# Resumo Programação de Dispositivos Móveis

## *Definição de Dispositivo Móvel e outros conceitos*

De um modo geral, um dispositivo móvel possui as seguintes características:

- capacidade de **processar e armazenar dados**;
- possui dispositivos de **entrada e saída de dados**;
- passíveis de ser **transportados com facilidade**;
- **dimensões e peso reduzidos**;
- **comunicação sem fios**.

Como exemplos comuns temos *smartphones*, telemóveis, consolas portáteis, *ultrabooks*, *notebooks* e *netbooks*.

As aplicações móveis **raramente serão executadas** (e testadas) **no mesmo ambiente em que foram criadas**. Devido a isto, as ferramentas usadas no seu desenvolvimento possuem **formas de virtualização desses ambientes de execução**.

Devido à necessidade de congregação de tantos recursos, usam-se **IDE**'s (***I**ntegrated **D**evelopment **E**nvironment*), tal como o *Android Studio* ou o *Xcode* (para *iOS*).

Para cada plataforma são precisos recursos e ferramentas adequados a si. A este conjunto de desenvolvimento dá-se o nome de ***SDK*** (***S**oftware **D**evelopment **K**it*).

Uma ***API*** (***A**pplication **P**rogramming **I**nterface*) é um conjunto de métodos e/ou funções que um software possui e que permitem o seu uso consistente, mas do qual o utilizador final não sabe nada!

Por curiosidade, o *Android* é baseado em ***Linux***, enquanto que o *iOS* é baseado em ***UNIX***.

## *Android Studio*

No caso do desenvolvimento *Android*, o ***layout* e nome** das aplicações que criamos (entre outros) encontram-se num ficheiro .xml.

**Extensible Markup Language -** possui um design de herança, pais e filhos, atributos e tags

Todos os documentos .xml possuem a seguinte estrutura base:

```
<?xml version="1.0" encoding="utf-8"?/>
<manifest (...)>
    <application (...)>
        <activity (...)>
            (...)
        </activity>
    </application>
</manifest>
```

**NOTA**

O elemento-raiz é sempre o **manifest** e todo o conteúdo e forma estão descritos dentro de si.

## *Desenvolvimento de Interfaces*

Uma interface de utilizador **estabelece a comunicação** entre o utilizador e o programa/máquina que está a utilizar.

1. Programas Sequenciais

Os programas baseados neste modelo **controlam o fluxo de interação com o utilizador**, ou seja, só quando os programas acabam de realizar alguma atividade é que pedem input ao utilizador.

De um modo geral:

```
while(1)
    pedir_input()
    ler_input()
    analisar_input()
    tomar_acao()
    (gerar saídas)
```

O problema deste tipo de interação é que se torna **complicado de escalar o sistema** para que aceite vários inputs diferentes do utilizador (mudanças de modo, etc...). O código pode **aumentar exponencialmente** para se adaptar a utilizadores mais exigentes.

2. Programas/Interfaces Orientados a Eventos

Alternativamente, este modelo, possibilitado pelo advento dos ecrãs/dispositivos táteis, **aguarda pelos inputs** do utilizador.

```
while(1)
    espera_por_evento()
    envia_evento_para_lista_de_eventos()
    analisar_evento_e_envia_para_programa()
    (se o programa estiver a dormir, acorda-o)
```

A rotina principal de captação de eventos é da responsabilidade do **SO**, que os coloca numa pilha (First In First Out).

## *Modelo-Visão-Controlador*

É uma arquitetura de desenvolvimento de software, dividida em 3 partes:

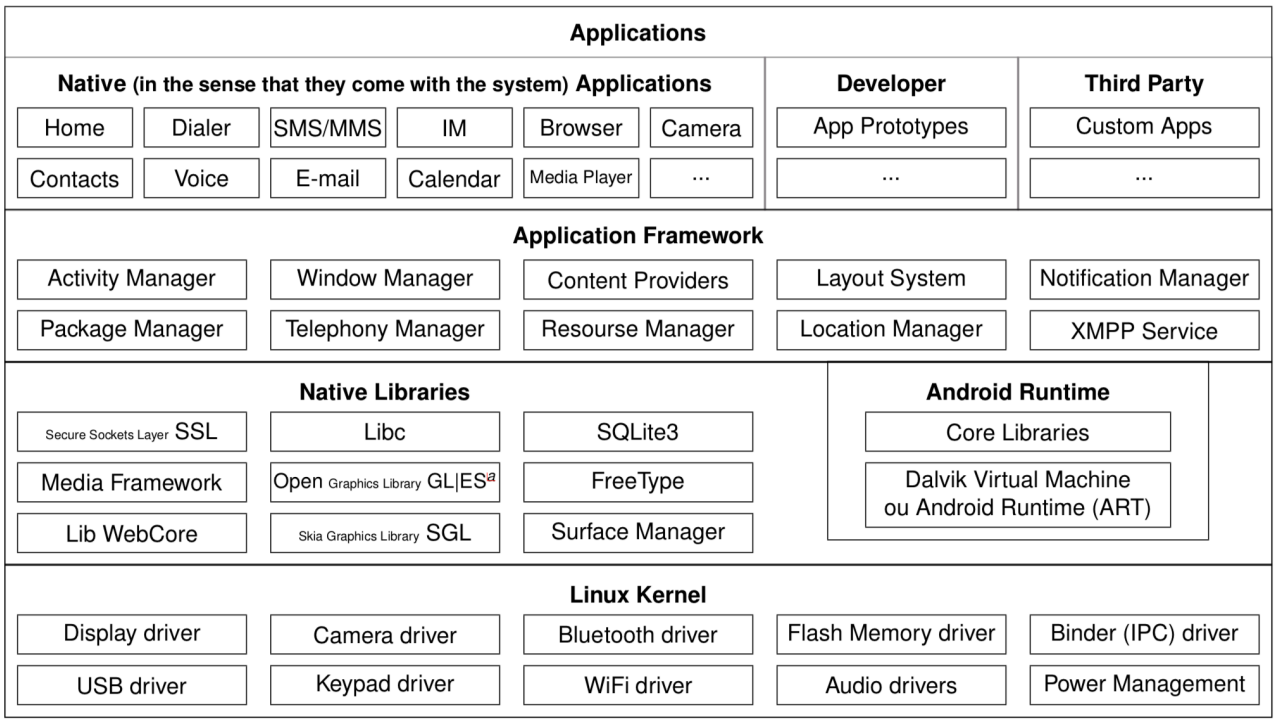
- **Modelo**: base da aplicação, nele estão definidos os métodos, funções, objetos e todo o processamento de dados;
- **Visão**: consiste numa representação dos dados da aplicação, incluindo as interfaces de utilizador;
- **Controlador**: inclui as rotinas de tratamento de eventos. Pode atualizar a visão e o modelo.

## *A plataforma Android*

Qualquer software assenta em vários pilares, que no caso do Android, são:

- **Pilha de software**: com várias camadas, permitindo o desenvolvimento de aplicações móveis;
- **SDK**;
- Extensa **documentação**.

Sendo baseado em Linux, o Android possui um kernel parecido ao das distribuições desktop, com algumas diferenças:


Applications
Native (in the sense that they come with the system) Applications
Developer
Third Party
Home
Dialer
SMS/MMS
IM
Browser
Camera
App Prototypes
Custom Apps
Contacts
Voice
E-mail
Calendar
Media Player
...
...
...
Application Framework
Activity Manager
Window Manager
Content Providers
Layout System
Notification Manager
Package Manager
Telephony Manager
Resourse Manager
Location Manager
XMPP Service
Native Libraries
Android Runtime
Secure Sockets Layer SSL
Libc
SQLite3
Core Libraries
Media Framework
Open Graphics Library GL|ES[a]
FreeType
Dalvik Virtual Machine
ou Android Runtime (ART)
Lib WebCore
Skia Graphics Library SGL
Surface Manager
Linux Kernel
Display driver
Camera driver
Bluetooth driver
Flash Memory driver
Binder (IPC) driver
USB driver
Keypad driver
WiFi driver
Audio drivers
Power Management