WebView

Android com Kotlin

Flávio Augusto de Freitas

WebView

Neste projeto, vamos entender como funciona o acesso a recursos da web, usando para isso o componente WebView. Neste exemplo de WebView, aprenderemos como construir aplicativos da web usando WebView no Android. Android WebView é usado principalmente para exibir páginas da web como parte de seu aplicativo. WebView é uma subclasse de View, escrito especialmente para mostrar uma página da web em sua atividade.

Neste artigo, criaremos um aplicativo que contém WebView com ProgressBar. Veremos como lidar com o evento voltar também.

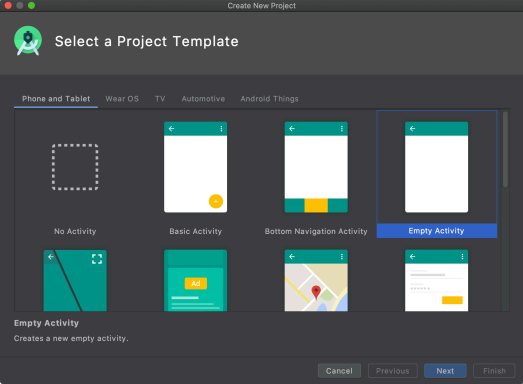
Usos do WebView

O WebView é muito útil se você deseja mostrar algum conteúdo HTML e JavaScript no aplicativo. Como um contrato de usuário final ou um guia do usuário, por exemplo.

Criando um app com WebView

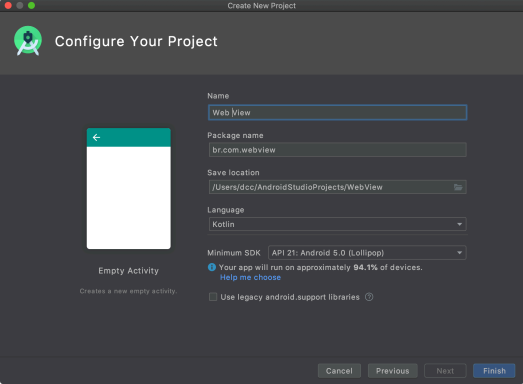
Crie um novo projeto

Abra o Android Studio. Clique em File > New... > New project... . e crie um novo projeto vazio baseado no template Empty Activity, como na figura abaixo. Clique em Next.

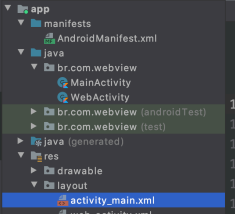


Na janela que se abre coloque os campos Name e Language como na figura seguinte, deixando os campos restantes como estão. Se quiser pode alterar o campo Save location como quiser. Clique em Finish.

2



Clique duas vezes em app > res > layout > activity\_main.xml.



No código que se abre, deixe-o como na listagem abaixo para termos um botão na tela principal.

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

<androidx.constraintlayout.widget.ConstraintLayout

xmlns:android="http://schemas.android.com/apk/res/android"

xmlns:app="http://schemas.android.com/apk/res-auto"

3

xmlns:tools="http://schemas.android.com/tools"

android:layout\_width="match\_parent"

android:layout\_height="match\_parent"

tools:context=".MainActivity">

<Button

android:id="@+id/buttonWebView"

android:layout\_width="204dp"

android:layout\_height="58dp"

android:text="Web View"

app:layout\_constraintBottom\_toBottomOf="parent"

app:layout\_constraintEnd\_toEndOf="parent"

app:layout\_constraintStart\_toStartOf="parent"

app:layout\_constraintTop\_toTopOf="parent" />

</androidx.constraintlayout.widget.ConstraintLayout>

Permitindo uso de internet

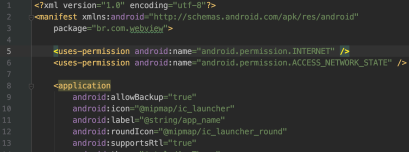
Toda vez que vamos usar algum recurso no Android precisamos autorizá-lo explicitamente no arquivo de manifesto da aplicação. Agora abra o arquivo de manifesto do Android clicando duas vezes em app > manifests > AndroidManifest.xml, como mostrado na figura abaixo.



No arquivo que se abre, adicione a permissão de uso para internet.

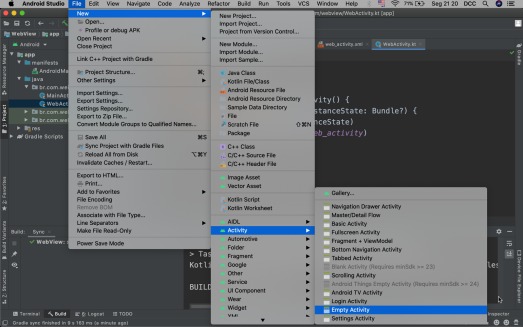
<uses-permission android:name="android.permission.INTERNET" />

<uses-permission android:name="android.permission.ACCESS\_NETWORK\_STATE" /> O arquivo de manifesto deve ficar como abaixo:

4

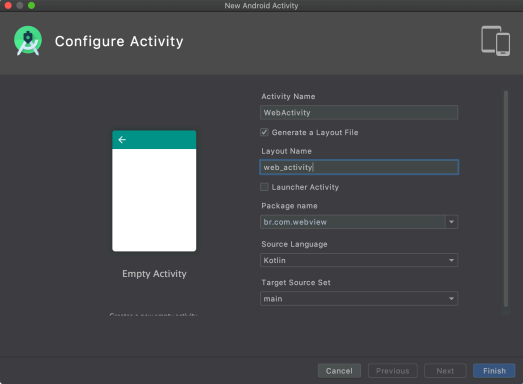
Pronto! Agora nossa aplicação pode acessar a internet.

Vamos criar uma WebActivity

Navegue até o menu File > New > Activity > Empty Activity, como na figura abaixo. 

Crie uma nova atividade com nome WebActivity e um arquivo de layout chamado web\_activity, como na figura abaixo. Clique no botão Finish.

5



Abra o arquivo de layout clicando duas vezes sobre app > res > layout > web\_activity.xml. Modifique-o para ficar como na listagem abaixo. Neste arquivo de layout estamos obtendo visualização da web e barra de progresso.

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

<androidx.constraintlayout.widget.ConstraintLayout

xmlns:android="http://schemas.android.com/apk/res/android"

xmlns:app="http://schemas.android.com/apk/res-auto"

xmlns:tools="http://schemas.android.com/tools"

android:layout\_width="match\_parent"

android:layout\_height="match\_parent"

tools:context=".MainActivity">

<ProgressBar

android:id="@+id/progressBar"

style="?android:attr/progressBarStyleHorizontal"

android:layout\_width="0dp"

android:layout\_height="wrap\_content"

android:layout\_margin="0dp"

android:minHeight="4dp"

android:padding="0dp"

app:layout\_constraintEnd\_toEndOf="parent"

app:layout\_constraintStart\_toStartOf="parent"

app:layout\_constraintTop\_toTopOf="parent" />

<WebView

android:id="@+id/webView"

android:layout\_width="0dp"

6

android:layout\_height="0dp"

app:layout\_constraintBottom\_toBottomOf="parent"

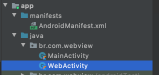
app:layout\_constraintEnd\_toEndOf="parent"

app:layout\_constraintStart\_toStartOf="parent"

app:layout\_constraintTop\_toBottomOf="@+id/progressBar" />

</androidx.constraintlayout.widget.ConstraintLayout>

Vamos passar para a WebActivity

Clique duas vezes sobre app > java > br.com.webview > WebActivity, como na figura abaixo. 

No arquivo fonte que se abre, dentro da classe WebActivity, mas abaixo do evento onCreate, coloque o código abaixo, que cria um initWebView. Note que dentro desta função há um webViewClient, que tem configurações de visualização da web que você pode alterar conforme usa. Mais uma coisa que faremos neste bloco de código é substituir onReceivedSslError.

@SuppressLint("SetJavaScriptEnabled")

private fun initWebView() {

webView.settings.javaScriptEnabled = true

webView.settings.loadWithOverviewMode = true

webView.settings.useWideViewPort = true

webView.settings.domStorageEnabled = true

webView.webViewClient = object : WebViewClient() {

override fun onReceivedSslError(view: WebView?, handler: SslErrorHandler?, error: SslError?) { handler?.proceed()

}

}

}

Defina WebChromeClient para WebView

Neste cliente Chrome, substituímos os métodos onProgressChanged (). Com base no progresso que estamos mostrando e ocultando a barra de progresso. Dentro da classe WebActivity, mas abaixo da função initWebView que acabamos de criar, coloque o código abaixo, que cria um webChromeClient.

private fun setWebClient() {

webView.webChromeClient = object : WebChromeClient() {

override fun onProgressChanged(view: WebView,

newProgress: Int) {

super.onProgressChanged(view, newProgress)

progressBar.progress = newProgress

if (newProgress < MAX\_PROGRESS && progressBar.visibility == ProgressBar.GONE) { progressBar.visibility = ProgressBar.VISIBLE

}

if (newProgress == MAX\_PROGRESS) {

progressBar.visibility = ProgressBar.GONE

}

7

}

}

}

Note que mesmo acertando os imports, a constante MAX\_PROGRESS ainda não existe e continuará reportando erro. Já vamos corrigir isto.

Logo abaixo da declaração da classe WebActivity, mas antes de onCreate, coloque o código abaixo:

companion object {

const val PAGE\_URL = "pageUrl"

const val MAX\_PROGRESS = 100

fun newIntent(context: Context, pageUrl: String): Intent { val intent = Intent(context, WebActivity::class.java)

intent.putExtra(PAGE\_URL, pageUrl)

return intent

}

}

private lateinit var pageUrl: String

Gerenciar o histórico da página da web

Dentro da classe WebActivity, mas abaixo da função setWebClient que criamos anteriormente, coloque o código abaixo.

override fun onKeyDown(keyCode: Int, event: KeyEvent?): Boolean {

// Verifica se o evento de tecla foi o botão Voltar e se há histórico

if (keyCode == KeyEvent.KEYCODE\_BACK && webView.canGoBack()) {

webView.goBack()

return true

}

// Se não é o botão Voltar ou não tem histórico de navegação, saia da activity return super.onKeyDown(keyCode, event)

}

Crie a função loadUrl

Dentro da classe WebActivity, mas abaixo da função onKeyDown que acabamos de criar, coloque o código abaixo.

private fun loadUrl(pageUrl: String) {

webView.loadUrl(pageUrl)

}

A função onCreate

Finalmente altere a função onCreate para o código abaixo.

8

override fun onCreate(savedInstanceState: Bundle?) {

super.onCreate(savedInstanceState)

setContentView(R.layout.web\_activity)

// pega a URL da página vindo da String

pageUrl = intent.getStringExtra(PAGE\_URL)

?: throw IllegalStateException("campo $PAGE\_URL não encontrado no Intent")

initWebView()

setWebClient()

handlePullToRefresh()

loadUrl(pageUrl)

}

E aproveite para acrescentar logo abaixo de onCreate o Código abaixo.

private fun handlePullToRefresh() {

}

Código-fonte completo da WebActivity

Esta é uma fonte completa desta atividade.

package br.com.webview

import android.annotation.SuppressLint

import android.content.Context

import android.content.Intent

import android.net.http.SslError

import android.os.Bundle

import android.view.KeyEvent

import android.webkit.SslErrorHandler

import android.webkit.WebChromeClient

import android.webkit.WebView

import android.webkit.WebViewClient

import android.widget.ProgressBar

import androidx.appcompat.app.AppCompatActivity

import kotlinx.android.synthetic.main.activity\_web.\*

class WebActivity : AppCompatActivity() {

companion object {

const val PAGE\_URL = "pageUrl"

const val MAX\_PROGRESS = 100

fun newIntent(context: Context, pageUrl: String): Intent { val intent = Intent(context, WebActivity::class.java)

intent.putExtra(PAGE\_URL, pageUrl)

return intent

}

}

private lateinit var pageUrl: String

override fun onCreate(savedInstanceState: Bundle?) {

super.onCreate(savedInstanceState)

setContentView(R.layout.activity\_web)

// pega a URL da página vindo da String

pageUrl = intent.getStringExtra(PAGE\_URL)

?: throw IllegalStateException("campo $PAGE\_URL não encontrado no Intent")

initWebView()

setWebClient()

handlePullToRefresh()

loadUrl(pageUrl)

}

9

private fun handlePullToRefresh() {

}

@SuppressLint("SetJavaScriptEnabled")

private fun initWebView() {

webView.settings.javaScriptEnabled = true

webView.settings.loadWithOverviewMode = true

webView.settings.useWideViewPort = true

webView.settings.domStorageEnabled = true

webView.webViewClient = object : WebViewClient() {

override fun onReceivedSslError(view: WebView?, handler: SslErrorHandler?, error: SslError?) { handler?.proceed()

}

}

}

private fun setWebClient() {

webView.webChromeClient = object : WebChromeClient() {

override fun onProgressChanged(view: WebView,

newProgress: Int) {

super.onProgressChanged(view, newProgress)

progressBar.progress = newProgress

if (newProgress < MAX\_PROGRESS && progressBar.visibility == ProgressBar.GONE) { progressBar.visibility = ProgressBar.VISIBLE

}

if (newProgress == MAX\_PROGRESS) {

progressBar.visibility = ProgressBar.GONE

}

}

}

}

override fun onKeyDown(keyCode: Int, event: KeyEvent?): Boolean {

// Verifica se o evento de tecla foi o botão Voltar e se há histórico

if (keyCode == KeyEvent.KEYCODE\_BACK && webView.canGoBack()) {

webView.goBack()

return true

}

// Se não é o botão Voltar ou não tem histórico de navegação, saia da activity return super.onKeyDown(keyCode, event)

}

private fun loadUrl(pageUrl: String) {

webView.loadUrl(pageUrl)

}

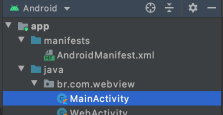
}

Executando a WebActivity

Do jeito que está nossa aplicação ainda não faz nada. Vamos criar o código do botão buttonWebView que criamos no layout activity\_main.xml no início deste tutorial.

Clique duas vezes em app > java > br.com.webview > MainActivity, como na figura abaixo.

10



No arquivo que se abre, altere o onCreate para chamar a WebActivity, como na listagem abaixo.

override fun onCreate(savedInstanceState: Bundle?) {

super.onCreate(savedInstanceState)

setContentView(R.layout.*activity\_main*)

// pega a referência do botão

val btnWebView = findViewById(R.id.*buttonWebView*) as Button

// cria on-click listener

btnWebView.setOnClickListener **{**

val intent = WebActivity.newIntent(this, "https://www.ifsudestemg.edu.br/riopomba") startActivity(intent)

**}**

}

Resumo

Chegamos ao final de mais um projeto! Foi um projeto bem simples, mas com muito conteúdo e aprendizado. Aqui, exploramos a biblioteca WebView para acesso à conteúdos online. Aprendemos os usos, a implementação e o gerenciamento do histórico de páginas da web.

Desafios

1. Coloque um EditText na MainActivity e ao invés de carregar a URL do Instituto Federal (https://www.ifsudestemg.edu.br/riopomba), carregue a URL que você quiser, por exemplo, https://www.google.com.

2. Ao invés de carregar uma URL fornecida, faça seu código usar o texto fornecido para buscar algo no Google. (Dica: ao buscar a palavra **kotlin** a URL do Google na barra de endereços de qualquer navegador fica: https://www.google.com/search?q=kotlin.)

11