Desenvolvimento de aplicações para Android

Criação de projetos Android

Ciclo de vida de um atividade

Objecto Application

Projeto Android

- Um projeto Android é constituído por:
  - Ficheiro de manifesto
    - Ficheiro XML com as configurações gerais da aplicação, registo das necessidades (permissões)
  - Número variável de componentes pertencentes aos seguintes 4 tipos possíveis:
    - `Activity`
    - `Service`
    - `BroadcastReceiver`
    - `ContentProvider`
  - Recursos (inc. imagens, *layouts* dos ecrãs, ...)
  - Ficheiros com configurações de compilação e dependências (ex: `build.gradle`)

Criação de projeto

- Executar Android Studio
- Escolher "**New Project**" para iniciar o *wizard* de criação do projeto
  - Escolher o tipo de aplicação
    - Plataforma: **Phone and Tablet**, Wear OS, TV, Automative)
    - **No Activity**
  - "**Next**"
  - Indicar o **nome da aplicação** (ex: Aula2)
  - Indicar o "**Package name**"
    - O *package name* é constituído pelo domínio da empresa, por ordem inversa, seguido pelo nome da aplicação (sem espaços), em letras minúsculas
    - O *package name* deve ser único
      - Vai ser usado para identificar a aplicação na *Google Play Store*
    - **Sugestão: pt.isec.a<numero_aluno>.<appname>**
  - Escolher a linguagem: **Kotlin**
  - Versão Android – API Level mínimo
    - Sugestão: **API 22** (ou outra experimentar opção "*Help me choose*")

Projeto

- Constituído por…
  - *Manifests*
    - `AndroidManifest.xml`
    - Definições gerais da aplicação
  - *Java/Kotlin*
    - Ficheiros *Java* ou *Kotlin* (.kt) organizados em *packages*
  - *Resources* (res)
    - *drawables, layouts, menus, values, mipmap, …*
  - *Gradle Scripts*
    - Configurações de compilação
      - Incluindo bibliotecas adicionais
- Durante o desenvolvimento do projeto poderão ser adicionados outros componentes

Testar o projeto criado

- Criar emulador
  - caso não tenha sido já criado

- Experimentar a executar no emulador
  - Opções de *Build* (*build, clean, rebuild*, ...)
  - Opções de execução (*run, debug*, ...)

  - Verificar que não mostra qualquer aplicação (ou, dá um erro a dizer que não tem qualquer atividade...dependendo da versão)
  - Abrir os *Settings* e encontrar entre as aplicações instaladas a aplicação criada

Compilação e execução

- Depois de um projeto ser compilado é gerado um "*package*" com a aplicação
  - O nome do pacote corresponderá ao definido para a aplicação
  - Extensão `.apk`
    - Corresponde a um ficheiro do tipo *jar*
- O ficheiro *apk* deverá ser transferido e instalado no dispositivo
- Ao escolher a opção de execução do *Android Studio*, o *apk* em causa é enviado para o dispositivo ligado ao computador ou é lançado um emulador para teste da aplicação (escolhido através de uma janela adicional)

Criação manual de atividade

- Criar na pasta de *resources* uma nova pasta de nome `layout`
- Criar um ficheiro *xml* de *layout* (`my_activity.xml`) dentro dessa pasta:

```
<?xml version="1.0" encoding="utf-8"?>
<FrameLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:background="#ffb040"
    android:layout_width="match_parent"
    android:layout_height="match_parent">
</FrameLayout>
```

- As duas ações anteriores podem ser realizadas de uma única vez com a opção de criar um *Resource File* do tipo *Layout*, indicando `FrameLayout` como *Root element*

Criação manual de atividade

- Criar uma classe *Kotlin*, `MyActivity`, derivada da classe `android.app.Activity`, no *package* já disponível
  - Processar o evento `onCreate`
    - No contexto da classe começar a escrever "`onCr`" e aceitar a sugestão da função `onCreate` que possui um parâmetro.
  - Adicionar uma linha com

    `setContentView(R.layout.my_activity)`

```
class MyActivity : Activity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.my_activity)
    }
}
```

Criação manual de atividade

- Registar a atividade no ficheiro de manifesto, no contexto da estrutura `application`

```
<activity android:exported="true" android:name=".MyActivity">
</activity>
```

- Adicionar `intent-filter` com a ação `MAIN` e categoria `LAUNCHER`

```
<intent-filter>
  <action android:name="android.intent.action.MAIN" />
  <category android:name="android.intent.category.LAUNCHER" />
</intent-filter>
```

Debug

- É possível efectuar o *debug* como em aplicações regulares *Java*
  - Inserir *breakpoints*, executar passo-a-passo, *watchs*,…
- Pode-se recorrer ao sistema de *logs* existente no *Android* para auxiliar nas tarefas de *debug*
  - Usar os métodos estáticos da classe `android.util.Log` (`Log.d`, `Log.i`, `Log.w`, …)
    - **Inserir no método `onCreate` uma linha de log**
      - `Log.i("AMovApp","onCreate: ")`
  - Consulta dos *logs* através do *Logcat*
    - Disponível no *Android Studio*

Debug

- Existem várias ferramentas disponíveis no *Android Studio* que nos podem auxiliar nas tarefas de construção e verificação das aplicações
  - Integradas no próprio ambiente
  - Executadas a partir da linha de comando
    - `adb`
      - Disponível na pasta `platform-tools`
      - Permite consultar *logs*, *upload* e *download* de ficheiros, executar uma *shell* no dispositivo, …
        - Exs:
          - `adb logcat`
          - `adb shell`

Ciclo de vida de uma atividade

- Fazer o processamento de eventos que ocorrem no ciclo de vida de uma atividade
  - Processar os seguintes métodos e gerar uma mensagem de *log* apropriada em cada um deles
    - `onCreate`
    - `onStart`
    - `onRestart`
    - `onResume`
    - `onPause`
    - `onStop`
    - `onDestroy`
    - `onSaveInstanceState`
    - `onRestoreInstanceState`

Inserir em cada método:

```
Log.i("AMovApp", "<nome do método>");
```

Ciclo de vida de uma atividade

- Com a ajuda do *logcat*, analisar a ordem das mensagens …
  - Iniciar a aplicação
  - Finalizar a aplicação
  - Reiniciar a aplicação
  - Carregar no botão *home*
  - Rodar o ecrã com a aplicação ativa (Ctrl+F11/F12)
  - Outras situações (por ex., efetuar uma chamada atendendo e recusando, *Google Assistant*, ...)

Application

- Criar um objeto do tipo `android.app.Application`
  - Dar nome `MyApp`
- Configurar o objeto no ficheiro de manifesto
  - Adicionar atributo "`name`" à *tag application*, indicando como valor o nome classe `Application` criada
- Colocar uma linha de *log* no método `onCreate`
  - Verificar a existência de outros métodos "*onXXX*"

Application

- Sugestão:
  - Adicionar um contador inteiro no objeto `Application`
    - Implementar com auxílio de uma propriedade *Kotlin* que incremente o valor automaticamente

```
private var _my_value = 0;
val my_value : Int
      get() = ++_my_value
```

  - Usar (e incrementar) esse contador em todos as linhas de *log* definidas anteriormente
    - Usar a propriedade `application` (Java:`getApplication()`) para aceder ao objeto `Application`
    - Utilizar uma variável *lazy* na classe `MyActivity` para aceder e fazer o *cast* do objeto `Application` para `MyApp`

```
val app : MyApp by lazy { application as MyApp }
```

Exercício com um *object* (Kotlin)

- Criar um contador similar ao colocado na classe `MyApp`, mas implementado através de um *Object kotlin*
  - Dar nome: `MyObject`

- Verificar as mensagens geradas no contexto do ciclo de vida da aplicação