# Activity Life Cycle

-A activity tem6 funções callback (chamados quando termina outra função). Cada um para um estado no activity:

**OnCreate:** É chamado quando a activity é criada. É chamado somente 1 vez.Este metodo tem parametro Bundle savedInstanceState, que contém o estado do activity anterior(null caso não existe). **Neste modo é definido qual Layout será usado, instânciar classes e variaveis e iniciar threads.**

**OnStart**: É chamado quando a acitivity fica visivél ao usuário. Depois vai para onResum **Neste Metodo é definido os BroadcastRecevier.**

**OnResume**: É chamado toda vez a activity fica em foreground e interativo ao usuário (receber clicks) ou quando o activity volta a ser visivel depois que **outra activity estava por cima**. A activity para neste metodo até que a activity não ficar mais no foreground(primeiro plano), então chama onPause().**Neste metodo tem que inicializar os componentes que foram parados no metodo on Pause() com verificação se é nulo do onPause**

**onPause**: É chamado toda vez que a activity não vai ficar visivel **Este metodo NÃO pode conter metodos pesados como salvar dados no banco de dados (que é feito no OnStop)**. **Neste metodo deve pausar /parar elementos iniciados onde pode ser inicados novamente em onResume, como pausar video, parar Broadcast Receivers, parar busca GPS, parar Camera ou qualquer outra coisa que pode gastar bateria enquando a activity não está visivel.**

**onStop**: É chamado toda vez a activity não está mais visivel, o usuario pode volta ou destruir a activity. **Aqui é executado as tarefas pesadas, como salvar dados no banco de dados,e remover todos os elementos inciados, por que se o Usuario destory o app não executa onDestory. Então aqui também pode parar os Boadcast Receivers.**

**onDestroy**: É chamado quando a activity é destroida como o metodo finish(), ou quando o Android ficar sem memoria e destroy a activity para salvar espaço na memoria. Este metodo para automaticamente os elementos não parados no onStop()

# OnsaveStateInstance and OnRestoreSateInstance

**- Used to Save Variables state when rotating the Screen, or if OS killed our app. Restoring can be in onCreate() with null verification or onRestoreStateInstance().**

-onSaveStateInstance() is called after onPause() and before Onstop () when the activity is not destroyed by clicking back button. In this method is put the variables with keys to the Bundle. **Important:** It must have super.onSaveStateInstance() because it saved the view’s state.

-onRestoreInstanceState()is only called when the screen is rotated or when app is killed by OS to free memory space

**To save persistent data**, such as user preferences or data for a database, you should take appropriate opportunities when your activity is in the foreground. If no such opportunity arises, you should save such data during the **onStop()** method.

# Android Intent Filters

- Application code run when app is running, but Mainfest code run even when application is closed that’s why we can use intent filters to open the app when other app Implicitly call it but an action, category, data….

- **There are two types of Intents**:

- **Explicit Intents:** Intents called by Class name, Example:

startActivity(**new** Intent(getApplicationContext(),MainActivity.**class**));

**- Implicit Intents**: Intents that can call any application by its action, data, category…etc

startActivity(**new** Intent(Intent.***ACTION\_DIAL***));

-Activity is a screen

**-Instant Run:** is a fast run that just update the current activity, **normally used in graphics testes**.

**-Usar tudo em dp menos para textos utiliza sp.**

**-string.format(“%.2f”,double); //arredonda um double para 2 numeros depois da virgula**

**-Todo double tem que usar o format para mostrar para usuario o numero arredondado**

# Java

-Pode criar uma classe interna dentro de: public static void main(String[] args){ }

**ArraysList:** List list = new Arraylist();

List.add(“”):

List.length

Log.i(“num”,list);

**Arrays:**  Int[] list = {1,2, 3, 4};

Log.i(“num”,list[0]);

**Map:** Map map = new Hashmap();

Map.put(“Father”, “Motassem”);

Map.put(“Mother”,”X”);

Map.remove(“Mother”);

Map.size();

Log.i(“test”,map.get(“Father”));

Log.i(“test”,map.toString());

**-Random number:** Random rand = new Random();

int n = rand.nextInt(50) + 1; //1 to 50

# Layout and Graphics

XML: (Vide o .doc da pasta: 1-Gaphic)

-**Melhor começar criando Relative Layout com width e height como fill\_partent**

**-Todos os elementos graficos implementam o Tipo: View.**

-Clica 2 vezes no elemento grafico como textview para **alterar as propridades** como: texto, id, margem,.. e outras propridades mais avançadas:

- **TextApperence ou style** podemos alterar o estilo do texto ou do elemento grafico.

- **Alfa**: Define a transparancia do texto e do elemeto grafico.

- **Backgound**: Define a cor de fundo do elemento.

- **FontFamily**: Define a fonte do texto

-**TextColor**: Cor do texto.

-**TextSize**: tamanho do texto: **tem que colocar em sp para definir uma unidade relativa para conseguir mudar para diferentes dispositivos.**

-**Gravity**: alinhar o conteudo do elemento dentro do elemento em Si, como alinha o um texto dentro de uma caixa Textview, para direita ou esquerda.

-**Clickable**: Deixar o elemento clicavél para utilizar eventos.

-Links Clickable: ?

-**Margin**: deslocar o objeto com um certa distância

-**Padding**: aumentar o objeto com um certo valor

-**OnClick:** definir nome do evento quando clica. Onde tem que criar um metodo no classe java, se não da crush:

**public void metodoBotao(View view){**

**//Aqui podemos obter qualquer elemento grafico apartir do metodo do botão**

**}**

-**Hint**: mostrar no campo de texto um texto highlight como dica o que ele pode fazer.

-Ao criar um **Relative Layout**, pode colocar objetos graficos relativos, no centro, corners, top, bottom...etc.

-Melhor **unidade grafica para Android é: dp** densidade independente de pixel. É uma unidade relativa ao numero de pixel do dispositivo do usuário.

# Linear Layout

È utilizado em layout simples com orientação vertical/horizontal. Onde pode definir para seus filhos o Atributo **Weight** que divide a tela em partes e define quantos partes cada elemento vai ocupar

Exemplo: TextView com 1 parte e EditText com 3 partes. (total 4 partes)

Podemos falar que o TextView está occupando ¼ da tela, enquanto o EditText 3/4 da tela.

# Animation

**-Fade Image**: imageView.animate().alpha(0f).setDuration(2000);

-**Translation**: **imgViewHomer.setX(0f);**

//Garantir que a imagem está nesta posição (O Usuario pode criar varias vezes e não dar tempo a chegar)

**imgViewHomer.animate().translationXBy(5000f).setDuration(1000);**

//Fazer fade in com a nova imagem do Homer

**-Rotate:** **imgViewBart.setRotation(0);**

//Garantir que a imagem está nesta posição (O Usuario pode criar varias vezes e não dar tempo a chegar)

**imgViewHomer.animate().rotate(180f) .setDuration(1000);** //Rotate 190º

**-Scale**: **imgViewBart.setScaleY(0f);**

//Garantir que a imagem está nesta posição (O Usuario pode criar varias vezes e não dar tempo a chegar)

**imgViewBart.setScaleX(0f);**

//Garantir que a imagem está nesta posição (O Usuario pode criar varias vezes e não dar tempo a chegar)

**imgViewBart.animate().scaleX(0.5f).scaleY(0.5f).setDuration(1000);**

//Deixar o Bart comScale no X e Y

# Relatar Textos Com Log

-Log.i(“tag”,”string”); //para informação mas é mais usado em todas as situações

-Log.e(“tag”,”string”); //para erro

-Log.wtf(“tag”,”string”); //para Bug

-System.out.println(“string”); //Mostrar no controle do sistema

# Mostrar Mensagens com Toast

Toast.makeText(MainActivivty.this, ”string”, Toast.Long).show();

# Imagens

-Para mostrar imagem tem que colocar a imagem na **pasta de drawable**

-Em XML: Cria ImageView - > Definir Height and Width -> definir src -> drawable/imgem

-Em Java: ImageView imagem = (ImageView) findViewById(R.id.imagem);

Imagem.setImageResource(R.drawable.pic);

**-scaleType: fillCenter** //imagem redemensionada para ficar no meio

**-scaleType: fitInside** //redemensionar imagem para caber no ImageView

-**scaleType:fitXy**  // redemensionar imagem para caber na tela

# Vídeos

1. Adicionar VideoView em XML
2. Criar arquivo raw em res e colocar o Video MP4
3. Código:

VideoView videoView = (VideoView) findViewById(R.id.***videoView***);  
videoView.setVideoPath(**"android.resource://"**+getPackageName()+**"/"**+R.raw.***motassem***);

*//Definir o Path do Arquivo*MediaController mediaController = **new** MediaController(**this**); *//Criar controlador do video*mediaController.setAnchorView(videoView); *//Definir o view do controlador*videoView.setMediaController(mediaController); *//Definir o controlador do view*videoView.start(); *//Play*

# Audios

1. Criar arquivo raw em res e colocar o audio MP3
2. código

MediaPlayer mplayer = MediaPlayer.*create*(**this**,R.raw.***laugh***);  
mplayer.start();  
*//mplayer.pause(); //Pausar o vídeo onde mplayer.start retorna   
//mplater.stop(); //Para o vídeo*

# Volume

AudioManager audio = (AudioManager) getSystemService(Context.***AUDIO\_SERVICE***); *//Obter o serviço do audio do sistema*maxvolume = audio.getStreamMaxVolume(AudioManager.***STREAM\_MUSIC***); *//Obter o volume máximo do Stream do music*

**audio**.setStreamVolume(AudioManager.***STREAM\_MUSIC***,progress,0); *//Definir o volume do stream music o progresso do seekbar com flag 0 que não é usada*

# Getting Resources by ID (economizar de metodos)

String strID = view.getResources().getResourceEntryName(view.getId());

//Obter o ID com o nome definido e não o numero do ID  
int resourceID = getResources().getIdentifier(strID, "raw", getPackageName());

//Obter o ID do som apartir do nome do id da tecla que foi identica ao nome do som

# ListView

1. Criar ListView em XML
2. Criar ArrayList com os itens desejados

ArrayList<String> itemsList = **new** ArrayList<String>(); itemsList.add(**"item 1"**);  
itemsList.add(**"item 2"**);  
itemsList.add(**"item 3"**);

1. Criar Um Array Adapter com estilo: android.R.layout.simple\_expandable\_list\_item\_1 (poderia ser outros estilos layout. ..)

ArrayAdapter<String> itemListAdapter = **new** ArrayAdapter<String>(**this**,android.R.layout.***simple\_expandable\_list\_item\_1***,itemsList);

1. Atribuir AdapterList para a listview

list.setAdapter(itemListAdapter);

1. Criar Evento de Click, Onde:

AdapterView = listview/Spinner.. que foi clicada

view = Row que foi clicado

Position = Row Number

list.setOnItemClickListener(**new** AdapterView.OnItemClickListener() {  
 @Override  
 **public void** onItemClick(AdapterView<?> parent, View view, **int** position, **long** id) {  
 Toast.*makeText*(getBaseContext(), itemsList.get(position), Toast.***LENGTH\_LONG***).show();  
 }  
});

1. Quando altera o Adapter é necessario executar o metodo para atualizar o ListView:

arrayAdapter.notifyDataSetChanged();

# Timers and Handlers

-Os **Handlers** controla o tempo de execução de códigos com delays periodicos, como: a cada 10 segundos.

-Os **Runnable**s permitam que exeuta um código a cada um periodo.

**-Exemplo 1** : Handler que executa Runnable a cada 1 segundo **em Loop**

**final** Handler handler = **new** Handler(); *//Os Handlers controla o tempo de execução de códigos com delays periodicos, como: a cada 10 segundos.*Runnable loopedRunnable = **new** Runnable() {  
 @Override  
 **public void** run() {  
 handler.postDelayed(**this**,1000); *//Cada Run vai executar Este metodo novamente a cada 1 segundo* }  
};  
handler.post(loopedRunnable); *//Executar o Handler para executar o Runnale*

**-Exemplo 2:** Count down com ticks que **roda somente 1 vez**

**new** CountDownTimer(10000,1000){ *//roda 1 vez durante 10 segundos que enquanto disso chama onTick a cada 1 segundo(as vezes coloca + 100 millisegundo para processar o primeiro tick se for mostrar no TextView)* @Override  
 **public void** onTick(**long** millisUntilFinished) {  
 Log.*i*(**"Tempo Restante: "**,millisUntilFinished/1000 + **""**);*//Metodo chamado a cada 1 tick dentro do tempo do CountDownTimer* }  
  
 @Override  
 **public void** onFinish() {  
 Log.*i*(**"Tempo Acabou"**,**"Passou 10 segundos"**);  
 }  
}.start();

# Download HTML with ASYNCTASK

-AsyncTask é uma classe que roda em background (Thread diferente do Main Thread) que pode executar código de processamento, download de paginas HTML, execução de tarefas...etc

- **Parâmetros da Classe**: O primeiro parâmetro é para passar URL, e segundo é para execução de uma tarefa como mostrar progresso do download, e o terceiro é o tipo do varivel que vai retornar do metodo do Async Task

**public class** DownloadpaginaWeb **extends** AsyncTask<String,Void, String>{

- Parâmetro do metodo doInBackgound: O parâmetros do metodo doInBackground é um array de Strings que é atribuido na hora de exeutar: AsyncTask.execute(“parâmetro1”,”Parametro2”);

**protected** String doInBackground(String... params) {

- Para criar um URL tem que ter a formação de endereço Web “http://www.example.com”

- Para poder baixar paginas web tem que adicionar permissão de acesso internet no Mainfest:

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

**- doInBackground() method não pode interagir com UI, enquanto o onPostExecute() method pode interagir com UI**

# Cortar Strings com Padrões

String t = **"missisipimissisomflpi"**;  
Pattern pattern = Pattern.*compile*(**"mi(.\*?)pi"**); *//Criar padrão para obter strings que começam com "mi" e terminam com "pi"*Matcher matcher = pattern.matcher(t); *//Definir o padrão para o string***while** (matcher.find()) { *//Se existe strings deste padrão, obter-los* System.***out***.println(matcher.group(1)); *//Retorna o primeiro str com o padrão e remove o primeiro para a proxima leitura*}

# JSON Javascript Object Notation

- JSON e XML são as linguagens mais famosas em passar dados ou parametros entre sistemas e paginas web. Usado em API também

**try** {  
 JSONObject json = **new** JSONObject(result); *//Criar Objeto JSON apartir do String obtido* String temperatura = json.getString(**"weather"**); *//obter somente a parte que tem a temperatura* JSONArray jsonArray = **new** JSONArray(temperatura); *//Desmontar a estrutura obtida de arrays* JSONObject jsonPart = jsonArray.getJSONObject(0); *//Obter o segundo dado do array* System.***out***.println(jsonPart.getString(**"description"**)); *//obter subparte do JSON*} **catch** (JSONException e) {  
 e.printStackTrace();  
}

# Esconder Teclado

InputMethodManager mgr = (InputMethodManager) getSystemService(Context.INPUT\_METHOD\_SERVICE); //Obter Entrada de dados do sistema

mgr.hideSoftInputFromWindow(cityName.getWindowToken(), 0); //esconder entrada de dados para o EditText

# Formatar URL em UTF8 Para subsituir espaços

String encodedCityName = URLEncoder.encode(cityName.getText().toString(), "UTF-8");

# Mapas

-Para obter lat e longitude, basta ir no google maps -> botão direito -> whats here

1. Criar Google Maps Activity
2. Copiar o endereço do google\_maps\_api.xml para navegador e colar o criar uma chave e colocar no XML (tem que está logado no google developer)
3. Copiar google\_maps\_api.xml e colocar em: /res/values
4. No MainFest, o value do meta-dada do google maps api tem que ser mudado para o código manualmente colocado e não apontado pelo xml.

<**meta-data  
 android:name="com.google.android.geo.API\_KEY"  
 android:value="AIzaSyC-4fXfoTz\_gMPnIFewOO80sgXfiOjBC-Y"** /> *<!-- Google Maps API -->*

1. Se for para limitar o API key tem que colocar o chave SHA-1 da assinatura e a chave de debug do aplicativo no api:

-para obter chave SHA-1 da Assinatura:

Cmd->cd c:\Program Files\Java\jdk1.8.0\_121\bin

-> keytool -list -keystore "D:\Programacao\Android\!- Meus dados e Assinatura\Assinatura.jks"

->colocar a senha da assinatura e aprecerá o chave

-Para obter chave SHA-1 do aplicativo: está presente no google\_maps\_api.xml

-No website da Google API - > Credentials -> selecionar api key -> limitar Android->

Colocar package name e SHA-1 do debugg - > adicionar mais um key para colocar da assinatura também.

1. Para Configurar mapa, Definir poisção, Custom marker, adicionar no OnMapReady

**mMap**.setMapType(GoogleMap.***MAP\_TYPE\_HYBRID***); *//Mapa completa com todos os recursos de satilite*

**mMap**.clear(); *//Remover todos os marcadores*

LatLng sydney = **new** LatLng(-34, 151);  
**mMap**.addMarker(**new** MarkerOptions().position(sydney).title(**"Marker in Sydney"**).icon(BitmapDescriptorFactory.*defaultMarker*(BitmapDescriptorFactory.***HUE\_CYAN***))); *//definir a posição na mapa e definir o titulo e o icon default azul  
// mMap.moveCamera(CameraUpdateFactory.newLatLng(sydney)); //Adicionar na mapa sem zoom*

mMap.moveCamera(CameraUpdateFactory.*newLatLngZoom*(sydney,20)); *//adicionar na mapa com zoom de x20*

# MultiActivities

**Atenção: Se volta de Activity 1 para activity 2 o metodo Oncreate não é executado**

1. Criar Activity: new -> Activity -> Gallary -> escolher Activity
2. Alterar Entre Atividades:

Intent intent = **new** Intent(getApplicationContext(), SecondActivity.**class**); *//Intent para mudar o atividade atual com a atividade do MainActivity*startActivity(intent); *//iniciar a Activity criada*

1. Passer Strings Entre Activities:

-na primeira Activity:

Intent intent = **new** Intent(getApplicationContext(), SecondActivity.**class**); *//Intent para mudar o atividade atual com a atividade do MainActivity*intent.putExtra(**"username"**,**"Motassem"**); *//Passar String para outra Atividade*startActivity(intent); *//iniciar a Activity criada*

-Na segunda Activity:

**public void** buttonListener(View view){ *//Evento quando o botão é clicado* Intent intent = **new** Intent(getApplicationContext(), MainActivity.**class**); *//Intent para mudar o atividade atual com a atividade do MainActivity* startActivity(intent); *//iniciar a Activity criada*}  
  
@Override  
**protected void** onCreate(Bundle savedInstanceState) {  
 **super**.onCreate(savedInstanceState);  
 setContentView(R.layout.***activity\_second***);  
 System.***out***.println(getIntent().getExtras().get(**"username"**)); *//Obter o String da outra Activity*}

1. **O metodo Finish() termina activitdade Atual e se tem outra atividade principal volta para ela**

# Shared Preference

- Shared Preferences é um sistema de amazenamento permanente em arquivo XML dentro do cache (Sand Box) do aplicativo “/Data/Data/com.example/shared\_prefs”

- Shared Preferences é usado para armazenar permanente os dados de pequena quantidade.

SharedPreferences sharedPreferences = **this**.getSharedPreferences(**"com.mathbox.pro.memories"**, Context.***MODE\_PRIVATE***); *//Colocar dados no pacote do aplicativo em modo privado (outros aplicativos não tem acesso)*sharedPreferences.edit().putString(**"key1"**,**"Test Strings"**).apply(); *//Colocar String dentro do chave: key1*sharedPreferences.getString(**"key1"**,**""**); *//Obter string com o chave: key1 e se não achar retorna ""*

# Shared Preference + ArrayLists

-Utilizar o código:

**public static void** saveNotes(Activity activity,ArrayList<String> notes){  
 SharedPreferences sharedPreferences = activity.getSharedPreferences(**"com.mathbox.pro.notes"**,***MODE\_PRIVATE***);  
 HashSet<String> hashSet = **new** HashSet<>(notes); *//Hashset para gravar ArrayList de Strings* sharedPreferences.edit().putStringSet(**"notes"**,hashSet).apply(); *//salvar Hashset do ArrayList*}  
**public static** ArrayList<String> getNotes(Activity activity){  
 SharedPreferences sharedPreferences = activity.getSharedPreferences(**"com.mathbox.pro.notes"**,***MODE\_PRIVATE***);  
 HashSet<String> arrayHashSet = (HashSet<String>) sharedPreferences.getStringSet(**"notes"**,**new** HashSet<String>(**new** ArrayList<String>())); *//Se achar hashset retorna ele, se não obter hashset com ArrayVazio* ArrayList<String> notes = **new** ArrayList<String>(arrayHashSet); *//Obter o ArrayList apartir do hashset* **return** notes;  
}

# Salvar e Ler Arquivos

Existe 4 tipos de arquivos:

1. Internal Cache (disponível somente para proprio app + apagado quando deleta app )
2. External Cache (disponível para todos os app + apagado quando deleta app )
3. External File Private (disponível somente para proprio app)
4. External File Public (disponível para todos os app)

- External File é definido pelo o usuario se é o phone ou SD Card

## armazenamento interno (Cache)

O arquivos salvos no armazenamento interno são privados do aplicativo e não podem ser acessados por outros aplicativo (nem pelo usuário). Quando o usuário desinstalar o aplicativo, esses arquivos serão removidos.

-Arquivos Salvos em: /data/data/com.example/files

- Mode Private: sobre escrever arquivo antigo

-Mode Append: escrever no arquivo antigo sem apagar

-Escrever:

fileOutputStream = openFileOutput(**"file.txt"**, ***MODE\_APPEND***); fileOutputStream.write(text.getBytes());

-Ler:

fileInputStream = openFileInput(**"file.txt"**);  
**while** ((caracter = fileInputStream.read()) != -1){  
 text += (**char**) caracter; *//Adicionar Caracter ASCII*}

# OnLongClick

Tem que colocar: return true para não executar onclick() também.

# Criar Menu Settings

-criar diretorio menu

new -> menu file resources

-colocar items com id

@Override

public boolean onCreateOptionsMenu(Menu menu) {

MenuInflater menuInflater = getMenuInflater(); //Obter linkador de menu

menuInflater.inflate(R.menu.menu\_settings,menu); //Linkar menu

return super.onCreateOptionsMenu(menu);

}

@Override

public boolean onOptionsItemSelected(MenuItem item) { //Metodo quando um item foi selecionado

super.onOptionsItemSelected(item);

switch (item.getItemId()){

case R.id.item1:

Toast.makeText(getBaseContext(),"1",Toast.LENGTH\_LONG).show();

return true;

case R.id.item2:

Toast.makeText(getBaseContext(),"2",Toast.LENGTH\_LONG).show();

return true;

default:

return false;

}

}

# SQLITE

**Notação SQL: tudo maiusculo menos nome do database e variaveis**

varchar = String  
 INT(3) = int com 3 caracteres  
  
 Query Criar Tabela se não tem:

sqLiteDatabase.execSQL("CREATE TABLE IF NOT EXISTS users (id INTEGER PRIMARY KEY, name VARCHAR, age INT(3))"); //Criar tabela no banco se não existe com campo de String e int de 3 caracteres

Queries Adição:

sqLiteDatabase.execSQL("INSERT INTO users (name, age) VALUES ('Assem', 19)"); //Inserir dados no banco de dados  
  
 1- Queries Seleção:

ArrayList<News> artigos = **new** ArrayList<>();

SQLiteDatabase database = activity.openOrCreateDatabase(**"Webpages"**, Context.***MODE\_PRIVATE***, **null**); *//Criar Database em modo privado se Não existe,e se existe abre-o*

database.execSQL(**"CREATE TABLE IF NOT EXISTS webpages (id INTEGER PRIMARY KEY, articalid INTEGER, title VARCHAR, webpage VARCHAR)"**);

Cursor curosr = database.rawQuery(**"SELECT \* FROM webpages"**,**null**); *//Criar um cursor que vai obter o resultado do Query*

**int** articalIDIndex = curosr.getColumnIndex(**"articalid"**); *//obter ID da coluna*

**int** webpageIndex = curosr.getColumnIndex(**"webpage"**);

**if**(curosr.moveToNext()){ *//se existe itens*

**do**{

String id = curosr.getString(articalIDIndex);

String test = curosr.getString(webpageIndex);

artigos.add(**new** News(id,test));

}**while** (curosr.moveToNext()); *//Obter dados enquando ainda tem*

}

2-Queries Seleção:

Cursor cursor = sqLiteDatabase.rawQuery("SELECT \* FROM users", null);  
 int idIndex = cursor.getColumnIndex("id");  
 int nameIndex = cursor.getColumnIndex("name"); //Obter o index (id da coluna de nome)  
 int ageIndex = cursor.getColumnIndex("age"); //Obter o index (id da coluna de idade)  
 cursor.moveToFirst(); //mover o cursor para o primeiro resultado  
 boolean containsResult = true;  
 while (cursor != null && cursor.getCount() > 0 && containsResult){ //Enquanto o cursor não está vazio (nem null, tem mais do que 0 result, e tem proximo result)  
 Log.e("Rows","Nome: "+ cursor.getString(nameIndex) + " Idade: "+cursor.getInt(ageIndex)); //tem que obter o tipo certo da coluna  
 containsResult = cursor.moveToNext(); //Ir para o próximo registro, e se não tem resultado retorna  
 }  
  
 -Obter Tudo: SELECT \* FROM users  
 -Obter informações especificas: SELECT \* FROM users WHERE name = 'Motasem' AND age = 21  
 SELECT \* FROM users WHERE age > 10  
 -Obter onde nome tem o carater: SELECT \* FROM users WHERE name LIKE '%o%'  
 -% indica qualquer coisa antes ou depois  
 -Obter dados contain caracter e obter somente o primeiro row: SELECT \* FROM users WHERE name LIKE '%o%' LIMIT 1  
  
 Queries de Alteração:

sqLiteDatabase.execSQL("UPDATE users SET age = 22 where id = 1");  
  
 Queries Deletação:

**SEMPRE Deleta pelo ID, por que é unico para cada dado. por que para outros campos pode apagar mais.**  
 -Deletar Dados: sqLiteDatabase.execSQL("DELETE FROM users WHERE id = 1"); //Deletar dados (SEMPRE Apagar utilizando id)  
  
 Query Deletar Tabela:  
 sqLiteDatabase.execSQL("DROP TABLE users"); //Deletar Tabela

# WebView

WebView é super simples, permite mostrar conteudo HTML,JavaScript.. no android

* pode carregar uma pagina apartir url
* pode criar código HTML,css,javascript e mostrar no aplicativo
* Pode construir o aplicativo usando WebView e JavaScript

webView.getSettings().setJavaScriptEnabled(**true**); *//Habilitar mostrar conteudo JavaScript no WebView* webView.setWebViewClient(**new** WebViewClient()); *//Criar Instancia para mostrar conteudo HTML no WebView* webView.loadUrl(**"http://www.google.com"**); *//Mostrar pagina Web no webView  
// webView.loadData("<html><body><h1>Hello</h1><p>This is my Page</p></body></html>","text/html","UTF-8"); //Mostrar uma pagina personalizada usando código HTML com códificação padrão UTF-8*

# Services

Services can perform long-running operations without offering user interface, it can handle network transactions, play music, perform file I/O, or interact with a content provider, all from the background.they can send **Toast Notifications** or **Status Bar Notifications**.

-If you must perform work outside of your main thread, but only while the user is interacting with your application, you should instead create a new thread.

**- Services run in your application's main thread** by default, so you should still create a new thread within the service if it performs intensive or blocking operations.

If a component starts the service by calling startService() (which results in a call to onStartCommand()), the service continues to run until it stops itself with stopSelf() or another component stops it by calling stopService().

If a component calls bindService() to create the service and onStartCommand() is not called, the service runs only as long as the component is bound to it. After the service is unbound from all of its clients, the system destroys it.

**IntentService** If you use an Intent service, onHandleIntent is called in a single daemon thread. Each new call to startService will appear as an in-order, queued call to onHandleIntent. The result is simple, well behaved, in-order execution of calls, on that daemon thread. Btw, the service will not stop itself until the queue is empty: no need to "recreate"

**Bound Service in same App** There are, actually, two cases for a bound service. If it is running in your process, the service will run on the UI thread. If you need stuff run *off* the UI thread, you will have to build your own thread (probably a Looper) and, thus, talk to it with a Messenger. The result is more flexible (# of threads, queueing order, etc. is up to you) but very similar to the IntentService.

|  |  |  |
| --- | --- | --- |
| **Unbound  Service** | **Bound  Service** | **Intent  Service** |
| Unbounded Service is used to perform long repetitive task | Bounded Service is used to perform background task in bound with another component | Intent Service is used to perform one time task i.e when the task completes the service destroys itself. |
| Unbound Service gets starts by calling **startService()**. | bounded Service gets starts by calling **bindService()**. | Intent Service gets  starts by calling **startService()**. |
| Unbound Service is stopped or destroyed explicitly  by calling **stopService().** | bounded Service is unbind or destroyed by calling **unbindService()**. | IntentService Implicitly calls **stopself()** to destroy |
| Unbound Service is independent of the component in which it is started. | bound Service dependents on the component in which it is started. | Intent Service is independent of the component in which it is started. |

There is 3 types of services:

**-Started**: Service started by component by executing startService() and keeps running indefinitely even if the component is destroyed. A started service performs a single operation and does not return a result to the caller. For example, it can download or upload a file over the network. **When the operation is complete, the service should stop itself.**

**-Bound**: Service bound to application by calling bindService(), Which allow the service offering client-server interface that allows the application components to interact with the service, send requests, get results, and even do so across processes with interprocess communication (IPC). A bound service runs only as long as another application component is bound to it. **Multiple components can bind to the service at once, but when all of them unbind, the service is destroyed.**

## Service Life Cycle



## Create a Service

Service **need to be stopped** when it finishes the action (IntentService automatically stop)

**Not used to MultiThreading (Use IntentService)**

1. Create Manifest code:

<**service android:name=".StartedService"  
 android:exported="false"** /> *<!-- Não permitir outro aplicativos acessar este serviço-->*

1. Create Java Class extends Service

-**Started Service:** onBind() retorna null, e usar onStartCommand()

**-Bind Service:** usar onBind, e não usarOnStartCommand()

**onStartCommand():** chamado quando o service em ativo e quando criado tem que retornar um dos três variaveis:

**-START\_NOT\_STICKY**: não reinicia o serviço quando o Android fecha.

**-START \_STICKY**: reinicia o serviço quando o Android fecha. Onde é chamado onStartCommand **com** **intent nulo**

-**START\_REDELIVER\_INTENT**: reinicia o serviço quando o Android fecha. Onde é chamado onStartCommand **com** **ultimo intent**. Normalmente usado para serviços de download. Onde não pode ser interrompido.

## Create an IntentService

**IntentService** Service is a base class for IntentService **Services that handle asynchronous requests or handle many startService() into one Service** (expressed as startService(Intent) calls) on demand; the service is started as needed, handles each Intent in turn using a worker thread, and stops itself when it runs out of work. Normally use to handle multithread Requests. Where **OnHandleIntent()** handle each request of start service individually and let the others wait until the current request is processed.

**Differences**

1. **Service class uses the application's main thread**, **IntentService** **Runs on Background).**
2. IntentService **Automatically stops** when it finishes onHandleIntent() handling all requests, while services need to be stopped
3. IntentService implements onBind() that returns null. This means that the IntentService **cannot be** bound by default.
4. IntentService implements onStartCommand() that sends Intent to queue and to onHandleIntent().
5. Create Manifest code:

<**service android:name=".StartedService"  
 android:exported="false"** /> *<!-- Não permitir outro aplicativos acessar este serviço-->*

1. Create Java Class extends IntentService with basic methods:

create onCreate(), OnHandleIntent(), onDestory()

## Create Bind Service

-It send requests, receive responses, and perform interprocess communication (IPC). A bound service typically lives only while it serves another application component and **run in MainAcitvity Thread**.

-When the **last client unbinds** from the service, the system **destroys the service**, **unless the service was also started by startService().**

- If you need to interact with the service only while your activity is visible, **you should bind during onStart()** and **unbind during** **onStop().**

-Multiple clients can connect to a service simultaneously. However, the system calls your service's onBind() method to retrieve the IBinder only when the first client binds. The system then delivers the same IBinder to any additional clients that bind, without calling onBind() again.

-Only activities, services, and content providers can bind to a service—you **can't** bind to a service from a broadcast receiver.

# Broadcast Receivers

- Broadcast receivers são listener de algum evento no android, ou por eventos criados. Existe 2 maneiras de declarar Broadcast Receivers:

**Java**: criar Instância do BroadcastReceiver e registrar no onResume() e tirar o registro no metodo onPause().

**Manifest**: criar classe Java que extende Broadcast receiver e no manifest declarar tag <receiver> com tag <intent-filter> com tag <action> onde será definido a ação. **IMPORTANTE, NÃO TODO BROADCAST RECEIVER ACEITA SER COLOCADO NO MANIFEST.**

# Content Provider

- Content Providers can provide to other apps data from shared preferences, SQLite, flat file…etc

- Content provider consuguem efetuar CRUD.

-Content providers are used to:

* You want to offer complex data or files to other applications.
* You want to allow users to copy complex data from your app into other apps.
* You want to provide custom search suggestions using the search framework.
* You want to expose your application data to widgets.
* You want to implement the [AbstractThreadedSyncAdapter](https://developer.android.com/reference/android/content/AbstractThreadedSyncAdapter.html), [CursorAdapter](https://developer.android.com/reference/android/widget/CursorAdapter.html), or [CursorLoader](https://developer.android.com/reference/android/content/CursorLoader.html) classes.