AGENDA 6

WIDGETS

GEEaD - Grupo de Estudos de Educação a Distância

Centro de Educação Tecnológica Paula Souza

GOVERNO DO ESTADO DE SÃO PAULO

EIXO TECNOLÓGICO DE INFORMAÇÃO E COMUNICAÇÃO

CURSO TÉCNICO EM DESENVOLVIMENTO DE SISTEMAS

PROGRAMAÇÃO MOBILE I

Expediente

Autor:

GUILHERME HENRIQUE GIROLLI

Revisão Técnica:

Eliana Cristina Nogueira Barion

Revisão Gramatical:

Juçara Maria Montenegro Simonsen Santos

Editoração e Diagramação:

Flávio Biazim

São Paulo – SP, 2020

MERGULHANDO NO TEMA...

O que é Widget?

O Widget, como sua tradução já diz, é uma ferramenta encontrada na tela inicial dos sistemas operacionais, para realizar determinadas funções de maneira rápida e ágil. Podemos dizer que ele é uma miniatura de um aplicativo ou programa, onde nessa miniatura contempla algumas das principais funções de um aplicativo ou programa maior.

Essa ferramenta surgiu nos sistemas operacionais para computador, e logo ganhou adeptos pela sua praticidade. Um dos primeiros Widgets desenvolvidos para computadores foi o famoso “bloco de anotações” ou conhecido pela marca “Post-it”. Esse programa é a versão virtual do bom e velho papel amarelo de anotações, que colamos em agendas, cadernos entre outros locais, para registrar informações momentâneas e importantes. A figura 1 mostra o programa de anotações que ganhou fãs em todo o mundo.

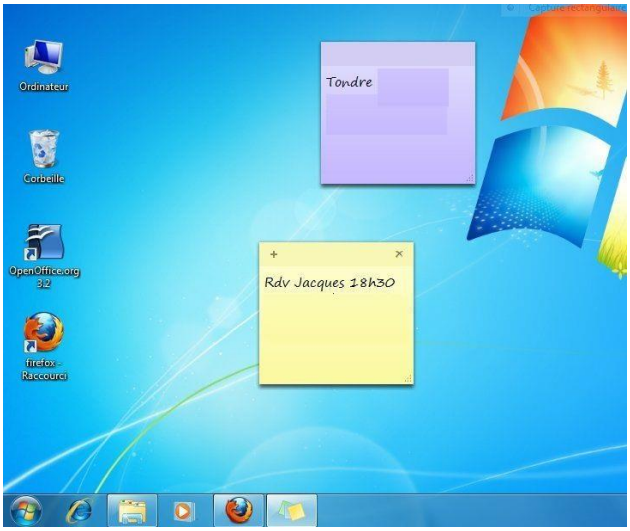


Figura 1 – Widget e anotações para o sistema operacional Windows 7.

Como os sistemas operacionais para dispositivos móveis geralmente implementam as ferramentas que tiveram sucesso nos sistemas para computadores, o sistema Android incorporou os Widgets em sua tela inicial, permitindo assim que os desenvolvedores explorem ao máximo esse recurso, proporcionando ótimas experiencias de utilização para os usuários em seus aplicativos.

Em um projeto Android, é necessário verificar alguns pontos importantes antes de desenvolver um Widget que trabalhe em conjunto com sua ferramenta principal. Primeiramente é fundamental verificar a real importância do desenvolvimento desta etapa, ou seja, se realmente o projeto necessita de um Widget, como por exemplo, um projeto de calculadora não tem a necessidade de receber uma implementação de um Widget, uma vez que isso só deixaria o projeto mais complexo e pesado para o dispositivo mobile. Em contrapartida, um projeto de aplicativo de vendas de uma loja, poderia contar com um Widget na tela inicial, informando os usuários das principais promoções dessa determinada loja.

Outro ponto importante é escolher qual tipo de Widget seu projeto necessita, a seguir vamos conhecer os principais tipos para o sistema operacional Android.

Widget de informação

Figura 2 – Widget de informação, sobre previsão do tempo.

Um Widget de informação nada mais é que um elemento na tela do usuário que apresenta informações sobre determinado tema, buscando informações em um banco de dados local ou remoto. Um exemplo simples e de fácil compreensão são os Widgets de previsão do tempo. Eles apresentam em intervalos de tempo atualizações sobre os dados meteorológicos de uma determinada cidade. A figura 2 mostra um Widget de informação.

Widget de coleção

As coleções são formadas através de galerias ou listas, e podemos definir um Widget de coleção pelo fato dele exibir mais de uma informação ao mesmo tempo. Podemos citar um exemplo de um aplicativo de E-mail que possui um Widget para a tela inicial do dispositivo, onde mostra as mensagens da caixa de entrada da conta de E-mail.

Desta forma o Widget pertence a um único aplicativo, e como mostra diversos e-mails ao mesmo tempo, ele é considerado uma coleção. A figura 3 mostra um Widget de coleção.

Este tipo de Widget conta com a função de rolagem vertical em seu sistema, podendo exibir muitas informações independente do seu tamanho, e após o usuário escolher qual informação deseja visualizar, apenas uma é aberta por vez.

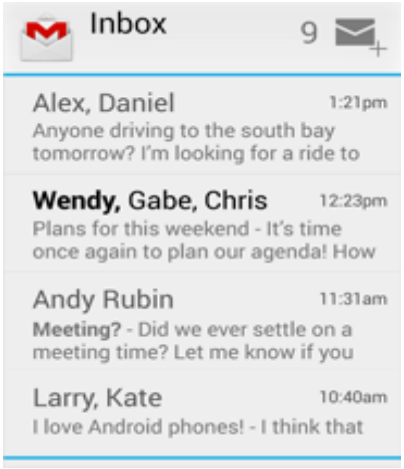


Figura 3 – Widget de coleção, Gmail da Google.

Widget de controle

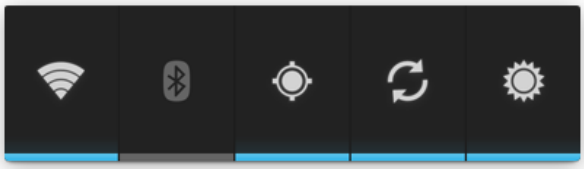


Figura 4 – Widget de controle.

Widgets de controle são aqueles utilizados como atalhos na tela, para gerar controle em um aplicativo maior. Geralmente esses Widgets são compostos apenas por botões não exibindo informações em seu contexto.

Um exemplo clássico de Widget de controle são os botões de avançar, voltar e pausar/tocar música de um player de áudio qualquer presente no mercado de aplicativos. Esse Widget fica na tela principal, e facilmente o usuário tem acesso as funções, sem a necessidade abrir o player.

A figura 4 mostra um exemplo de Widget de Controle presente na maioria dos celulares, esse conjunto de botões ativa ou desativa alguns recursos do aparelho, sem a necessidade do usuário navegar pelas configurações do sistema operacional.

Widget híbridos

Alguns Widgets podem se enquadrar em mais de um tipo, neste caso ele passa a se chamar Widget híbrido. Um Exemplo fácil e claro é de um player de música, que além de ter os botões de controle, exibe as informações sobre qual a faixa de áudio que está sendo executa. Neste caso ele assumiu as funções de Widget de controle e Widget de informação. A figura 5 exibe um exemplo de Widget Híbrido.

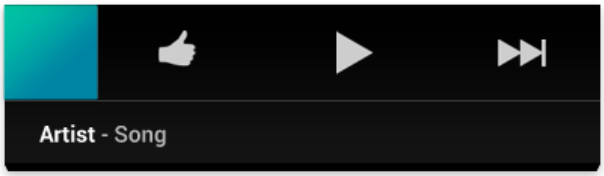


Figura 5 – Widget Híbrido.

Como os Widgets ficam localizados na tela do usuário, independentemente do tipo eles possuem apenas duas formas de gestos utilizadas pelo usuário, ou seja, o usuário apenas pode tocar (clicar na tela) ou deslizar verticalmente (rolar). Essas regras são estabelecidas para que o Widget não atrapalhe outros sistemas, uma vez que o usuário poderia gerar outros toques involuntários na tela tentando aplicar outros gestos no Widget.

Desenvolvendo um App Widget

AppWidgetProvider

Antes de desenvolver nosso primeiro Widget é necessário compreender a sua estrutura principal, que é composta por classe e arquivos XML.

O AppWidgetProvider é uma classe que é herdada pela classe que desenvolve as ações do nosso Widget, essa classe é chamada de "Provedor de Widget". Ela possui métodos que auxiliam no processo de implementação das ações do aplicativo. A classe AppWidgetProvider possuí alguns métodos padrões que podem ser sobrescritos para atender a necessidade do desenvolvedor. A seguir vamos aprender sobre os principais métodos dessa classe e suas respectivas funções.

onUpdate: Esse método é utilizado pelo provedor em todas as vezes que o sistema solicita uma nova atualização da tela do Widget.

onReceive: O método onReceive é responsável por capturar e encaminhar para outros métodos do AppWidgetProvider as ações feitas no Widget.

onRestored: Utilizado para ações que serão realizadas caso o Widget seja restaurado de um backup realizado no provedor.

onEnabled: Responsável pelas ações que serão executadas quando o usuário criar o Widget na tela.

onDisabled: Responsável pelas ações que serão executadas quando o usuário desabilitar o Widget da tela.

A classe Java que prove o servidor de Widget fica localizada no projeto em **App > Java > Com.example.** Na figura 6 encontramos um exemplo de código de uma classe que recebe a herança da AppWidgetProvider.

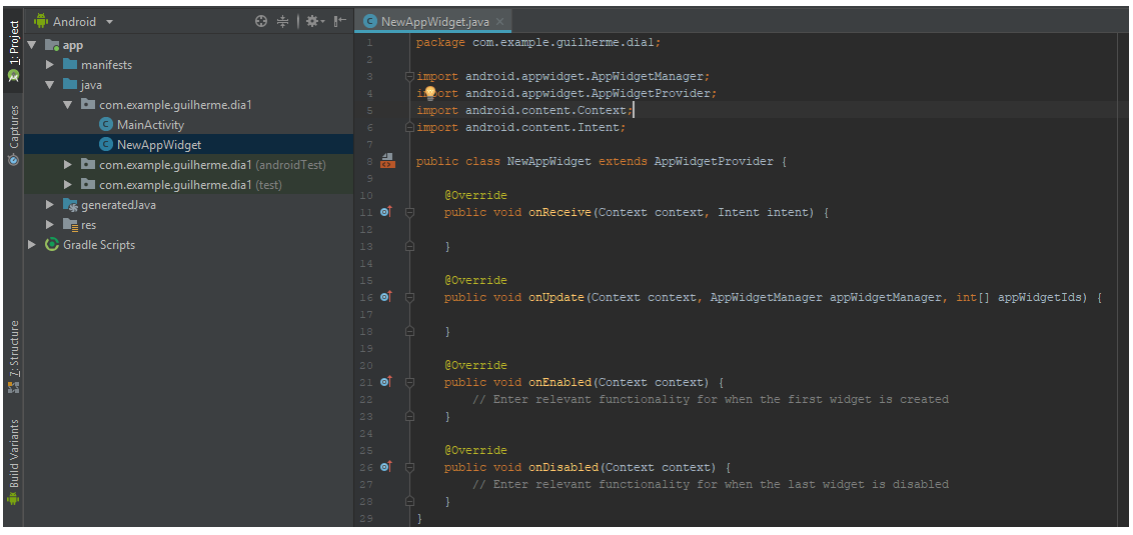


Figura 6 – Classe Java com a herança da classe AppWidgetProvider.

AppWidgetProviderInfo

Esse arquivo XML fica localizado no projeto em **App > Res > XML**, e sua função principal é dar estrutura para nosso Widget. É esse arquivo que vai determinar o tamanho, o intervalo de tempo para atualizações do conteúdo do Widget, entre outras características.

Na figura 7 encontramos um exemplo de arquivo AppWidgetProviderInfo com algumas características de exemplo de um Widget.

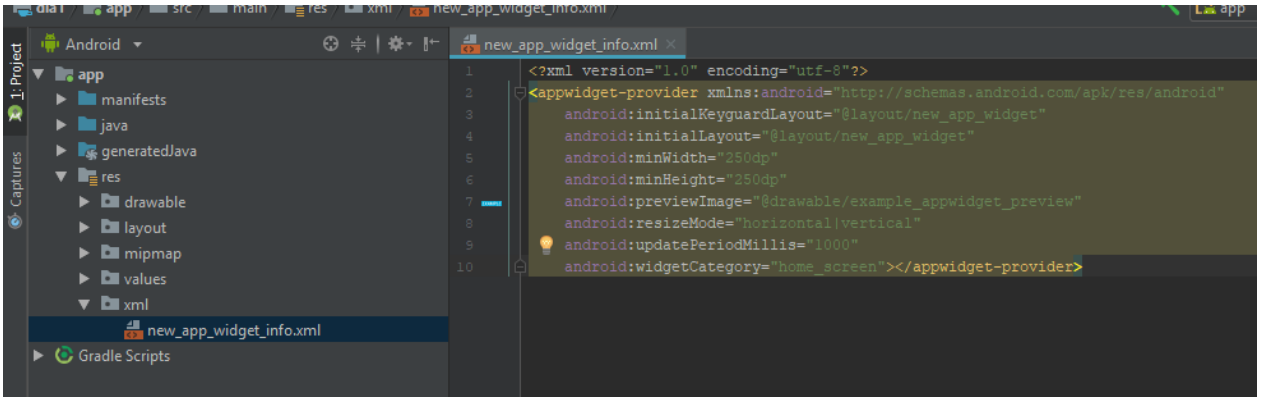


Figura 7 – Arquivo XML do AppWidgetProviderInfo.

Declaração do Widget no arquivo AndroidManifest.xml

O Widget necessita ser declarado no arquivo AndroidManifest, localizado em **App > Manifests.** Essa declaração é necessária para especificar as ações do provedor do Widget e a localização do AppWidgetProviderInfo. A figura 8 exemplifica essa declaração do arquivo Manifest.

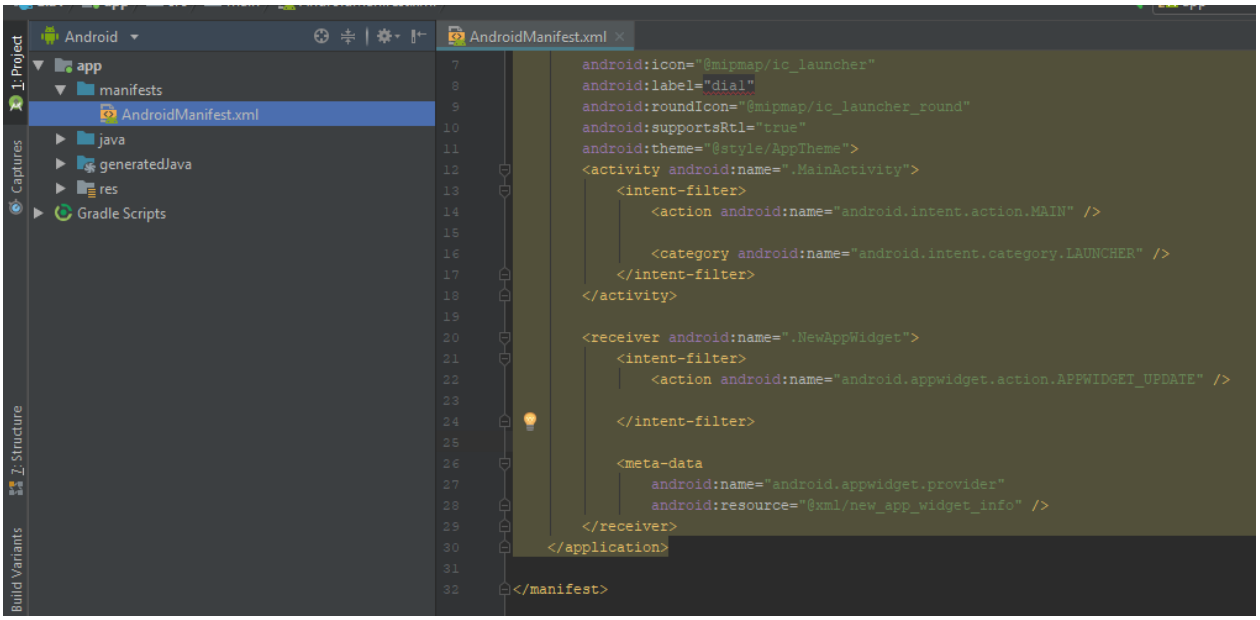
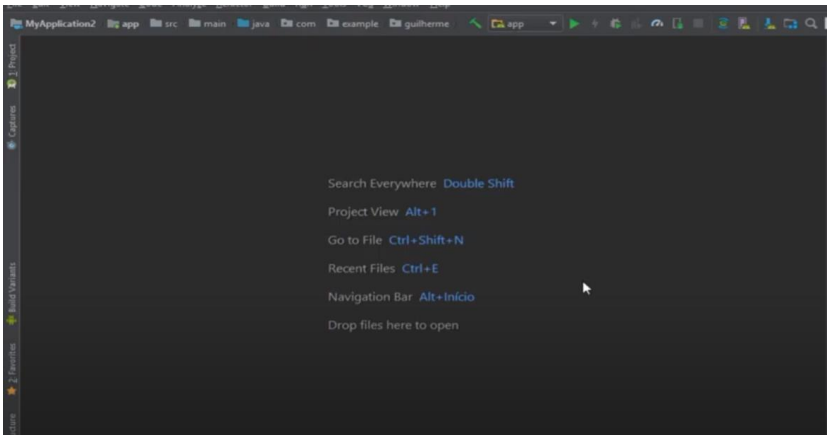
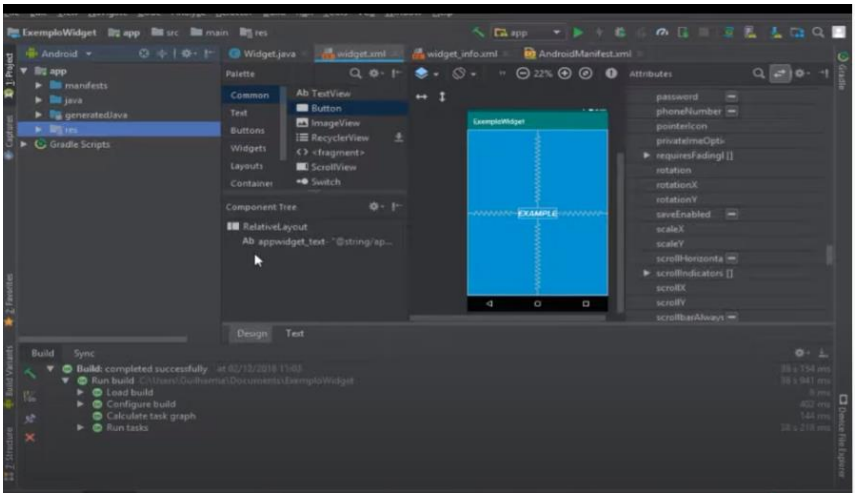


Figura 8 – Declaração do Widget no arquivo AndroidManifest.xml.

Agora chegou a hora de colocar em prática essa teoria adquirida, vamos criar um projeto na API 24, com o nome de “ExemploWidget”. Acompanhe nos vídeos a seguir o desenvolvimento do Widget.



Disponível em https://www.youtube.com/watch?v=q5iKANKqUr4



Disponível em https://www.youtube.com/watch?v=j73Uy8f2SYk

Para criar a Activity do Widget utilizamos o seguinte arquivo **widget.xml.** Localizado na pasta **App > Layout > widget.xml.**

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="#09C"
    android:padding="@dimen/widget_margin">

    <TextView
        android:id="@+id/appwidget_text"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_centerHorizontal="true"
        android:layout_centerVertical="true"
        android:layout_margin="8dp"
```

```
        android:background="#09C"
        android:contentDescription="@string/appwidget_text"
        android:text="@string/appwidget_text"
        android:textColor="#ffffff"
        android:textSize="24sp"
        android:textStyle="bold|italic" />

    <Button
        android:id="@+id/btnVerificar"
        android:layout_below="@id/appwidget_text"
        android:layout_centerHorizontal="true"
        android:layout_centerVertical="true"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Verificar Bateria" />

</RelativeLayout>
```

A declaração dos arquivos **AndroidManifest.xml** e **AppWidgetProviderInfo.xml** são automáticas, e são geradas no processo de adição do **AppWidget.**

A classe Widget.java ficou da seguinte forma, e é localizada em **App > Java > Com.example > Widget.java.**

```
package com.example.guilherme.exemplowidget;

import android.app.PendingIntent;
import android.appwidget.AppWidgetManager;
import android.appwidget.AppWidgetProvider;
import android.content.ComponentName;
import android.content.Context;
import android.content.Intent;
import android.content.IntentFilter;
import android.os.BatteryManager;
import android.widget.RemoteViews;

/**
 * Implementation of App Widget functionality.
 */
public class Widget extends AppWidgetProvider {

    private static final String VERIFICAR = "Verificar";

    public void updateAppWidget(Context context, AppWidgetManager appWidgetManager,
                                int appWidgetId) {

        CharSequence widgetText = context.getString(R.string.appwidget_text);
        // Construct the RemoteViews object
        RemoteViews views = new RemoteViews(context.getPackageName(),
R.layout.widget);
        views.setTextViewText(R.id.appwidget_text, widgetText);
        //Quando o botao receber um click
        views.setOnClickPendingIntent(R.id.btnVerificar,
getPendingSelfIntent(context, VERIFICAR));
        // Instruct the widget manager to update the widget
        appWidgetManager.updateAppWidget(appWidgetId, views);
```

```
    }

    protected PendingIntent getPendingSelfIntent(Context context, String action) {
        Intent intent = new Intent(context, getClass());
        intent.setAction(action);
        return PendingIntent.getBroadcast(context, 0, intent, 0);
    }

    @Override
    public void onUpdate(Context context, AppWidgetManager appWidgetManager, int[]
appWidgetIds) {
        // There may be multiple widgets active, so update all of them
        for (int appWidgetId : appWidgetIds) {
            updateAppWidget(context, appWidgetManager, appWidgetId);
        }
    }

    @Override
    public void onReceive(Context context, Intent intent) {
        // TODO Auto-generated method stub
        super.onReceive(context, intent);

        if (VERIFICAR.equals(intent.getAction())) {
            AppWidgetManager appWidgetManager =
AppWidgetManager.getInstance(context);
            RemoteViews remoteViews;
            ComponentName watchWidget;
            remoteViews = new RemoteViews(context.getPackageName(),
R.layout.widget);
            watchWidget = new ComponentName(context, Widget.class);
            final String[] value = new String[1];
            final int[] level = new int[1];
            final IntentFilter ifilter;
            final Intent[] batteryStatus = new Intent[1];
            ifilter = new IntentFilter(Intent.ACTION_BATTERY_CHANGED);
            batteryStatus[0] =
context.getApplicationContext().registerReceiver(null, ifilter);
            level[0] = batteryStatus[0].getIntExtra(BatteryManager.EXTRA_LEVEL, -
1);
            value[0] = Integer.toString(level[0]);
            remoteViews.setTextViewText(R.id.appwidget text, "Bateria: " + value[0]
+ "%");
            appWidgetManager.updateAppWidget(watchWidget, remoteViews);
        }
    }


    @Override
    public void onEnabled(Context context) {
        // Enter relevant functionality for when the first widget is created
    }

    @Override
    public void onDisabled(Context context) {
        // Enter relevant functionality for when the last widget is disabled
    }
}
```