Android com Kotlin

Capítulo 1: Iniciando com Android e Kotlin

https://github.com/rafael-vieira-coelho/android

Capítulo 1: Iniciando com Android e Kotlin

- Aprender como Kotlin e Android trabalham juntos
- Como configurar a IDE Android Studio
- Compreender o que é um Android app
- Como fazer o deploy de um app em um emulador Android
- Como rodar um app em um celular

Por que não usar Java com Android?

- Android existe desde 2008, sendo grátis e aberta.
- O Software Development Kit (SDK) do Android é implementado em boa parte em Java.
- Independente se o seu app foi implementado em Java ou Kotlin, gera-se um código DEX (Dalvik Executable) para rodar no celular.
- Diferentemente de Java, Kotlin é uma linguagem sucinta e de curva de aprendizado mais acentuada. E ela visa fazer mais com menos código.
- Como exemplos de apps implementados em Kotlin: Kindle, Twitter e Netflix.

O que são os recursos de Android?

- Além do código-fonte em linguagem Kotlin, podemos ter arquivos auxiliares em nossos projetos Android.
- Podem ser imagens, arquivos de som, layout, etc.
- Para organizar estes arquivos podemos separá-los pacotes (packages).
- Os pacotes que são disponibilizados pela API Android: “https://developer.android.com/reference/packages

Como Android e Kotlin trabalham juntos?

- Android roda em uma versão simplificada do Sistema Operacional Linux.
- Precisamos aprender a usar as classes da API do Android.
- O código DEX (juntamente com outros recursos) são armazenados em um arquivo APK (Android Application Package).
- Iremos programar na IDE Android Studio.

INSTALANDO IDE ANDROID STUDIO

Como Configurar o Android Studio?

https://developer.android.com/studio?hl=pt

- Inicialmente, precisamos baixar o software no site.
- Ele detectará automaticamente o seu SO.

- Aceitar os termos e condições

- Iniciar a instalação (next).

- Selecionar os componentes que devem ser instalados.

- Definir a pasta destino da instalação.

- Definir qual pasta do menu iniciar será colocado seu atalho.

- Esperar a finalização da instalação e clicar em next.

- Finalizar a instalação.
- Mantenha marcada a caixa “Start Android Studio”.

- Esta é a janela de boas vindas que será executada somente na primeira vez.

- Escolhemos o tipo de instalação padrão e clicamos em next.

- Escolhemos o tema da IDE (escuro ou claro).

- É realizada uma verificação da instalação de seus componentes.

CRIANDO UM NOVO PROJETO

- E é mostrada a tela inicial do Android Studio.

- Quando criamos um projeto novo (primeira opção da tela anterior), devemos escolher qual tipo de projeto é inicialmente.

- Precisamos configurar o projeto.

- Quando o projeto terminar de ser criado, acontecerá um erro pois o sdk do Android ainda não foi baixado.

- É necessário clicar no link em azul para instalar.

Android Virtual Device Manager
Your Virtual Devices
Android Studio
Type
Name
Play Store
Resolution
API
Target
CPU/ABI
Size on Disk
Actions
Pixel 2 XL API 28
1440 × 2880: 560dpi
28
Android 8.+ (Google APIs)
x86
10 GB
Create Virtual Device...

Actions

File
Edit
View
Navigate
Code
Analyze
Refactor
Build
Run
Tools
VCS
Window
Help
app

- Este é o layout do nosso celular do emulador.

PrimeiroApp [~/Desktop/Curso LibGDX/ud405-master/PrimeiroApp] - .../app/src/main/java/com/example/primeiroapp/MainActivity.kt

PrimeiroApp > app > src > main > java > com > example > primeiroapp > MainActivity

Add Configuration... | Pixel 2 API 29

Android

- PrimeiroApp ~/Desktop/Curso LibGDX/ud405-
  - .gradle
  - .idea
  - app
    - libs
    - src
      - androidTest
      - main
        - java
          - com
            - example
              - primeiroapp
                - MainActivity
      - res
      - AndroidManifest.xml
    - test
    - .gitignore
    - build.gradle
    - proguard-rules.pro
  - gradle
  - .gitignore
  - build.gradle
  - gradle.properties
  - gradlew
  - gradlew.bat
  - local.properties
  - settings.gradle
  - Gradle Scripts

activity_main.xml | MainActivity.kt

```kotlin
package com.example.primeiroapp

import ...

class MainActivity : AppCompatActivity() {

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)
    }
}
```

- Este é o código-fonte padrão em linguagem Kotlin que carrega a view.

```kotlin
package com.example.primeiroapp

import androidx.appcompat.app.AppCompatActivity
import android.os.Bundle

class MainActivity : AppCompatActivity() {

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)
    }
}
```

Arquivo AndroidManifest.xml

- Este arquivo descreve algumas informações básicas sobre o nosso App.
- Caso nossa aplicação necessite de permissões especiais para rodar, deve estar declarado no manifesto.

```xml
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.primeiroapp">

    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="PrimeiroApp"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:theme="@style/AppTheme">
        <activity android:name=".MainActivity">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>

</manifest>
```

AndroidManifest.xml

Text Merged Manifest

View activity_main.xml

1. A **barra de ferramentas** permite realizar diversas ações, inclusive executar apps e inicializar ferramentas do Android.
2. A **barra de navegação** ajuda a navegar pelo projeto e a abrir arquivos para edição. Ela oferece uma visualização mais compacta da estrutura visível na janela **Project**.
3. A **janela do editor** é onde você cria e modifica o código. Dependendo do tipo de arquivo atual, o editor pode mudar. Por exemplo, ao visualizar um arquivo de layout, o editor abre o Editor de layout.
4. A **barra de janela de ferramentas** fica fora da janela do ambiente de desenvolvimento integrado e contém os botões que permitem expandir ou recolher as janelas de cada ferramenta.
5. As **janelas de ferramentas** permitem acessar tarefas específicas, como gerenciamento de projetos, pesquisa e controle de versões, entre outras. As janelas podem ser expandidas e recolhidas.
6. A **barra de status** exibe o status do projeto e do próprio ambiente de desenvolvimento integrado, bem como todos os avisos ou mensagens.

Alguns Atalhos

Janela de ferramentas	Windows e Linux	Mac
Projeto	Alt + 1	Command + 1
Controle de versões	Alt + 9	Command + 9
Executar	Shift + F10	Control + R
Depurar	Shift + F9	Control + D
Logcat	Alt + 6	Command + 6
Voltar ao editor	Esc	Esc
Ocultar todas as janelas de ferramentas	Control + Shift + F12	Command + Shift + F12

Como funciona um Projeto Gradle?

Sistema de compilação Gradle

O Android Studio usa o Gradle como o sistema de compilação de base, com outros recursos específicos do Android disponibilizados pelo Plug-in do Android para Gradle. Esse sistema de compilação é executado como uma ferramenta integrada no menu do Android Studio e de forma independente na linha de comando. Você pode usar os recursos do sistema de compilação para fazer o seguinte:

- Personalizar, configurar e ampliar o processo de programação.
- Criar diversos APKs para seu app com diferentes recursos usando o mesmo projeto e os mesmos módulos.
- Reutilizar código e recursos nos conjuntos de origem.

A flexibilidade do Gradle permite que você faça tudo isso sem modificar os arquivos de origem principais do seu app. Os arquivos de versão do Android Studio recebem o nome de `build.gradle`. Eles são arquivos de texto simples que usam a sintaxe do Groovy para configurar a criação com elementos fornecidos pelo Plug-in do Android para Gradle. Cada projeto tem um arquivo de compilação de nível superior para todo o projeto e arquivos de compilação de módulo separados para cada módulo. Quando você importa um projeto existente, o Android Studio gera automaticamente os arquivos de compilação versão.

Para saber mais sobre o sistema de compilação e como configurá-lo, consulte Configurar sua compilação.

Estrutura do Projeto Android

A visualização de projeto Android

Para ver a estrutura de arquivos real do projeto, incluindo todos os arquivos ocultos na visualização do Android, selecione **Project** no menu suspenso na parte superior da janela Project.

Quando você seleciona a visualização **Project**, pode ver um número muito maior de arquivos e diretórios. Os mais importantes são:

module-name/

`build/`

Contém saídas de compilação.

`libs/`

Contém bibliotecas privadas.

`src/`

Contém todos os arquivos de código e recursos do módulo nos seguintes subdiretórios:

`androidTest/`

Contém o código dos testes de instrumentação executados em um dispositivo Android. Para ver mais informações, consulte a documentação do Android Test.

Estrutura do Projeto Android

`main/`

Contém os arquivos do conjunto de origem "main": o código Android e os recursos compartilhados por todas as variantes de compilação. Os arquivos para outras variantes de compilação ficam em diretórios irmãos, como `src/debug/` para o tipo de compilação de depuração.

`AndroidManifest.xml`

Descreve a natureza do aplicativo e de cada um dos componentes dele. Para ver mais informações, consulte a documentação AndroidManifest.xml.

`java/`

Contém os códigos-fonte Java.

`jni/`

Contém o código nativo que usa a Java Native Interface (JNI). Para ver mais informações, consulte a documentação do Android NDK.

`gen/`

Contém os arquivos Java gerados pelo Android Studio, como o arquivo `R.java` e as interfaces criadas de arquivos AIDL.

Estrutura do Projeto Android

`res/`

Contém recursos de aplicativos, como arquivos drawable, arquivos de layout e strings de IU. Consulte Recursos de aplicativo para ver mais informações.

`assets/`

Contém o arquivo que precisa ser compilado em um arquivo `.apk` no estado em que está. Você pode navegar nesse diretório da mesma maneira que um sistema de arquivos típico usando URIs e ler arquivos como um fluxo de bytes usando `AssetManager`. Por exemplo, esse é um bom local para texturas e dados de jogos.

`test/`

Contém código para testes locais executados na JVM host.

`build.gradle` **(módulo)**

Define as configurações de compilação do módulo.

`build.gradle` **(projeto)**

Define a configuração de compilação que se aplica a todos os módulos. Esse arquivo faz parte do projeto, então precisa ser mantido no controle de revisões em conjunto com todo o código-fonte restante.
Para saber mais sobre outros arquivos de compilação, consulte Configurar sua compilação.

Rodando o App em um Celular Real

Como rodar um App em um celular?

1. Deve-se acessar o site do fabricante do seu celular para verificar se não é necessário baixar e instalar drivers para o respectivo celular em sua máquina (principalmente se for um celular antigo e no Windows).

2. Antes de conectar o seu celular via USB na máquina, precisa acessar suas configurações e habilitar o modo desenvolvedor (Developer Options). Ele deve estar no modo DEBUG.

3. Conecte via USB o seu celular.

4. Instale o seu App no celular. Caso não tenha sucesso, reconstrua o projeto (rebuild)

For ADB to recognize your device as a target for deploying debuggable APKs, you must first enable USB debugging in the on-device developer options.

Depending on the version of Android you're using, proceed as follows:

- On Android 8.0 and higher, go to **Settings > System > About phone** and tap **Build number** seven times.
- On Android 4.2 through 7.1.2, go to **Settings > About phone** and tap **Build number** seven times.

Return to the main **Settings** menu to find **Developer options** at the bottom. In the **Developer options** menu, scroll down and enable **USB debugging**.

LINKS

https://www.oracle.com/java/technologies/javase-downloads.html

https://developer.android.com/studio

https://developer.android.com/studio/run/emulator.html