

Macoratti.net Xamarin Android - Apresentando a plataforma Android

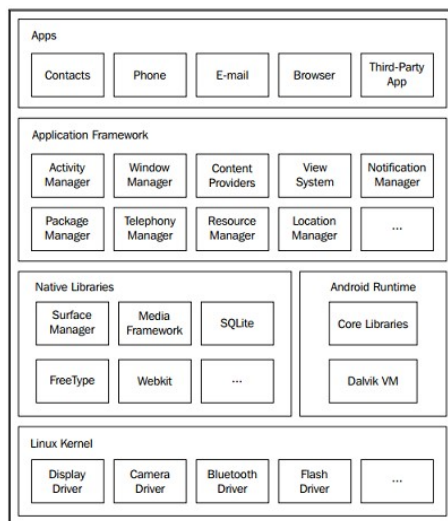


Neste artigo vou apresentar uma visão geral resumida sobre a plataforma Android.

O **Android** é um sistema operacional baseado no Linux para dispositivos móveis : *smartphones, tablets, etc.*, e foi desenvolvido pela [Open Handset Alliance](#), liderada pelo Google e outras empresas.

O Android oferece uma abordagem unificada para o desenvolvimento de aplicações para dispositivos móveis, o que significa que os desenvolvedores só precisam desenvolver para Android, e suas aplicações serão capazes de rodar em diferentes dispositivos alimentados por Android.

A plataforma Android é uma das plataformas mais bem sucedidas desenvolvidas nos últimos anos, e fornece aos desenvolvedores muitos serviços e recursos necessários para criar ricas aplicações para dispositivos móveis. O diagrama a seguir fornece um visão de alto nível de como a plataforma Android esta organizada :

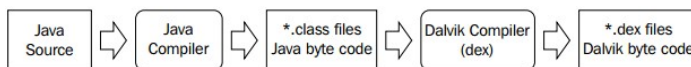


O runtime Android

Aplicativos Android são executados dentro da [Dalvik Virtual Machine \(Dalvik VM Dalvik\)](#), que é semelhante a uma máquina virtual Java, mas foi otimizado para dispositivos com memória e capacidade de processamento limitados.

A Dalvik VM faz uso de recursos básicos do Linux como gerenciamento de memória e multithreading, o que é intrínseco na linguagem Java. A [Dalvik VM](#) permite que cada aplicativo Android possa ser executado em seu próprio processo, com a sua própria instância da máquina virtual Dalvik.

Os aplicativos Android são inicialmente compilados para *byte code* Java usando o compilador Java, mas eles têm uma etapa adicional de compilação que transforma o *byte code* Java para o *byte code* Dalvik, adequado para executar dentro da [Dalvik VM](#).

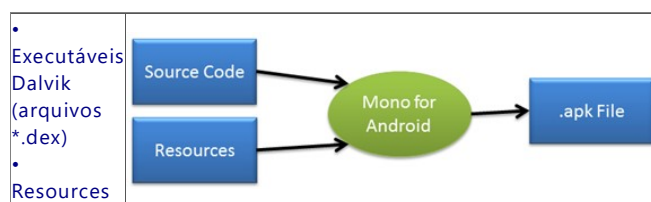


A Dalvik é entregue com as bibliotecas do núcleo do Android. Essas bibliotecas não se alinham com uma plataforma Java específica (JSE, JEE, ou JME), mas atuam como uma plataforma híbrida mais alinhada com JSE sendo que a [Android Application Framework \(AAF\)](#) fornece um meio alternativo de criação de interfaces de usuário.

Os pacotes Android (.apk)

As aplicações são entregues para a instalação em um formato de pacote Android.

Um pacote Android é criado como resultado da compilação de um aplicativo para Android e é um arquivo com uma extensão **.apk** que contém todo o código e os arquivos de suportes necessários para executar uma única aplicação, incluindo o seguinte:



- bibliotecas nativas
- O manifesto do aplicativo

os pacotes do Android podem ser instalados diretamente via e-mails, URLs, ou cartões de memória. Eles também pode ser instalados indiretamente por meio de lojas de aplicativos como Google Play.

O manifesto do aplicativo

Todos os aplicativos do Android tem um arquivo de manifesto (**AndroidManifest.xml**) que informa para a plataforma Android tudo o que ela precisa saber para executar com êxito a aplicação.

Dentre as informações do arquivo de manifesto temos:

- O Nível mínimo da API requerido pela aplicação;
- Os recursos de hardware/software usados ou exigidos pelo aplicativo;
- As permissões exigidas pela aplicação;
- A tela inicial (atividade Android) apresentada quando a aplicação for iniciada;
- As Bibliotecas exigidas pela aplicação (Exceto a AAF);






As versões do Android

Identificar a versão do Android pode ser um pouco confuso; Existe um **número de versão, um nível da API, e um apelido**, e esta nomenclatura são, por vezes, utilizadas de forma intercambiáveis.

O número da versão representa uma versão da plataforma. Às vezes, uma nova versão é criada para fornecer novas capacidades, embora às vezes ela é criada para corrigir bugs.

O nível da API representa um conjunto de capacidades. Medida que o nível da API aumenta, novas capacidades são entregues para o desenvolvedor.

Para você ter uma ideia, abaixo temos duas imagens exibindo o número de algumas [versões do Android, o API Level e apelido](#):

| | | | | | Platform version | API level | Nickname | Released |
|-------------------------------------------------------------------------------------|-------------------------------------------------------------------------------------|-------------------------------------------------------------------------------------|-------------------------------------------------------------------------------------|-------------------------------------------------------------------------------------|-------------------|-----------|--------------------|------------|
|  |  |  |  |  | 4.4 | 19 | KitKat | 10/31/2013 |
| Cupcake Android 1.5 | Donut Android 1.6 | Eclair Android 2.0/2.1 | Froyo Android 2.2.x | Gingerbread Android 2.3.x | 4.3 | 18 | Jelly Bean | 07/24/2013 |
| | | | | | 4.2, 4.22 | 17 | | 11/13/2012 |
| | | | | | 4.1, 4.11 | 16 | | 07/09/2012 |
| | | | | | 4.0.3, 4.0.4 | 15 | Ice Cream Sandwich | 12/16/2011 |
| | | | | | 4.0, 4.01, 4.02 | 14 | | 10/19/2011 |
| | | | | | 3.2 | 13 | Honeycomb | 07/15/2011 |
| | | | | | 3.1.x | 12 | | 05/10/2011 |
| | | | | | 3.0.x | 11 | | 02/22/2011 |
| | | | | | 2.3.3, 2.3.4 | 10 | Gingerbread | 02/02/2011 |
| | | | | | 2.3, 2.3.1, 2.3.2 | 9 | | 12/06/2010 |
| | | | | | 2.2.x | 8 | Froyo | 05/20/2010 |
| | | | | | 2.1.x | 7 | Éclair | 01/12/2010 |
| | | | | | 2.0.1 | 6 | | 12/03/2009 |
| | | | | | 2.0 | 5 | | 10/26/2009 |
| | | | | | 1.6 | 4 | Donut | 09/15/2009 |

Para mais detalhes sobre o histórico das versões do Android consulte : https://pt.wikipedia.org/wiki/Hist%C3%B3rico_de_vers%C3%B5es_do_Android

Aplicações Android

Uma das partes mais importantes de uma aplicação Android é a Atividade (**Activity**).

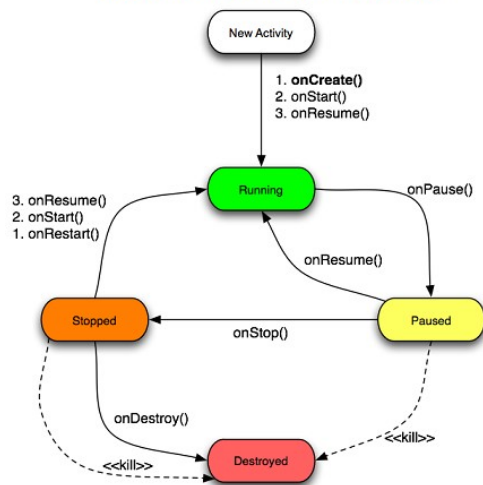
Uma atividade fornece uma única função que um usuário pode realizar com um aplicativo tal como: listar produtos, incluir produtos, exibir mapas, etc.

Uma única aplicação é composta de muitas atividades, e um usuário interage com uma atividade por meio de uma ou mais **views** ou visões. Se você estiver familiarizado com o padrão **MVC**, você vai notar que as atividades cumprem um papel parecido como *Controller*.

As atividades possuem um ciclo de vida bem definido que pode ser descrito em termos de estados, transições e eventos.

O diagrama abaixo fornece uma visão do ciclo de vida de uma atividade:

Ciclo de Vida de uma Atividade



Em outro artigo vou abordar com mais detalhes os estados e eventos de uma atividade.

As Views e as ViewGroups

Tudo o que você vê em um aplicativo Android é uma **View** : botões, etiquetas, caixas de texto e botões de rádio são todos exemplos de **views**.

As **Views** são organizadas em uma hierarquia utilizando diversos tipos de **ViewGroups**. Uma **ViewGroup** é um tipo especial de **View** que é usado para organizar (*layout*) as outras views na tela.

As **Views e as ViewGroups** podem ser criadas usando dois métodos diferentes: **programaticamente ou declarativamente**.

- Ao usar uma abordagem **programática**, o desenvolvedor faz chamadas de API para criar e posicionar cada **View** individual na interface.
- Ao usar uma abordagem **declarativa**, o desenvolvedor cria arquivos de layout XML que especificam como as **Views** devem organizadas.

A abordagem declarativa possui a seguintes vantagens:

- Proporciona uma melhor separação do design visual de um aplicativo da sua lógica de processamento;
- Permite que vários layouts sejam criados para apoiar vários dispositivos ou configurações de dispositivos com uma única base de código;
- Ferramentas de desenvolvimento, tais como Android Studio, Xamarin Studio e o plugin Android para Eclipse, permitem visualizar a interface do usuário, sem a necessidade de compilar e executar o aplicativo após cada alteração;

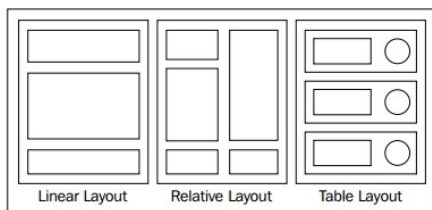
Embora a abordagem declarativa seja a mais indicada às vezes uma combinação das abordagens se faz necessária.

Interface do Usuário (Views, Widgets) e Layouts

O Android oferece um conjunto abrangente de **widgets** de interface que podem ser usados para construir uma rica experiência do usuário. Todos estes elementos são subtipos de views e podem ser organizados em layouts sofisticados que utilizam vários tipos de **ViewGroups**. Todos os **widgets** de interface podem ser encontrados no pacote **android.widget** no Application Framework.

O Application Framework possui um número ode subclasses de **ViewGroup**, cada uma fornece uma única forma de organizar o conteúdo da interface.

Os layouts mais comuns podem ser vistos na figura abaixo:



Cada layout pode ser usado para um propósito específico.

| Layout | Descrição | Cenário |
|------------------------|--------------------------------------------------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Linear Layout | Organize seus ' <i>filhos</i> ' em uma única linha horizontal ou vertical e cria uma barra de rolagem quando necessário. | Indicado quando os widgets fluem horizontal ou verticalmente |
| Relative Layout | Organiza os objetos ' <i>filhos</i> ' relativamente uns aos outros ou em relação ao pai. | Indicado quando as posições dos widgets podem ser melhor descritas em relação a outro elemento (à esquerda) ou à área de fronteira do pai (lado direito, ou centrado) |

| | | |
|---------------------|-----------------------------------------------|---------------------------------------------------------------------------------------|
| Table Layout | Organiza os seus 'filhos' em linhas e colunas | Indicado quando as posições dos widgets naturalmente se encaixam em linhas e colunas. |
|---------------------|-----------------------------------------------|---------------------------------------------------------------------------------------|

Para layouts que são orientados a uma fonte de dados dinâmica o Application Framework possui um conjunto de classes derivada de **AdapterView**.

Nota: Um AdapterView é uma view cujos filhos são determinados por um Adapter que atua como uma ponte entre a view e os dados relacionados.

Dentre esse layouts os mais comuns são :

- **List View** - Organiza o conteúdo da fonte de dados em uma lista de coluna única com rolagem.
- **Grid View** - Organiza o conteúdo da fonte de dados em uma grade de colunas e linhas

Conforme mostra a figura a seguir:



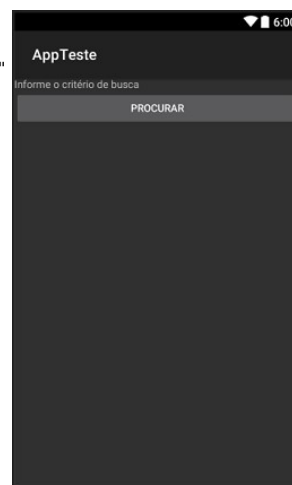
Arquivos de layouts XML

Para criar uma interface do usuário usando a abordagem declarativa, O Android fornece um vocabulário XML com etiquetas que definem os vários tipos de elementos que podem compor uma **View**.

O conceito por trás arquivos de layout XML Android é muito semelhante à maneira como as tags HTML são usadas para definir as páginas web ou as tags XAML da Microsoft são usados para definir as interfaces do usuário em aplicações *WPF (Windows Presentation Foundation)*.

O exemplo a seguir mostra uma view simples usando um layout linear e contendo um campo de entrada de pesquisa e botão de pesquisa e a sua renderização na tela do dispositivo:

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent">
    <TextView
        android:text="Informe o critério de busca"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:id="@+id/criterioBuscaTextView" />
    <Button
        android:text="Procurar"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:id="@+id/procurarButton" />
</LinearLayout>
```



Você deve tomar cuidado ao alinhar os nomes dos elementos e atributos no vocabulário XML com os nomes das classes e dos métodos no Application Framework.

Neste exemplo os nomes dos elementos **LinearLayout**, **TextView** e **Button** correspondem aos nomes das classes no Application Framework. Da mesma forma no elemento Button, o atributo **android:text** corresponde ao alocador **setText()** na classe **Button**.

Os arquivos de layout XML definem as views da interface e estão contidos na pasta **Resources\layout** do projeto no **Xamarin Studio** ou no **Visual Studio**.

Cada **View** pode ter um número inteiro **ID** único a ela associado que pode ser utilizado para referenciar a **View** a partir do código do aplicativo.

No arquivo XML, o ID é especificado como um nome de texto amigável. Por exemplo, considere a seguinte linha de código:

android: id = "@+id/procurarButton"

Neste exemplo, o operador @ diz ao analisador que se deve tratar o restante da cadeia como um recurso ID; o símbolo + instrui o analisador de que este é um novo nome de recurso que deve ser adicionado ao arquivo de recurso.

Dessa forma criar aplicações Android envolve muito mais do que apenas escrever código. Uma aplicação para celular robusta requer recursos

como imagens, áudio, arquivos, menus e estilo.

O **Application Framework** fornece as APIs que podem ser usadas para carregar e utilizar vários tipos de recursos com suas aplicações Android que aliado ao seu código vai compor a sua aplicação.

E concluímos aqui esse resumo onde eu tentei fornecer uma descrição dos principais recursos da plataforma Android e no próximo artigo vou tratar do **Xamarin.Android** e seus recursos para o desenvolvimento Android usando a plataforma .NET e a linguagem C#.

Aguarde mais artigos sobre o desenvolvimento de aplicativos Android usando o Visual Studio 2015 e o Xamarin.

Porque a palavra da cruz é loucura para os que perecem; mas para nós, que somos salvos, é o poder de Deus.

1 Coríntios 1:18

[Veja os Destaques e novidades do SUPER DVD Visual Basic \(sempre atualizado\) : clique e confira !](#)

Quer migrar para o VB .NET ?

- Veja mais sistemas completos para a plataforma .NET no [Super DVD .NET](#) , confira...
- [Curso Básico VB .NET - Vídeo Aulas](#)

Quer aprender C# ??

- Chegou o [Super DVD C#](#) com exclusivo material de suporte e vídeo aulas com curso básico sobre C#.
- [Curso C# Basico - Vídeo Aulas](#)

Quer aprender os conceitos da Programação Orientada a objetos ?

- [Curso Fundamentos da Programação Orientada a Objetos com VB .NET](#) NEW

Quer aprender o gerar relatórios com o ReportViewer no VS 2013 ?

- [Curso - Gerando Relatórios com o ReportViewer no VS 2013 - Vídeo Aulas](#) NEW

Referências:

- [Seção VB .NET do Site Macoratti.net](#)
- [Super DVD .NET - A sua porta de entrada na plataforma .NET](#)
- [Super DVD Vídeo Aulas - Vídeo Aula sobre VB .NET, ASP .NET e C#](#)
- [Super DVD C# - Recursos de aprendizagens e vídeo aulas para C#](#)
- [Seção C# do site Macoratti.net](#)
- [Seção ASP .NET do site Macoratti .net](#)
- [Curso Básico VB .NET - Vídeo Aulas](#)
- [Curso C# Básico - Vídeo Aulas](#)
- [Curso Fundamentos da Programação Orientada a Objetos com VB .NET](#) NEW
- [Macoratti .net | Facebook](#)
- [macoratti - YouTube](#)
- [Jose C Macoratti \(@macoratti\) | Twitter](#)
- [Xamarin - Desenvolvimento Multiplataforma com C# ... - Macoratti.net](#)
- [Xamarin - Apresentando Xamarin.Forms - Macoratti.net](#)
- [Xamarin.Forms - Olá Mundo - Criando sua primeira ... - Macoratti.net](#)
- [Xamarin.Forms - Olá Mundo - Anatomia da aplicação - Macoratti.net](#)

José Carlos Macoratti