Geração de APK

Computação para Dispositivos Móveis

Prof. Gustavo Custodio
gustavo.custodio@ulife.com.br

Geracao de APK

Criando um Emulador

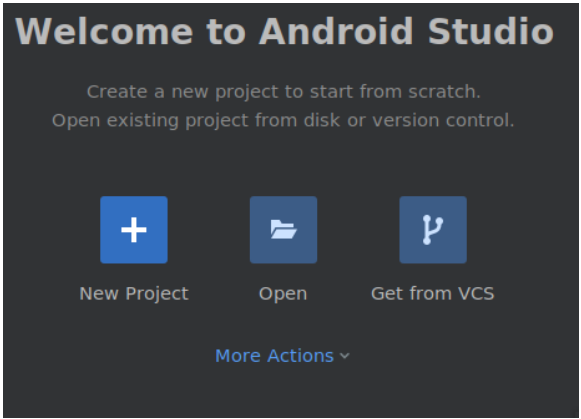
Emulador Android

- O `Android Studio` nos fornece o `AndroidSDK`, um conjunto de ferramentas para desenvolvimento de software.
- Além disso, ele também nos fornece um **emulador** de Android
  - que permite trabalharmos com diversos dispositivos `android` diferentes.

Emulador Android

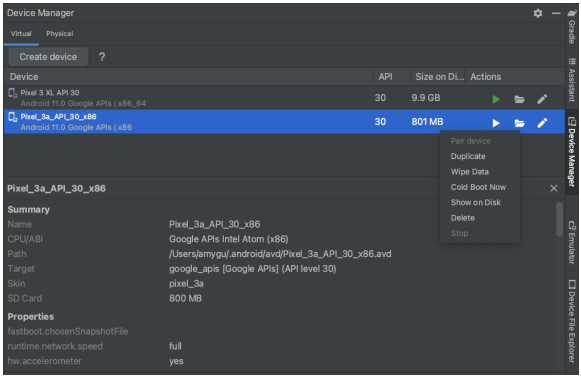
- Se você instalou o `Android Studio`, provavelmente já existe uma opção de emulador habilitada.
- Abra o `Android Studio`
  - Clique em **more actions** (mais ações).
  - Selecione **Virtual Device Manager**.

Emulador Android



Emulador Android

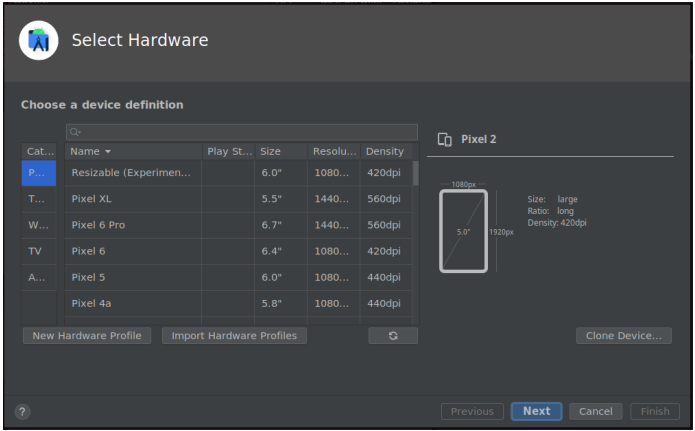
- Possivelmente o `Android Studio` terá o **Pixel 3** disponível como opção de emulador.



Emulador Android

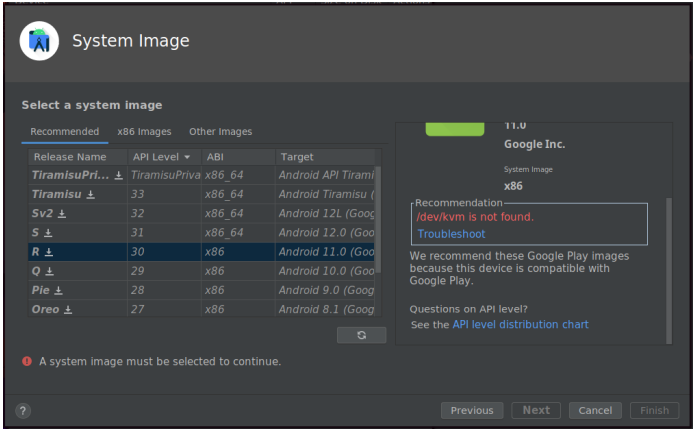
- Se não existe uma opção de emulador, você pode criar um emulador novo clicando em **Create Device**.
- Escolha a opção **Pixel 3a**.

Emulador Android



Emulador Android

- Clique no símbolo de download da *release* **R** para baixar a *system image*.



Emulador Android

- É necessário ter a **tecnologia de virtualização** da Intel habilitada para rodar o emulador.
  - Se o emulador não iniciar, esse pode ser o problema.
  - Entre na **BIOS** de seu computador e habilite a opção **Intel Virtualization Technology** (VT-x).

Emulador Android

- Com o emulador pronto, podemos começar a criar a APK.
- Siga o processo de criação de APK utilizando o aplicativo que você preferir.
  - Caso prefira, pode baixar o aplicativo disponível para essa aula no site da disciplina.
  - Quando baixar o aplicativo, dentro da pasta do projeto:
    - `flutter clean`
    - `flutter pub get`

Geracao de APK

Gerando uma APK

Geração de APKs

- **APK** é a sigla para *Android Application Pack*.
  - É como se fosse o arquivo executável (**.exe**) do `Android`.
- O `Flutter` facilita bastante a geração e instalação de APKs.

Assinando o App

- Para publicar seu aplicativo na *Play Store*, você precisa dar seu aplicativo uma **assinatura digital**.
- Use a linha de comando para gerar uma assinatura digital.

Assinando o App

- No Mac/Linux, use o seguinte comando:
  - `keytool -genkey -v -keystore upload-keystore.jks -keyalg RSA -keysize 2048 -validity 10000 -alias upload`
- No Windows, use o comando:
  - `keytool -genkey -v -keystore upload-keystore.jks -storetype JKS -keyalg RSA -keysize 2048 -validity 10000 -alias upload`

Assinando o App

- Siga todos os passos, adicionando senhas e informações e será gerado um arquivo chamado **upload-keystore.jks**
- Agore crie um arquivo chamado:
  - `[pasta-projeto]/android/key.properties`
  - Ele vai conter a referência para a chave gerada.

Assinando o App

- Adicione o seguinte conteúdo nesse arquivo:

```
storePassword=<senha escolhida no passo anterior>
keyPassword=<senha escolhida no passo anterior>
keyAlias=upload
storeFile=<caminho-app>/upload-keystore.jks
```

Configurando a Assinatura no Gradle

- Configure o `gradle` para utilizar sua chave na hora de gerar a versão de distribuição (*release*) do app.
- Abra o arquivo:
  - **`[pasta-projeto]/android/app/build.gradle`**

Configurando a Assinatura no Gradle

- Adicione a **informação de armazenamento da chave** do seu arquivo `key.properties`.
  - Antes do bloco android, adicione:

```
def keystoreProperties = new Properties()
def keystorePropertiesFile = rootProject.file('key.properties')
if (keystorePropertiesFile.exists()) {
    keystoreProperties.load(new FileInputStream(keystorePropertiesFile))
}
```

Configurando a Assinatura no Gradle

- Agora procure o **buildTypes** no mesmo arquivo:

```
1     buildTypes {
2         release {
3             // TODO: Add your own signing config for the release build
4             // Signing with the debug keys for now,
5             // so `flutter run --release` works.
6             signingConfig signingConfigs.debug
7         }
8     }
```

Configurando a Assinatura no Gradle

- Substitua pelo seguinte trecho:

```
signingConfigs {
    release {
        keyAlias keystoreProperties['keyAlias']
        keyPassword keystoreProperties['keyPassword']
        storeFile keystoreProperties['storeFile'] ? file(keystoreProperties['storeFile']) : null
        storePassword keystoreProperties['storePassword']
    }
}
buildTypes {
    release {
        signingConfig signingConfigs.release
    }
}
```

Revisando o Manifest

- Se seu aplicativo utiliza a Internet em algum momento, é necessário adicionar essa permissão no **AndroidManifest.xml**
- Abra o arquivo:
  - **<aplicativo>/android/app/src/main/AndroidManifest.xml**
- Antes de **application**, adicione a seguinte linha:
  - `<uses-permission android:name="android.permission.INTERNET"/>`

Revisando o Manifest

- Atualize o `android:label` dentro da tag `application`
  - Escolha o nome final do app.
  - Versão de distribuição.

Gerando a APK

- Com seu aplicativo assinado, é hora de gerar a APK do projeto.
- Da linha de comando do seu projeto:
  - `flutter build apk --split-per-abi`
- Serão gerados múltiplas versões diferentes para diferentes processadores.

Rodando seu projeto

- Conecte um dispositivo `Android` por USB no seu computador.
  - Ative o mode desenvolvedor.
  - E a depuração por USB.
- Se preferir, pode rodar seu aplicativo diretamente no emulador.
  - Para isso, tenha um emulador conectado e funcionando.

Rodando seu Projeto

- Da linha de comando do seu projeto:
  - `flutter install`
- Seu projeto será instalado no dispositivo conectado.
- Navegue pelo dispositivo e abra seu aplicativo.

Geracao de APK

Alguns detalhes adicionais

Mudando o Ícone de sua Aplicação

- Por padrão, seu app utilizará o símbolo do `Flutter`.
- É possível mudar isso utilizando o pacote `flutter_launcher_icons`.

Mudando o Ícone de sua Aplicação

- Na raiz do seu projeto:
  - `flutter pub add flutter_launcher_icons`
- Supondo que seu arquivo de ícone se chama `icone_pizza.png`
  - adicione o código no **pubspec.yaml**.

Mudando o Ícone de sua Aplicação

```
flutter_icons:
  android: "launcher_icon"
  ios: true
  image_path: "images/icone_pizza.png"
  min_sdk_android: 21 # android min sdk min:16, default 21
  web:
    generate: true
    image_path: "images/icone_pizza.png"
    background_color: "#FFFFFF" # fundo branco
    theme_color: "#FFFFFF"
  windows:
    generate: true
    image_path: "images/icone_pizza.png"
    icon_size: 48 # min:48, max:256, default: 48
  macos:
    generate: true
    image_path: "images/icone_pizza.png"
...
flutter:
```

Mudando o Ícone de sua Aplicação

- Por fim, mude os ícones de sua aplicação rodando os comandos:
  - `flutter pub get`
  - `flutter pub run flutter_launcher_icons`
- Gere novamente a APK e instale para ver as diferenças.

Publicando

- Para publicar seu aplicativo, siga o procedimento em:
  - https://developer.android.com/distribute/best-practices/launch

Referências


Build and release an android app.
`https://docs.flutter.dev/deployment/android`.
Acessado 16/11/2022.


Simone Alessandria and Brian Kayfitz.
Flutter Cookbook: Over 100 proven techniques and solutions for app development with Flutter 2.2 and Dart.
Packt Publishing Ltd, 2021.

Conteúdo

https://gustavotcustodio.github.io/dispomoveis.html

Obrigado

gustavo.custodio@ulife.com.br