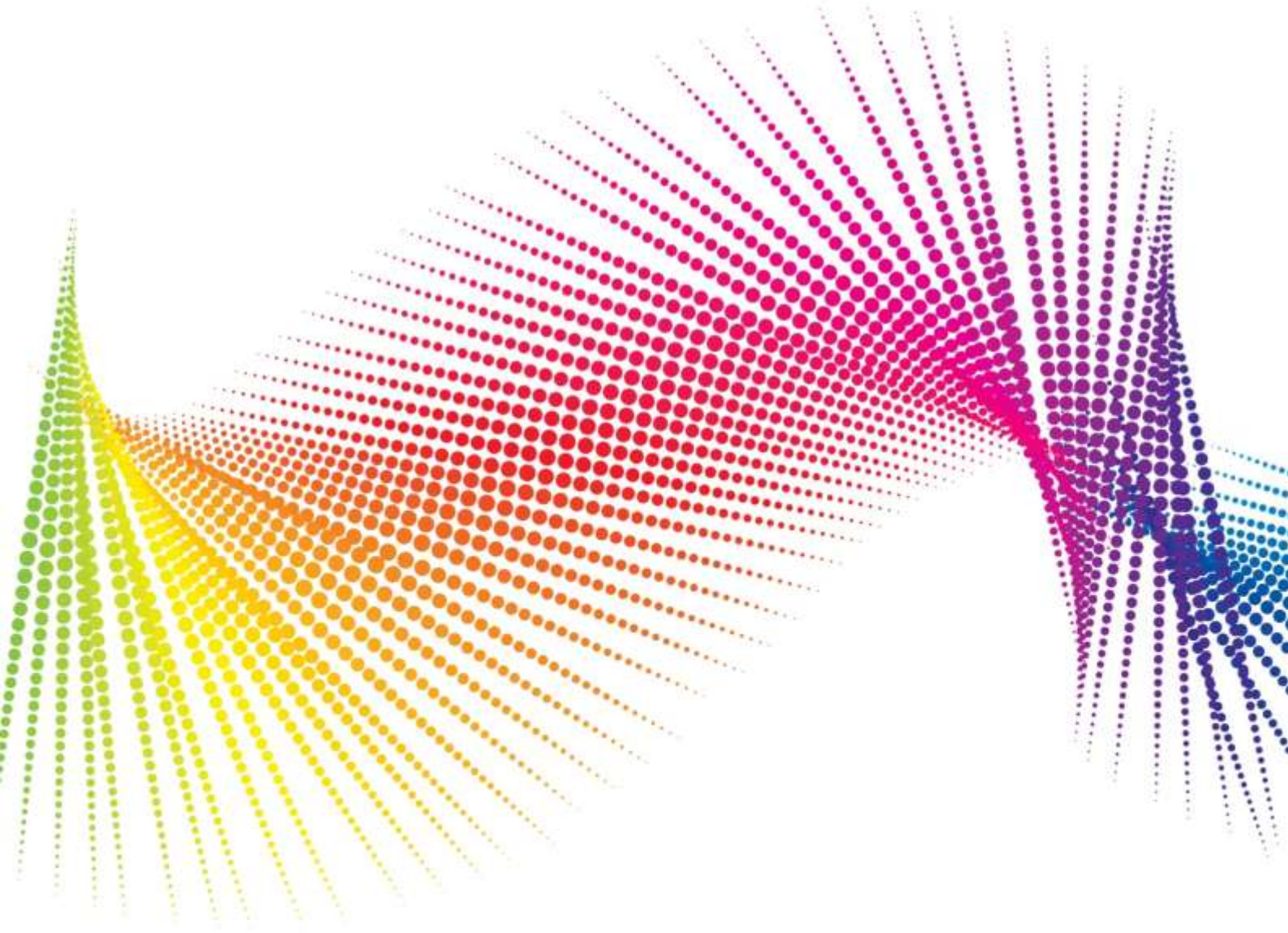
Computação Móvel

Aula 19



Este material é parte integrante da disciplina oferecida pela UNINOVE.
O acesso às atividades, conteúdos multimídia e interativo, encontros virtuais, fóruns de discussão e a comunicação com o professor devem ser feitos diretamente no ambiente virtual de aprendizagem UNINOVE.

Uso consciente do papel.
Cause boa impressão, imprima menos.

Aula 19: Aplicações para web e Android

Objetivo: Introduzir a utilização de acessos à web em aplicações Android que necessitam de informações remotas no aplicativo. Para tanto, abrangeremos a utilização do objeto WebView e seus recursos.

Criando aplicativos web

Se desejarmos utilizar uma aplicação web, como parte de um aplicativo cliente, podemos fazer isso usando WebView. A classe WebView é uma extensão da classe Android View, que permite visualizar páginas da web como parte de um layout em uma atividade. Ele não inclui quaisquer características de um navegador totalmente desenvolvido, como controles de navegação ou uma barra de endereços.

Um cenário comum em que o uso WebView é útil, é quando desejamos fornecer informações em seu aplicativo que precisamos atualizar. Dentro de sua aplicação Android, podemos criar uma atividade que contém um WebView, em seguida, usar isso para exibir o documento que está hospedado online.

Outro cenário em que WebView pode ajudar é se o aplicativo fornece dados para o usuário que sempre requer uma ligação à internet para obter dados, como e-mail. Neste caso, podemos achar que é mais fácil construir um WebView em sua aplicação Android que mostra uma página web com todos os dados do usuário, em vez de executar uma solicitação de rede, em seguida, analisar os dados e torná-lo em um layout Android. Em vez disso, podemos criar uma página web que é adaptado para dispositivos Android e depois implementar um WebView em sua aplicação Android, que carrega a página web.

Adicionando um WebView ao aplicativo

Por exemplo, aqui está um arquivo de layout em que o WebView ocupa a tela:

```
<?xml version="1.0" encoding="utf-8"?>
<WebView  xmlns:android="http://schemas.android.com/apk/res/android"
   android:id="@+id/webview"
   android:layout_width="fill_parent"
   android:layout_height="fill_parent"
/>
```

Para carregar uma página da web no WebView, use loadUrl(). Por exemplo:

```
WebView myWebView = (WebView) findViewById(R.id.webview);
myWebView.loadUrl("http://www.example.com");
```

Para que se possa ter acesso à internet, solicitar a permissão INTERNET em seu arquivo de manifesto. Por exemplo:

```
<manifest ... >
   <uses-permission android:name="android.permission.INTERNET" />
   ...
</manifest>
```

Usando JavaScript no WebView

Se a página da web que planejamos carregar necessita do uso de JavaScript, devemos habilitá-lo para que funcione no WebView. Uma vez que o JavaScript esteja habilitado, também podemos criar interfaces entre o código do aplicativo e o código JavaScript.

Ativando JavaScript

JavaScript está desativado em um WebView por padrão. Podemos habilitá-lo através dos WebSettings anexados ao WebView. Podemos recuperar WebSettings com getSettings(), então habilitar o JavaScript com setJavaScriptEnabled().

Por exemplo:

```
WebView myWebView = (WebView) findViewById(R.id.webview);
WebSettings webSettings = myWebView.getSettings();
webSettings.setJavaScriptEnabled(true);
```

WebSettings fornece acesso a uma variedade de outras configurações que podem ser úteis. Por exemplo, se estamos desenvolvendo uma aplicação web, que é projetado especificamente para o WebView no seu aplicativo Android, então podemos definir uma sequência de agente de usuário personalizada com setUserAgentString(), em seguida, consultar o agente de usuário personalizada na página web para verificar se o cliente que está solicitando sua página web é realmente uma aplicação Android.

Para vincular uma nova interface entre o JavaScript e o código do Android, chamando addJavascriptInterface(), passando uma instância de classe para ligar para o JavaScript e um nome de interface que com o JavaScript podemos chamar para acessar a classe.

Por exemplo, podemos incluir a seguinte classe em uma aplicação Android:

```
public class WebAppInterface {
    Context mContext;

    /** Instantiate the interface and set the context */
    WebAppInterface(Context c) {
        mContext = c;
    }

    /** Show a toast from the web page */
    @JavascriptInterface
    public void showToast(String toast) {
        Toast.makeText(mContext, toast, Toast.LENGTH_SHORT).show();
    }
}
```

Isso cria uma interface Android para JavaScript executando o WebView. Neste ponto, a aplicação web tem acesso à classe WebAppInterface. Por exemplo, aqui está um pouco de HTML e JavaScript que cria uma mensagem de brinde usando a nova interface quando o usuário clica em um botão:

```
<input type="button" value="Say hello" onClick="showAndroidToast('Hello Android!')" />

<script type="text/javascript">
    function showAndroidToast(toast) {
        Android.showToast(toast);
    }
</script>
```

Manuseio de navegação de página

Quando o usuário clica em um link de uma página da web no WebView, o comportamento padrão do Android é mostrar uma aplicação que lida com URLs. Normalmente, o navegador-padrão é aberto e carrega o URL de destino. No entanto, podemos substituir esse comportamento para o WebView, para abrir as solicitações dentro de sua WebView. Sua história página web que é mantido por seu WebView.

Para abrir links clicados pelo usuário, basta fornecer um WebViewClient para o WebView, usando setWebViewClient(). Por exemplo:

```
WebView myWebView = (WebView) findViewById(R.id.webview);
myWebView.setWebViewClient(new WebViewClient());
```

Se queremos mais controle sobre onde a carga um link clicado, crie a sua própria WebViewClient que utilize o método shouldOverrideUrlLoading(). Por exemplo:

```
private class MyWebViewClient extends WebViewClient {
    @Override
    public boolean shouldOverrideUrlLoading(WebView view, String url) {
        if (Uri.parse(url).getHost().equals("www.example.com")) {
            // This is my web site, so do not override; let my WebView load the page
            return false;
        }
        // Otherwise, the link is not for a page on my site, so launch another Activity that handles URLs
        Intent intent = new Intent(Intent.ACTION_VIEW, Uri.parse(url));
        startActivity(intent);
        return true;
    }
}
```

Em seguida, criar uma instância desta WebViewClient novo para o WebView:

```
WebView myWebView = (WebView) findViewById (R.id.webview);
myWebView.setWebViewClient (novo MyWebViewClient ());
```

Histórico da página navegando na web

Quando seu WebView substitui carregamento URL, ele automaticamente acumula um histórico de páginas visitadas. Podemos navegar para trás e para frente ao longo da história com o GoBack() e GoForward().

Por exemplo, veja como a atividade pode usar o botão voltar para navegar para trás:

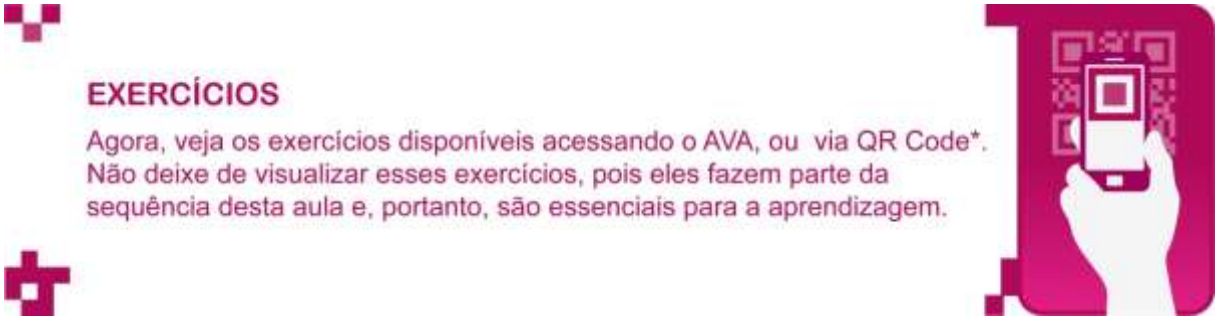
```
    @Override
    public boolean onKeyDown(int keyCode, KeyEvent event) {
        // Check if the key event was the Back button and if there's history
        if ((keyCode == KeyEvent.KEYCODE_BACK) &&
myWebView.canGoBack()) {
            myWebView.goBack();
            return true;
        }
        // If it wasn't the Back key or there's no web page history, bubble up to the
default
        // system behavior (probably exit the activity)
        return super.onKeyDown(keyCode, event);
    }
```

Próxima aula

Para concluirmos nossos conhecimentos sobre o Android, na nossa última aula – **aula 20** – aprenderemos sobre funcionamento do processo de distribuição de sistemas feitos para Android.

Exercício

Para termos certeza de que entendemos a aula, vamos realizar uma atividade sobre aplicações Android que possuem acesso a web.



***** O QR Code é um código de barras que armazena links às páginas da web. Utilize o leitor de QR Code de sua preferência para acessar esses links de um celular, tablet ou outro dispositivo com o plugin Flash instalado.

REFERÊNCIAS

LECHETA, Ricard R. *Android*: aprenda a criar aplicações para dispositivos móveis com o android SDK*. 2*. Ed. São Paulo: Novatec, 2010.

ROGERS, Rick *et al*. *Desenvolvimento de Aplicações Android*. São Paulo: Novatec, 2009.