# Desenvolvimento para Dispositivos Móveis
# Mobile Apps Development

## Generalidades sobre Android (1)

# Android : ~kernel

- Corre em cima de um Linux Kernel modificado
- Porquê o Linux Kernel?
  - Boa gestão de memória e de processos
  - Modelo de segurança baseado em permissões
  - Modelo de drivers maduro
  - Suporte a libraries partilhadas
  - Open source
- Não é literalmente Linux
  - Não tem glibc
  - Não tem diversos utilitários Linux
- Kernel = gestão de consumo de energia + drivers para display, camera, bluetooth, memória partilhada, usb, keypad, wifi, áudio… binder = mecanismo próprio para comunicação entre processos (IPC) e RPC

# Android : ~kernel

- Modificações ao Kernel
  - Aspetos de gestão de energia
    - Limitações de baterias
    - Componentes pedem para manter-se abastecidos via um mecanismo chamado “Wake Locks”
  - Logger
  - Debugger
  - Low memory killer
  - Alarm
  - Binder
    - Apps em processos separados mas comunicação por memória partilhada
- Android kernel @ https://android.googlesource.com/kernel/common/

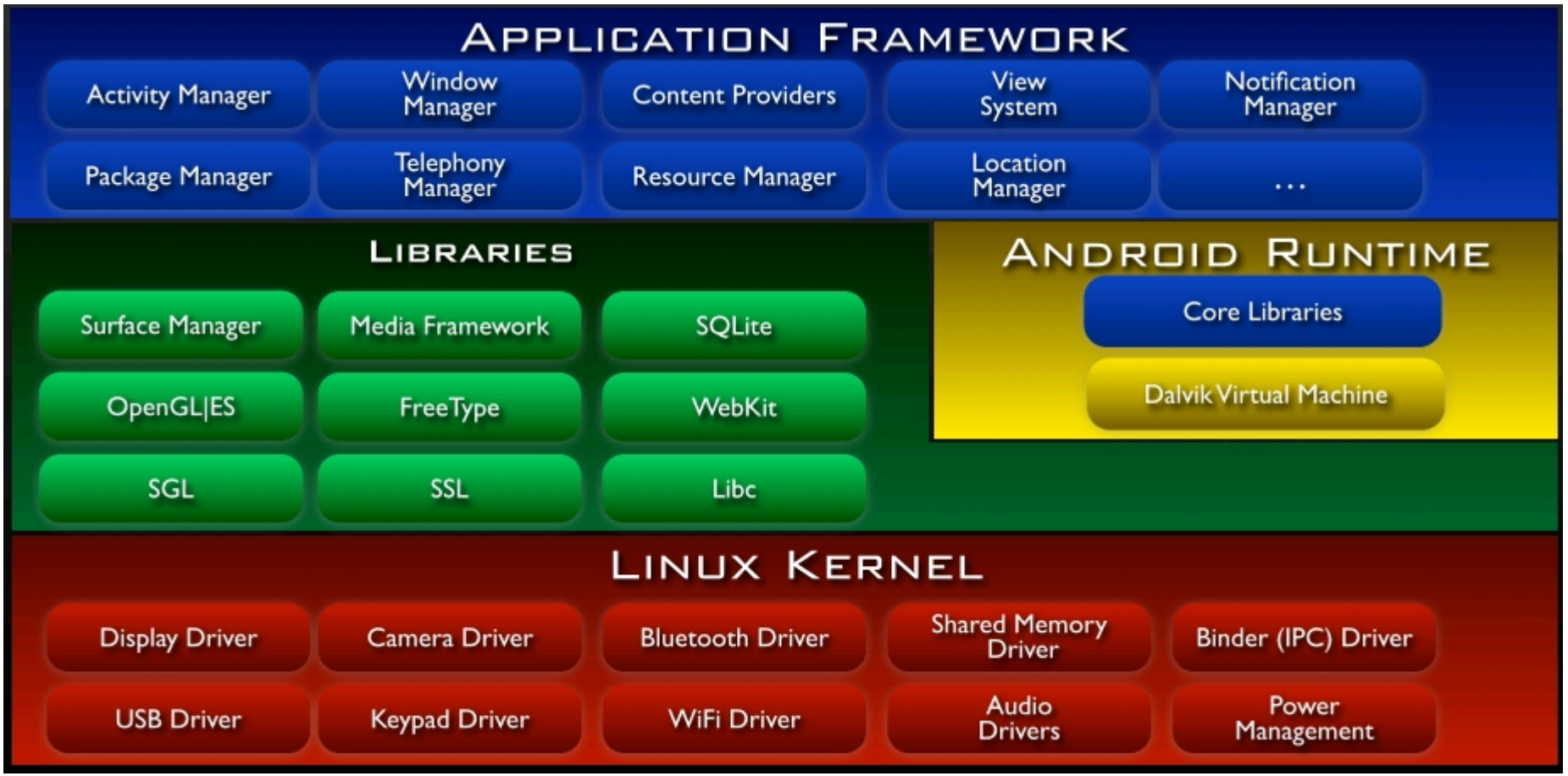
# Android : libraries nativas

- Surface manager (2D, 3D, frame buffering)
- Media framework
  - Baseada em “PacketVideo OpenCORE framework”
    - Suporta plug-ins hw/sw
- SQLite (suporta a generalidade dos dados do sistema)
- OpenGL
- FreeType
- WebKit (web browser engine) @ https://webkit.org
- SGL
- SSL
- LibC
  - Bionic – LibC específica para Android que foge à licença GPL… (usa BSD), implementa acesso a serviços do sistema e oferece ferramentas para logging

# Android : Dalvik + Art VMs

- Dalvik VM (até Android KitKat)
  - Corre .dex e Dalvik bytecode
  - Java .class e .jar são convertidas em .dex
- Android Runtime (ART, desde Android 5)
  - também corre .dex

# Android : Application framework



# Termos : activity(ies)

- Activities – analogia mais simples: como que “as páginas da app”. Uma activity contém a sua interface com o utilizador e o código que faz essa interface responder
- Em termos de código, todas as activities deverão ser classes derivadas da classe “Activity”
- Quase todas as activities têm interface full screen, mas via “activity group”, uma activity pode parecer embebida noutra

# Termos: activity + activity stack

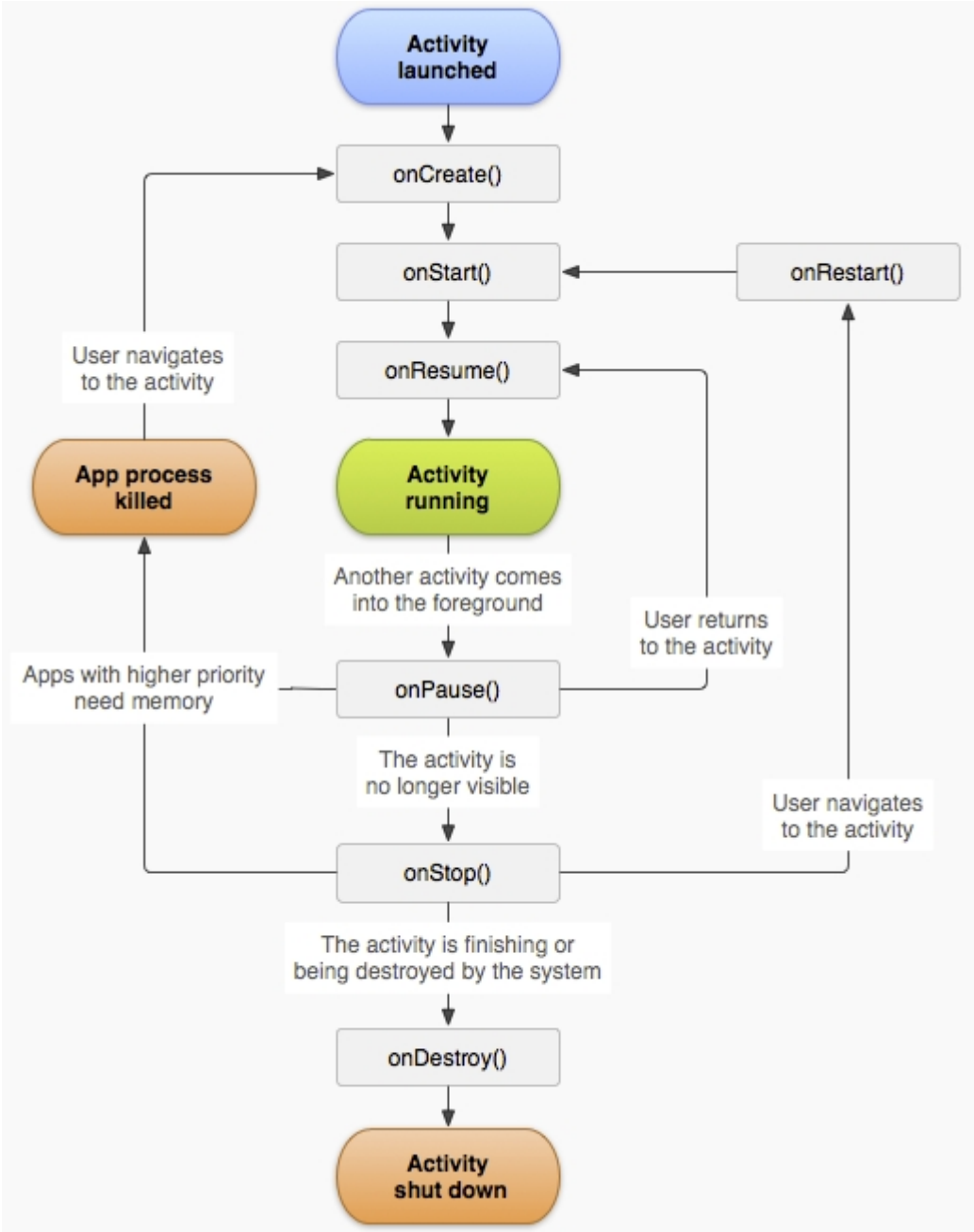
- Activities:  os 2 métodos mais importantes
  - onCreate
    - Chamado quando a activity é inicializada:
      - Aqui deve escolher-se um recurso de layout como “entry point”
  - onPause
    - Chamado quando o utilizador deixa a activity
      - Aqui deve memorizar-se o “estado” da activity, para eventualmente retomá-la
- Activity stack
  - Quando uma activity é criada vai para o topo da stack
  - A activity que funcionava antes passa para #2 da stack

# Activity – estados principais

- **Active / running** – quando está em foreground, no topo da stack de activities
- **Paused** – quando perde o foco mas permanece visível (uma atividade não full screen ganhou o foco entretanto)
- **Stopped** – quando deixa de ser visível. Retém informação de estado, mas não é visível ao utilizador. Provavelmente será killed quando os recursos por ela consumidos forem necessários para algo
- **Created / resumed** – quando transita de paused or stopped para visível, implicando restauração de estado

# Activities – “ciclo de vida”

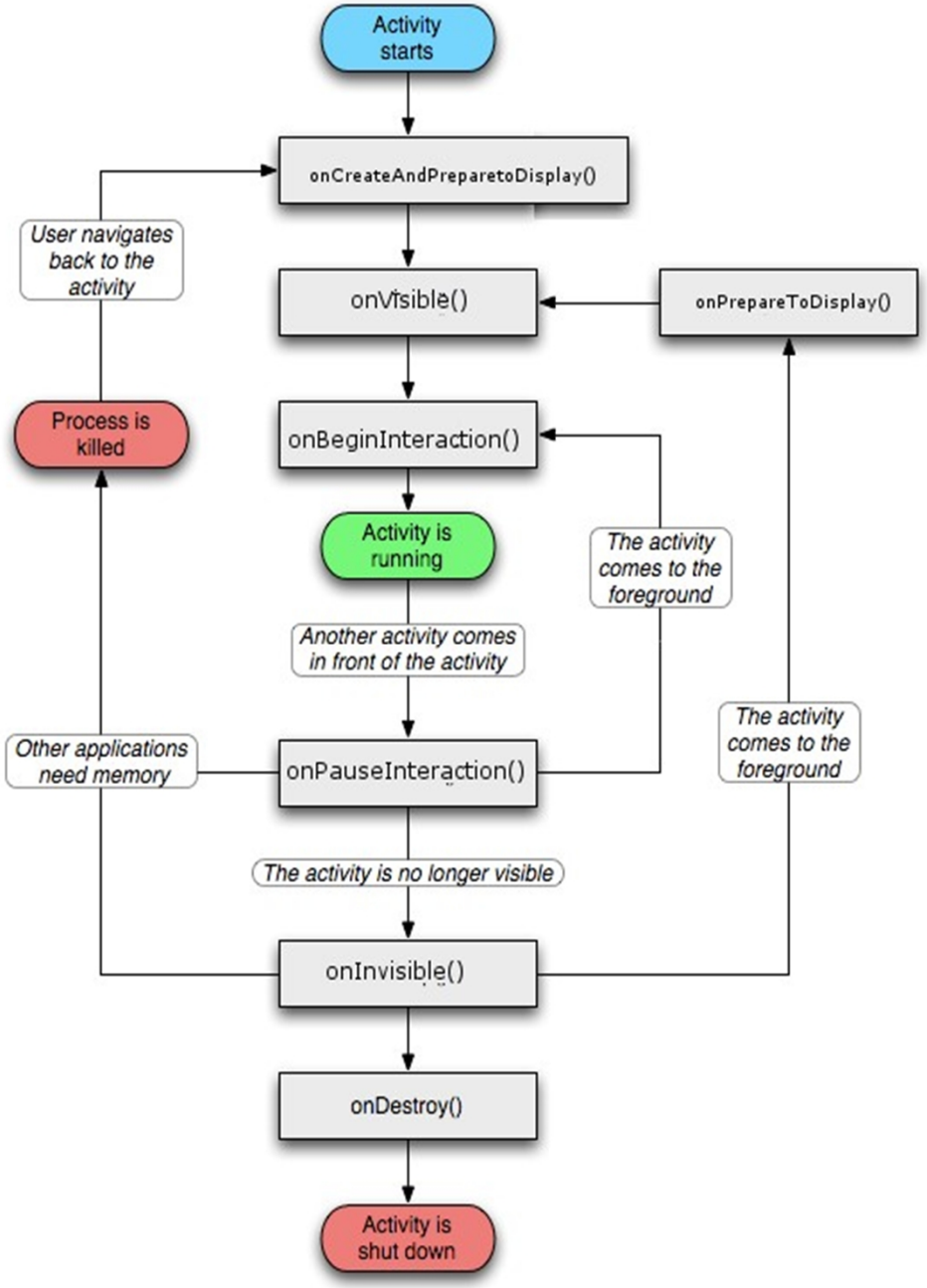
http://developer.android.com/reference/android/app/Activity.html#ProcessLifecycle




Os retângulos correspondem a “callbacks”: métodos que serão **automaticamente** chamados aquando dos eventos correspondentes e que podem/devem ser implementados para customizar a resposta da app a esses eventos

# Activities – “ciclo de vida”

http://developer.android.com/reference/android/app/Activity.html#ProcessLifecycle




Ignorando detalhes com “fragments” talvez outros nomes fossem mais inteligíveis.

http://stackoverflow.com/questions/8515936/android-activity-life-cycle-what-are-all-these-methods-for

# Activities

- Exemplo: estamos a usar um programa que permite tomar pequenos apontamentos…
- Ao começar o programa, acontecem:
  - **onCreate() → onStart() → onResume()**
- Se durante o uso acontece, por exemplo, um alarme, a activity de tomar apontamentos vai para background, sendo chamados:
  - **onPause() → onStop()**
- Se o utilizador regressar, acontecem:
  - **onRestart() → onStart() → onResume()**
- Quando o utilizador quiser sair dos apontamentos:
  - **onStop() → onDestroy()**

# Termos : intent(s)

- Intents – o sistema de mensagens em Android. Um intent tem duas partes:
  - Ação/action: o que fazer quando a mensagem é recebida?
  - Dados/data: dados complementares necessários para executar a ação
- Uma app pode
  - Fazer broadcast de um intent
    - Pedindo uma nova activity na mesma app
    - Começando uma app terceira
  - Receber um intent
  - É necessário as apps registarem-se como “intent receivers” para poderem responder ao intent
  - Se mais do que uma app souber responder, deve acontecer um “chooser”

# Recursos sem cursor

- Usam-se os dedos para interagir com recursos da interface
- O equivalente a um right-click em Windows é um “long press”, que poderá fazer aparecer um menu de opções, sensíveis ao contexto, se tal tiver sido programado

# Views

- Peças básicas de componentes da user interface em Android
- Exemplos
  - TextView
  - ImageView
  - Layout
  - Button
  - Etc
- Definidas nas packages (conjunto de classes) “android.widget” e “android.view”
- Ocupam uma área retangular
- Responsáveis por:
  - Desenhar/atualizar/drawing nessa área
  - Responder aos eventos de utilizador na área
- http://developer.android.com/reference/android/widget/package-summary.html

# Views

- Um objeto View deve ser casted para o tipo específico correto, para poder ser trabalhado
  - Exº: (TextView)v;
- LayoutParams obrigatórios
  - Layout_width
  - Layout_height
  - Esquecer um deles = crash da app
- Valores +frequentes para os LayoutParams
  - match_parent (ou fill_parent)
    - Dizer ao controlo para ocupar toda a área do seu pai/contentor
  - wrap_content
    - Dizer ao controlo para se auto-dimensionar na área disponível

# Alguns layouts comuns

- ConstraintLayout
- LinearLayout
  - Filhos/componentes arrumados numa só linha ou coluna
- RelativeLayout
  - Filhos/componentes descritos relativamente a outros filhos ou ao pai/contentor
- GridLayout
  - Filhos/componentes organizados em matriz

# Chamadas assíncronas

- Pode usar-se a class “AsyncTask” para correr múltiplas operações em simultâneo, sem que o programador tenha que gerir a thread correspondente manualmente
- Mais simples, mais limitado, usar “loaders”
- A utilização de uma solução assíncrona poderá permitir evitar situações “ANR” – app not responding
- Boas práticas para apps fluídas:
  - http://developer.android.com/guide/practices/design/responsiveness.html

# Fragmentos (Android 3+)

- Como que sub-activities
  - Parte de uma activity
  - Com ciclo de vida próprio
- Com Android 3.0/"JellyBean", passaram a ser suportados "tablet devices", com maior écran do que os "phone devices" típicos
- Não é trivial mostrar duas activities em Android
- Uma solução é "fragmentar" uma só activity

# Loaders (Android 3+)

- Boa prática: cautela com I/O na thread principal, que faz interface com o utilizador
- O I/O tem potencial para causar ANR
- Um loader é uma solução para fazer I/O em background, em thread própria

# Fragments, Loaders, etc em Android <3 ?

- Usar a “Android Support Library” para retro-compatibilidade
- Ver
  - http://developer.android.com/tools/support-library/index.html

# Action bar

- Recurso de interface normalmente sempre visível
- Elementos da “action bar”, da esquerda para a direita:
  - Logo da app (ou “up button”)
    - Faz o user regressar à activity anterior da app corrente
      - Exemplo: web page anterior quando se está num web browser
    - Diferente do “back button”
      - Faz o user regressar à activity anterior, independentemente da app
        - » Exemplo: app anterior, quando se está num web browser
  - Nome da page @ app OU tabs intra app
  - Buttons para actions diversas
  - http://developer.android.com/guide/topics/ui/actionbar.html

# Holo?

- Themes em Android 3+ que garantem consistência de interface entre dispositivos
  - Holo Dark
  - Holo Light

# HW

- Acelerómetro (accelerometer) – para medir aceleração; por exemplo, indica se o dispositivo está em movimento
- Bluetooth – para comunicações de rádio entre dispositivos compatíveis
- Bússola (compass) – indica a direção do dispositivo
- “Camera” – fotos e vídeo
- Sensor de proximidade – para indicar se o dispositivo está de écran para cima ou para baixo, por exemplo
- GPS – para localização geográfica

# Segurança

- Apps necessitam de pedir explicitamente certas permissões, por exemplo
  - para aceder à Internet
  - para ler/escrever no cartão SD

# Referências

- http://developer.android.com