01

Cordova e além

Transcrição

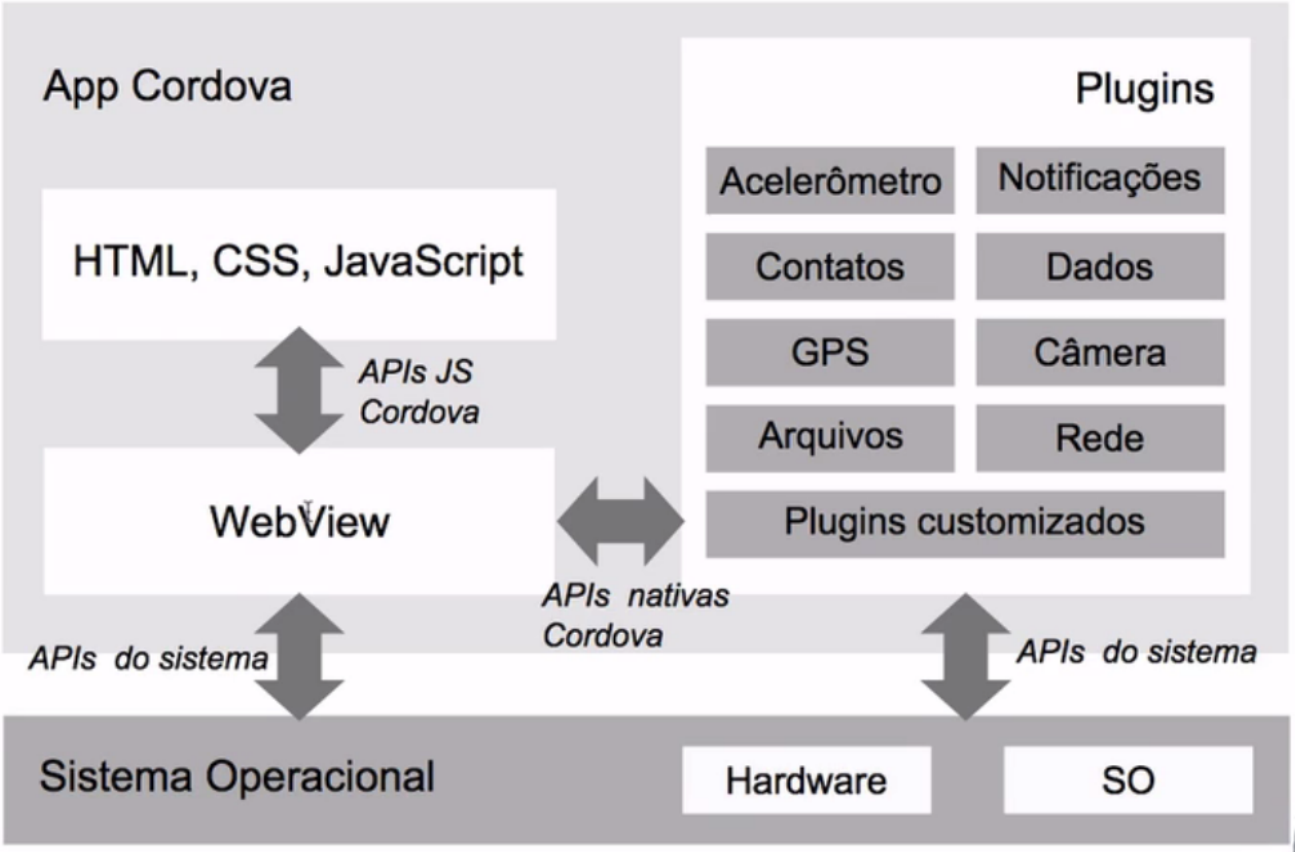
Estamos chegando ao final do curso de Cordova & PhoneGap, mas gostaria de sedimentar alguns conhecimentos interessantes. O nosso aplicativo está rodando no dispositivo móvel e tem a funcionalidade de anotar os pedidos dos clientes, além de se comunicar com o Ajax e ler códigos de barras com a câmera. Nós conseguimos!

Mas afinal, o que é o Cordova e quais problemas ele irá resolver? Com o Cordova, criamos um aplicativo multiplataforma - que roda em Android, WindowsPhone e iOS -, baseado em HTML, CSS e JavaScript. O segredo do êxito do Cordova é a forma como ele renderiza estas linguagens, que são as *WebViews*. Este é um termo que representa um motor de renderização de HTML, dentro das plataformas nativas. Tanto o iOS, como o Android e WindowsPhone possuem além de um navegador usado pelos usuários, um renderizador destas linguagens. Ou seja, ele é um navegador sem interface. O WebView é uma tela em branco, capaz de executar um código HTML.

Diversos apps nativos costumam usar as WebViews em diferentes partes. Por exemplo, quando clicamos em um link no aplicativo do Facebook, ele abre uma WebView dentro do post específico. Porém, este é um app nativo. A grande inovação feita pelo Cordova & PhoneGap foi a possibilidade de renderizar o app completo em uma WebView. Você até pode utilizar elementos nativos na sua interface, mas com o Cordova isto é opcional. Ele permite que você tenha a interface inteira escrita em HTML e CSS, renderizado dentro da WebView. E ele irá renderizar o conteúdo.

Podemos ainda acrescentar em cima deste HTML e CSS, novos plugins - que darão novas capacidades que podem eventualmente, comunicar-se com o sistema operacional. Mostramos a página de plugins do site do Cordova, onde podemos encontrar vários plugins. Vários dos que foram utilizados por nós aqui, fazem parte da lista dos plugins mais utilizados. Vimos como é o processo de busca e instalação dos plugins e como é possível encontrar orientações consultando as documentações disponibilizadas no site. Para todas as funcionalidades que quisermos implementar, procuramos por um plugin e descobrimos como utilizá-lo.

Abaixo temos um esquema, para entender o funcionamento de um app Cordova.



Observe, nós escrevemos o HTML, CSS e JavaScript, que serão renderizados na WebView do nosso sistema operacional. No caso do iOS, o WebView será baseado no WebKit do Safari. Nas versões mais recentes do Android, teremos um WebView baseado no Chromium - projeto do Google, por trás do Chrome. No Windows 10 temos uma WebView baseada no Microsoft Edge. Geralmente, elas são baseadas no navegador padrão de cada plataforma. A WebView irá renderizar o HTML, CSS e JS como se fosse um arquivo local (`file:`), sem um servidor. O usuário acessará APIs JavaScript do Cordova, para ter acesso a funções nativas dos plugins. Vimos isto com o `bascodescanner` , por exemplo. Nós acessávamos o `cordova-plugin-barcodescanner` , que era feito através do JS acessando algum elemento no sistema operacional.

Podemos instalar vários plugins, como os de: **Notificações, Acelerômetro, Contatos, GPS e muitos outros. Inclusive, temos a opção de criar os nossos próprios plugin customizado. Se ele for usar elementos nativos, precisaremos conhecer programação nativa. Para Android, conhecer Java, para iOS, conhecer Objective C. Se você tem este tipo de conhecimento, poderá criar um plugin também. A sua Webview poderá acessar os plugins instalados. Além disso, tanto um como outro, permitiram o acesso ao sistema operacional, onde poderemos utilizar elementos da plataforma, de Hardware - incluindo a câmera e a vibração. Nosso esquema parece complexo, mas basicamente, o que fazemos é escrever o HTML, CSS e JavaScript.

Outro detalhe que gostaria de recapitular é o estado atual do nosso projeto. Quero que você entenda a estrutura de arquivo que o projeto Cordova tem. A maior parte do tempo, iremos trabalhar com a pasta `www` , onde encontramos os arquivos `css` , `js` , `font` , `icons` . Seguindo o padrão, criamos um `index.html` que será a página raiz, a partir dele, você criará a sua app. Nela serão importadas o `css` , o `js` , as imagens, os ícones. Uma curiosidade é que o nome da pasta `www` parece estar relacionado com Web, mas é apenas uma impressão. Cordova não é Web.* *Inclusive, preferia que ela tivesse recebido outra nomeação. O Cordova é um aplicativo, sem `http` , `url` , `link` . Nenhuma parte integrante do *Core* da Web integra também o Cordova. Ele está mais próximo para App, do que para Web, a maior diferença é que usamos as mesmas linguagens.

Outro arquivo que também será bastante utilizado é o `config.xml` , nos falamos muito sobre ele no curso. É nele que incluímos nossas configurações e necessidades.

O projeto também contêm outras pastas. Temos a `resources`, onde geramos os ícones e *splash screens* para Android e iOS. Na pasta `plugins`, estão os plugins instalados. Não é recomendável "comitar" esta pasta no seu repositório. É preferível deixar o nome do plugin no arquivo `xml` e assim, ela será gerada novamente por quem baixar o projeto, a partir do arquivo `xml`. A pasta `platforms` também não é interessante ser comitada. Nela, temos uma lista de todas plataformas que estão sendo usadas: Android, iOS e outras. Nós não abordamos tanto no curso, mas dentro das pastas temos os projetos criados para Android Studio, Xcode e VisualStudio. Ou seja, você tem disponível todos os projetos nativos, caso queira fazer alguma customização.

Temos ainda a pasta `hooks`, nós não a usamos, ela está vazia. Se você tiver o interesse em pesquisar, poderá incluir alguns *scripts* que serão executados automaticamente pelo Cordova e que terá determinadas utilidades. Por exemplo, antes de buildar um Android, ele rodará um certo script, ou antes de executar o iOS, ele rodará um outro script. Com isto, automatizamos algumas tarefas, mas no nosso caso, não foi necessário.

Revisamos nossa estrutura de arquivos, agora, gostaria que pensássemos em como escrever um bom app híbrido para o Cordova. Observe que estamos utilizando as linguagens para Web (HTML e CSS), mas não estamos na Web. Quando trabalhamos com o Cordova, temos que gerar algo específico para o formato de aplicativo.

Por esta razão, no app "Só de Cenoura", usamos o Materialize - o *framework* de Material Design. O nosso projeto ganhou uma "cara" de app, com características que os usuário de Android, por exemplo, estão acostumados a encontrar. Mesmo se o aplicativo for usado em iOS, ele terá um comportamento de app. Então, nós precisamos ter um design de app, mas não necessariamente um design nativo. Podemos usar um design próprio - inclusive, é um cenário preferível -, mas ele precisará ser reconhecido como app. Um dos comportamentos mais característicos do app é que ele não tem navegação. Quando trocamos de abas, não vamos para outro HTML. Qualquer interação com o app será feita no mesmo HTML. Em um projeto mais complexo, não poderemos utilizar apenas um único arquivo HTML. Mas podemos simular este tipo de comportamento, usando telas secundárias através de *Ajax* - é o que chamamos de "Single-page Applications". O importante, é não termos a sensação de *refresh* na tela do aplicativo. Caso isto aconteça, o usuário terá a sensação de Web e isto não é positivo.

O nosso projeto era simples e o Materialize supriu nossas necessidades, mas em outros mais complexos, talvez utilizar o ***Angular*** fosse recomendável. Temos um curso específico sobre *AngularJS* (https://www.alura.com.br/curso-online-angularjs-mvc), do Alura. Inclusive, se você quiser um framework, que já tenha um Angular e que tenham componentes visuais característicos para dispositivos mobile, você pode usar frameworks mais completos. Um deles é o Ionic, que possui inúmeras utilidades. Com ele, você poderá criar aplicativos mobile baseados em Angular, com componentes parecidos aos nativos. O Ionic é o mais famoso, mas existem outros frameworks.

O essencial é tentar pensar que o seu projeto tenha um perfil de app, com a sensação de app. Isto determinará o sucesso para os eu usuário.

Parabéns por ter concluído o curso. Sucesso na construção dos seus apps! E não deixe de fazer os exercícios do capítulo.