Minicurso introdutório de desenvolvimento para dispositivos Android

Cristiano Costa

Desenvolvimento da interface estática

Sobre mim

- Graduando em Engenharia Eletrônica - UNIFEI

- Trabalhou em desenvolvimento de sistemas empresariais (Base em servidores LAMP)

- Começou a desenvolver para Android na API 8 (Android 2.2)

História do Android

- Criado autonomamente em 2003 na recém-fundada Android Inc.
- Comprado pelo Google em 2005
- Primeira versão publica lançada em 2008 no dispositivo HTC Dream
- Atualmente o Android está na versão 5.1

Android Studio

- Nova IDE de desenvolvimento para Android
- Baseado na versão open-source do IntelliJ (IDE de desenvolvimento Java)
- Possui editor visual para as telas, similar ao método empregado no Visual Studio, QT Creator e similares

APIs de Desenvolvimento

- Cada versão do Android usa uma versão da API de Desenvolvimento (Ex: API 8 – 2.3, API 15 – 4.0.3, API 19 – 4.4)
- Aparelhos com uma determinada versão do Android executam aplicativos criados com a API equivalente em versão atual ou anterior
- Desenvolver utilizando APIs novas trazem mais funcionalidades, porem menor compatibilidade
- Desenvolver utilizando APIs antigas trazem maior compatibilidade, porem menos funcionalidades

A máquina virtual Dalvik

- Máquina virtual que executa os aplicativos no Android
- Executa bytecode semelhante à máquina virtual JAVA (JVM)
- Permite execução dos processos em ambiente com **chroot**, garantindo a segurança do sistema
- Assim como a JVM, o bytecode é traduzido Just-in-Time, aumentando o gasto de recursos e processamento
- Vem sendo substituída pela máquina virtual ART, que possui compilação Ahead-of-Time, onde parte do bytecode é compilado para a linguagem nativa do hardware no momento da instalação do aplicativo

Localização dos arquivos no armazenamento

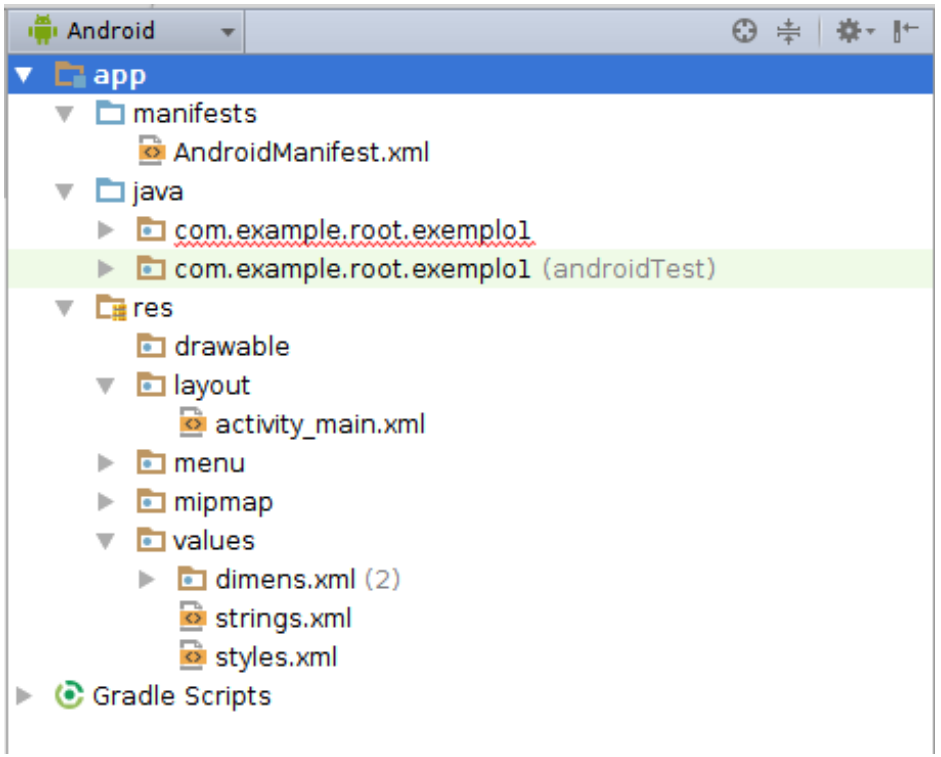
- Bytecode, arquivos de recursos e bases de dados são armazenados na pasta raiz do aplicativo, localizada em:

  `/data/data/`**`br.com.pacote/`**

- Recursos armazenados na pasta raiz são fáceis de ser acessados e estão seguros (os aplicativos não conseguem acessar a pasta raiz uns dos outros)

- Recursos também podem ser armazenados no cartão de memória (`/sdcard`), porém ficam de maneira insegura, pois todos os aplicativos tem acesso completo

Estrutura dos arquivos no pacote



- **java:** Contém os arquivos de código .java estruturados no padrão de pacotes
- **res:** Contém os arquivos de recursos, subdivididos em:
- **drawables:** Arquivos de imagem usados na interface
- **layout:** Arquivos usados para configurar as opções de estilo do sistema (barras superiores e afins)
- **menu:** Arquivos relacionados ao menu padrão do aplicativo
- **mipmap:** Ícones
- **values:** Arquivos de valores que podem ser alterados por condições do aparelho

(parênteses) O XML

- e**X**tensible **M**arkup **L**anguage
- Linguagem de marcação baseada em **tags**
- Surgiu como uma especificação do SGML, mesma linguagem em que o HTML se baseia
- Tags podem ser aninhadas umas às outras e possuem propriedades

(parênteses) O XML

Exemplo:

```
<exemplo>
    <TagAninhada propriedade="valor">Texto</TagAninhada>
    <TagSemTexto></TagSemTexto>
    <TagFechada />
</exemplo>

<carro>
    <carroceria cor="azul">
        <porta lado="esquerdo" posicao="frente">Porta 1</porta>
        <porta lado="direito" posicao="frente">Porta 2</porta>
        <porta lado="esquerdo" posicao="tras">Porta 3</porta>
        <porta lado="esquerdo" posicao="tras">Porta 4</porta>
    </carroceria>
    <rodas aro="20" />
</carro>
```

Exemplo de **dois** arquivos XML

- Cada arquivo XML pode ter somente UMA tag superior
- As tags podem ser aninhadas infinitamente, assim como as tags podem ter infinitas propriedades
- As tags podem se “auto-fechar”, caso não necessitem de conteúdo textual
- Facilidade para descrever objetos, suas propriedades e sua hierarquia

A pasta **res**

- Pasta onde se localizam os arquivos de recursos
- Só pode conter pastas nomeadas de acordo com o recurso que conterá (como consta na documentação) e os sufixos permitidos de acordo com o tipo de recurso.
  Ex: `values`, `values-fr`, `drawable`
- Pastas que contém sufixos de localização, resolução de tela e afins serão utilizadas preferencialmente, fazendo com que as pastas sem sufixo sejam default (sendo utilizada caso não exista uma pasta específica para as configurações do aparelho)

Arquivos de recursos

- Para minimizar a complexidade do código e facilitar a visualização da hierarquia das estruturas, a aplicação conta com arquivos XML para descrever as telas, strings para internacionalização, configurações gerais entre outras coisas
- Arquivos ficam separados nas respectivas pastas e podem ser acessados pela seguinte sintaxe dentro do código:
  `R.pasta.recurso`
  Exceto a pasta *values*, onde seus arquivos são acessados diretamente
- Devido o acesso ser feito através do objeto **R**, não é possível acessar os recursos fora das suas respectivas pastas (como em algumas linguagens, onde se usa caminhos relativos)

Arquivos de estilos

- Ficam na pasta **layout**
- Cada tela, caixa de diálogo, widget e similares possui um XML de estilo associado.
- O arquivo de estilo define layouts e elementos de UI (como botões, entradas de texto, textos ou seja, tudo o que é visto na tela)
- Cada elemento possui um **id** que será usado para se referenciar ao elemento no código para que seja possível manipulá-lo e adicionar funcionalidades à ele

Arquivos de estilos

Exemplo:

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:orientation="vertical" >

    <TextView
        android:id="@+id/selectDialogTextView"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="" />

    <ListView
        android:id="@+id/selectDialogListView"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content" >
    </ListView>

</LinearLayout>
```

Arquivos de estilo

Aonde buscar a documentação?

- O arquivo de estilo possui um editor gráfico. Rapidamente é possível descobrir todos os elementos de UI que o Android possui
- Para a documentação completa de cada elemento, contendo TODOS os métodos e propriedades disponíveis pode ser consultado em: http://developer.android.com/reference/android/package-summary.html

Elementos de UI

- Todo elemento de UI é uma **classe** que estendida da classe **View**
  http://developer.android.com/reference/android/view/View.html
- Devido a hierarquia compartilhada, grande parte dos métodos e propriedades dos elementos são idênticos facilitando a manipulação
- No código, quando requisitamos a instância de um elemento, nos é retornado com o tipo **View** e devemos fazer o *casting* para o tipo filho correto

Arquivo de menu

- Ficam na pasta **menu**
- Determina o menu **padrão** do android (Aquele que pode ser acessado segundando a tecla de switch ou clicando no ícone superior direito)
- Cada item do menu é equivalente à um botão da interface e ao ser pressionado, pode executar um método
- O padrão android prevê apenas um menu para a aplicação, não sendo recomendado (para não sair dos padrões de desenvolvimento e causar incompatibilidades), implementar um sistema de menu usando elementos de UI não destinados à isso

Arquivo de menu

Exemplo:

```
<menu xmlns:android="http://schemas.android.com/apk/res/android" >

    <item
        android:id="@+id/action_settings"
        android:orderInCategory="100"
        android:showAsAction="never"
        android:title="@string/action_settings"/>

</menu>
```

Arquivos de menu

Aonde buscar a documentação?

http://developer.android.com/guide/topics/ui/menus.html

Arquivos gráficos

- Ficam na pasta **drawable**
- Todos os formatos comuns de imagem são suportados (jpg, png, gif, consultar documentação para mais formatos)
- Geralmente divididos em pastas com o sulfixo de resolução (hdpi, ldpi, mdpi, ex: drawable-hdpi)
- Ao se usar diferentes sufixos nas pastas, é necessário que todas as imagens sejam replicadas com resoluções diferentes, pois o android escolherá a resolução da imagem mais compatível com a resolução de tela do aparelho

A pasta **value**

- Contém os XML referentes à internacionalização, dimensionamento do conteúdo conforme resolução de tela e arquivos de estilo das partes geridas pelo sistema
- Usualmente é separada usando sufixos para fins propostos, por exemplo:
  Pode-se usar *values-en* para adicionar o arquivo de string para traduzir o aplicativo em inglês nos dispositivos configurados nessa língua, pois, mesmo que na pasta *value* default conter o arquivo de string em português, ele será sobrecarregado
- Seus arquivos podem ser acessados diretamente pelo objeto **R** no código, por exemplo:
  R.string.**id**

Arquivos de strings

- Para a organização de strings estáticas e para facilitar a internacionalização, elas devem ser colocadas no arquivo strings.xml
- Elas devem ser colocadas como um elemento derivado diretamente do elemento raiz, segundo o seguinte modelo XML:
  `<string name="id">A frase aqui</string>`
- No código, a string pode ser acessada como um recurso normal, usando:
  `R.string.id`

Arquivo de strings

Exemplo:

```xml
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string name="app_name">Water Control</string>
    <string name="action_settings">Settings</string>
    <string name="title_section1">General</string>
    <string-array name="quantity_spinner">
        <item>50 ml</item>
        <item>100 ml</item>
        <item>150 ml</item>
        <!-- ... -->
        <item>950 ml</item>
        <item>1000 ml</item>
    </string-array>
    <string name="changelog_full_title">Change Log</string>
    <string name="changelog_title">What\'s New</string>
    <string name="changelog_ok_button">OK</string>
    <string name="changelog_show_full">more&#8230;</string>
</resources>
```

Arquivo de strings

Aonde buscar a documentação?

http://developer.android.com/guide/topics/resources/string-resource.html

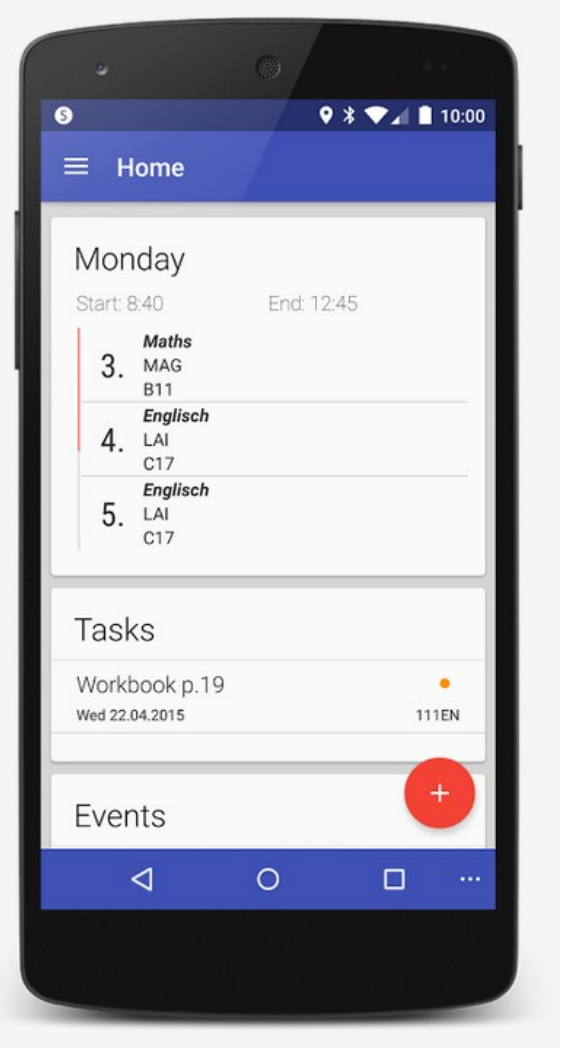
Arquivos adicionais

- Os outros arquivos presentes na pasta de recursos são usados somente para fins muito específicos, portanto não serão abordados nesse minicurso
- A documentação para eles pode ser encontrada facilmente no site do android developer

O arquivo manifest

- Contém as configurações de execução do aplicativo, permissões, definições sobre os processos foreground e background, listeners de eventos do sistema e definições das telas
- Uma abordagem mais completa do arquivo AndroidManifest.xml será feita na próxima aula, quando trabalharemos com o código

Material design



- http://www.google.com/design/spec/material-design/introduction.html
- Nova padronização da interface proposto pelo google
- Usa elementos simples, com uma paleta de cor determinada, estilos e sombras
- Todas as diretrizes de estilização estão dispostas no link

Exemplo prático

Exemplo prático