PROGRAMAÇÃO MOBILE

SOLUÇÕES EDUCACIONAIS INTEGRADAS

Web apps com Apache Cordova

Hygo Sousa de Oliveira

OBJETIVOS DE APRENDIZAGEM

- Caracterizar os aplicativos *web apps* e *progressive web apps*.
- Descrever o Apache Cordova, sua arquitetura e seus componentes.
- Desenvolver um projeto de *web app* com Apache Cordova.

Introdução

Os aplicativos para *smartphone* têm ganhado diversos aprimoramentos, principalmente em relação à forma como eles passaram a ser produzidos, deixando de ser construídos pelos métodos tradicionais, por meio de códigos nativos. Nesse sentido, é comum haver similaridade entre aplicativos *web* e aplicativos nativos, com a diferença de que aplicativos nativos normalmente disponibilizam o melhor dos recursos do dispositivo, enquanto os dispositivos *web* alcançam bem mais dispositivos, devido à sua natureza multiplataforma. *Sites* responsivos também são bastante similares a aplicativos nativos e, principalmente, a aplicativos *web*.

Neste capítulo, você vai estudar as aplicações *web apps* e *progressive web apps*. Vai ver a arquitetura e os componentes do Apache Cordova, além de desenvolver um *web app* com o Apache Cordova.

Aplicativos *web apps* e *progressive web apps*

Aplicações *web* tradicionais utilizam um servidor, responsável por dar suporte a acessos ou requisições a páginas nele contidas, quando há requisições de usuários. Essas páginas podem ser páginas estáticas com HTML puro ou páginas dinâmicas; também podem existir seções de códigos, responsáveis por acessar uma base de dados para atualizar determinado conteúdo. O processo de requisição acontece da seguinte forma: um usuário faz uma requisição, via navegador de sua preferência, ao servidor, que, por sua vez, recupera o conteúdo armazenado e o envia para o navegador requerente. Isso significa que o navegador do cliente não faz qualquer esforço com as informações, tendo apenas a tarefa de apresentar as informações recuperadas pelo servidor.

Uma forma de atribuir maior poder de manipulação aos navegadores cliente está na possibilidade de maior interação entre os navegadores cliente e os servidores que o suporte à linguagem JavaScript oferece. De acordo com Wargo (2013), com o advento da Web 2.0, a carga de dados ao servidor tendeu a diminuir, considerando que código com JavaScript em execução no navegador do cliente pode gerenciar a requisição e a apresentação dos dados. Aplicações *web* com JavaScript são possíveis graças à adição da interface de programação de aplicações (API, *application programming interface*) denominada XMLHTTPRequest (XHR), a qual permite que uma aplicação *web* submeta requisições assíncronas para um servidor e processe dados que retornam do servidor sem interrupção da atividade do usuário com a aplicação.

Esse tipo de aplicação se torna cada vez mais interessante, à medida que ganha comportamento e aparência de aplicações nativas. Em aplicações voltadas para dispositivos móveis, a Web 2.0 tem gerado muitos benefícios, principalmente pelo controle das informações repassadas. Além disso, o HTML5 trouxe novos recursos, que permitem que um aplicativo opere com mais eficiência em um dispositivo móvel (ou dispositivos com conectividade limitada) e utilize um banco de dados do lado do cliente para armazenar dados do aplicativo, pois o HTML5 suporta a adição do arquivo manifesto, responsável por listar todos os arquivos que compõem o aplicativo *web* (WARGO, 2013).

Os aplicativos *web apps* se distinguem das aplicações nativas porque, basicamente, um aplicativo nativo já está traduzido em instruções específicas da plataforma e instalado em uma plataforma específica do usuário, podendo estar em uma plataforma iOS ou Android, por exemplo (STEYER, 2017). Enquanto isso, aplicativos *web apps* são independentes de plataformas

e são escritos em três linguagens (HTML, CSS e JavaScript). Além disso, existem os *progressive web apps*, ou *web apps* progressivos, que não precisam ser instalados, compilam muito mais rápido e não requerem aprovação de alguma loja para ser baixado.

Aplicações híbridas

Um aplicativo híbrido é um aplicativo móvel nativo que usa um navegador *web*, geralmente chamado de *webview*, para executar os aplicativos *web*. Ele funciona por meio de um *wrapper* de aplicativo nativo, que interage entre o dispositivo nativo e o *webview*. Como os aplicativos móveis *web*, a maior parte do código pode ser implementada em várias plataformas. Ao desenvolver em uma linguagem comum, é mais fácil manter a base de código, e não há necessidade de aprender uma linguagem de programação completamente nova.

Ao contrário dos aplicativos *web* para celular, os aplicativos híbridos têm acesso total aos recursos do dispositivo, o qual é fornecido por meio de algum *plug-in*. Umas das principais desvantagens desse tipo de solução é que o aplicativo é apenas *web*, ou seja, utilizado pelo navegador, sendo limitado pelo desempenho e pelos recursos do navegador no dispositivo.

Uma tecnologia bastante utilizada em aplicativos híbridos é o Ionic. Conforme Griffith (2017), Ionic Framework adota a abordagem de aplicativo híbrido, com a utilização de componentes de interface de usuário (UI, *user interface*) baseados na *web* que parecem e funcionam como suas contrapartes nativas. Com a estrutura do Apache Cordova e sua biblioteca de *plug-ins*, recursos como o acesso a componentes como câmera do celular, acelerômetro, bússola e etc., foi resolvida.

Saiba mais

Se quiser saber mais sobre Ionic Framework, acesse o *site* oficial.

As aplicações com Ionic Framework são construídas como parte de três camadas de tecnologia: Ionic, Angular (https://angular.io/) e Cordova/Capacitor. Além dessas ferramentas, Ionic suporta tecnologias de React e Vue.

Ionic Framework tem como principal recurso o fornecimento de componentes da interface do usuário que não estão disponíveis para o desenvolvimento de aplicativos baseados na *web*, como, por exemplo, uma barra de guias. Isso só é possível porque Ionic estende a biblioteca HTML para criar uma. Esses componentes são construídos com uma combinação de HTML, CSS e JavaScript, e se comportam e se parecem com os controles nativos que estão recriando. Angular é um projeto de código aberto com suporte, principalmente, do Google; seu objetivo é fornecer uma estrutura MVW (*model-view-whatever*) para construir aplicativos *web* complexos em uma única página.

Saiba mais

Se quiser saber mais sobre Angular, acesse o *site* oficial.

O Apache Cordova é uma estrutura de *software* livre que permite que desenvolvedores de aplicativos móveis utilizem seu conteúdo HTML, CSS e JavaScript para criar um aplicativo híbrido para uma variedade de dispositivos móveis. Conforme Griffith (2017), o Cordova renderiza um aplicativo *web* em um *webview* nativo. Um *webview* corresponde a um componente de aplicativo nativo (como um botão ou uma barra de guias) e é usado para exibir o conteúdo da *web* em um aplicativo nativo. Aplicativos *web apps* em execução no contêiner do Cordova funcionam como qualquer outro aplicativo *web* que seria executado em um navegador móvel: eles podem abrir páginas HTML adicionais, executar código JavaScript, reproduzir arquivos de mídia e se comunicar com servidores remotos, denominados aplicativos híbridos.

Fique atento

Uma das principais diferenças entre Cordova e Capacitor é que o Capacitor exige que os desenvolvedores do aplicativo manuseiem o projeto a partir do aplicativo nativo.

Apache Cordova: arquitetura e componentes

Na última década, houve uma explosão de dispositivos móveis, de *smartphones* a *tablets*, criando um ecossistema de aplicativos que cobrem tudo. Aprender a desenvolver para as plataformas móveis é uma habilidade importante a se adquirir; no entanto não é algo que você possa aprender rapidamente. Além disso, geralmente, as plataformas de desenvolvimento têm suas particularidades, como, por exemplo, o sistema do Google, que utiliza Android SDK e a linguagem Java para o desenvolvimento de aplicações, e o da Apple, que oferece ferramentas para iOS e permite usar Objective-C ou Swift (LOPES, 2016).

A programação nativa de aplicativos móveis (*apps*) significa que você se concentra no *hardware*, incluindo um sistema operacional especial ou apenas uma versão especial dele. Para oferecer suporte a vários sistemas, pode ser necessário criar um aplicativo separado para cada sistema de destino, que é funcionalmente idêntico a um aplicativo existente para outro sistema de destino. Muito provavelmente, você terá que criar o aplicativo para uma plataforma na linguagem de programação A e para o outra plataforma na linguagem de programação B, e isso pode significar um esforço imenso (STEYER, 2017). Para diminuir esse esforço existe o Apache Cordova, que permite o desenvolvimento de aplicativos móveis com linguagens de programação *web*.

Cordova é uma estrutura de código aberto que permite converter HTML, JavaScript e *cascading style sheets* (CSS) em um aplicativo nativo, que pode ser executado em iOS, Android e outras plataformas móveis (CAMDEN, 2016). O Apache Cordova usa um *wrapper* nativo em torno de um *webview* (uma espécie de navegador integrado), comumente chamado de aplicativo móvel híbrido. Além disso, o Cordova oferece acesso a recursos de *hardware*, como por exemplo, câmera, acelerômetro, GPS, contatos. Visualize, na Figura 1, como os aplicativos móveis híbridos funcionam.

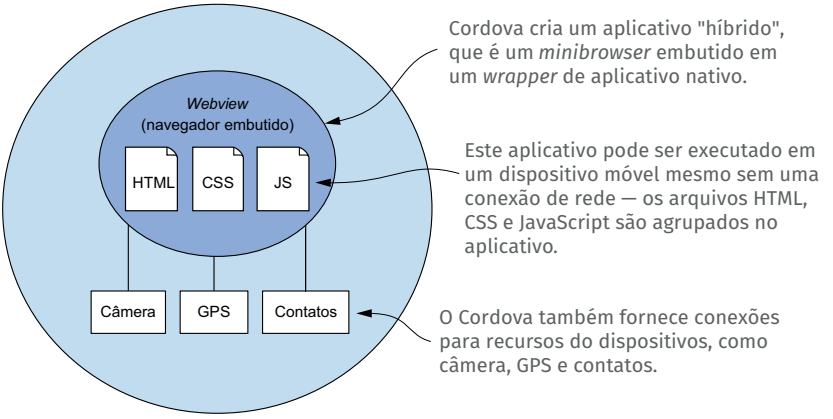

Figura 1. Funcionamento de um aplicativo híbrido.
Fonte: Camden (2016, p. 4).

O Cordova gera aplicativos nativos a partir de código *web* puro; nas outras estruturas mencionadas, tais aplicativos podem ser usados exclusivamente por meio de um navegador na plataforma móvel. Para esse propósito, uma interface de função externa (FFI, *foreign function interface*) é fornecida a um motor WebView ou WebKit embutido (grosso modo, o navegador *web*) no dispositivo móvel (STEYER, 2017). O FFI é um mecanismo com o qual um programa pode chamar ou usar rotinas ou serviços que foram escritos em outra linguagem de programação. Existem vários componentes para um aplicativo Cordova. A Figura 2 mostra uma visão de alto nível da arquitetura do aplicativo Cordova.

O componente **WebView** do Cordova pode fornecer ao aplicativo toda a interface de usuário em outras plataformas. Ele também pode ser um componente dentro de um aplicativo híbrido maior, que combina o WebView com componentes de aplicativos nativos (STEYER, 2017). Outro componente muito importante é **Web App**, responsável por conter o código do aplicativo implementado como uma página *web*, geralmente, um arquivo local chamado index.html, que faz referência a CSS, JavaScript, imagens, arquivos de mídia ou outros recursos necessários para sua execução. Conforme Cordova (2020), o aplicativo é executado em um *webview* dentro do *wrapper* de aplicativo nativo, que você distribui para lojas de aplicativos. Os *plug-ins* fornecem uma interface para que o Cordova e os componentes nativos se comuniquem entre si e façam ligações para APIs de dispositivo padrão.

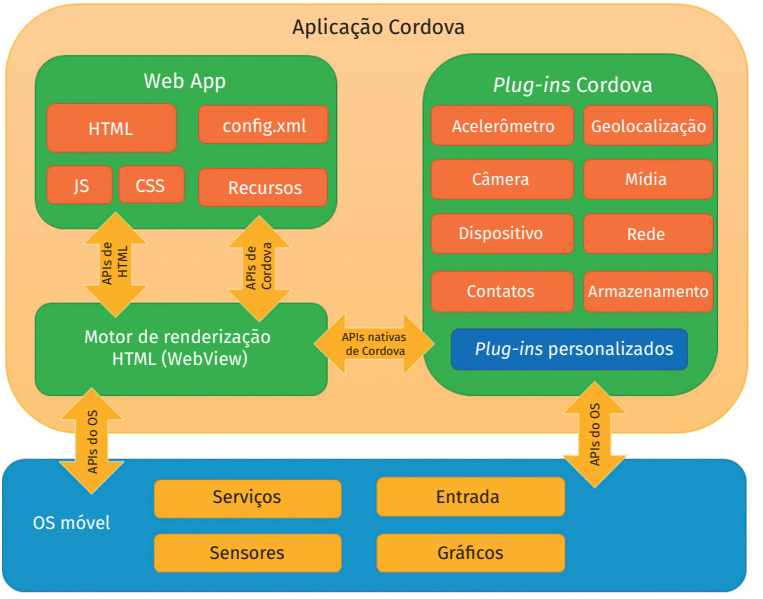

Figura 2. Arquitetura de Apache Cordova.
Fonte: Documentação (c2015, documento *on-line*).

Saiba mais

Existem outras opções para construir aplicativos móveis sem código nativo, como, por exemplo, o Titanium da Appcelerator. Pesquise mais a respeito na *web*.

Desenvolvimento de um *web app* com Apache Cordova

Instalação do Cordova

Para iniciar o desenvolvimento de um *web app* com o Apache Cordova, antes, é preciso instalar o Cordova, caso ainda não esteja instalado. Para realizar a instalação do Cordova, siga as etapas apresentadas a seguir, conforme sua documentação.

Saiba mais

Você encontra a documentação do Apache Cordova no *site* oficial do *framework*.

1. Baixe e instale Node.js. Na instalação, você deve ser capaz de invocar o *node* `npm` na linha de comando. Após instalado, verifique se a instalação foi bem-sucedida, verificando a versão dele com o comando `node -v`.
2. Instale o módulo Cordova utilizando o comando `npm` utilitário do Node.js. O módulo Cordova será baixado automaticamente pelo `npm` utilitário.
   - Para sistemas OS X e Linux (terminal):

```
$ sudo npm install -g cordova
```

   - Para Windows (*prompt* de comando):

```
C:\>npm install -g cordova
```

3. Digite o comando `cordova` no terminal/*prompt* para verificar se o Cordova foi corretamente instalado.

Saiba mais

Para baixar o Node.js, acesse o *site* oficial Nodesj.

Criação do aplicativo

Para a criação do aplicativo, usaremos um exemplo fornecido por Lopes (2016): a criação de um cardápio em um aplicativo *mobile* para uma *startup* de confeitaria especializada em bolos de cenoura *gourmet*. O objetivo do aplicativo é utilizar as imagens do cardápio da confeitaria, frente e verso, como mostrado na Figura 3. Nosso aplicativo precisa mostrar essas imagens e oferecer uma forma simples de navegação para o usuário trocar de página.

Figura 3. Imagens para o aplicativo.
Fonte: Lopes (c2021, documento *on-line*).

O primeiro passo para a criação do aplicativo é ir para o diretório onde você mantém seu código-fonte e criar um projeto Cordova, digitando o seguinte comando no CMD:

```
C:\Users\usuario>cordova create cenoura com.example.cenoura
Cenoura
```

Isso cria a estrutura de diretório necessária para seu aplicativo Cordova. Por padrão, o Cordova Create gera uma estrutura de aplicativo baseado na *web* cuja *home page* é o www/index.html. Para realizar a troca de páginas do cardápio, é possível utilizar dois *radio buttons* com seus respectivos *labels*. Agora você vai editar o arquivo index.html, deixando-o com a estrutura apresentada na Figura 4.

```html
<!DOCTYPE html>
<html>
    <head>
        <meta charset="utf-8">
        <meta name="viewport" content="width=device-width,initial-scale=1.0">
        <link rel="stylesheet" href="menu.css">
    </head>
    <body>

        <input type="radio" name="opcao" id="opcao-bolos" checked>
        <label for="opcao-bolos">Bolos</label>

        <input type="radio" name="opcao" id="opcao-bebidas">
        <label for="opcao-bebidas">Bebidas</label>

        <div class="container-menus">
            <img src="imagens/menu-bolos.png" alt="Nossos bolos" id="menu-bolos" class="menu">
            <img src="imagens/menu-bebidas.png" alt="As Bebidas" id="menu-bebidas" class="menu">
        </div>

    </body>
</html>
```

Figura 4. Edição do arquivo index.html.

Feita a estrutura HTML, agora você vai criar o arquivo CSS. Usando CSS, é possível mostrar somente a imagem que estiver selecionada, utilizando seletores avançados do CSS3. Com a pseudoclasse `:checked`, você sabe qual opção está selecionada, como mostra a Figura 5.

```css
/* funcionamento */
#opcao-bolos:checked ~ #menu-bebidas,
#opcao-bebidas:checked ~ #menu-bolos {
    display: none;
}
```

Figura 5. Criação do arquivo CSS e pseudoclasse `:checked`.

Para evitar o estouro de pixels das imagens, você vai configurar a largura máxima delas e esconder os *input radio*, para usar apenas o *label* para a escolha de opção, como mostra a Figura 6.

```css
input[type=radio] {
    display: none;
}
.menu {
    width: 100%;
}
```

Figura 6. Configuração da largura máxima das imagens.

No próximo passo, você vai deixar os *labels* com cara de botões, colocar alguns ícones bonitos e acertar outras coisas decorativas. Acompanhe a Figura 7.

```
/* decoracao */
body {
    background: #3D1A11;
    font-family: sans-serif;
    margin: 0;
    text-align: center;
}

label {
    background: center top no-repeat #563429;
    background-size: 4em;
    color: white;
    display: block;
    font-size: 75%;
    padding: 4em 0 1em;
    text-transform: uppercase;
}
    label[for=opcao-bolos] {
        background-image: url(imagens/icone-bolos.svg);
    }
    label[for=opcao-bebidas] {
        background-image: url(imagens/icone-bebidas.svg);
    }
    :checked + label {
        background-color: #E4876D;
    }
```

Figura 7. Ajustes de elementos decorativos.

Por fim, você vai posicionar os botões fixamente, na parte inferior da tela, de lado a lado, como mostra a Figura 8.

```
/* posicionamento */
label {
    width: 50%;

    position: fixed;
    bottom: 0;
    z-index: 1;
}
    label[for=opcao-bolos] {
        left: 0;
    }
    label[for=opcao-bebidas] {
        right: 0;
    }

.menu {
    margin-bottom: 100px;
}
```

Figura 8. Posicionamento dos botões.

Agora você vai adicionar alguns efeitos, por exemplo, fazer uma transição entre as duas telas, escorregando da direita para a esquerda, com CSS Transitions. Veja a Figura 9.

```
/* animado */
html,
body {
    overflow-x: hidden;
    width: 100%;
}
.container-menus {
    -webkit-transform: translateX(0);
            transform: translateX(0);
    -webkit-transition: transform 300ms ease;
            transition: -webkit-transform 300ms ease;
    width: 200%;
}
.container-menus .menu {
    float: left;
    width: 50%;
}
#opcao-bebidas:checked ~ .container-menus {
    -webkit-transform: translateX(-50%);
            transform: translateX(-50%);
}
}
```

Figura 9. Criação de efeito com CSS Transitions.

Com a página HTML criada e personalizada com o CSS, você vai voltar para o CMD. Todos os comandos subsequentes precisam ser executados no diretório do projeto ou em qualquer subdiretório. Para isso, basta digitar os seguintes comandos:

```
C:\Users\usuario>cd cenoura
```

Adicione as plataformas que você deseja direcionar ao seu aplicativo. Vamos adicionar a plataforma "android":

```
C:\Users\usuario\cenoura>cordova platform add android
```

Para verificar se você atende aos requisitos de construção da plataforma, digite o seguinte:

```
C:\Users\usuario\cenoura>cordova requirements
```

É preciso instalar todas as dependências indicadas pelo resultado da execução do comando. Execute o seguinte comando para construir o projeto para todas as plataformas:

```
C:\Users\usuario\cenoura>cordova build
```

Execute um comando como o seguinte para recriar o aplicativo e visualizá-lo dentro de um emulador de plataforma específica:

```
C:\Users\usuario\cenoura>cordova emulate android
```

Visualize, na Figura 10, uma ilustração de como ficou a aplicação desenvolvida com o Apache Cordova.

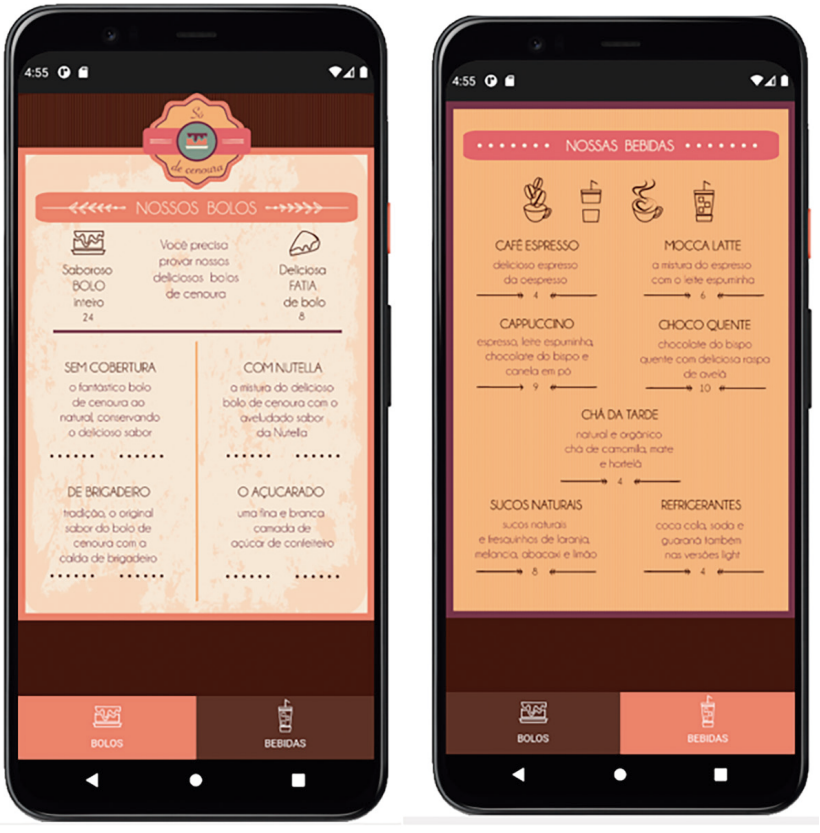

Figura 10. Aplicativo Cardápio.

A utilização do Cordova pode diminuir esforços no desenvolvimento de aplicativos para múltiplas plataformas móveis. Ao criar aplicativos por meio do Cordova, você pode utilizar linguagens *web* sem a necessidade de dominar cada uma das plataformas de desenvolvimento de aplicativos móveis, o que possibilita criar uma única versão, que pode ser usada para as plataformas escolhidas durante o desenvolvimento do aplicativo com o Cordova.

Referências

CAMDEN, R. C. *Apache Cordova in Action*. 1st ed. Shelter Island: Manning Publications, 2016.

DOCUMENTAÇÃO. *Apache Cordova*, c2015. Disponível em: https://cordova.apache.org/docs/en/latest/guide/overview/index.html. Acesso em: 28 dez. 2020.

GRIFFITH, C. *Mobile App Development with Ionic*. Sebastopol, CA: O'Reilly Media, 2017.

LOPES, S. *Aplicações mobile híbridas com Cordova e PhoneGap*. 1. ed. São Paulo: Casa do Código, 2016.

LOPES, S. cenourapp. *Why GitHub*, c2021. Disponível em: https://github.com/sergiolopes/cenourapp/tree/master/imagens. Acesso em: 20 jan. 2021.

STEYER, R. *Cordova*: Entwicklung plattform neutraler Apps. Switzerland AG: Springer Vieweg, 2017.

WARGO, J. M. *Apache Cordova 3 programming*. Upper Saddle River, New Jersey: Addison-Wesley, 2013.

Fique atento

Os *links* para *sites* da *web* fornecidos neste capítulo foram todos testados, e seu funcionamento foi comprovado no momento da publicação do material. No entanto, a rede é extremamente dinâmica; suas páginas estão constantemente mudando de local e conteúdo. Assim, os editores declaram não ter qualquer responsabilidade sobre qualidade, precisão ou integralidade das informações referidas em tais *links*.

Conteúdo:

SAGAH
SOLUÇÕES EDUCACIONAIS INTEGRADAS