Introdução ao desenvolvimento Android

Programação para dispositivos móveis

Prof. Allan Rodrigo Leite

Desenvolvimento de aplicativos Android

- Ferramentas e linguagens de programação para desenvolvimento de aplicativos nativos para Android
  - Suporta linguagens de programação Java e Kotlin
  - Aplicativos são compilados utilizando o JDK (Java Development Kit)
    - Java é a mais utilizada
    - Execução é realizada pela JVM (Java Virtual Machine)
    - Arquivos compilados são reunidos em um arquivo Android Package (APK)

  - Ferramentas populares utilizadas para o desenvolvimento Android
    - Android Studio
    - Android Emulator
    - Android Virtual Device
    - Android Debug Bridge

Ambiente de desenvolvimento Android Studio

- Ambiente de desenvolvimento integrado especificamente projetado para desenvolvimento de aplicações Android
  - Disponível em Windows, Linux e MacOS
  - Substituto do Eclipse Android Development Tools (ADT)
  - Baseado no ambiente de desenvolvimento IntelliJ

  - Principais recursos disponíveis
    - Suporte para diferentes dispositivos (celular, tablet, TV, veículo, relógio)
    - Integração com builders (gradlew)
    - Integração com Android Emulator
    - Debugging
    - Monitoramento e logging

Ambiente de desenvolvimento Android Studio

- Estrutura básica de um projeto Android
  - `java`
    - Armazena códigos da aplicação (classes, interfaces, enums, etc.)
    - Todos os artefatos de código são organizados em pacotes
    - Existem pacotes para testes (testes unitários e ou mais amplos)
  - `res`
    - Armazena recursos da aplicação (layouts, textos, ícones, cores, etc.)
    - Textos têm suporte a internacionalização (i18n)
    - Layouts e ícones têm suporte para diferentes resoluções de dispositivos
  - Gradle Scripts
    - Gradle é o gerenciador de tarefas para construção e testes da aplicação
    - Tarefas comuns são `build` e `test`

Ferramentas de desenvolvimento

- Android Emulator
  - Ferramenta que permite emular um dispositivo Android
    - É possível executar diferentes versões de um dispositivo Android
    - Oferece quase todos os recursos de um dispositivo Android real

- Android Virtual Device (AVD)
  - Configuração que define características de um dispositivo Android
    - Celular, tablet, relógio, TV, veículo, etc.
    - Resolução do dispositivo
    - Capacidades computacionais (armazenamento, processamento, modelo)

Ferramentas de desenvolvimento

- Android Debug Bridge (ADB)
  - Ferramenta CLI que permite a comunicação com um dispositivo
  - Facilita e permite a automação de algumas tarefas como
    - Instalação e desinstalação de aplicativos
    - Depuração de código
    - Monitoramento e logging
    - Roteamento da comunicação com um dispositivo (port forward e reverse)

Fundamentos de aplicativos Android

- O sistema operacional Android é um sistema Linux multiusuário
  - Cada aplicativo é um usuário diferente
    - O sistema atribui a cada aplicativo um código de usuário do Linux exclusivo
    - O sistema define permissões para todos os arquivos em um aplicativo
    - Somente o código de usuário atribuído àquele aplicativo pode acessá-los
  - Cada processo tem a própria máquina virtual
    - O código de um aplicativo é executado isoladamente de outros aplicativos
    - Porém, cada aplicativo é executado no próprio processo do Linux
    - O Android inicia o processo quando é preciso executar algum componente do aplicativo

Fundamentos de aplicativos Android

- O sistema Android implementa o princípio do privilégio mínimo
  - Há privilégios apenas nos componentes necessários para a execução
  - Um aplicativo pode solicitar permissões adicionais ao usuário
    - O usuário precisa conceder essas permissões de forma explícita

Principais componentes de um aplicativo

- Activities
  - Ponto de entrada para a interação com o usuário
    - Representa uma tela única com uma interface do usuário
    - As activities podem interagir, porém, são independentes entre si
  - Exemplo de activities em um aplicativo de e-mails
    - Pode ter uma atividade que mostra uma lista de novos e-mails
    - Outra activity para escrever e enviar novos e-mails
    - Outra activity para ler e-mails

Principais componentes de um aplicativo

- Services
  - Forma para manter um aplicativo em execução no segundo plano
    - Executa em segundo plano para realizar operações de execução longa
    - Também utilizado em trabalhos para processos remotos
    - Services não apresentam uma interface do usuário, embora seja possível ter interação de uma activity com um service
  - Exemplo de service em um aplicativo de e-mails
    - Service para monitorar o recebimento de novos e-mails e notificar o usuário

Principais componentes de um aplicativo

- Intents
  - Mecanismo no sistema operacional Android para coordenar funções em diferentes activities
    - Objeto de mensagem para intercomunicação entre diferentes aplicativos
    - Geralmente utilizado para executar activities e como cola entre activities
  - Exemplo de service em um aplicativo de e-mails
    - Anexar uma foto capturada e anexá-la no envio de um e-mail

Principais componentes de um aplicativo

- Broadcast receiver
  - Componente que responde à eventos do sistema
    - Isto é, são eventos entregues ao aplicativo fora de fluxo de usuários comum
    - Cada transmissão é entregue como um objeto intent

  - Exemplo de service em um aplicativo de e-mails
    - Desligamento da tela durante a escrita de um novo e-mail pode transmitir um evento para salvar a mensagem como rascunho

Principais componentes de um aplicativo

- Content providers
  - Gerencia um conjunto compartilhado de dados do aplicativo
  - Dados podem ter sido armazenados em
    - Sistema de arquivos do dispositivo
    - Banco de dados SQLite
    - Algum serviço Web
    - Qualquer local de armazenamento persistente acessível pelo aplicativo
  - Exemplo de service em um aplicativo de e-mails
    - Gerenciamento dos anexos baixados e salvos localmente no dispositivo

Activities

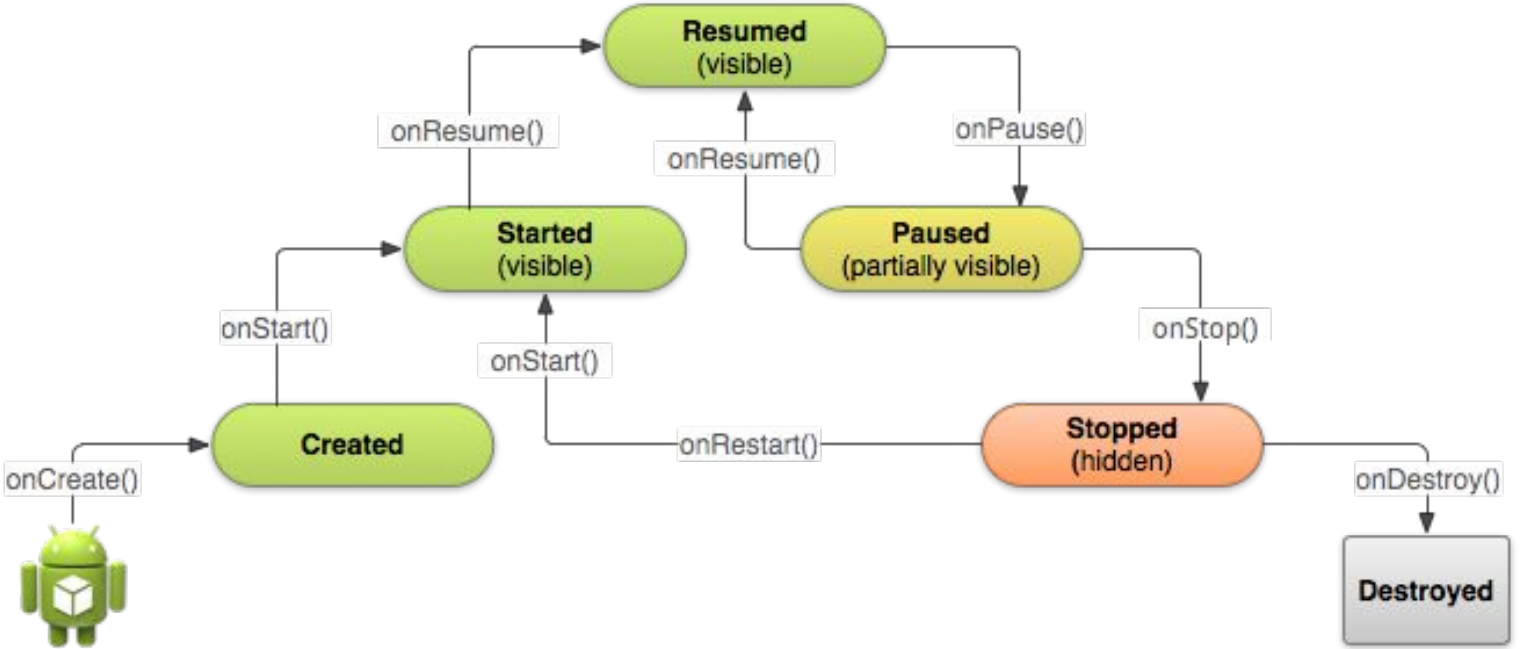
- Activities são implementados como subclasse da classe `Activity`

```
public class MainActivity extends AppCompatActivity {

  @Override
  protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
  }
}
```

Activities

- Ciclo de vida de uma activity



Activities

- Activities são implementados como subclasse da classe `Activity`

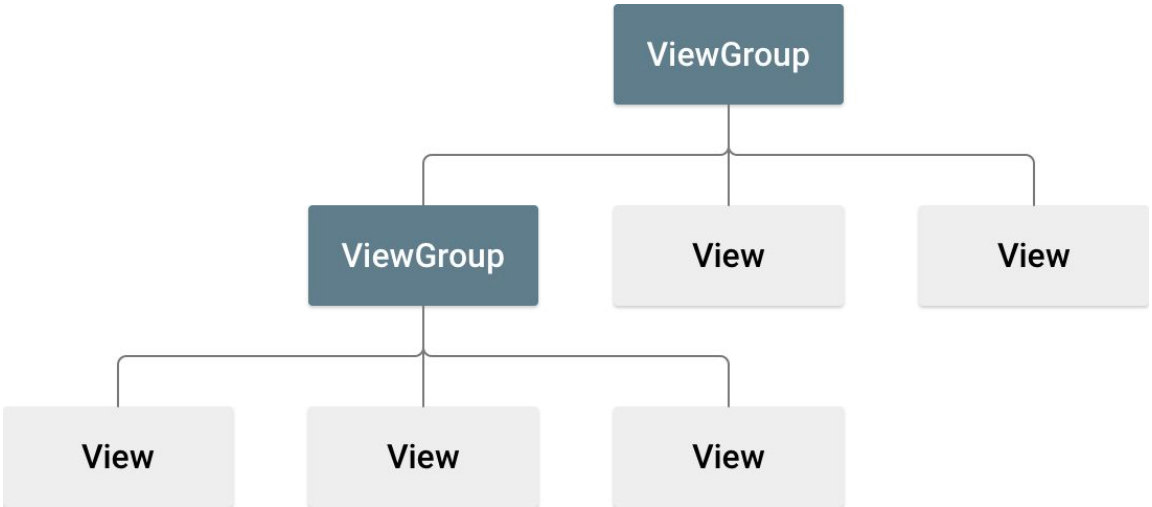
```
public class MainActivity extends AppCompatActivity {
  private static final String TAG = "LIFE_CYCLE";

  @Override
  protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
    Log.i(TAG, "onCreate");
  }

  @Override
  protected void onStart() {
    super.onStart();
    Log.i(TAG, "onStart");
  }
}
```

Definindo layouts de uma activity

- Define a estrutura de uma interface do usuário em uma activity
  - Elementos do layout são organizados em hierarquia de:
    - `View`: componente visual que o usuário pode ver e interagir
    - `ViewGroup`: container invisível que define o layout de um grupo de views

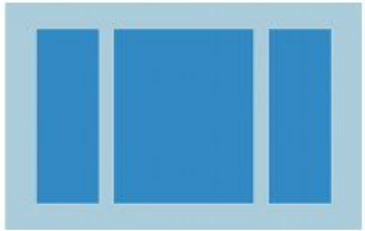


Definindo layouts de uma activity

- Um layout pode ser declarado de duas maneiras
  - Declarar componentes visuais em XML
    - Android fornece um vocabulário XML direto
    - Representação de elementos XML para widgets e layouts
  - Instanciar componentes visuais via código
    - Manipulação dos componentes visuais por classes e subclasses
    - Classes e subclasses correspondem a componentes visuais

Definindo layouts de uma activity

- Principais layouts
  - Layout linear (`LinearLayout`)
    - Organiza os filhos em uma única linha horizontal ou vertical
    - Cria-se uma barra de rolagem se o comprimento exceder a tela



  - Layout relativo (`ConstraintLayout`)
    - Localização de componentes filhos relativos entre si ou relativos aos pais
    - Exemplo: filho A à esquerda do filho B, alinhado na parte superior do pai

Definindo layouts de uma activity

- Componentes visuais em XML

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
              android:layout_width="match_parent"
              android:layout_height="match_parent"
              android:orientation="vertical" >
    <TextView android:id="@+id/text"
              android:layout_width="wrap_content"
              android:layout_height="wrap_content"
              android:text="Um texto qualquer" />
    <Button android:id="@+id/button"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="Um botão qualquer" />
</LinearLayout>
```

Definindo layouts de uma activity

- Atributos de componentes visuais em XML
  - `orientation`
    - Define a orientação dos componentes no layout (vertical ou horizontal)
  - `layout_width` e `layout_height`
    - Definem a largura e altura do componente visual
    - `wrap_content` define dimensões do componente pelo seu conteúdo
    - `match_parent` define dimensões do componente pelo container pai
    - É possível usar unidades absolutas como pixel, porém, não se recomenda

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
              android:layout_width="match_parent"
              android:layout_height="match_parent"
              android:orientation="vertical" >
...
</LinearLayout>
```

Definindo layouts de uma activity

- Componentes visuais em XML
  - Cada arquivo de layout XML é compilado em um recurso `View`
  - O layout é carregado pelo callback `Activity.onCreate()`
    - Deve ser invocado o método `setContentView()`
    - O argumento da invocação é a referência do layout XML

```
@Override
protected void onCreate(Bundle savedInstanceState) {
  super.onCreate(savedInstanceState);
  setContentView(R.layout.activity_main);
}
```

Definindo layouts de uma activity

- Classe R
  - Referência estática para o diretório de recursos (resources) do projeto
  - Esta classe é criada estaticamente durante o build da aplicação
  - Principais métodos
    - `id`: referência de identificadores únicos para cada objeto disponível
    - `layout`: acessa os layouts disponíveis
    - drawable: acessa gráficos vetoriais disponíveis
    - `mipmap`: acessa ícones disponíveis

Adicionando ações em um componente visual

- Exemplo para adicionar uma ação em um botão
  - Necessário determinar um identificador para o botão
    - O identificador é utilizado para obter o objeto (instância) do botão
    - Em seguida é feito o binding do comportamento da ação com o objeto
  - Recomenda-se usar o resource string para todos os textos dos widgets
    - Todas as strings precisam estar declaradas no XML strings

```
<Button android:id="@+id/button_main"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/button_text" />
```

Adicionando ações em um componente visual

- Exemplo para adicionar uma ação em um botão (cont.)
  - A interface `OnClickListener` implementa a ação de click do botão

```
@Override
protected void onCreate(Bundle savedInstanceState) {
  super.onCreate(savedInstanceState);
  setContentView(R.layout.activity_main);

  Button button1 = findViewById(R.id.button_main_1);
  button1.setOnClickListener(view ->
    Toast.makeText(
      MainActivity.this, R.string.alert_text, Toast.LENGTH_LONG).show()
  }
}
```

Adicionando ações em um componente visual

- Mapeando a ação diretamente no XML
  - Não é necessário adicionar o identificador do componente
  - Usa-se o atributo onClick para especificar o método a ser invocado

```
<Button android:onClick="showToast"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/button_text" />
```

Adicionando ações em um componente visual

- Mapeando a ação diretamente no XML (cont.)
  - Não é necessário adicionar o identificador do componente
  - Usa-se o atributo onClick para especificar o método a ser invocado

```
@Override
protected void onCreate(Bundle savedInstanceState) {
  ...

  public void showToast(View view) {
    Toast.makeText(
      MainActivity.this, R.string.alert_text, Toast.LENGTH_LONG).show()
  }
}
```

Construindo layouts programaticamente

- Criando instâncias de componentes visuais
  - Utilizar `this` como referência de contexto
  - Obter instância do componente pai e adicioná-lo no layout

```
@Override
protected void onCreate(Bundle savedInstanceState) {
  super.onCreate(savedInstanceState);
  setContentView(R.layout.activity_main);

  LinearLayout layout = findViewById(R.id.linear_layout_main);

  Button button = new Button(this);
  button.setText("Botão adicionado dinâmicamente");
  button.setLayoutParams(new ViewGroup.LayoutParams(
      ViewGroup.LayoutParams.MATCH_PARENT,ViewGroup.LayoutParams.WRAP_CONTENT));
  layout.addView(button);
}
```

Exercícios

1. Desenvolva um aplicativo para o jogo da velha, considerando pessoas como ambos os jogadores. O jogo deve iniciar com a jogada do jogador X. O jogo deve possuir uma ação para reiniciar a partida a qualquer momento. Dica: utilize `TableLayout` para organizar o tabuleiro do jogo.

2. Altere o aplicativo anterior para um único jogador e o oponente ser o próprio dispositivo.

Introdução ao desenvolvimento Android

Programação para dispositivos móveis

Prof. Allan Rodrigo Leite