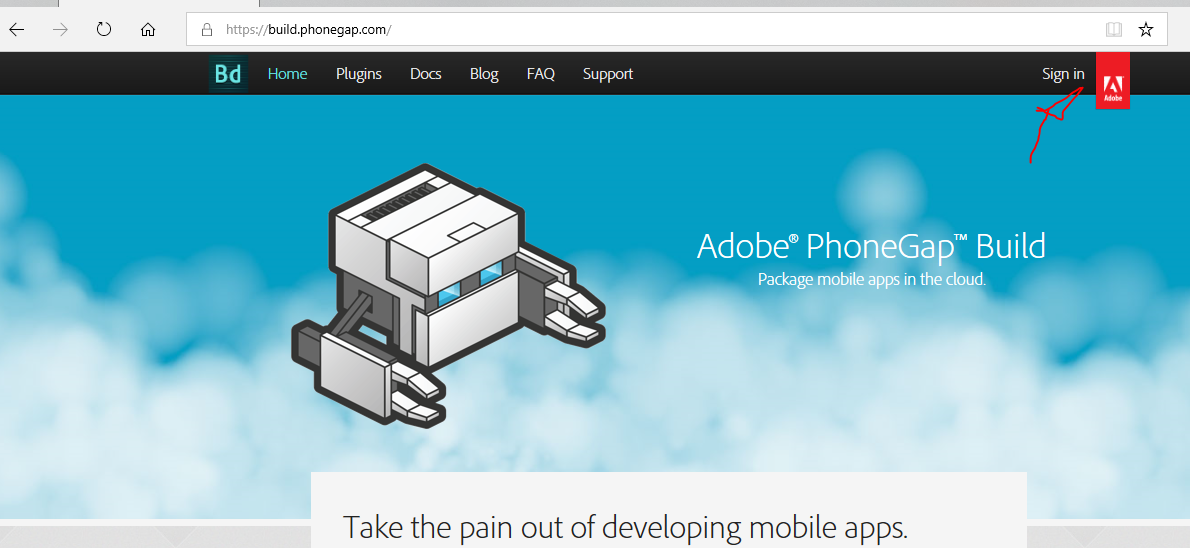
**App Android com PhoneGap Build**

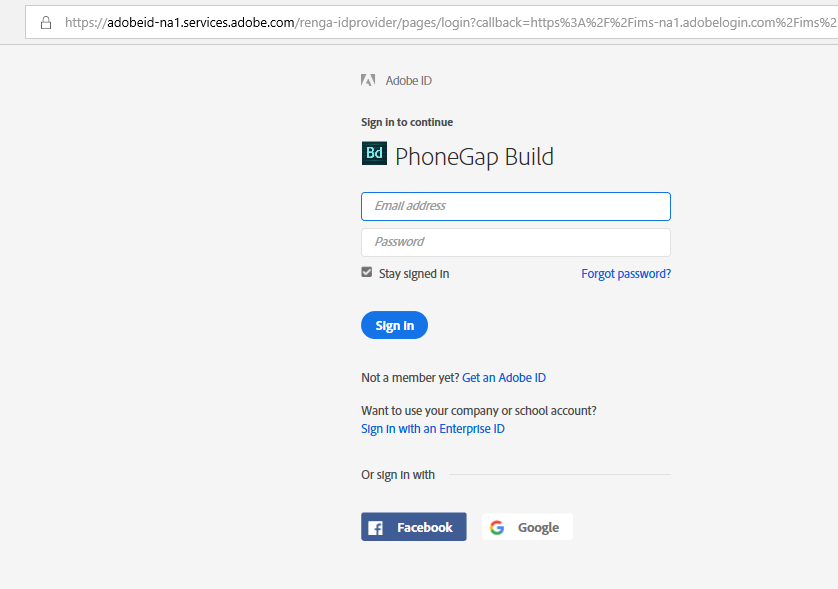
 O PhoneGap é um produto da Adobe, com base no Cordova, mas com alguns serviços adicionais. Um deles é o Adobe PhoneGap Build que permite empacotar os apps mobiles na nuvem.

Ele provê a estrutura nativa do Android e iOS instalado em suas máquinas e através da nuvem, enviamos nosso aplicativo para ele e receberemos de volta nosso projeto "buildado". Sem precisarmos executar tarefas do nosso computador, como instalar SDK, por exemplo.

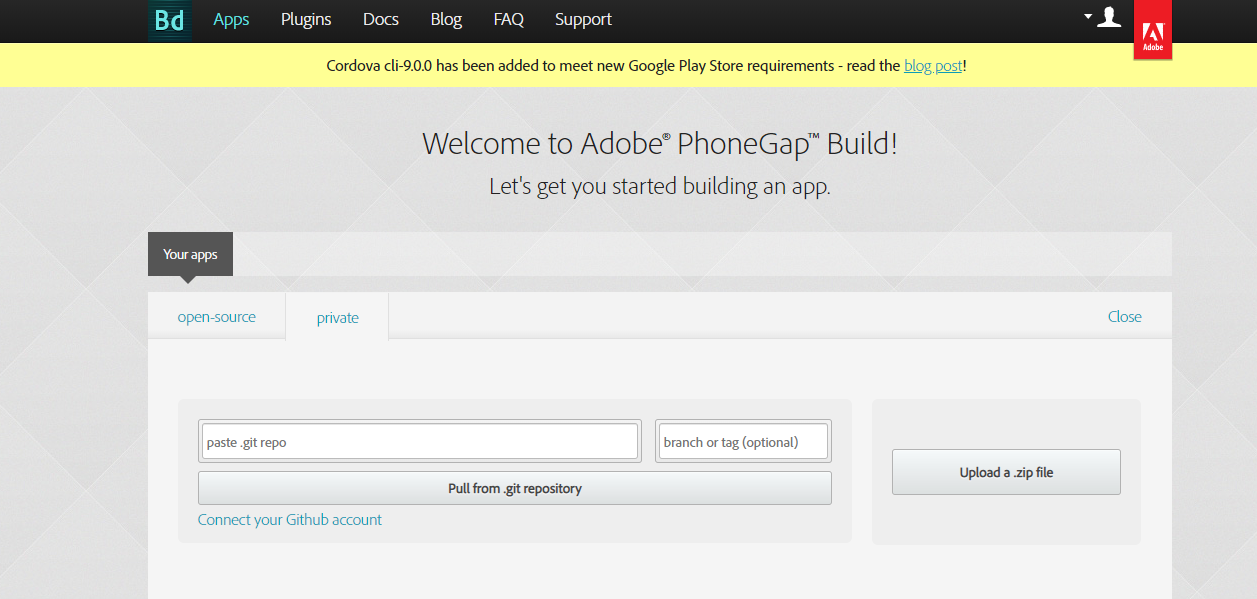
O PhoneGap Build oferece muitas facilidades: compila, disponibiliza múltiplas plataformas... O serviço é pago, mas oferece uma opção de plano gratuito que já será o bastante para o nosso aplicativo.

Para utilizar o serviço, você precisará de um cadastro. Ele irá te pedir uma senha da Adobe ID. Eu já tenho meu cadastro, poderemos seguir para a próxima página. <https://build.phonegap.com/>

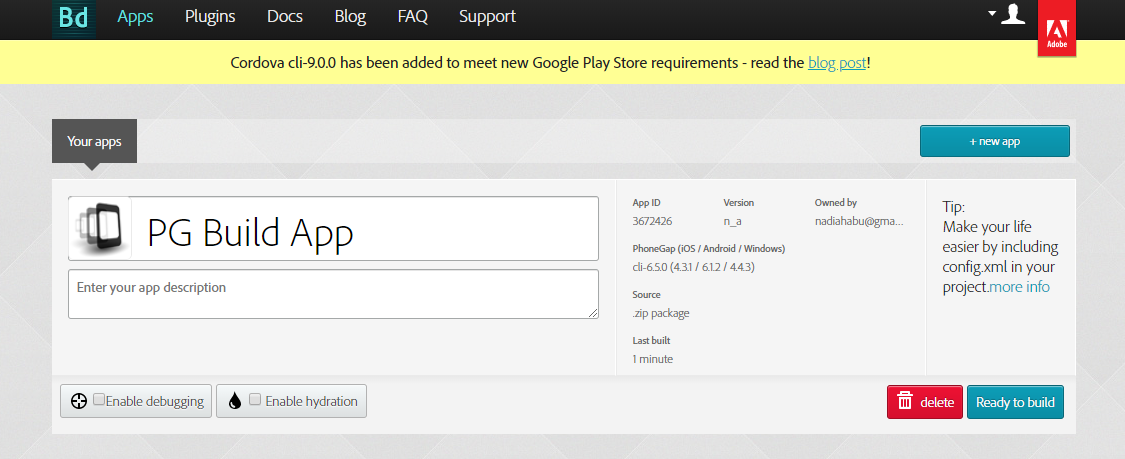




Nós temos a opção de fazer infinitas apps open-source ou apenas uma privada. Com a open-source precisaremos trabalhar com o Github. Nós faremos uma privada, que conta com uma facilidade interessante: podemos simplesmente fazer o upload de um arquivo .zip.



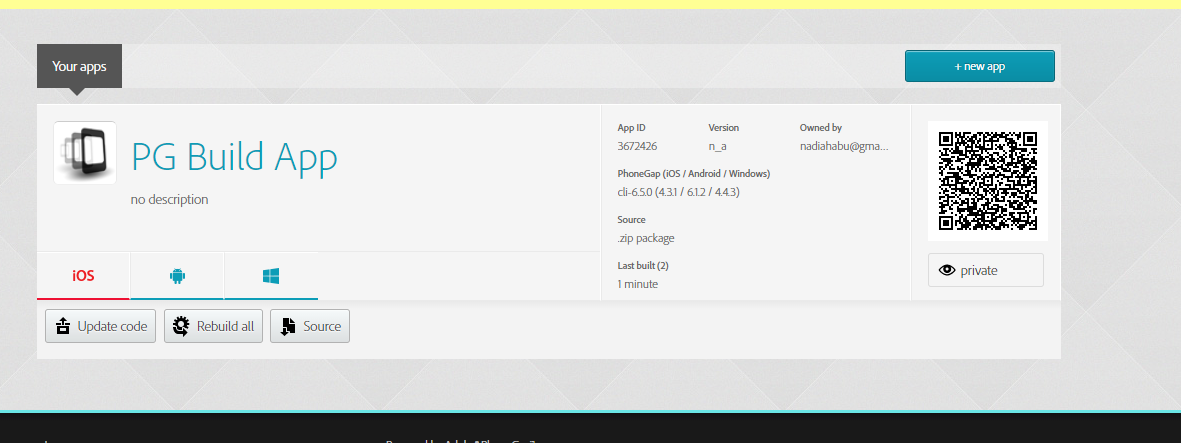
Criado o arquivo comprimido, voltaremos ao PhoneGap Build e clicaremos em Upload. Ele abrirá uma nova janela, onde poderemos procurar o arquivo .zip que acabamos de criar. Após selecionar o arquivo, clicamos em open e ele começará a fazer o uploading. Aguardaremos um pouco, ele indicará que importou o arquivo e depois, clicaremos em Ready to Build.



Agora, ele irá "buildar" o aplicativo em três plataformas que ele suporta por padrão (iOS, Android e Windows Forms). Até aqui, apenas subimos um arquivo .zip. No entanto, percebemos que o ícone de iOS está vermelho, indicando que ele não conseguiu "buildar".

Mas ele consegue com as outras plataformas. Se clicarmos no ícone do Android e irá baixar o arquivo .apk (que é o formato de apps do Android para rodar no celular).

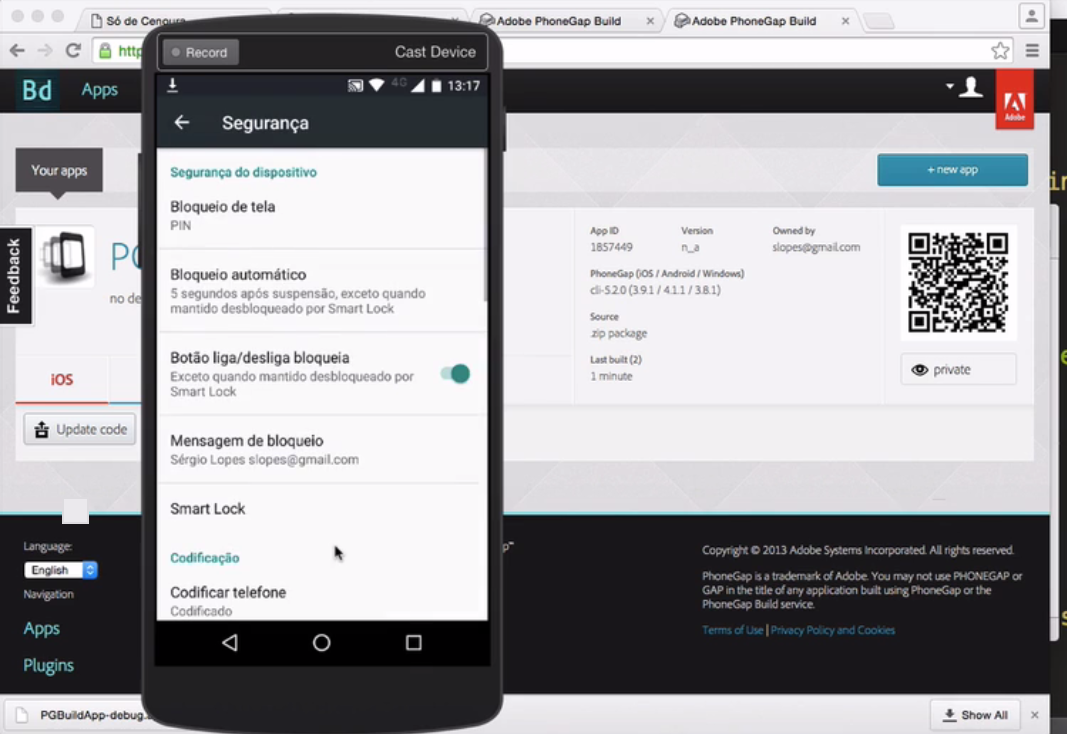
Agora queremos testar o aplicativo em um aparelho de celular. Veremos a tela do meu celular. Aproveitaremos para instalar o app no aparelho. o PhoneGap Buil oferece a facilidade de já disponibilizar o QR Code, que podemos scanear com o celular.



Ele irá executar o QR Code, iremos executar, e o celular irá abrir o arquivo .apk que nós geramos no computador. Ele pedirá permissão para fazer o download, daremos "OK" e o arquivo será baixado. Depois, faremos a instalação.

Para fazer o download de um aplicativo fora da loja, teremos que ir até as "Configurações" do seu celular, até me "Segurança".

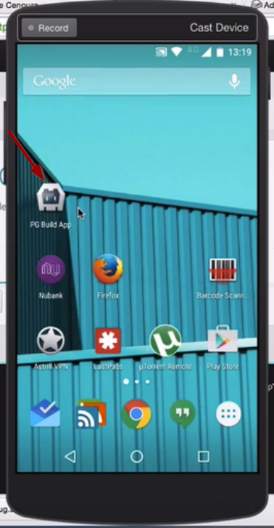
Iremos até a opção "Fontes desconhecidas". Com isso, poderemos abrir a instalação para aplicativos fora da loja. Aparecerá uma mensagem de segurança, mas o nosso aplicativo é seguro e podemos desabilitar a opção. Ele fará o processo de instalação no aparelho.



Vamos abrir o aplicativo. Veremos adiante como configurar nome e ícone.



A mesma app que desenvolvemos em HTML e CSS que funcionou no navegador, também rodou no aplicativo. Se buscarmos o app no menu do celular, ele terá um nome esquisito de **"PG Build App"**, mas podemos colocar o ícone na Homescreen.



Ele terá o funcionamento normal de um app nativo, sendo irrelevante se foi feito em HTML e CSS.

Resumindo, o PhoneGap Build é uma forma fácil de "buildar" o app, tornando desnecessário a configuração da geração de ambiente do Android ou iOS no computador. Podemos usar o serviço da Adobe, que tem a opção gratuita, simples de trabalhar.

**Configurando o PhoneGap**

No site do PhoneGap Build, vemos que ele gerou um nome padrão, "PG Build App", e inseriu um ícone esquisito. Nós podemos configurar estes elementos. Veremos como fazer tais ajustes.

No terminal, abriremos o projeto, em que temos HTML e o CSS. Criaremos um arquivo novo chamado config.xml, com algumas configurações para o nosso aplicativo.

Vamos criar o arquivo xml, que irá configurar o início do arquivo. Para isto, ele seguirá a especificação do WDC, com a tag widget - que receberá algumas configurações como o nome (name) .

<?xml version="1.0" encoding="UTF-8" ?>

<widget>

<name>Só de Cenoura</name>

</widget>

Quando trabalhamos com um xml precisamos indicar os cabeçalhos, com os namespaces. Iremos adicionar o xmlnss com o endereço do <http://www.w3.org/ns/widgets>.

<?xml version="1.0" encoding="UTF-8" ?>

<widget xmlns="http://www.w3.org/ns/widgets">

<name>Só de Cenoura</name>

</widget>

Também vamos passar outro namespace, chamado gap, onde posso colocar algumas extensões específicas do phonegap.

<?xml version="1.0" encoding="UTF-8" ?>

<widget xmlns="http://www.w3.org/ns/widgets" xmlns:gap="http://phonegap.com/ns/1.0"

>

<name>Só de Cenoura</name>

</widget>

Podemos colocar os namespaces na mesma linha.

Existem outros dois parâmetros importantes que precisamos incluir. O primeiro é a versão (version) do app.

Para termos o controle da quantidade de versões do aplicativo PG Build App, vamos especificar o número do app no código. Neste caso, iremos usar a versão "1.0.0", mas podemos ir implementando outros números como "1.0.1" ou "1.0.2". Outro atributo importante é id, que irá configurar qual será o nome do pacote do app. Esse pacote deve ser igual para todos os aplicativos. Geralmente, utilizamos o nome do pacote ao contrário. Por exemplo, o meu site chama sergiolopes.org, logo usarei org.etec e depois acrescentaremos nome do app cenouraapp.

<?xml version="1.0" encoding="UTF-8" ?>

<widget xmlns="http://www.w3.org/ns/widgets" xmlns:gap="http://phonegap.com/ns/1.0" version="1.0.0" id="org.etec.cenouraapp"

>

<name>Só de Cenoura</name>

</widget>

Sugerimos que quando você for fazer o exercício, utilize um id diferente, baseado no seu próprio domínio. Caso contrário, se você tentar subir seu aplicativo para um loja, ele não será aceito.

Após configurado o cabeçalho, já sabemos como trabalhar com o name, podemos incluir outras tags. Vamos trabalhar com a description que permitirá incluir descrições maiores na configuração do aparelho.

<description>

Menu do restaurante Só de Cenoura

</description>

Podemos incluir o nome do autor do app.

<author href="http://etec.org" email="qualquer@etec.org">

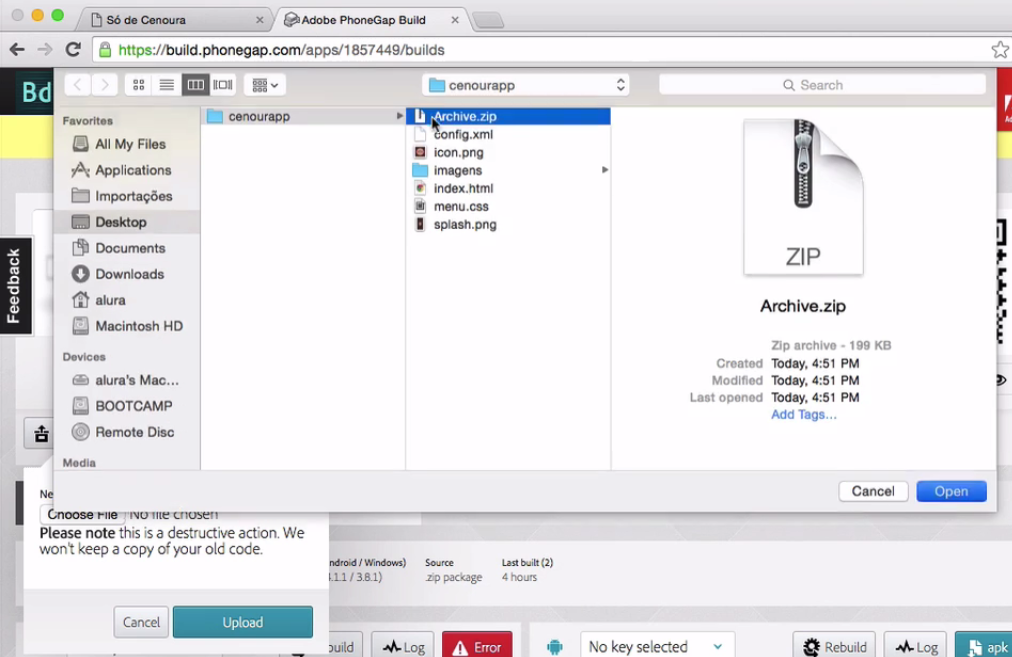
Etec

</author>

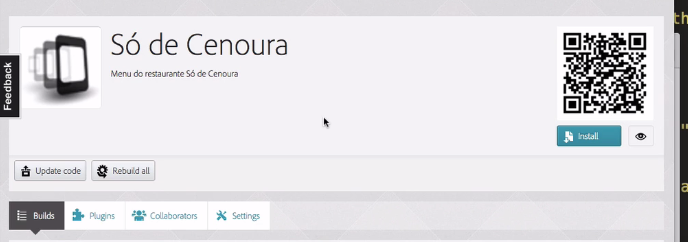
Meu xml ficou na raiz do projeto, junto com o HTML.

Para subir o projeto no PhoneGap Build, iremos zipar o arquivo novamente, mas incluindo o arquivo config.xml.

Voltaremos para o site do PhoneGap Build, clicaremos em Update code, depois em Choose File e selecionaremos o arquivo para o upload.

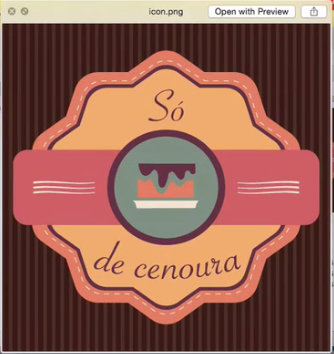


Nosso arquivo irá aparecer já com o nome e a descrição alterados.

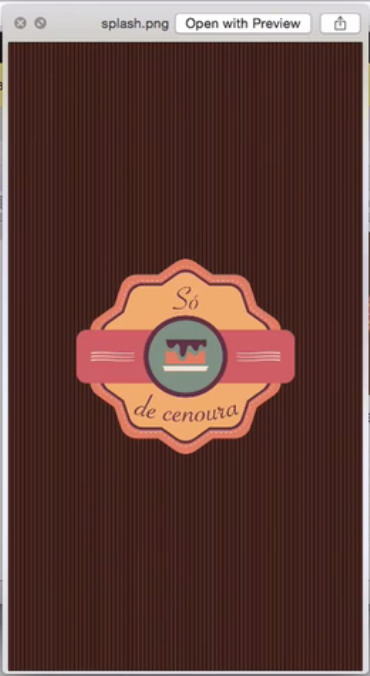


Se fizéssemos a instalação do aplicativo, ele teria o nome novo.

Em seguida iremos configurar o ícone. No projeto, temos um arquivo chamado icon.png, que tem uma imagem do "Só de Cenoura".



Este será o ícone que estará na Home do usuário. Temos também outro arquivo, chamado splash.png, que é a tela de carregamento (splash screen).



A nossa imagem foi criada com as dimensões para Android.

Para usar o ícone, precisamos voltar na aba XML e utilizar uma configuração apontando a imagem explicitamente. Criaremos a tag icon, que irá receber o source. Acrescentaremos também o nome do arquivo.

<icon src="icon.png" />

Para trabalhar com a splashscreen, usaremos a tag do PhoneGap, gap:splash, com o nome do arquivo.

<plugin name="cordova-plugin-splashscreen" />

<splash src="splash.png" />

Nosso xml está assim:

<?xml version="1.0" encoding="UTF-8" ?>

<widget xmlns="http://www.w3.org/ns/widgets" xmlns:gap="http://phonegap.com/ns/1.0" version="1.0.0" id="org.etec.cenouraapp"

>

<name>Só de Cenoura</name>

<description>

Menu do restaurante Só de Cenoura

</description>

<author href="http://etec.org"

email="qualquer@etec.org"> Etec

</author>

<icon src="icon.png" />

<plugin name="cordova-plugin-splashscreen" />

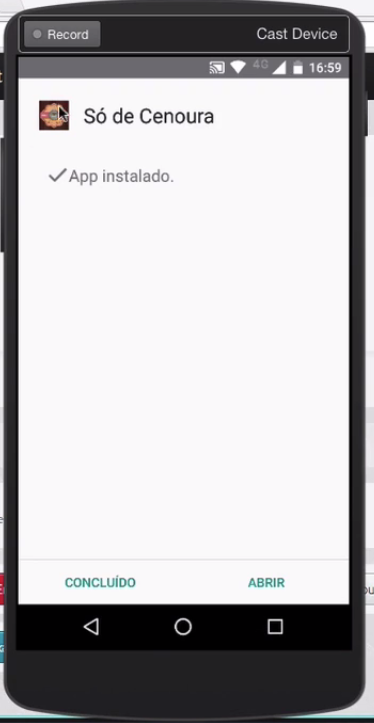
<splash src="splash.png" />

</widget>

Vamos testar o novo código. Vamos apagar o arquivo .zip anterior e criar outro com as alterações feitas no xml. Depois, faremos o upload da versão mais recente do PhoneGap Build. Com a nova configuração, agora aparece o ícone do "Só de Cenoura".



Se baixarmos a última versão do arquivo .apk no celular, também veremos o ícone na Home. Repetiremos o processo de instalação feito antes, com o QR Code. Ele irá abrir o navegador e irá baixar o arquivo. Iremos instalá-lo e já poderemos visualizar o ícone.

Qual é a grande diferença agora? Se formos no menu, o novo ícone já estará disponível para colocarmos na Home.



Ao clicarmos no ícone, veremos que o aplicativo funciona corretamente.

Ainda temos outras configurações adicionais que podem ser feitas no .xml, como a tag preference. Ela permite criar especificações do app, por exemplo, que o usário não rotacione a página e utilize apenas o modo retrato. Iremos incluir uma name="orientation" com o valor igual a retrato (portrait).

<preference name="orientation" value="portrait" />

Se "buildarmos" novamente o app, o usuário não poderá mais rotacionar a visualização. Podemos criar outra preferência para que o aplicativo seja fullscreen e ocupe a tela completa - o que pode ser uma opção interessante para um cardápio.

<preference name="orientation" value="portrait" />

<preference name="fullscreen" value="true" />

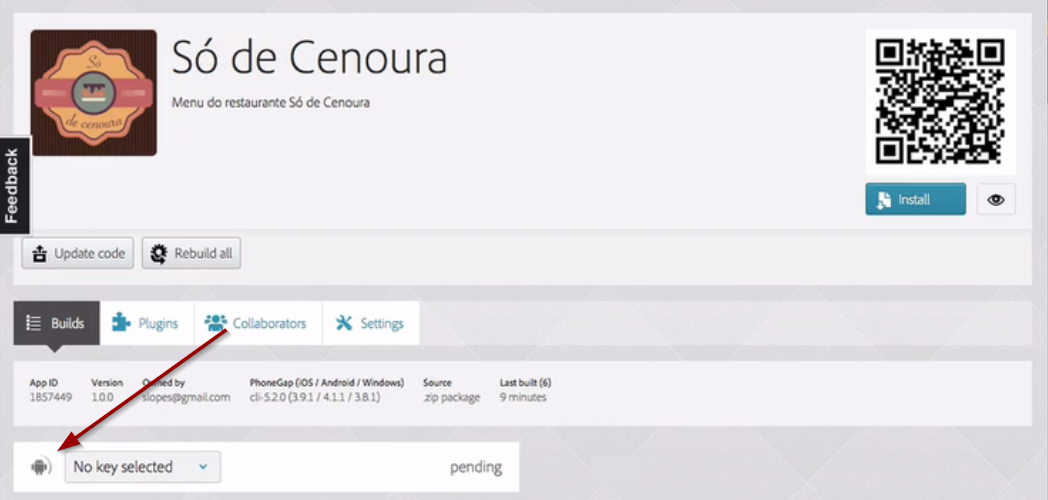
Temos outras configurações, para alterar as cores, por exemplo. Ainda que o PhoneGap Build nos ofereça a opção de "buildar" o aplicativo para outras plataformas, podemos especificar no código que ele funcionará apenas para Android.

<platform name="android" />

Ele irá "buildar" apenas para plataforma Android

Você encontrará mais detalhes sobre preferências, nos [exercícios](https://cursos.alura.com.br/course/cordova-phonegap/task/14251).

Agora vamos subir a nova versão do arquivo no PhoneGap Buil. Observe que na plataforma, ele só irá permitir "buildar" para Android. Ele seguiu corretamente o que configuramos no xml.



Vamos repetir novamente o processo de instalação com o QR Code, ele baixará a versão mais recente do aplicativo. Após a instalação ser finalizada, iremos abrir o aplicativo. Agora, o aplicativo estará fullscreen.



Não podemos visualizar o botão de voltar, nem a barra de status. Essa preferência não é recomendável para qualquer tipo de app, mas no nosso caso, queremos oferecer uma experiência imersiva. Por exemplo, o restaurante disponibilizará tablets que serão usados simplesmente como cardápios de pedidos. Não é necessário que o usuário saia do aplicativo. Também, não será possível rotacionar o aplicativo. Neste caso, trabalhar com estas preferências pode ser interessante.