Faculdade de Engenharia da Universidade do Porto

Agente de Contexto para Dispositivos Móveis

Pedro Manuel Neves Garrido

Dissertação realizada no âmbito do
Mestrado Integrado em Engenharia Electrotécnica e de Computadores
Major Telecomunicações

Orientador: Prof. Dra. Maria Teresa Andrade

Julho de 2011

Resumo

As soluções actuais de Mobile TV efectuam a transmissão de vídeo sem olhar ao contexto em que se encontra o utilizador final. Esse contexto pode ser muito diversificado nomeadamente no desempenho do equipamento, nas características do ecrã, largura de faixa disponível, perfil do utilizador entre outras características que podem influenciar substancialmente a qualidade de experiência do utilizador.

Com esta dissertação pretendeu-se desenvolver um Agente de Contexto que informe o servidor de Streaming de vídeo para que este faça a adaptação em tempo real da transmissão de vídeo às condições do contexto do utilizador final. Optou-se por concentrar o desenvolvimento para a plataforma Android.

Pretende-se que o Agente de contexto gere informação em tempo real do estado em que se encontra o utilizador final a vários níveis como, Contexto computacional (hardware e software), Contexto do utilizador (perfil do utilizador), Contexto físico (ambiente) e Contexto temporal.

Pretende-se que o Agente de Contexto gere informação normalizada e após estudo das normas existentes optou-se pela norma UAProf que adopta CC/PP e fornece um vocabulário concreto visando principalmente dispositivos móveis.

A plataforma Android, sendo uma novidade para mim, ocupou uma boa parte do tempo disponível, optando-se por isso por criar alguma documentação que poderá vir a ser útil para trabalho futuro nesta plataforma.

Relativamente aos resultados, desenvolveram-se alguns módulos que recolhem informação do sistema como por exemplo informação do software e hardware como o CPU utilizado, características do ecrã, ligações de rede disponíveis e seu desempenho. Desenvolveu-se e documentou-se ainda algum *software* para visualização de vídeo por streaming com vista a futuros testes da eficiência do Agente de contexto.

Não se chegou pois a criar propriamente o Agente de contexto porque não houve possibilidade de o fazer em tempo útil pelo facto de ter sido necessário dispensar um tempo maior do que o previsto para a familiarização com a plataforma Android. No entanto foi criada uma boa base para trabalho futuro.

Glossário

AAF - Advanced Authoring Format

API - Application Programming Interface

AVD - Android Virtual Devices

CMU - Context toolkit Architecture

CoBrA - Context Broker Architecture

DCO - Delivery Context Ontology

dpi – dots per inch

GNSS - Global Navigation Satellite Systems

CC/PP – Composite Capabilities/Preference Profiles

hdpi – high dots per inch

HTTP - Hypertext Transfer Protocol

IDE - Integrated Development Environment

ldpi – low dots per inch

mdpi – medium dots per inch

MPEG - Moving Picture Experts Group

MXF - Material Exchange Format

RDF - Resource Description Framework

RIM - Research In Motion Limited

iOS - iPhone Operating System

OS – Operating System

PDA - Personal Digital Assistants

RTSP - Real Time Streaming Protocol

SDK - Software Development Kit

SOCAM - Service Oriented Context aware Middleware

TVA - TV-Anytime

UaProf - User Agent Profile

UED - Usage Environment Description

UMID - Unique Material Identifier

WAP - Wireless Application Protocol

WOL - Web Ontology Language

xhdpi – extra high dots per inch

JDK – Java Development Kit

Agradecimentos

Agradeço à minha esposa, Aida e aos meus filhos, João Pedro e Inês por me terem acompanhado e compreendido na realização desta tarefa.

Gostaria também de agradecer de uma forma especial à minha orientadora, Professora Doutora Maria Teresa Andrade por toda a disponibilidade e apoio na realização da dissertação.

Conteúdo

1. INTRODUÇÃO 1

1.1 ENQUADRAMENTO 1
1.2 MOTIVAÇÃO 3
1.3 OBJECTIVOS DA DISSERTAÇÃO 3
1.4 COMPOSIÇÃO DO SISTEMA 4
1.5 ESTRUTURA DO DOCUMENTO 5

2. ESTADO DA ARTE 6

2.1 METADADOS 6
2.2 *CONTEXT-AWARENESS* 8
2.3 CONCLUSÕES 15

3. TECNOLOGIAS 17

3.1 ANDROID 17
3.2 JAVA 18
3.3 ECLIPSE 19
3.4 CONCLUSÃO 19

4. O ANDROID 21

4.1 DESENVOLVIMENTO DE APLICAÇÕES ANDROID 21
4.2 INSTALAÇÃO 21
4.3 EMULAÇÃO - ANDROID VIRTUAL DEVICES - AVD 24
4.4 CRIAÇÃO DE UM NOVO PROJECTO 27
4.5 SUPORTE 29
4.6 PROJECTOS ANDROID 29
4.7 REQUISITOS DE SISTEMA 30
4.7.1 SISTEMAS OPERATIVOS SUPORTADOS 30
4.7.2 REQUISITOS DE HARDWARE 31
4.8 ANDROIDMANIFEST 31
4.8.1 ESTRUTURA DO FICHEIRO ANDROIDMANIFEST 31
4.8.2 PERMISSÕES 33
4.9 NÍVEIS DE API NO ANDROID 34
4.9.1 NÍVEIS DE API 34
4.9.2 UTILIZAÇÃO DE NÍVEL DE API DE ANDROID 35
4.9.3 COMPATIBILIDADE DE APLICAÇÕES PARA A FRENTE 35
4.9.4 UTILIZAÇÃO ACTUAL DAS VERSÕES DA PLATAFORMA 36
4.10 ANDROID VIRTUAL DEVICES - AVD 37
4.10.1 CRIAR UM AVD 38
4.10.2 INICIAR E PARAR O EMULADOR 40
4.10.3 CONTROLAR O EMULADOR 40
4.10.4 OPÇÕES DE INICIALIZAÇÃO DO EMULADOR 41
4.10.5 EMULAÇÃO DE GPS 44
4.10.6 EMULAÇÃO DE VELOCIDADES DE REDE 45
4.10.7 LIMITAÇÕES DO EMULADOR 45
4.11 SUPORTE DE MÚLTIPLOS ECRÃS 46
4.11.1 VISÃO GERAL DOS ECRÃS 46
4.11.2 TERMOS E CONCEITOS DOS ECRÃS 46
4.11.3 GAMA DE ECRÃS SUPORTADOS 47
4.11.4 INDEPENDÊNCIA DA DENSIDADE 48
4.11.5 SUPORTAR PARA MÚLTIPLOS ECRÃS 49
4.11.6 QUALIFICADORES PARA DIFERENTES ECRÃS 50
4.11.7 TESTE DA APLICAÇÃO EM MÚLTIPLOS ECRÃS 51

4.11.8 Estado Actual de Tamanhos e Densidades de Ecrãs 53
4.12 Áudio e Vídeo 53
4.12.1 Reprodução a Partir de um Recurso Local 54
4.12.2 Reprodução a Partir de um Ficheiro ou Stream 54
4.12.3 Protocolos de Rede 54
4.12.4 Formatos de Multimédia 55
4.12.5 Codificações de Vídeo 56
4.13 Conclusões 57
5. ARQUITECTURA E IMPLEMENTAÇÃO 58
5.1 Cenários 58
5.2 Informação Contextual 59
5.3 Implementação de Aplicações de Testes 60
5.3.1 Contexto do Hardware 61
5.3.2 Contexto - diversos 62
5.3.3 Contexto de Resolução e Densidade do Ecrã 63
5.3.4 Slideshow para Diferentes Resoluções de Densidades de Ecrã 64
5.3.5 VideoPlayer 65
5.3.6 Estado das Ligações GSM e Wifi 66
5.3.7 Contexto do GSM e Wifi 67
5.3.8 ConnectivityDemo 68
5.4 Conclusões 68
6. CONCLUSÕES E TRABALHO FUTURO 69
6.1 Conclusões 69
6.2 Trabalho Futuro 69
BIBLIOGRAFIA 70
REFERÊNCIAS 72
ANEXO A – CLASSES DE API ANDROID USADAS 73
ANEXO B – LISTAGENS DOS MÓDULOS 86

Índice de Figuras e Tabelas

Figura 1. Gráfico de mercado de plataformas para smartphones 2
Figura 2. Gráfico de estimativa de mercado de plataformas para tablets 2
Figura 3. Diagrama do sistema a desenvolver 4
Figura 4. Normas de Metadados [1] 7
Figura 5. Comparação de ecrã e de memória em diferentes tipos de dispositivos [3] 9
Figura 6. Listing User Agent Profile with two components [3] 11
Figura 7. Listing Simple UED example [3] 12
Figura 8. Arquitectura de Context toolkit Architecture (CMU) [4] 13
Figura 9. Arquitectura CoBrA [4] 13
Figura 10. Arquitectura CMF [4] 14
Figura 11. Arquitectura SOCAM [4] 15
Tabela 1. Comparação de Frameworks [4] 15
Figura 12. Mercado de Dispositivos móveis [5] 17
Figura 13. Arquitectura da plataforma Android [3] 18
Figura 14. Janela de instalação do Eclipse 22
Figura 15. Janela de preferencias do Eclipse 23
Figura 16. Janela no Eclipse de Android SDK and AVD Manager de Packages disponiveis 23
Figura 17. Janela no Eclipse de escolha de instalação de Packages 24
Figura 18. Janela no Eclipse de Android SDK and AVD Manager de Virtual Devices (AVD) 24
Figura 19. Janela no Eclipse de Create new AVD 25
Figura 20. Janela no Eclipse de Android SDK and AVD Manager com Virtual Devices (AVD) criado 25
Figura 21. Janela no Eclipse de Opções para executar o AVD 26
Figura 22. Janela do AVD 26
Figura 23. Janela no Eclipse para criar um novo projecto Android 27
Figura 24. Janela (parcial) no Eclipse da árvore de um projecto Android 28
Tabela 2. Memória típica ocupada pelos diversos components de Android [8] 31
Figura 25. Estrutura do ficheiro AndroidManifest.xml [8] 32
Tabela 3. Tabela (parcial) de permissões usadas no ficheiro AndroidManifest.xml [8] 33
Tabela 4. Níveis de API em Android [8] 34
Figura 26. Gráfico e tabela de níveis de API dos dispositivos móveis no acesso ao Android Market [8] 36
Figura 27. Gráfico de níveis de API dos dispositivos móveis no acesso ao Android Market em 6 meses [8] 37
Figura 28. Janela no Eclipse de Android SDK and AVD Manager 39
Figura 29. Janela no Eclipse para criar novo Android Virtual Devices (AVD) 39
Figura 30. Tabela de teclas para controlar o AVD [8] 41
Figura 31. Tabela das opções de arranque do AVD [8] 44
Figura 32. Tabela de comandos para a emulação de GPS [8] 44
Figura 33. Tabela de comandos para a emulação de direntes velocidades de rede [8] 45
Figura 34. Diagrama de tamanhos e densidades de ecrãs [8] 48
Figura 35. Imagem de aplicação em ecrãs de diferentes densidades sem independência da densidade [8] 48
Figura 36. Imagem de aplicação em ecrãs de diferentes densidades com independência da densidade [8] 49
Figura 37. Tabela de qualificadores de ecrãs [8] 51
Figura 38. Janela em Eclipse com diferentes AVD para diferentes ecrãs 51
Tabela 5. Tabela de comparação de ecrãs ao nível da densidade e tamanho [8] 51
Figura 39. Janela em Eclipse de opções de densidade e tamanho para o AVD 52
Figura 40. Gráfico de ecrãs dos dispositivos móveis no acesso ao Android Market [8] 53
Figura 41. Tabela de formatos multimédia utilizados no Android [8] 56
Tabela 6. Tabela de codificações de vídeo utilizados no Android [8] 56
Figura 42. Captura de ecrã do AVD da aplicação ContextBuildClass 61
Figura 43. Captura de ecrã do AVD da aplicação ContextContextClass 62
Figura 44. Captura de ecrã do AVD da aplicação ContextViewDisplay 63
Figura 45. Captura de ecrã do AVD da aplicação ContextMultiRes 64
Figura 46. Captura de ecrã do AVD da aplicação ContextVideoPlayer 65
Figura 47. Captura de ecrã do AVD da aplicação ContextConnectivityManager 66
Figura 48. Captura de ecrã do AVD da aplicação ContexWifiInfo 67
Figura 49. Captura de ecrã do AVD da aplicação ContextConnectivityDemo 68
Figura 50. Tabela de campos da Classe: android.os.Build [8] 74

Figura 51. Tabela de campos da Classe: android.content.Context [8] 76
Figura 52. Tabela de campos da Classe: android.view.Display [8] 77
Figura 53. Tabela de campos da Classe: android.media.MediaPlayer [8] 79
Figura 54. Tabela de campos da Classe: android.widget.VideoView 80
Figura 55. Tabela de campos da Classe: android.net.wifi.WifiManager [8] 83
Figura 56. Tabela de campos da Classe: android.net.ConnectivityManager [8] 84
Figura 57. Tabela de campos da Classe: android.net.NetworkInfo [8] 85

1. Introdução

1.1 Enquadramento

Os grandes avanços tecnológicos que se tem vindo a observar nos últimos anos e os preços cada vez mais interessantes, têm levado à proliferação de dispositivos portáteis com capacidades de acesso à Internet e de reprodução de conteúdos multimédia, nomeadamente TV Digital.

A par disto, observou-se um aumento exponencial de conteúdos multimédia disponíveis em rede. Estes factores levaram a uma crescente procura e consequente consumo de conteúdos multimédia nos ambientes mais variados.

Embora as características destes dispositivos tenham evoluído bastante têm sempre algumas limitações face a um PC Desktop ou Notebook.

De uma forma geral, os terminais móveis são muito heterogéneos em termos de tamanho de ecrã, resolução, capacidade de processamento, capacidade de armazenamento, *software* utilizado entre outras características.

Há ainda uma grande diversidade quer no ambiente onde o utilizador se encontra, quer nos perfis desse mesmo utilizador.

Sendo que estes dispositivos fazem o seu acesso à Internet recorrendo a redes sem fios, que se caracterizam de certo modo pelo grande grau de variabilidade da largura de faixa, dependendo do número instantâneo de utilizadores e das condições ambientais, torna-se difícil seleccionar parâmetros adequados/óptimos de codificação levando-nos à inevitável consideração de que tais parâmetros têm de ser adaptados à medida que variam as condições do contexto de consumo.

Há pois a necessidade criação de ferramentas que assegurem que um determinado conteúdo seja correctamente visualizado pelo utilizador final de uma forma automática e dependente do seu contexto.

Actualmente os dispositivos móveis podem-se dividir em dois grandes grupos os *smartphones* e *tablets*.

Nos smartphones existem três plataformas dominantes. A Google (Android) recentemente passou a ser que tem a que tem maior quota, seguida da RIM (BlackBerry) e da Apple (iOS) conforme dados recentes da www.zdnet.com como se pode ver no gráfico da figura 1.

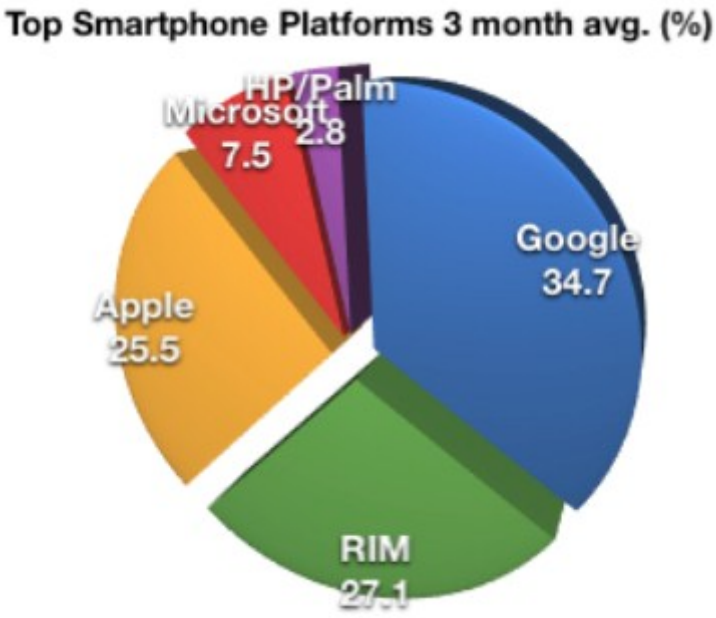


Figura 1. Gráfico de mercado de plataformas para smartphones

Relativamente ao tablets o mercado ainda é maioritariamente Apple (iPad) mas com uma forte tendência para recuperação por parte da Google (Android) como comprova o gráfico da figura 2.

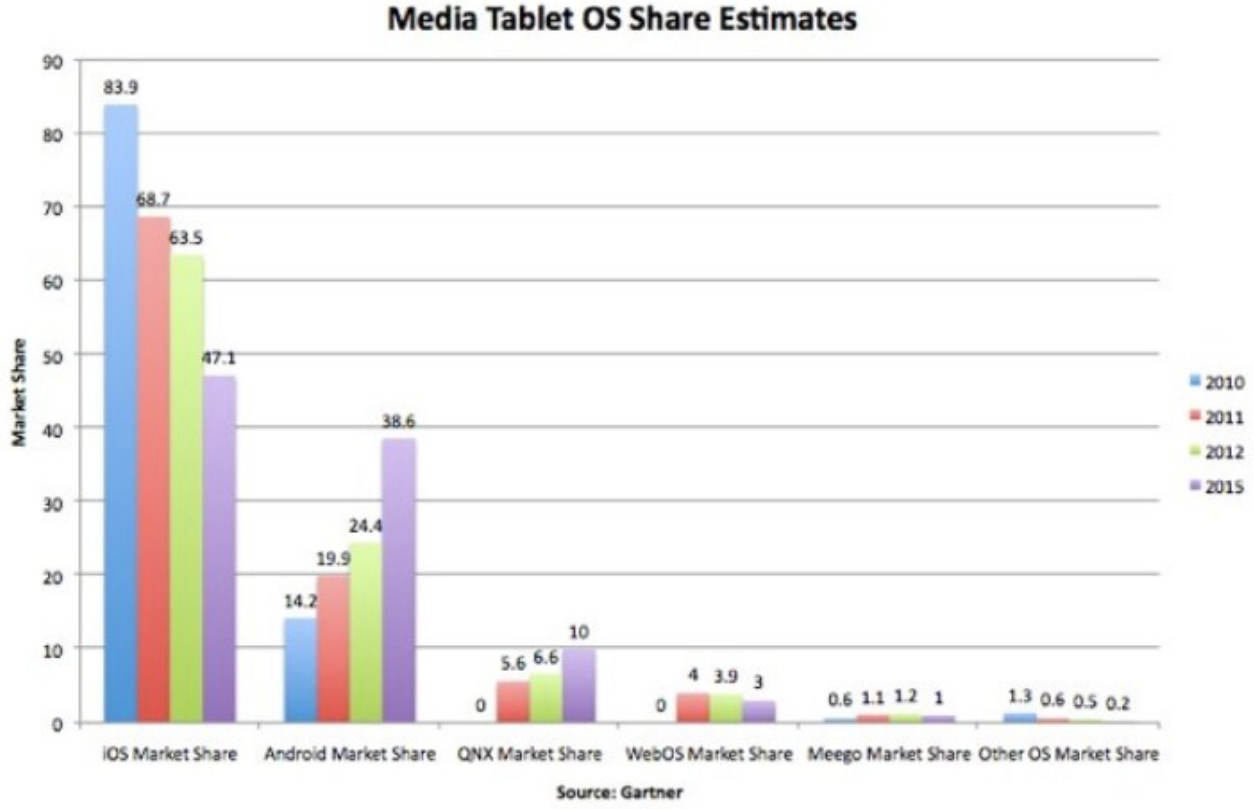


Figura 2. Gráfico de estimativa de mercado de plataformas para tablets

Devido a estes dados penso que a plataforma da Google (Android) será uma plataforma muito interessante nos próximos anos e tem ainda a mais-valia de ser uma plataforma o*pen*

source.

Por estas razões optou-se por eleger a plataforma Google Android para o desenvolvimento deste trabalho

1.2 Motivação

Durante a realização da Unidade Curricular de Televisão Digital e Novos Serviços foram abordadas bastantes questões acerca da tecnologia de Televisão Digital. Estes assuntos suscitaram muito interesse aliado ao fascínio que tenho por tecnologias móveis como são os actuais dispositivos móveis quer *smartphones* quer *tablets* nomeadamente em plataforma Android que se distingue pelo seu factor inovador de ter uma plataforma *open source*.

1.3 Objectivos da Dissertação

O objectivo deste trabalho consiste no desenvolvimento de um módulo *software* para terminais móveis (PDA e/ou telefones móveis) que execute as funções de um agente de contexto para a plataforma Google Androide.

Um agente de contexto é um módulo que consegue extrair de uma forma automática características sobre recursos *hardware* e *software* do terminal em que reside e de sinais gerados por componentes ou periféricos desse terminal.

Recursos de *hardware* incluem o tipo de interface de rede disponível, o tipo de processador, existência e tipo de câmara de vídeo, microfones, capacidade de memória, etc.

Recursos de *software* incluem tipos de *codecs* multimédia instalados, tipo e versão do navegador *Web*, aplicações de *software* instaladas no terminal, etc. Sinais gerados por componentes ou periféricos do terminal podem incluir sinais de vídeo captados por uma câmara ou sinais de áudio captados por um microfone, etc.

Esta informação é designada de informação de contexto e pode ser utilizada para auxiliar a tomada de decisão sobre a necessidade de adaptar conteúdos que são enviados para o terminal de modo a satisfazer limitações ou características do terminal e do contexto de utilização (como por exemplo condições ambientais – luminosidade extrema, ruído ambiental elevado, etc.).

O trabalho proposto nesta tese pretendia assim demonstrar a viabilidade de desenvolver e instalar um agente para plataformas móveis que fosse capaz de recolher de uma forma automática informações sobre o contexto de utilização durante o consumo de conteúdos multimédia. Consequentemente, que esse agente fosse capaz de processar e utilizar esse conhecimento para tomar decisões sobre a necessidade de adaptar conteúdo multimédia às restrições do ambiente de consumo, contribuindo dessa forma para melhorar a qualidade de experiência do utilizador.

1.4 Composição do Sistema

Neste trabalho pretendeu-se desenvolver um agente de contexto para dispositivos móveis que suportem a plataforma Androide.

Esse agente de contexto deverá ser capaz de utilizar os sensores e mecanismos disponíveis no dispositivo móvel para recolher informações que descrevam o ambiente de utilização em que o terminal está inserido.

Essas informações deverão ser formatadas de acordo com normas e protocolos existentes e enviadas para um servidor multimédia juntamente com um pedido do utilizador de consumo de recursos multimédia ou sempre que se verifiquem variações no contexto.

O servidor interpretará as informações de contexto recebidas e actuará da forma que julgar necessário para que o conteúdo seja entregue ao utilizador na forma que melhor se adequa às condições de utilização com vista a maximizar a qualidade de experiência do utilizador.

O sistema proposto para a realização do estudo é composto por 3 blocos: Servidor de *Streaming* Vídeo com adaptação e um dispositivo móvel com Android conectados por uma rede WiFi, como podemos ver na figura 3.

Figura 3. Diagrama do sistema a desenvolver

Servidor de *Streaming* de Vídeo - Representa uma máquina na rede com capacidade de emissão de um *stream* vídeo para a rede. Este bloco tem ainda de possuir a capacidade de recepção e interpretação de mensagens provenientes do terminal de consumo, a fim de alterar os parâmetros de transcodificação do vídeo.

Dispositivo Móvel em Android – Equipamento que funciona como receptor de um *stream* de vídeo, onde se pretende implementar o Agente de Contexto.

Rede WiFi - Representa a rede em que as máquinas acima referidas se interligam. A importância deste bloco tem a ver com o facto do seu desempenho desta ser um factor ou elemento de contexto.

1.5 Estrutura do Documento

Este documento é composto por seis capítulos.

O primeiro capítulo encontra-se composto pelo enquadramento da dissertação, os seus objectivos e a composição do sistema.

O capítulo número dois representa o estado da arte, ou seja a revisão bibliográfica sobre os temas abordados no desenvolvimento desta tese.

No terceiro capítulo identificam-se as tecnologias utilizadas para o desenvolvimento do trabalho.

O capítulo quatro contém uma descrição com algum pormenor sobre a plataforma Android que foi uma grande parte do estudo aquando do desenvolvimento deste trabalho, tratando com mais enfoque as áreas relacionadas com o tema.

No capítulo cinco descreve-se a arquitectura e a implementação bem como os testes desenvolvidos.

No capítulo seis são apresentadas as conclusões desta dissertação, os objectivos cumpridos e os pontos que ficam em aberto para trabalho futuro.

Para terminar, são apresentadas as referências bibliográficas utilizadas no estudo desta dissertação.

2. Estado da Arte

Este capítulo apresenta o levantamento do estado da arte sobre metadados e *Context-awareness*, efectuado antes do início da implementação.

Na última década, algumas entidades criaram esquemas normalizados de metadados e de formatos de ficheiros que permitem combinar conteúdo ("essência") e metadados num único ficheiro.

Entidades como W3C e MPEG têm desenvolvido várias normas no âmbito do *Context-awareness*, sendo no entanto algumas delas incompatíveis entre si.

As abordagens encontradas contemplam *middlewares,* mas, em caso algum, com aplicação específica à plataforma Android nem para situações específicas de *live streaming*.

2.1 Metadados

Metadados são dados sobre dados. Existem vários tipos de metadados. Há metadados que descrevem as características técnicas, outros que descrevem o conteúdo e outros ainda que descrevem o contexto.

Podemos dividir os metadados em metadados de conteúdo (quando são criados na produção) e metadados de contexto (quando são gerados no lado do consumidor).

Metadados de conteúdo:

Na produção de conteúdos, podem descrever qualquer atributo do conteúdo e podem ser divididos em 2 níveis:

- Alto nível - Título de um filme, realizador, nome dos actores, data de produção, local de filmagem, etc. Estes metadados são úteis para efectuar pesquisa;
  - Qualidade, resolução espacial e temporal, espaço de cores, esquema e codificação, direitos de autor, etc. Estes metadados são úteis para efectuar adaptação do conteúdo.
- Baixo nível - Relacionadas com a codificação e estrutura do *bitstream;*
  - Relacionadas com características intrínsecas do conteúdo:
    - movimento (info sobre vectores de movimento), texturas, cores, etc;
    - útil para efectuar pesquisa (por comparação ou "por exemplo").

Metadados de contexto:

No lado do consumidor, podem descrever qualquer atributo relativo ao ambiente de consumo:

- Capacidades do equipamento terminal

Tipos de descodificadores, capacidade de processamento, memória, armazenamento, dimensões dos ecrãs, etc.;

- Características do meio ambiente:

  Luminosidade, ruído ambiente, indoor/outdoor, etc.;

- Preferências do utilizador:

  Tipo de média preferido, tipo de conteúdo, dificuldades de audição ou de visão, etc.;

- Características das ligações

  Capacidade da rede, protocolo, etc. [1].

Para a representação semântica de metadados existem várias linguagens.

A seguir apresento um esquema de posicionamento das Entidades/ Normas e ciclo produção de conteúdos

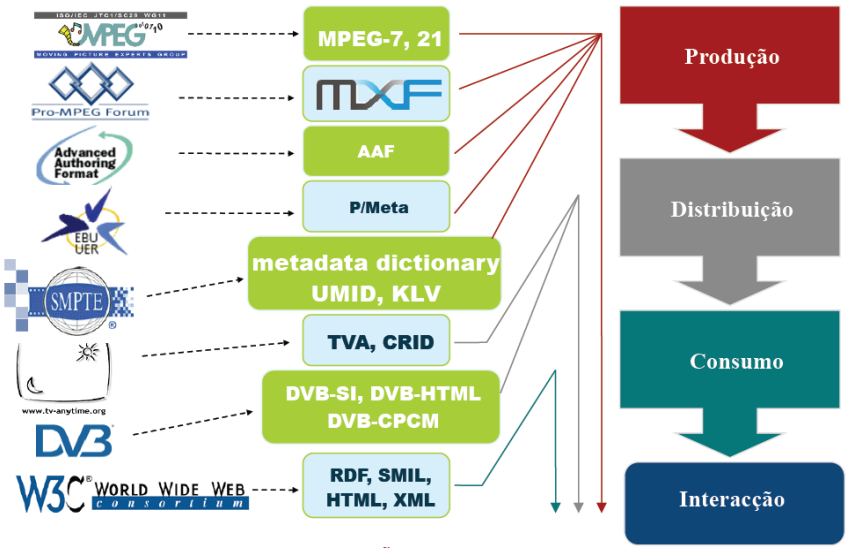


Figura 4. Normas de Metadados [1]

As mais relevantes são:

- UMID, MXF, AAF, TVA - Foram desenvolvidos especificamente para TV;
- XML - Uso genérico e é utilizado por outros formatos de metadados (RDF, MPEG-7, MPEG-21);
- MPEG-7 - Conjunto de descritores e de esquemas de descrição para conteúdo multimédia com vista a facilitar a pesquisa e acesso. Contém descritores de conteúdo e de contexto;
- MPEG-21 - Conjunto de ferramentas de descrição para permitir o uso alargado de conteúdos multimédia em ambientes heterogéneos. Tem uma estrutura de objectos multimédia complexos. Inclui direitos de autor, adaptação de conteúdos, etc. [1].

2.2 *Context-awareness*

A informação de contexto são metadados.

Existem diferentes tipos de contexto, a saber:

- **Contexto computacional** - descreve os recursos disponíveis nos dispositivos: conectividade de rede, os custos de comunicação, e largura de faixa de comunicação, bem como impressoras disponíveis, visores e ecrãs.

  O perfil do terminal (expresso como CC/PP) especifica as capacidades técnicas do terminal. Estes incluem: tamanho do ecrã (em pixels), tipo de ecrã (número de cores), suporte de recursos (WAP e Java), memória máxima para plataformas WAP e Java MIDlets, suporte para diferentes tipos de conteúdo.
- **Contexto do utilizador** - pode conter o perfil do utilizador, a localização, as pessoas próximas e até mesmo a actual situação social.
- **Contexto físico** – refere-se à iluminação, níveis de ruído, as condições de tráfego e temperatura. A informação de contexto físico podem ser obtidas a partir de sensores como termómetros, sensores de luz (dentro / fora, bolso / mesa), microfones (níveis de ruído) ou os acelerómetros (gestos, movimento, orientação). A informação destes sensores pode ser utilizada, por exemplo, adivinhando a actual situação social, identificando gestos e movimentos, e determinar a localização do dispositivo.
- **Contexto temporal** - Contém hora do dia, semana, mês e época do ano. Pode ser usado em conjunto com outro contexto para produzir notificações baseadas em contexto de pesquisa e serviço. Por exemplo, pesquisa de desportos, eventos, exposições, horários de abertura, etc. [2].

Os terminais móveis podem ser classificados no que concerne aos seus recursos de *hardware* e *software*. Este tipo de informação fornece-se-nos uma estimativa grosseira dos requisitos das normas de entrega da descrição do contexto do ponto de vista do terminal.

Uma comparação de ecrã e da memória é apresentada na Figura 5.

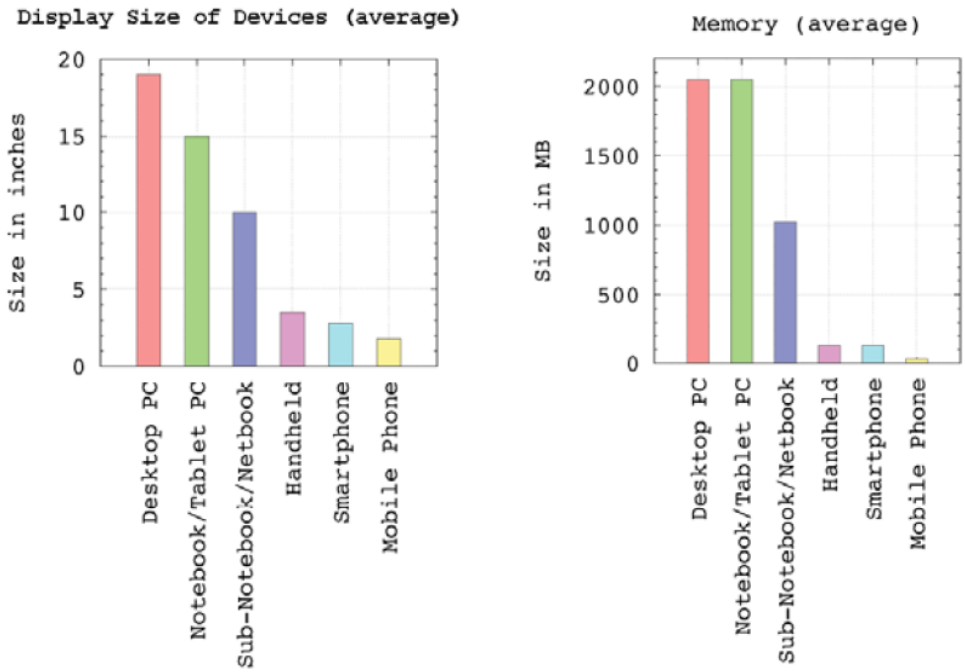


Figura 5. Comparação de ecrã e de memória em diferentes tipos de dispositivos [3]

Como se pode ver, há ainda uma enorme lacuna entre os dispositivos móveis e os dispositivos que possam ter alimentação completa.
No entanto, não só as características de *hardware* são importantes, também o *software* como sistemas operativos, codecs suportados, etc. são cada vez mais importantes [3].

Existe uma grande variedade de formatos de descrição do contexto.
No entanto, há dois mais relevantes que foram concebidos para atender aos requisitos de descrições de dispositivo e de utilizador.
O primeiro foi lançado pelo WAP Forum (agora a Open Mobile Alliance) e é chamado User Agent Profile (**UAProf**) e a segunda é a Usage Environment Description (**UED**) que foi padronizada em MPEG-21 [3].
Apresenta-se resumidamente quatro normas que têm actualmente algum relevo.

- A Composite Capabilities/Preference Profiles (**CC/PP**) inclui descrições Resource Description Framework (RDF), que abrangem as capacidades do dispositivo e preferências do utilizador através da introdução de uma hierarquia de dois níveis composta por componentes e atributos. Os componentes são os grupos de atributos relacionados com o significado, como o *software* ou as propriedades de *hardware* de um terminal.
- O User Agent Perfil (**UAProf**) foi definido pela Open Mobile Alliance (OMA), que se baseia em CC/PP, e define um vocabulário para descrever as características e capacidades de dispositivos móveis. Os componentes podem ser agrupados em `<HardwarePlatform>`, `<SoftwarePlatform>`, `<BrowserUA>`, `<NetworkCharacteristics>`, `<WapCharacteristics>`, `<PushCharacteristics>`.

- O Usage Environment Description (**UED**) é definido na Parte 7 do MPEG-21, i.e., no capítulo sobre Digital Item Adaptation. O UED é um vocabulário muito abrangente, baseado em XML Schema, e suas propriedades podem ser divididas em quatro categorias: características do utilizador, recursos do terminal, características da rede e características do ambiente natural.
- O Delivery Context Ontology (**DCO**) é baseado na Web Ontology Language (WOL) e fornece um conjunto de características que descrevem o contexto em que os recursos de media são consumidos, a saber: audio, vídeo e formatos de imagem suportados, resolução em píxeis, suporte de marcadores e localização, provedor; e informações sobre o dispositivo, ambiente e utilizador [3].

Análise e comparação das normas

A UED é baseado em XML Schema enquanto que CC/PP e UAProf são baseadas em RDF. Assim, observa-se uma incompatibilidade ao nível de tecnologia utilizada para estes formatos de descrição, principalmente entre XML Schema e RDF.

O CC/PP define apenas uma estrutura básica (ou seja, os componentes e atributos), sem especificar um vocabulário específico de termos.

A UAProf adopta CC/PP e fornece um vocabulário concreto visando principalmente WAP.

A UED define tanto a estrutura como o vocabulário abrangente.

O padrão mais recente é o DCO que adoptou OWL que é baseado em RDF. O DCO define uma ontologia que inclui não só um vocabulário de contexto dos termos de entrega, mas também unidades de medida de base [3].

De seguida apresentam-se listagens de dois exemplos de código, uma para UAProf e outra para UED.

```
1  <?xml version="1.0"?>
2  <! DOCTYPE rdf:RDF [
3    <!ENTITY ns-rdf "http://www.w3.org/1999/02/22-rdf-syntax-ns#">
4    <!ENTITY ns-prf "http://www.openmobilealliance.org/tech/profiles/UAPROF/ccppschema
                      -20030226#">
5    <!ENTITY prf-dt "http://www.openmobilealliance.org/tech/profiles/UAPROF/xmlschema
                      -20030226#">
6  ]>
7  <rdf:RDF xmlns:rdf="&ns-rdf;"
8           xmlns:prf="&ns-prf;">
9   <rdf:Description rdf:ID="MyDeviceProfile">
10   <prf:component>
11    <rdf:Description rdf:ID ="HardwarePlatform">
12     <rdf:type rdf:resource="&ns-prf;HardwarePlatform"/>
13     <prf:ScreenSizeChar rdf:datatype="&prf-dt;Dimension">15x6</prf:ScreenSizeChar>
14     <prf:BitsPerPixel rdf:datatype="&prf-dt;Number">2</prf:BitsPerPixel>
15     <prf:ColorCapable rdf:datatype="&prf-dt;Boolean">No</prf:ColorCapable>
16     <prf:TextInputCapable rdf:datatype="&prf-dt;Boolean">Yes</prf:TextInputCapable>
17     <prf:ImageCapable rdf:datatype="&prf-dt;Boolean">Yes</prf:ImageCapable>
18     <prf:Keyboard rdf:datatype="&prf-dt;Literal">PhoneKeypad</prf:Keyboard>
19     <prf:NumberOfSoftKeys rdf:datatype="&prf-dt;Number">0</prf:NumberOfSoftKeys>
20    </rdf:Description>
21   </prf:component>

22   <prf:component>
23    <rdf:Description rdf:ID="SoftwarePlatform">
24     <rdf:type rdf:resource="&ns-prf;SoftwarePlatform"/>
25     <prf:AcceptDownloadableSoftware
            rdf:datatype="&prf-dt;Boolean">No</prf:AcceptDownloadableSoftware>
26     <prf:CcppAccept-Charset>
27      <rdf:Bag>
28       <rdf:li rdf:datatype="&prf-dt;Literal">US-ASCII</rdf:li>
29       <rdf:li rdf:datatype="&prf-dt;Literal">ISO-8859-1</rdf:li>
30       <rdf:li rdf:datatype="&prf-dt;Literal">UTF-8</rdf:li>
31       <rdf:li rdf:datatype="&prf-dt;Literal">ISO-10646-UCS-2</rdf:li>
32      </rdf:Bag >
33     </prf:CcppAccept-Charset>
34    </rdf:Description>
35   </prf:component>
36  </rdf:Description>
37 </rdf:RDF>
```

Figura 6. Listing User Agent Profile with two components [3]

```
1  <?xml version="1.0"?>
2  <DIA xmlns="urn:mpeg:mpeg21:2003:01-DIA-NS" xmlns:mpeg7="urn:mpeg:mpeg7:schema:2001"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
3   <Description xsi:type="UsageEnvironmentType">
4    <UsageEnvironmentProperty xsi:type="UsersType">
5     <User xsi:type =" UserType ">
6      <UserCharacteristic xsi:type="UserInfoType">
7       <UserInfo xsi:type="mpeg7:PersonType">
8        <mpeg7:Name xsi:type =" mpeg7:PersonNameType ">
9         <mpeg7:GivenName xsi:type="mpeg7:NameComponentType">John</mpeg7:GivenName>
10        <mpeg7:FamilyName xsi:type="mpeg7:NameComponentType">Doe</mpeg7:FamilyName>
11       </mpeg7:Name>
12      </UserInfo>
13     </UserCharacteristic>
14    </User>
15   </UsageEnvironmentProperty>
16   <UsageEnvironmentProperty xsi:type="TerminalsType">
17    <Terminal>
18     <TerminalCapability xsi:type="DisplaysType">
19      <Display xsi:type="DisplayType">
20       <DisplayCapability xsi:type="DisplayCapabilityType" colorCapable="false"
                            bitsPerPixel="2">
21        <Mode>
22         <SizeChar horizontal="15" vertical="6"/>
23        </Mode>
24        <CharacterSetCode>US-ASCII</CharacterSetCode>
25        <CharacterSetCode>ISO-8859-1</CharacterSetCode>
26        <CharacterSetCode>UTF-8</CharacterSetCode>
27        <CharacterSetCode>ISO-10646-UCS-2</CharacterSetCode>
28       </DisplayCapability>
29      </Display>
30     </TerminalCapability>
31     <TerminalCapability xsi:type="UserInteractionInputsType">
32      <UserInteractionInput>
33       <UserInteractionInputSupport xsi:type="MicrophoneType"/>
34       <UserInteractionInputSupport xsi:type="KeyInputType">
35        <KeyInput href="urn:mpeg:mpeg21:2003:01-DIA-KeyInputCS-NS:2"/>
36       </UserInteractionInputSupport>
37      </UserInteractionInput>
38     </TerminalCapability>
39    </Terminal>
40   </UsageEnvironmentProperty>
41  </Description>
42 </DIA>
```

Figura 7. Listing Simple UED example [3]

Existem já bastantes sistemas desenvolvidos na área de Context-Awareness.

Serão expostos de seguida quatro exemplos que mostram diferentes abordagens à gestão do contexto.

• Context Toolkit - É um conjunto de ferramentas para auxiliar no desenvolvimento e implantação de serviços sensíveis ao contexto. As informações de contexto reflectem o ambiente da aplicação tal como é percebido pelo dispositivo. O Context Toolkit consiste em *widgets* de texto e uma infra-estrutura distribuída que hospeda os *widgets*. Os *widgets* de contexto são componentes de *software* que fornecem aplicações com acesso a informações de contexto ao esconder os detalhes dos sensores de contexto [4].

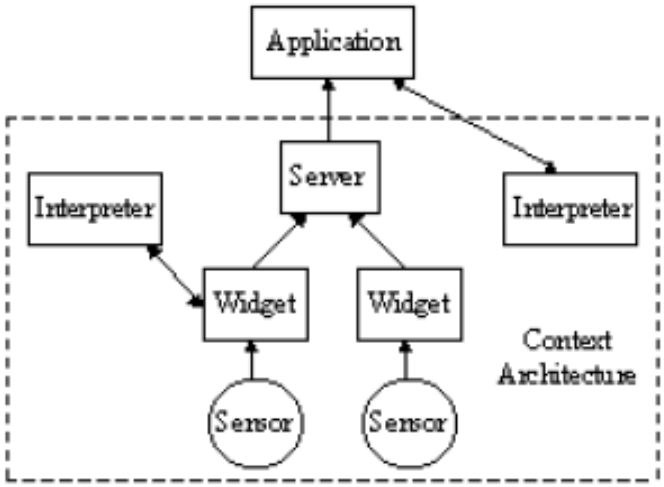


Figura 8. Arquitectura de Context toolkit Architecture (CMU) [4]

• Context Broker Architecture (CoBrA) - Arquitectura de intermediação de contexto para os espaços inteligentes. É uma arquitectura baseada em agentes para apoiar os sistemas sensíveis ao contexto em espaços inteligentes (por exemplo, salas de reuniões inteligentes, casas inteligentes e veículos inteligentes). O centro da arquitectura é um Context Broker que mantém um modelo partilhado do contexto sob o nome de uma comunidade de agentes, serviços e dispositivos no espaço e fornece protecção à privacidade de utilizadores aplicando as regras de política que eles definem.

O CoBrA fornece um exemplo de um cenário de reunião inteligente representada no esquema OWL [4].

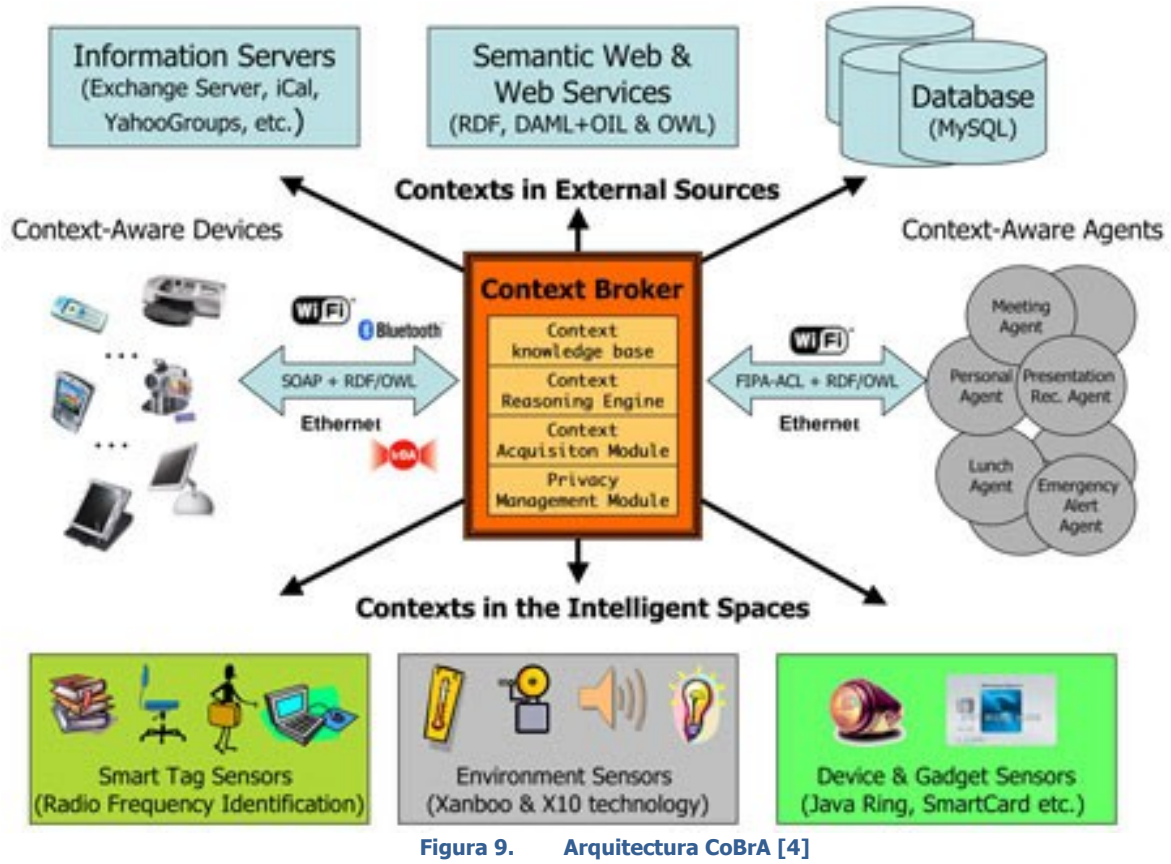


Figura 9. Arquitectura CoBrA [4]

• Context Mediated Framework (CMF) - Interface de programação de aplicações para a gestão de informações de contexto. O CMF é um ambiente de trabalho que tem quatro componentes principais: gestor de contexto, gestor de recursos do servidor, serviço de reconhecimento de contexto e serviço de aplicação. O gestor de contexto funciona como um servidor central, enquanto outras entidades (com excepção da segurança) atuam como clientes e serviços de uso do servidor fornece. O gestor de contexto, todos os servidores de recursos e aplicações executados no próprio dispositivo móvel e os serviços sejam distribuídos ou local [4].

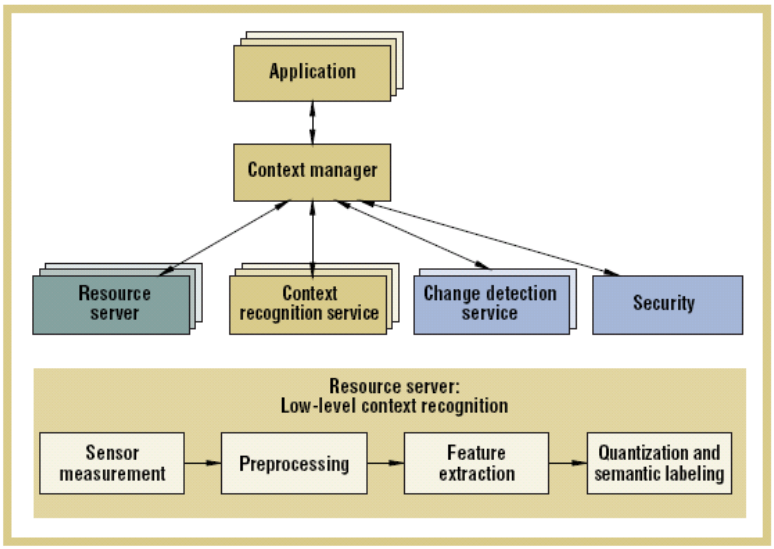


Figura 10. Arquitectura CMF [4]

• Service Oriented Context Aware Middleware (SOCAM) - A arquitetura SOCAM tem como objectivo fornecer um suporte de infra-estrutura eficiente para a construção de serviços sensíveis ao contexto em ambientes de computação ubíqua. É um *middleware* distribuído que converte vários espaços físicos, a partir dos quais os contextos são adquiridos, num espaço semântico onde os contextos podem ser partilhados e acedidos pelos serviços sensíveis ao contexto. É constituída pelos seguintes componentes, que atuam como componentes de serviço independentes: *context providers, context interpreter, context database, context aware services* e *service-locating service* [4].

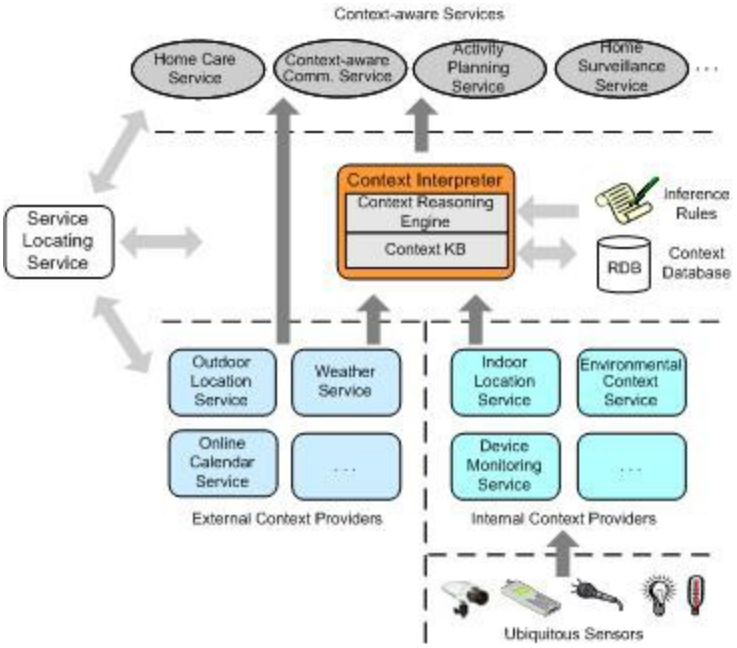


Figura 11. Arquitectura SOCAM [4]

Tabela 1. Comparação de Frameworks [4]

Arquitectura	Framework	Representação de Contexto	Conhecimento	Diversos
Context-Toolkit	Baseado em *widget*	Valor do atributo	Agregação de contexto, interrogação por atributo	*Design* simples para ajuda ao desenvolvimento de serviços de *context aware*
CoBrA	Agente Java, agente centralizado de contexto	OWL ontology, SOUPA	Raciocínio por OWL schema, regras motor de dedução	Arquitectura de *context aware* para espaços inteligentes
SOCAM	Serviços Web, arquitectura distribuída	OWL ontology	Motor de regras de encadeamento directo e inverso	Ambiente de casa inteligente
CMF	Arquitectura centralizada	Baseado em ontology	Deteção de característica e *fuzzy logic* para para criar conceitos de alto nível por um incerto sensor de dados	API para *framework* de *context aware* baseada em sensores
STU21	Arquitectura distribuída baseada em agente	OWL ontology	Motor de regras DROOLS	Espaços inteligentes em instuições de educação

2.3 Conclusões

Apresentou-se aqui o estado da arte relativo às normas de metadados e a alguns ambientes de trabalho na área do *context-awareness*.

Dos estudos realizados, a partir da extensa bibliografia referida, e observa-se que a maioria foram realizados entre os anos 2000 e 2006 e que, mais recentemente há menos documentação, possivelmente porque os trabalhos mais actuais já não tem âmbito

académico mas sim empresarial.

Relativamente à manipulação de contexto em plataforma Android, não encontrei qualquer referência a criação aplicações do tipo agente mas apenas aplicações para utilizador final com intuito meramente informativo.

O esquema a usar neste trabalho para representação destes metadados será o **UAProf** porque fornece um vocabulário para descrever as características e capacidades de dispositivos móveis.

3. Tecnologias

Para o desenvolvimento do trabalho foram estudadas e utilizadas várias tecnologias, nomeadamente um Software Development Kit (SDK) Android, a linguagem de programação Java e um Integrated Development Environment (IDE) (Eclipse) como ambiente de desenvolvimento.

O Android é uma plataforma para dispositivos móveis recente e muito promissora relativamente à qual não tinha experiência, tornando-se por isso a grande área de trabalho desta dissertação. Por esta razão dedico o capítulo seguinte exclusivamente a este tema.

Aqui faço uma breve referência às três principais tecnologias utilizadas neste trabalho.

3.1 Android

Android é um sistema operativo *open source* concebido para dispositivos móveis, em particular, telemóveis. Foi inicialmente desenvolvido pela Google e posteriormente adoptado pela Open Handset Alliance.

As aplicações Android, programadas em Java, correm numa máquina virtual chamada Dalvik. Esta incorpora grande parte da funcionalidade das máquinas virtuais de Java dos computadores de secretária, e está particularmente optimizada para dispositivos móveis.

A importância actual do Android é enorme sendo perceptível na seguinte notícia da BBC:

2 August 2010 Last updated at 15:26 GMT

Google Android phone shipments increase by 886%

Shipments of Google's Android mobile operating system have rocketed in the last year, figures suggest [5]

Worldwide smartphone market					
OS	Q2 2010 shipments	% share	Q2 2009 shipments	% share	Growth
Symbian	27,129,340	43.5	19,178,910	50.3	41.5
RIM	11,248,830	18.0	7,975,950	20.9	41
Android	10,689,290	17.1	1,084,240	2.8	885.9
Apple	8,411,910	13.5	5,211,560	13.7	61.4
Microsoft	3,083,060	4.9	3,431,380	9.0	-10.2
Others	1,851,830	3.0	1,244,620	3.3	48.8
Total	62,414,260	100	38,126,660	100	63.3

Figura 12. Mercado de Dispositivos móveis [5]

O Android SDK inclui documentação, código e utilitários para que os programadores consigam desenvolver as suas aplicações de acordo com um padrão de desenvolvimento para o sistema operativo em questão. Nele temos então bibliotecas, ferramentas e emulador Android.

A arquitectura Android é constituída por várias camadas de componentes como podemos na Figura 13.

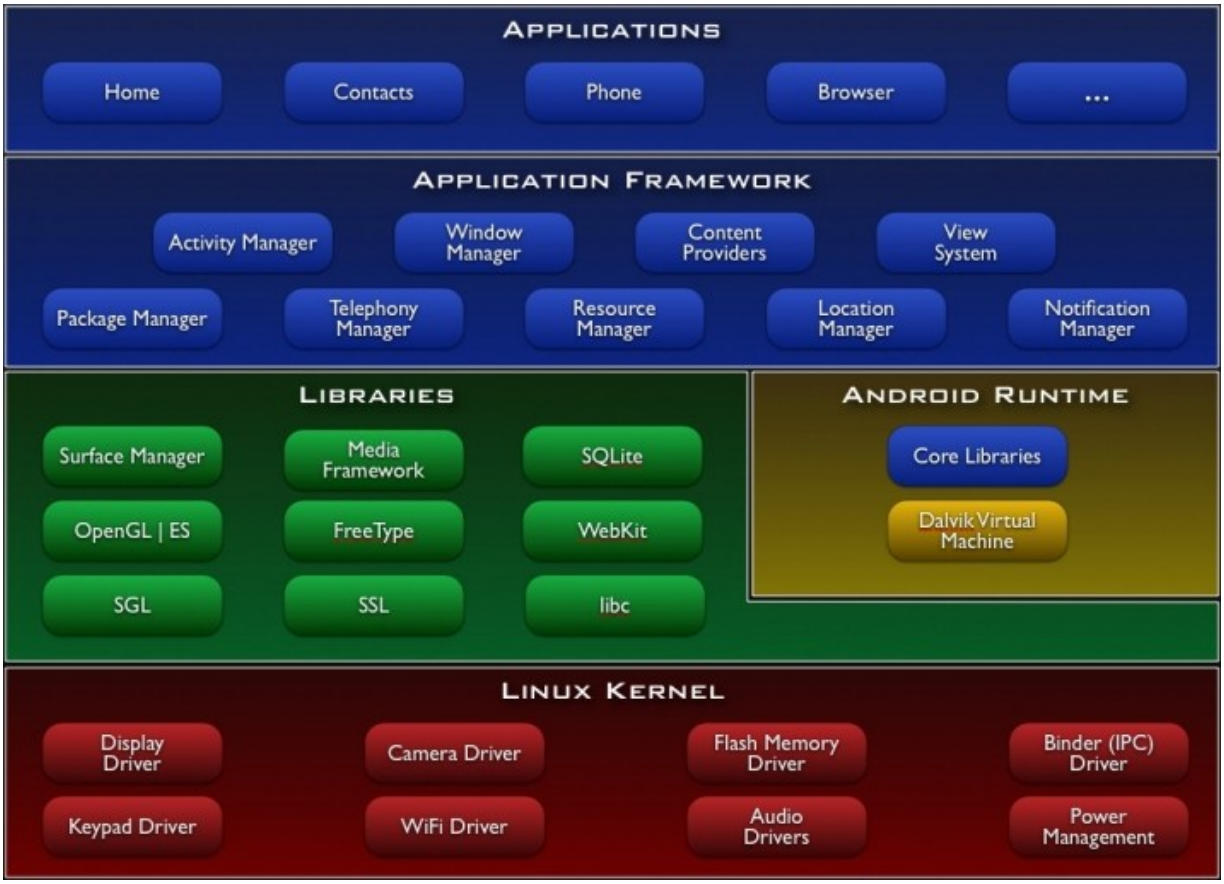


Figura 13. Arquitectura da plataforma Android [3]

3.2 Java

Java é uma linguagem de programação e plataforma de computação lançado pela Sun Microsystems em 1995. É a tecnologia subjacente a poderosos state-of-the-art incluindo programas utilitários, jogos e aplicações de negócios. Java é executado em mais de 850 milhões de computadores pessoais no mundo todo, e em milhões de outros dispositivos, incluindo dispositivos móveis e TV [6].

O Java é uma linguagem de programação que permite a criação de programas em várias plataformas sem necessidade de recompilação.

Os programas escritos em Java são compilados num código intermédio (*bytecode*) que é depois (aquando da execução do programa) interpretado e transformado em código binário.

O Java é usado como linguagem de desenvolvimento preferencial nas plataformas Android.
Para o desenvolvimento em Java há disponíveis uma máquina virtual de execução e um conjunto de ferramentas de desenvolvimento.

3.3 Eclipse

O Eclipse é uma comunidade *open source*, cujos projectos são focados na construção de uma plataforma de desenvolvimento aberta extensível composta por um ambiente de trabalho, ferramentas e ambientes de execução para a construção, implantação e gestão de *software* em todo o ciclo de vida. A Eclipse Foundation é uma organização sem fins lucrativos e ajuda a cultivar tanto a comunidade open source e um ecossistema de produtos e serviços complementares [7].
O Eclipse é um dos IDE mais utilizados no desenvolvimento de aplicações Java. Temos pois um editor, compilador, depurador entre outras funcionalidades.
Para o desenvolvimento para a plataforma Android, o Eclipse dispõe de um *plugin.*Android Development Tools (ADT) é um *plugin* para o Eclipse que é projectado para dar-lhe um ambiente poderoso, integrado, para a criação de aplicaçõesdo Android [8].

Foram instaladas e testadas as seguintes ferramentas:

- ✓ Java SE Runtime (JRE) (Version: Java SE 6 Update 23) e o Java SE Development kit (JDK)
- ✓ Eclipse IDE for Java Developers
- ✓ Plugin Android Development Tools (Version 9.0.0 January 2011)

A sua instalação é substancialmente demorada e, no caso do plugin Android Development Tools 9.0.0, a sua instalação não foi bem sucedida na primeira tentativa porque a documentação disponível é ambígua.

3.4 Conclusão

O Android, como foi referido, é uma tecnologia de que não se tinha experiência, tornando-se portanto o grande foco de estudo. Fiquei muito desperto para todo o potencial da plataforma que me parece irá ser nos próximos anos a grande rival do iPhone/iPad.Relativamente ao Java, esta é uma linguagem muito sólida e bastante utilizada.
No entanto, existe uma grande quantidade de programadores para Web que utilizam tipicamente Hypertext Markup Language (HTML), Cascading Style Sheets (CSS) e/ou JavaScript. Estes têm a possibilidade de manter as linguagens utilizando uma Open Source

Mobile Framework – PhoneGap (http://www.phonegap.com) que me parece extremamente interessante.

Quanto ao Eclipse, é um IDE muito abrangente, robusto e com excelente suporte.

Para o servidor de vídeo foi identificada uma ferramenta FFmpeg (http://www.ffmpeg.org) de *streaming* com *open source* que apesar de não ter sido possível testar, é de grande importância para esta dissertação, que suporta as funcionalidades de gravação, conversão e *streaming* de áudio e vídeo. Possui uma vasta gama de codificadores e descodificadores, permitindo assim a alteração dos parâmetros de vídeo a transmitir.

4. O Android

O Android é uma plataforma recente e muito prometedora mas, dada a minha falta de experiência, tornou-se uma grande área de trabalho nesta dissertação. Como fiz um trabalho extenso de aprendizagem neste tema, reflicto aqui a documentação que criei.
O Android é um sistema operativo baseado em Linux com uma interface de programação Java.Ele já tem uma boa quota de mercado nos Smartphones e com uma forte tendência de crescimento como se pode ver no capítulo anterior.
Temos ferramentas como, por exemplo, um compilador, um depurador e um emulador de dispositivo, bem como a sua própria máquina virtual Java (Dalvik).
O Android é criado pela Open Handset Alliance, que é liderada pela Google. Suporta gráficos 2D e 3D utilizando as bibliotecas OpenGL, e efectua o armazenamento de dados numa Base de Dados SqlLite.
Para o desenvolvimento de aplicações, a Google oferece o *plugin* Android Development Tools (ADT) para o Eclipse.

4.1 Desenvolvimento de aplicações Android

- Java Development Kit – JDK
  http://www.oracle.com/technetwork/java/javase/downloads/index.html
- Eclipse (IDE for Java Developers)
  http://www.eclipse.org/downloads/
- Base Android SDK -
  http://developer.android.com/sdk/index.html
- Android Development Tools (ADT) (Plugin para o Eclipse)
  http://developer.android.com/sdk/eclipse-adt.html

4.2 Instalação

- A instalação do Java Development Kit – JDK é bastante simples e tradicional. Fazer descarga da versão pretendida e executa-se o instalador. A versão actual (Kit 6 update 25) suporta os sistemas operativos Linux, Solaris e Windows, com versões de 32 b e 64 b.

- O Eclipse para ser instalado, deve-se fazer a descarga da versão pretendida (Windows, Mac OS X ou Linux , versões de 32 b ou 64 b) e executar o instalador. Esta instalação é algo demorada - no meu caso levou cerca de 1 h.

- Para SDK do Android, descarrega-se o ficheiro zip e extrai-se para qualquer lugar no sistema de ficheiros, por exemplo, para "C: \ Program Files \ android-sdk-windows".

- A instalação do ADT, visto ser um *plugin* para o Eclipse, instala-se dentro do Eclipse com Help->Install New Software

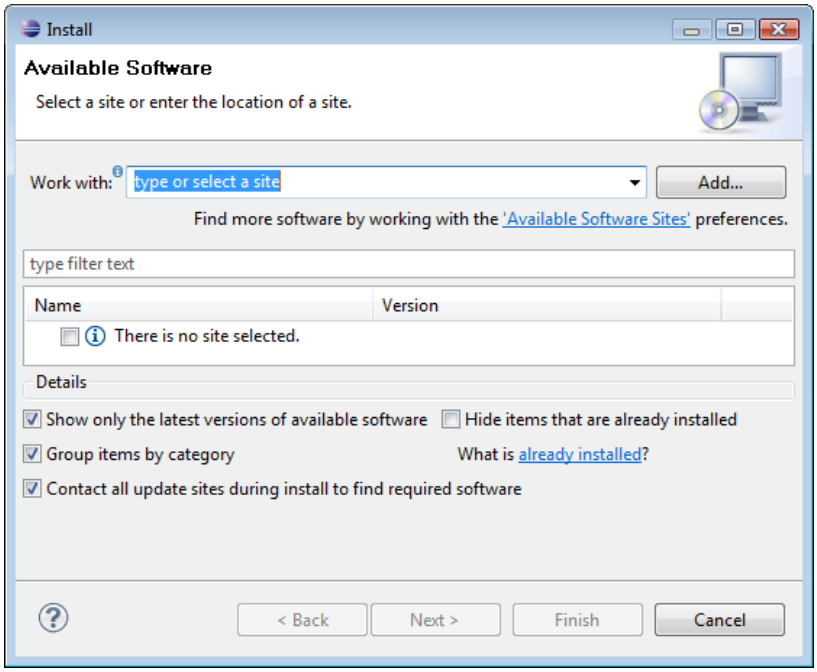


Figura 14. Janela de instalação do Eclipse

e fornece-se a URL: https://dl-ssl.google.com/android/eclipse/ , seguindo-se as instruções.

Deve reiniciar-se o Eclipse para ele arrancar com o *plugin* ADT.

De seguida, no Eclipse, abrir as Preferências através do Windows -> Preferences , seleccionar Android e digitar o caminho de instalação do SDK do Android – ver figura 15.

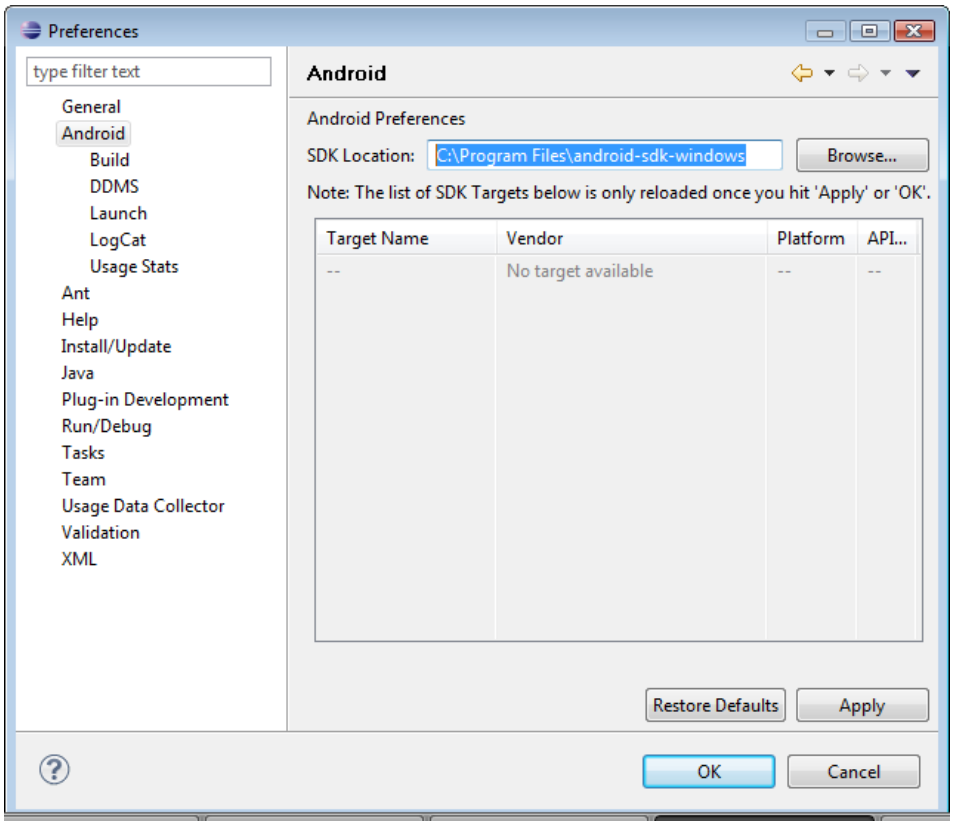


Figura 15. Janela de preferencias do Eclipse

Seleccionar Window > Android SDK and AVD Manager a partir do menu.

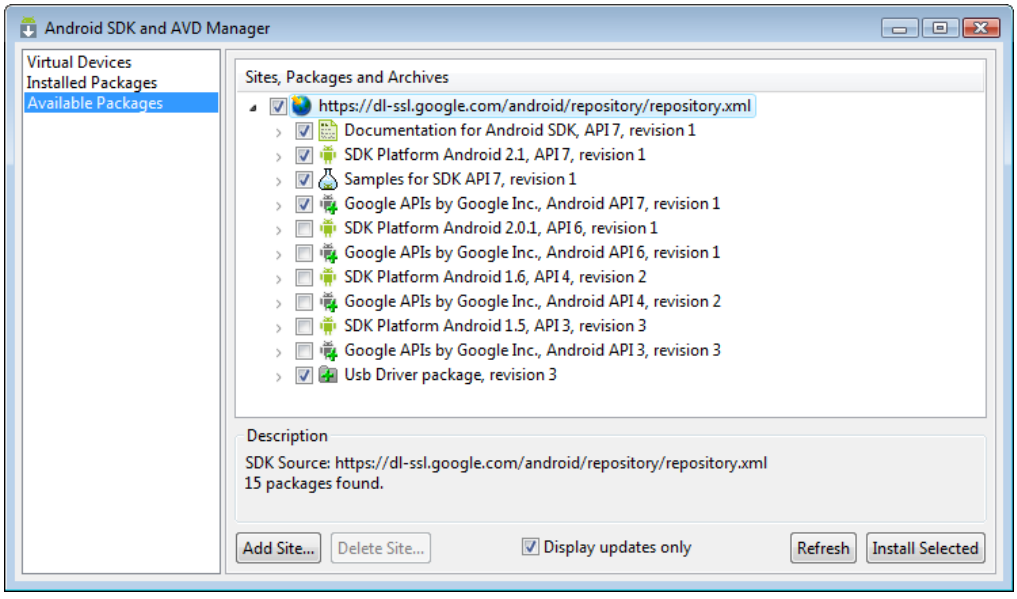


Figura 16. Janela no Eclipse de Android SDK and AVD Manager de Packages disponiveis

Pressionar o botão "Install Selected" e aceitar todos os pacotes.

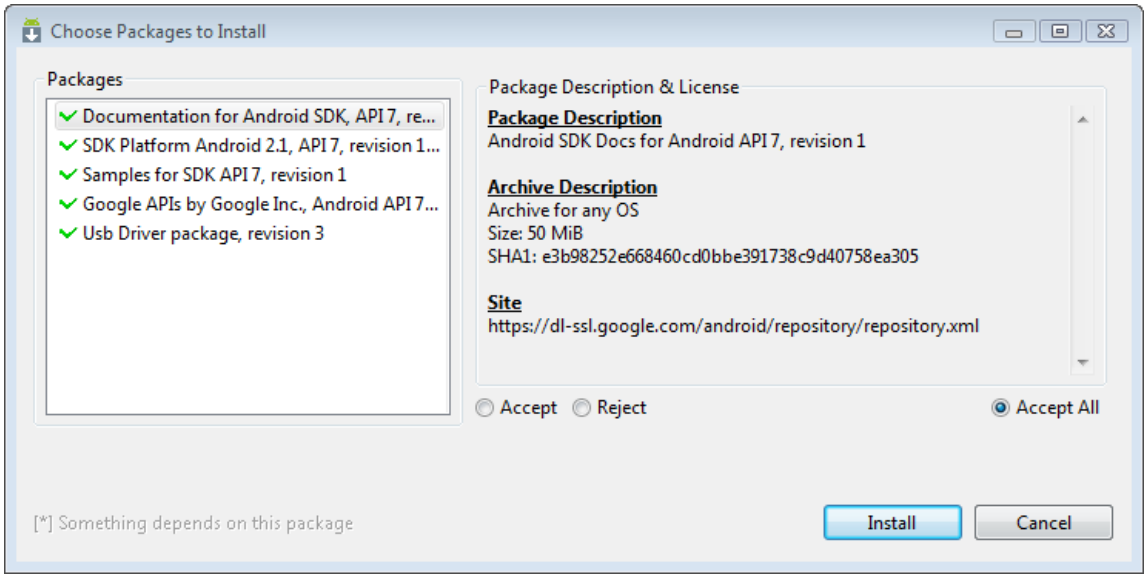


Figura 17. Janela no Eclipse de escolha de instalação de Packages

Após a instalação, reiniciar o Eclipse.

4.3 Emulação - Android Virtual Devices - AVD

Para usar o emulador, é necessário definir um dispositivo. Para tal selecciona-se Window-> Android SDK and AVD Manager a partir do menu.

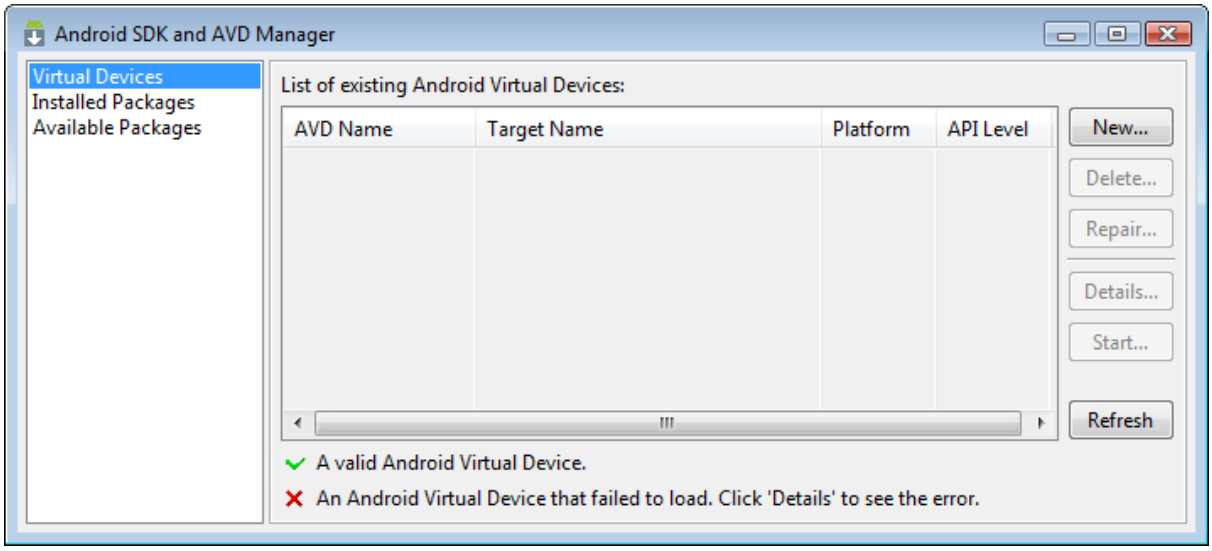


Figura 18. Janela no Eclipse de Android SDK and AVD Manager de Virtual Devices (AVD)

Pressiona-se o botão "New"

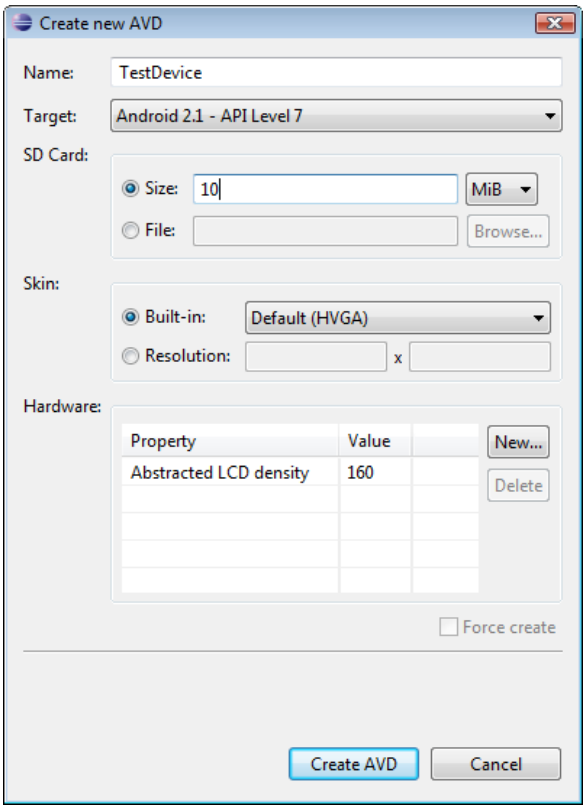


Figura 19. Janela no Eclipse de Create new AVD

Para se testar, selecciona-se o dispositivo e pressiona-se "Start".

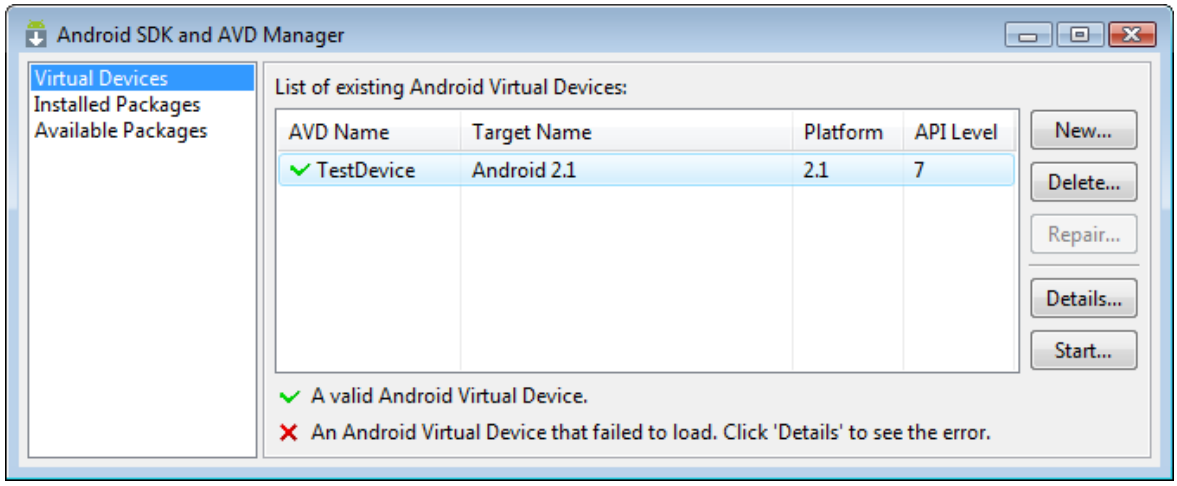


Figura 20. Janela no Eclipse de Android SDK and AVD Manager com Virtual Devices (AVD) criado

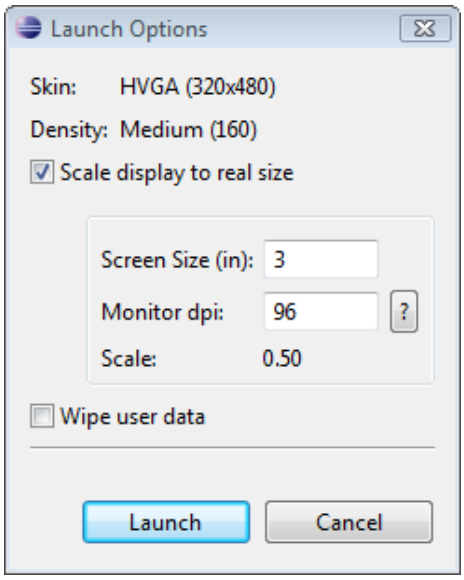


Figura 21. Janela no Eclipse de Opções para executar o AVD

Figura 22. Janela do AVD

4.4 Criação de um novo projecto

Para criar um novo projecto em Android deve-se fazer File > New > Project > Android project

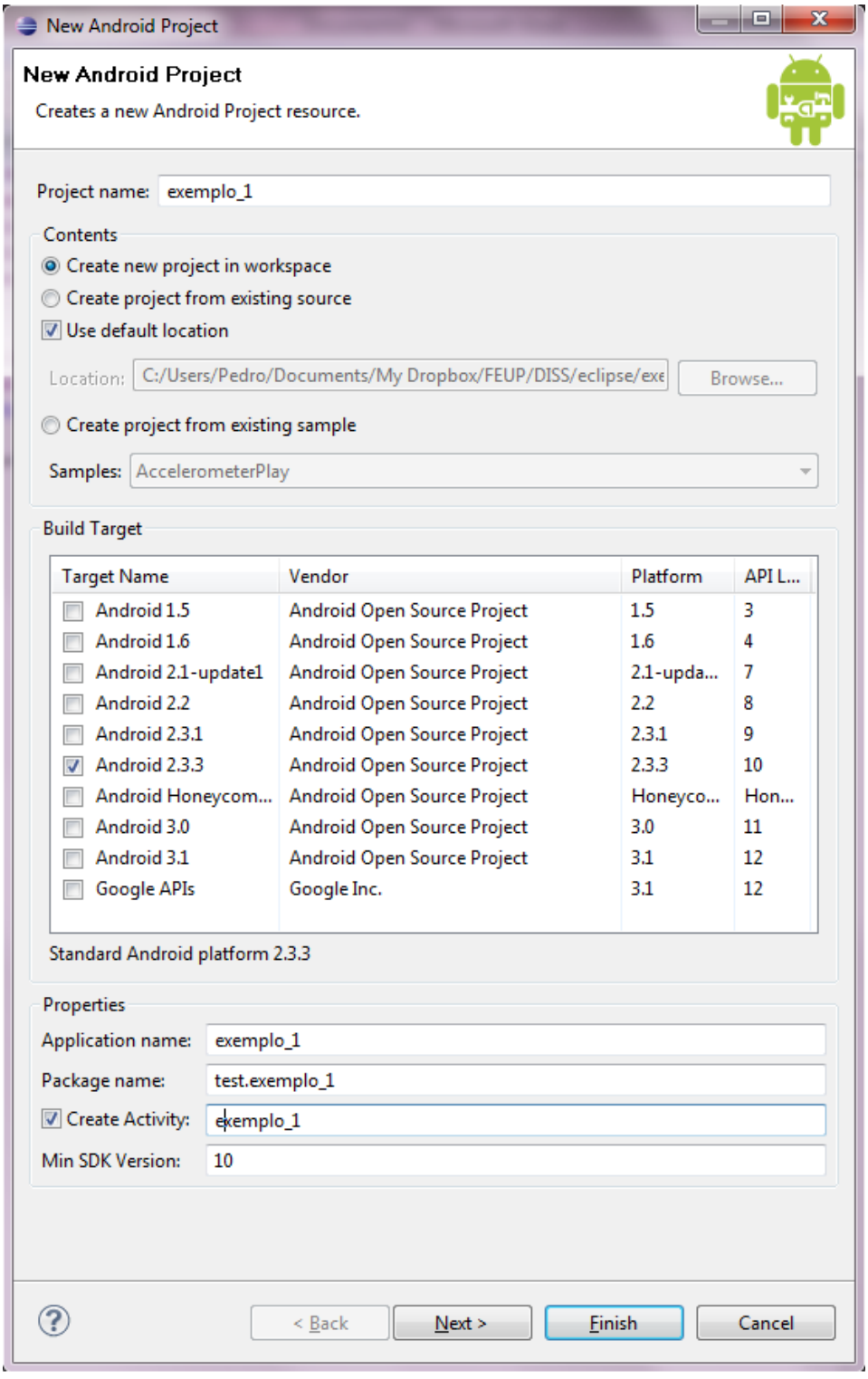


Figura 23. Janela no Eclipse para criar um novo projecto Android

E deve ver a seguinte estrutura no Package Explorer.

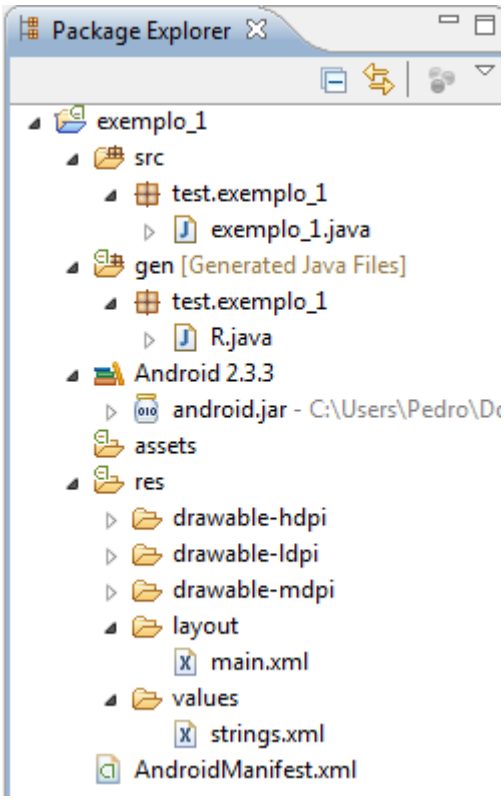


Figura 24. Janela (parcial) no Eclipse da árvore de um projecto Android

quando cria um novo projecto Android, tem vários itens no directório raiz do projecto:

- **src** /
  Pasta que contém o código-fonte Java
- **gen** /
  Este é o lugar onde as ferramentas do Android irão colocar o código fonte que eles geram. R.java é uma classe gerada, que contém o texto e os elementos da interface. Os projectos Android vêm com esta classe livre 'R', que é a abreviação de recursos. Esta é uma classe gerada automaticamente que nós podemos usar os recursos de referência no nosso projecto. Não devemos tentar modificar essa classe manualmente.
- **assets** /
  Uma pasta que contém outros ficheiros estáticos.
- **res** /
  Uma pasta que contém recursos como ícones, interface gráfica de utilizador (GUI), layouts, etc, que são empacotados com o Java compilado no aplicativo.
- **AndroidMainFest.xml**
  Um ficheiro XML que descreve a aplicação que está sendo construído e quais os componentes (actividades, serviços e assim por diante), estão sendo fornecidos pelo aplicativo. Este ficheiro é a base para qualquer aplicação Android.

4.5 Suporte

A Google disponibiliza *online* em http://developer.android.com/index.html todo o suporte a nível de ferramentas de desenvolvimento, guia de desenvolvimento, referência de código Java e recursos como tutoriais, exemplos e fóruns.

4.6 Projectos Android

Os projectos Android são construídos num ficheiro `.apk` que se instala num dispositivo. Eles contêm o código fonte da aplicação e os ficheiros dos recursos.

Um projecto Android é composto pelos seguintes directórios e ficheiros.

`src/`
Contém o ficheiro de Actividade, que é armazenado em `src/your/package/namespace/ActivityName.java` . Todos os outros ficheiros de origem de código (como o ficheiros `.java` ou `.aidl`) estão aqui também.

`bin`
Directório de saída da compilação. Isto é onde se pode encontrar o ficheiro final `.apk` e outros recursos compilados.

`jni`
Contém código fonte nativo desenvolvido utilizando o NDK Android.

`gen/`
Contém os ficheiros Java gerado pelo ADT como o ficheiro `R.java` e interfaces criadas a partir de ficheiros .aidl.

`assets/`
Pode-se usar para armazenar outros ficheiros. Ficheiros que se salva aqui são compilados num ficheiro `.apk` tal como está, e o nome do ficheiro original é preservado. Pode-se navegar nesse directório da mesma forma como um sistema de ficheiro típico usando URI e ler ficheiros como um fluxo de bytes. Por exemplo, este é um bom local para texturas e dados do jogo.

`res/`
Contém os recursos do aplicativo, como ficheiros *drawable*, ficheiros de *layout*, e os valores de *string*.

`anim/`
Para os ficheiros XML que são compilados em objectos de animação.

`color/`
Para os ficheiros XML que descrevem cores

`drawable/`
Para ficheiros *bitmap* (PNG, JPEG, 9 Patch ou GIF) e ficheiros XML que descrevem formas *Drawable* ou objectos *Drawable* que contêm vários estados (normal, pressionado, ou com foco).

`layout/`
Ficheiros XML que são compiladas em *layouts* de ecrã

`menu/`
Para os ficheiros XML que definem menus de aplicativos.

`raw/`
Salvar ficheiros multimédia aqui em vez do directório `assets/`, só difere na forma como se acede. Esses ficheiros são processados por AAPT e deve ser referenciado a partir da aplicação usando um identificador de recurso de classe `R`. Por exemplo, este é um bom lugar para ficheiros de multimédia, como ficheiros MP3 ou Ogg.

`values/`
Para os ficheiros XML que são compilados em vários tipos de recursos.

`xml/`
Para diversos ficheiros XML que configuram os componentes da aplicação.

`libs/`
Contém livrarias privadas.

`AndroidManifest.xml`
O ficheiro de controle que descreve a natureza do pedido e cada um de seus componentes. Veja o capítulo AndroidManifest.xml para mais informações .

`build.properties`
Propriedades personalizáveis para o sistema de compilação. Pode-se editar este ficheiro para substituir as configurações padrão usadas pelo Ant. Se se usa Eclipse, este ficheiro não é usado.

`build.xml`
A construção ficheiro Ant para o projecto. Só é aplicável para projectos criados com a linha de comando.

`default.properties`
Este ficheiro contém as configurações do projecto. Este ficheiro é parte integrante do projecto, como tal, deve ser mantido. Não editar o ficheiro manualmente.

4.7 Requisitos de Sistema

As secções seguintes descrevem os requisitos de sistema e software para desenvolver aplicações Android utilizando o Android SDK.

4.7.1 Sistemas Operativos Suportados

- Windows XP (32 bits), Vista (32 bits ou 64 bits) ou Windows 7 (32 bits ou 64 bits)
- Mac OS X 10.5.8 ou posterior
- Linux (testado no Ubuntu Linux, Lucid Lynx)
  - GNU C Library (glibc) 2.7 ou posterior

- No Ubuntu Linux, versão 8.04 ou posterior

4.7.2 Requisitos de Hardware

O SDK do Android requer o armazenamento em disco de todos os componentes a instalar. A tabela 26 fornece uma ideia aproximada dos requisitos de espaço em disco, com base nos componentes que se planeie usar.

Tabela 2. Memória típica ocupada pelos diversos components de Android [8]

Component type	Approximate size	Comments
SDK Tools	35 MiB	Required.
SDK Platform-tools	6 MiB	Required.
Android platform (each)	150 MiB	At least one platform is required.
SDK Add-on (each)	100 MiB	Optional.
USB Driver for Windows	10 MiB	Optional. For Windows only.
Samples (per platform)	10 MiB	Optional.
Offline documentation	250 MiB	Optional.

4.8 AndroidManifest

Cada aplicação deve ter um ficheiro AndroidManifest.xml na pasta raiz. O AndroidManifest.xml apresenta informações essenciais sobre a aplicação para o sistema Android, o sistema de informações deve ter antes que ele possa executar qualquer código do aplicativo. Entre outras coisas, o AndroidManifest.xml faz o seguinte:

- Indica o pacote de Java para a aplicação. O nome do pacote serve como um identificador exclusivo para o aplicativo.
- Descreve os componentes da aplicação - as actividades, serviços, receptores de radiodifusão e fornecedores de conteúdos que a aplicação utiliza. Ele cita as classes que implementam cada um dos componentes e publica as suas capacidades.
- Determina que processos acolhem os componentes da aplicação.
    - Declara que as permissões da aplicação deve ter para aceder a partes protegidas da Application Programming Interface (API) e interagir com outras aplicações.
- Declara o nível mínimo da API Android que a aplicação requer.
- Lista as bibliotecas usadas.

4.8.1 Estrutura do Ficheiro AndroidManifest

O diagrama da figura 25 mostra a estrutura geral do AndroidManifest.xml e cada elemento que ele pode conter. Cada elemento, juntamente com todos os seus atributos, está documentado na íntegra num ficheiro separado. Para ver informações detalhadas sobre qualquer elemento, clique sobre o nome do elemento no diagrama, na lista alfabética de elementos que segue o diagrama, ou em qualquer outra menção do nome do elemento.

```
<?xml version="1.0" encoding="utf-8"?>

<manifest>

    <uses-permission />
    <permission />
    <permission-tree />
    <permission-group />
    <instrumentation />
    <uses-sdk />
    <uses-configuration />
    <uses-feature />
    <supports-screens />
    <compatible-screens />
    <supports-gl-texture />

    <application>

        <activity>
            <intent-filter>
                <action />
                <category />
                <data />
            </intent-filter>
            <meta-data />
        </activity>

        <activity-alias>
            <intent-filter> . . . </intent-filter>
            <meta-data />
        </activity-alias>

        <service>
            <intent-filter> . . . </intent-filter>
            <meta-data/>
        </service>

        <receiver>
            <intent-filter> . . . </intent-filter>
            <meta-data />
        </receiver>

        <provider>
            <grant-uri-permission />
            <meta-data />
        </provider>

        <uses-library />

    </application>

</manifest>
```

Figura 25. Estrutura do ficheiro AndroidManifest.xml [8]

4.8.2 Permissões

A *permissão* é uma restrição que limita o acesso a uma parte do código ou dados no dispositivo. A limitação é imposta para proteger dados críticos e código que poderiam ser utilizadas para distorcer ou danificar a experiência do utilizador.

Cada permissão é identificada por um rótulo único. Muitas vezes, o rótulo indica a acção que é restrita.

Um recurso pode ser protegido por, no máximo, uma permissão.

Se um aplicativo precisa de acesso a um recurso protegido por uma permissão, deve declarar que exige que a permissão com um elemento `<uses-permission>` no AndroidManifest.xml. Então, quando a aplicação é instalado no dispositivo, o instalador determina se concede ou não a permissão solicitada, verificando as autoridades que assinaram os certificados da aplicação e, em alguns casos, pedindo ao utilizador. Se a permissão for concedida, a aplicação é capaz de usar os recursos protegidos. Se não, as suas tentativas de acesso a esses recursos simplesmente falharam sem qualquer notificação ao utilizador.

Um aplicativo também pode proteger seus próprios componentes (actividades, serviços, receptores de radiodifusão e fornecedores de conteúdos) com permissões. Ele pode empregar qualquer uma das permissões definidas pelo Android (listados na `android.Manifest.permission`)

Esta lista é muito extensa e pode ser vista em
http://developer.android.com/reference/android/Manifest.permission.html

Apresento no entanto aqui a título de exemplo algumas permissões que usei:

Tabela 3. Tabela (parcial) de permissões usadas no ficheiro AndroidManifest.xml [8]

Constants		
String	ACCESS_NETWORK_STATE	Allows applications to access information about networks
String	ACCESS_WIFI_STATE	Allows applications to access information about Wi-Fi networks
String	CHANGE_NETWORK_STATE	Allows applications to change network connectivity state
String	CHANGE_WIFI_STATE	Allows applications to change Wi-Fi connectivity state
String	INTERNET	Allows applications to open network sockets.
String	MODIFY_AUDIO_SETTINGS	Allows an application to modify global audio settings

Por exemplo, aqui estão algumas permissões definidas pelo Android:

```
android.permission.ACCESS_NETWORK_STATE
android.permission.CHANGE_WIFI_STATE
```

4.9 Níveis de API no Android

É útil compreender a abordagem geral da plataforma para API. É também importante compreender o identificador de Nível API e o papel que ela desempenha para garantir a compatibilidade da aplicação com dispositivos em que pode ser instalado.

As seções seguintes fornecem informações sobre o Nível API e como isso afecta suas aplicações.

4.9.1 Níveis de API

O Nível API é um valor inteiro que identifica a revisão API oferecida por uma versão da plataforma Android.

A plataforma Android oferece uma API quadro que os aplicaçõespodem usar para interagir com o sistema Android subjacente. A estrutura API consiste em:

- Um conjunto de pacotes e classes
- Um conjunto de elementos XML e atributos para declarar um ficheiro de manifesto
- Um conjunto de elementos XML e atributos para declarar e aceder aos recursos
- Um conjunto de Intenções
- Um conjunto de permissões que as aplicações podem solicitar

Cada versão sucessiva da plataforma Android pode incluir actualizações para a aplicação Android API que ele oferece.

Actualizações para o API são projectados para que a nova API permanece compatível com versões anteriores do API. Ou seja, a maioria das mudanças na API são aditivos e introduzir novas funcionalidades ou substituição. Como partes da API são actualizados, as partes mais velhas substituídas estão obsoletas, mas não são removidos, de modo que as aplicações existentes ainda podem usá-los.

A estrutura API que proporciona uma plataforma Android é especificada usando um identificador inteiro chamado de "Nível de API". Cada versão da plataforma Android suporta exactamente um nível API, embora o apoio está implícita para todos os níveis anteriores API (até API Nível 1). A versão inicial da plataforma Android desde API Nível 1 e versões posteriores têm incrementado o nível API.

A tabela a seguir especifica o nível API suportados por cada versão da plataforma Android.

Tabela 4. Níveis de API em Android [8]

Versão plataforma	Nível API
Android 3.1	12
Android 3.0	11
Android 2.3.4	10
Android 2.3.3	
Android 2.3	9
Android 2.2	8

Android 2.1	7
Android 2.0.1	6
Android 2.0	5
Android 1.6	4
Android 1.5	3
Android 1.1	2
Android 1.0	1

4.9.2 Utilização de Nível de Api de Android

Os aplicaçõespodem usar um elemento no AndroidManifest.xml fornecida pela API `<uses-sdk>` para descrever os níveis API mínima e máxima em que eles são capazes de executar, bem como o nível API preferível que eles são projectados para suportar. O elemento oferece três principais atributos:

- `android:minSdkVersion` - especifica o nível API mínimo em que a aplicação é capaz de executar. O valor padrão é "1".
- `android:targetSdkVersion` - especifica o nível API em que a aplicação foi projectado para ser executado. Em alguns casos, isso permite que a aplicação para usar elementos de AndroidManifest.xml ou comportamentos definidos no Nível API alvo, ao invés de ser limitado a utilizar apenas os definidos para o nível API mínimo.
- `android:maxSdkVersion` - especifica o nível API máximo no qual a aplicação é capaz de executar.

Quando declarados no AndroidManifest.xml de um aplicativo, um elemento `<uses-sdk>` tem esta aparência:

```
<manifest>
 <uses-sdk android:minSdkVersion="5" />
 ...
<manifest/>
```

A razão principal que o pedido iria declarar uma API em Nível `android:minSdkVersion` é dizer ao sistema Android que ele está usando APIs que foram *introduzidas* no nível API especificado. Se o pedido fosse de alguma forma instalado em uma plataforma com um menor nível de API, então deixaria de funcionar durante a execução quando ele tentar aceder às APIs que não existem. O sistema impede, não permitindo que a aplicação seja instalado se o nível de API for maior que a versão da plataforma no dispositivo de destino.

4.9.3 Compatibilidade de Aplicações para a Frente

Os aplicaçõesdo Android são geralmente compatível para a frente com a nova versão da plataforma Android porque quase todas as alterações na API são aditivas.

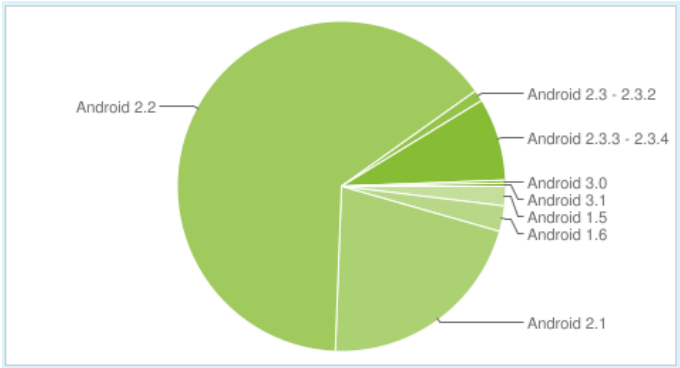
A compatibilidade para a frente é importante porque muitos dispositivos Android fazem actualizações do sistema. O utilizador pode instalar a aplicação e usá-lo com sucesso, e depois receber uma actualização para uma nova versão da plataforma Android. Uma vez que a actualização é instalada, a aplicação será executado em uma versão run-time do novo ambiente, mas que tem a capacidade e o sistema de API que a aplicação depende.

4.9.4 Utilização Actual das Versões da Plataforma

Aqui temos dados sobre o número relativo de dispositivos activos em execução numa determinada versão da plataforma Android. Isto ajuda-nos a entender a paisagem da distribuição de dispositivos e decidir como dar prioridade ao desenvolvimento das funções da aplicação para os dispositivos actualmente nas mãos dos utilizadores.

Distribuição actual

O gráfico e a tabela da figura 26 são baseados no número de dispositivos Android que acederam ao Android Market dentro de um período de 14 dias que terminou em 1 de Junho deste ano.



Platform	API Level	Distribution
Android 1.5	3	1.9%
Android 1.6	4	2.5%
Android 2.1	7	21.2%
Android 2.2	8	64.6%
Android 2.3 - Android 2.3.2	9	1.1%
Android 2.3.3 - Android 2.3.4	10	8.1%
Android 3.0	11	0.3%
Android 3.1	12	0.3%

Data collected during a 14-day period ending on June 1, 2011

Figura 26. Gráfico e tabela de níveis de API dos dispositivos móveis no acesso ao Android Market [8]

Distribuição histórica

O gráfico seguinte fornece um histórico do número relativo de dispositivos Android activos usando diferentes versões da plataforma Android. Ele também proporciona uma valiosa perspectiva de quantos dispositivos de aplicação é compatível com a versão da plataforma.

As versões da plataforma são empilhados em cima uns dos outros com a versão mais antiga activa no topo. Este formato indica a percentagem total de dispositivos activos que são compatíveis com uma determinada versão do Android. Por exemplo, se desenvolver uma aplicação para a versão que está no topo do gráfico, então a sua aplicação é compatível com 100% de dispositivos activos (e todas as versões futuras), porque todas as APIs do Android são compatíveis para a frente.

Cada conjunto de dados na linha do tempo é baseado no número de dispositivos Android que acedeu ao Android Market dentro de um período de 14 dias que termina na data indicada no eixo-x.

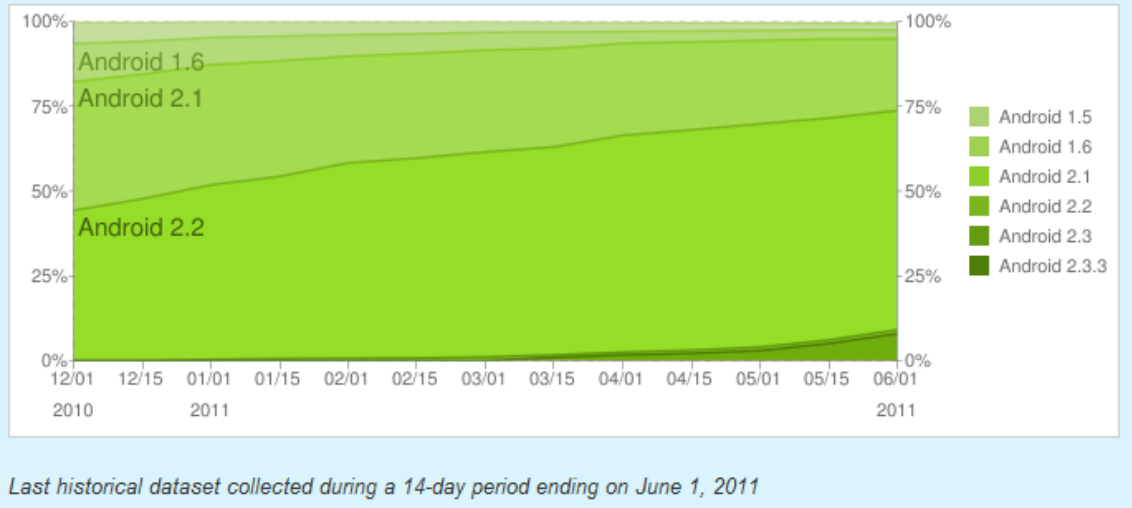


Figura 27. Gráfico de níveis de API dos dispositivos móveis no acesso ao Android Market em 6 meses [8]

Do gráfico gostaria de destacar três situações:

- Actualmente cerca de 85% dos utilizadores estão a usar dispositivos com versão Androide 2.1 ou 2.2.

- Nota-se que no último meio ano que representa o gráfico, as versões Android 1.5 e 1.6 que representavam cerca de 20% em Dezembro de 2010, diminuíram bastante e representam em Junho de 2011 cerca de 5%

- Nos últimos 3 meses com aparecimento das versões Android 2.3 e 2.3.3, o número de utilizadores destas versões teve um crescimento muito acentuado representando já cerca de 9%.

4.10 Android Virtual Devices - AVD

Um AVD (Android Virtual Devices) é um emulador que permite modelar um dispositivo real, definindo opções de hardware e software para ser emulado pelo Android.

Pode criar um AVD de duas formas, a primeira a partir de Eclipse, clicando em ***Window> Android SDK and AVD Manager.*** Também pode criar um AVD a partir da linha de comando, chamando o emulator no ***directório*** tools do SDK do Android.

Um AVD consiste em:

- Um perfil de hardware: Define as características de hardware do dispositivo virtual. Por exemplo, você pode definir se o dispositivo possui uma câmara, se ele usa um teclado QWERTY físico ou um teclado de marcação, a quantidade de memória que tem, entre outros. Um mapeamento de uma imagem do sistema: Você pode definir qual a versão da plataforma Android será executado no dispositivo virtual. Outras opções: Você pode especificar a capa do emulador que você deseja usar

com o AVD, que lhe permite controlar as dimensões da tela e a aparência. Também pode especificar a emulação do cartão SD para usar com o AVD.

- Uma área de armazenamento de desenvolvimento: os dados do dispositivo do utilizador como aplicaçõesinstalados, configurações e a emulação cartão SD são armazenados nesta área.

  Pode criar vários AVDs com base nos tipos de dispositivo que você deseja modelar. Para testar exaustivamente o aplicativo, deve-se criar um AVD para cada configuração do dispositivo geral (por exemplo, diferentes tamanhos de tela e as versões plataforma) com o qual a aplicação é compatível e testar a sua aplicação em cada um. Deve-se criar pelo menos um AVD que utiliza um API de nível é maior do que a exigida pela sua aplicação, pois permite testar a compatibilidade para a frente de sua aplicação.

O emulador Android suporta muitos recursos de hardware que pode ser encontrado em dispositivos móveis, incluindo:

- CPU ARMv5 e a unidade de gestão de memória (MMU)
- Display LCD 16-bit
- Um ou mais teclados (um teclado Qwerty-based e associados Dpad / Telefone botões)
- Controlador de áudio com capacidade de saída e de entrada
- Partições de memória Flash (emulado através de ficheiros de imagem de disco)
- Um modem GSM, incluindo um cartão SIM simulado

As seções seguintes fornecem mais informações sobre o emulador e como usá-lo para desenvolvimento de aplicaçõesAndroid.

4.10.1 Criar um AVD

Para criar um AVD no Eclipse :

Inicie o gestor de AVD no Eclipse: seleccione ***Janela> Android SDK e AVD Manager***

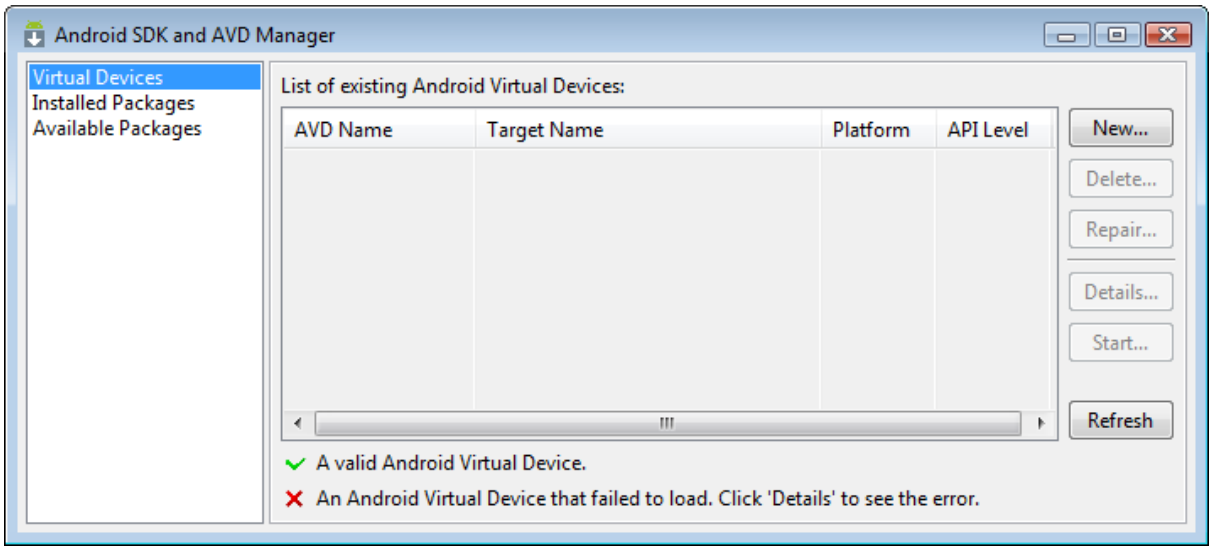


Figura 28. Janela no Eclipse de Android SDK and AVD Manager

No painel Virtual Devices, verá uma lista de AVDs existente. Clique em ***New*** para criar um AVD novo. A **C**aixa de diálogo Create new AVD aparece.

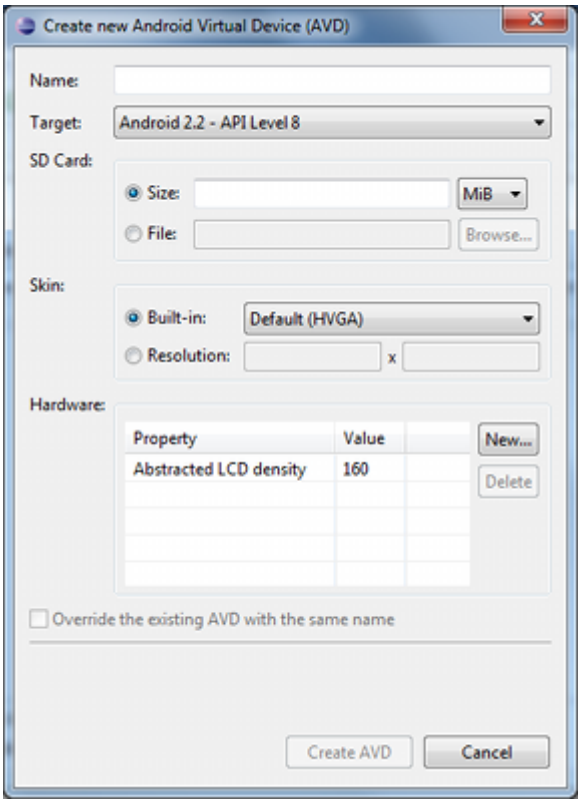


Figura 29. Janela no Eclipse para criar novo Android Virtual Devices (AVD)

Preencha os detalhes para a AVD.

Dar-lhe um nome, escolha a plataforma, um tamanho de cartão SD, e o ecrã (HVGA é o padrão). Também pode adicionar recursos de hardware específicos do dispositivo emulado, clicando no botão ***New*** e seleccionando o recurso.

Clique em ***Create AVD.***

O AVD está pronto e pode fechar o Gerenciador de SDK e AVD, criar mais AVDs, ou lançar um emulador com o AVD seleccionando um dispositivo e clicando em ***Start.***

4.10.2 Iniciar e Parar o Emulador

Pode-se iniciar o emulador como um aplicativo independente, a partir de uma linha de comando, ou pode usá-lo como parte de seu ambiente de desenvolvimento Eclipse. Em ambos os casos, você especifica a configuração AVD para carregar e quaisquer opções de inicialização que você deseja usar.

Pode-se executar a aplicação numa única instância do emulador ou, dependendo de suas necessidades, pode iniciar múltiplas instâncias do emulador e executar a aplicação em mais de um dispositivo emulado.

Para iniciar uma instância do emulador da linha de comando, mude para a pasta `tools/` do SDK e digite o comando `emulator` como este:

```
emulator -avd <avd_name>
```

Isso inicializa o emulador e carrega uma configuração AVD e abrirá uma janela do emulador.

Se estiver a trabalhar no Eclipse, o plugin para o Eclipse ADT instala a aplicação e inicia o emulador automaticamente, quando executar ou depurar o aplicativo.

Podemos especificar opções de inicialização do emulador na caixa de diálogo Run / Debug. Quando o emulador estiver a funcionar, podemos emitir comandos da consola, para alterar momentaneamente o seu estado como veremos posteriormente.

Para parar uma instância do emulador, basta fechar a janela do emulador.

4.10.3 Controlar o Emulador

Quando o emulador estiver a funcionar, podemos interagir com o dispositivo móvel emulado como se fosse um dispositivo móvel real, usa-se o ponteiro do rato para "tocar" o touchscreen e para "pressionar" as teclas do dispositivo simulado.

A figura 30 resume os mapeamentos entre as teclas e emulador e as teclas do seu teclado.

Emulated Device Key	Keyboard Key
Home	HOME
Menu (left softkey)	F2 *or* Page-up button

Star (right softkey)	Shift-F2 *or* Page Down
Back	ESC
Call/dial button	F3
Hangup/end call button	F4
Search	F5
Power button	F7
Audio volume up button	KEYPAD_PLUS, Ctrl-5
Audio volume down button	KEYPAD_MINUS, Ctrl-F6
Camera button	Ctrl-KEYPAD_5, Ctrl-F3
Switch to previous layout orientation (for example, portrait, landscape)	KEYPAD_7, Ctrl-F11
Switch to next layout orientation (for example, portrait, landscape)	KEYPAD_9, Ctrl-F12
Toggle cell networking on/off	F8
Toggle code profiling	F9 (only with `-trace` startup option)
Toggle fullscreen mode	Alt-Enter
Toggle trackball mode	F6
Enter trackball mode temporarily (while key is pressed)	Delete
DPad left/up/right/down	KEYPAD_4/8/6/2
DPad center click	KEYPAD_5
Onion alpha increase/decrease	KEYPAD_MULTIPLY(*) / KEYPAD_DIVIDE(/)

Figura 30. Tabela de teclas para controlar o AVD [8]

Note-se que, para usar as teclas do teclado, você deve primeiro desligar o NumLock no computador

4.10.4 Opções de Inicialização do Emulador

O emulador suporta uma grande variedade de opções que pode especificar aquando do seu lançamento para controlar sua aparência ou comportamento. Aqui está o uso de linha de comando para iniciar o emulador com opções:

```
emulator -avd <avd_name> [-<option> [<value>]] ... [-<qemu args>]
```

A tabela da figura 31 resume as opções disponíveis.

Category	Option	Description

Help	`-help`	Print a list of all emulator options.
	`-help-all`	Print help for all startup options.
	`-help-<option>`	Print help for a specific startup option.
	`-help-debug-tags`	Print a list of all tags for `-debug <tags>`.
	`-help-disk-images`	Print help for using emulator disk images.
	`-help-environment`	Print help for emulator environment variables.
	`-help-keys`	Print the current mapping of keys.
	`-help-keyset-file`	Print help for defining a custom key mappings file.
	`-help-virtual-device`	Print help for Android Virtual Device usage.
AVD	`-avd <avd_name>` or `@<avd_name>`	**Required**. Specifies the AVD to load for this emulator instance.
Disk Images	`-cache <filepath>`	Use <filepath> as the working cache partition image.
	`-data <filepath>`	Use <filepath> as the working user-data disk image.
	`-initdata <filepath>`	When resetting the user-data image (through `-wipe-data`), copy the contents of this file to the new user-data disk image. By default, the emulator copies the `<system>/userdata.img`.
	`-nocache`	Start the emulator without a cache partition.
	`-ramdisk <filepath>`	Use <filepath> as the ramdisk image.
	`-sdcard <filepath>`	Use <file> as the SD card image.
	`-wipe-data`	Reset the current user-data disk image (that is, the file specified by `-datadir` and `-data`, or the default file). The emulator deletes all data from the user data image file, then copies the contents of the file at `-inidata` data to the image file before starting.
Debug	`-debug <tags>`	Enable/disable debug messages for the specified debug tags.
	`-debug-<tag>`	Enable/disable debug messages for the specified debug tag.
	`-debug-no-<tag>`	Disable debug messages for the specified debug tag.
	`-logcat <logtags>`	Enable logcat output with given tags.
	`-shell`	Create a root shell console on the current terminal.
	`-shell-serial <device>`	Enable the root shell (as in `-shell` and specify the QEMU character device to use for communication with the shell.
	`-show-kernel <name>`	Display kernel messages.
	`-trace <name>`	Enable code profiling (press F9 to start), written to a specified file.
	`-verbose`	Enable verbose output.

Media	`-audio <backend>`	Use the specified audio backend.
	`-audio-in <backend>`	Use the specified audio-input backend.
	`-audio-out <backend>`	Use the specified audio-output backend.
	`-noaudio`	Disable audio support in the current emulator instance.
	`-radio <device>`	Redirect radio modem interface to a host character device.
	`-useaudio`	Enable audio support in the current emulator instance.
Network	`-dns-server <servers>`	Use the specified DNS server(s).
	`-http-proxy <proxy>`	Make all TCP connections through a specified HTTP/HTTPS proxy
	`-netdelay <delay>`	Set network latency emulation to <delay>.
	`-netfast`	Shortcut for `-netspeed full -netdelay none`
	`-netspeed <speed>`	Set network speed emulation to <speed>.
	`-port <port>`	Set the console port number for this emulator instance to `<port>`.
	`-report-console <socket>`	Report the assigned console port for this emulator instance to a remote third party before starting the emulation.
System	`-cpu-delay <delay>`	Slow down emulated CPU speed by <delay>
	`-gps <device>`	Redirect NMEA GPS to character device.
	`-nojni`	Disable JNI checks in the Dalvik runtime.
	`-qemu`	Pass arguments to qemu.
	`-qemu -h`	Display qemu help.
	`-radio <device>`	Redirect radio mode to the specified character device.
	`-timezone <timezone>`	Set the timezone for the emulated device to <timezone>, instead of the host's timezone.
	`-version`	Display the emulator's version number.
UI	`-dpi-device <dpi>`	Scale the resolution of the emulator to match the screen size of a physical device.
	`-no-boot-anim`	Disable the boot animation during emulator startup.
	`-no-window`	Disable the emulator's graphical window display.
	`-scale <scale>`	Scale the emulator window.
	`-raw-keys`	Disable Unicode keyboard reverse-mapping.
	`-noskin`	Don't use any emulator skin.

	`-keyset <file>`	Use the specified keyset file instead of the default.
	`-onion <image>`	Use overlay image over screen.
	`-onion-alpha <percent>`	Specify onion skin translucency value (as percent).
	`-onion-rotation <position>`	Specify onion skin rotation.
	`-skin <skinID>`	This emulator option is deprecated.
	`-skindir <dir>`	This emulator option is deprecated.

Figura 31. Tabela das opções de arranque do AVD [8]

4.10.5 Emulação de GPS

consola podemos fornecer comandos que lhe permita estabelecer a posição geográfica utilizados por um emulador de dispositivo emulado. Podemos usar o comando `geo` para enviar uma posição GPS simples para o emulador. O comando é:

```
geo <fix|nmea>
```

O comando `geo` suporta os subcomandos listados na tabela da figura 32.

Subcommand	Description	Comments
`fix <longitude> <latitude> [<altitude>]`	Send a simple GPS fix to the emulator instance.	Specify longitude and latitude in decimal degrees. Specify altitude in meters.
`nmea <sentence>`	Send an NMEA 0183 sentence to the emulated device, as if it were sent from an emulated GPS modem.	`<sentence>` must begin with '$GP'. Only '$GPGGA' and '$GPRCM' sentences are currently supported.

Figura 32. Tabela de comandos para a emulação de GPS [8]

Qualquer aplicação pode consultar o gestor de localização para obter a posição GPS atuais para o dispositivo emulado pelo telefone:

```
LocationManager.getLastKnownLocation ("gps")
```

4.10.6 Emulação de Velocidades de Rede

O emulador também permite simular várias taxas de transferência de rede. Podemos definir uma taxa de transferência na inicialização do emulador ou podemos usar a consola para alterar a taxa de forma dinâmica, enquanto a aplicação está funcionar no emulador.

Para definir a velocidade da rede na inicialização do emulador, use a opção `-netspeed` emulador com um valor `<speed>`, conforme referido acima. Aqui estão dois exemplos:

```
emulator                                -netspeed                                gsm
emulator -netspeed 14,4 80
```

Para fazer alterações dinâmicas para a velocidade da rede, enquanto o emulador está executando, ligue para o console e usar o `netspeed` comando com um apoio `<speed>` valor da tabela da figura 33.

```
network speed 14,4 80
```

O formato de rede `<speed>` é um dos seguintes (os números são em Kib/s):

Value	Description	Comments
`gsm`	GSM/CSD	(Up: 14.4, down: 14.4)
`hscsd`	HSCSD	(Up: 14.4, down: 43.2)
`gprs`	GPRS	(Up: 40.0, down: 80.0)
`edge`	EDGE/EGPRS	(Up: 118.4, down: 236.8)
`umts`	UMTS/3G	(Up: 128.0, down: 1920.0)
`hsdpa`	HSDPA	(Up: 348.0, down: 14400.0)
`full`	no limit	(Up: 0.0, down: 0.0)
`<num>`	Set an exact rate used for both upload and download.	
`<up>:<down>`	Set exact rates for upload and download separately.	

Figura 33. Tabela de comandos para a emulação de direntes velocidades de rede [8]

4.10.7 Limitações do Emulador

As limitações do emulador incluem:

- Não há suporte para fazer ou receber chamadas para um telefone real.
- Não há suporte para conexões USB
- Não há suporte para câmara / captura de vídeo (de entrada).
- Não há suporte para headphones.
- Não há suporte para determinar o estado ligado

- Não há suporte para determinar o nível de carga da bateria e AC estado de carga
- Não há suporte para cartão SD determinar inserir / ejectar
- Não há suporte para Bluetooth

4.11 Suporte de Múltiplos Ecrãs

O Android funciona numa grande variedade de dispositivos que oferecem diferentes tamanhos de ecrã e densidades. Para criar aplicações, o sistema Android fornece um ambiente de desenvolvimento consistente em dispositivos e o sistema realiza a maioria do trabalho para ajustar cada interface de utilizador da aplicação para a ecrã em que é exibido. Ao mesmo tempo, o sistema fornece APIs que permitem que você controle a UI da aplicação em tamanhos de ecrã e densidades específicas, a fim de modificar e optimizar o seu design de interface do utilizador para configurações de ecrã diferente. Por exemplo, podemos querer uma UI para os tablets diferente do design para telefones.

Embora o sistema execute o redimensionamento para tornar a aplicação a trabalhar em diferentes ecrãs, deve-se fazer um esforço para optimizar a aplicação para diferentes tamanhos de ecrã e densidades. Ao fazê-lo, maximiza-se a experiência do utilizador para todos os dispositivos e os utilizadores acreditam que foi realmente concebido para o *seu* dispositivo, ao invés de simplesmente esticar para caber no ecrã.

Pode-se criar um aplicativo que exibe correctamente e fornece uma experiência de utilizador optimizada em todas as configurações de ecrã suportada, usando um único ficheiro `.apk`.

4.11.1 Visão Geral dos Ecrãs

Esta seção fornece uma visão geral de apoio Android para múltiplos ecrãs, incluindo: uma introdução aos conceitos e termos usados neste documento e na API, um resumo das configurações de ecrã que o sistema suporta, e uma visão geral da API e ecrã subjacente compatibilidade características.

4.11.2 Termos e Conceitos dos Ecrãs

Tamanho do ecrã
Tamanho físico real, medida diagonal do ecrã (em polegadas).

Para simplificar, os grupos Android todos os tamanhos de ecrã real em quatro tamanhos generalizada: pequeno, normal, grande e extra grande.

Densidade de ecrã
A quantidade de pixels dentro de uma área física do ecrã, geralmente referida como dpi (dots per inch). Por exemplo, um ecrã de densidade "baixa" tem menos pixels dentro de uma determinada área física, comparado a um "normal" ou ecrã de densidade "alto".

Para simplificar, os grupos Android todas as densidades ecrã real em quatro densidades generalizada: alta, baixa, média, alta e extra.

Orientação
A orientação da ecrã do ponto do de vista do utilizador. Este é paisagem ou retrato, o que significa que a relação de aspecto do ecrã é ou largura ou altura, respectivamente. Estar ciente de que não só diferentes dispositivos operam em diferentes orientações, por omissão, mas a orientação também pode mudar durante o tempo de execução quando o utilizador gira o dispositivo.

Resolução
O número total de pixels em um ecrã físico. Ao adicionar o suporte para múltiplos ecrãs, as aplicações não trabalham directamente com a resolução, as aplicações devem-se preocupar apenas com o tamanho da ecrã e densidade, conforme especificado pelo tamanho generalizada e grupos de densidade.

Densidade-independente pixel (dp)
Uma unidade de pixel virtual que se deve usar para definir o *layout* da interface do utilizador, para expressar dimensões do *layout* ou a posição de uma forma independente da densidade.

A densidade de pixels independente é equivalente a um pixel física num ecrã de 160 dpi, que é a densidade de referência assumido pelo sistema para um ecrã de densidade "médio". Em tempo de execução, o sistema transparente lida com qualquer escala das unidades dp, se necessário, com base na densidade real do ecrã em uso. A conversão de unidades para dp pixels do ecrã é simples: `px = dp * (dpi / 160)`. Por exemplo, num ecrã de 240 dpi, 1 dp equivale a 1,5 pixels físicos. Deve-se usar sempre as unidades dp na definição de interface do utilizador da aplicação, para garantir a exibição correta da interface do utilizador em ecrãs com diferentes densidades.

4.11.3 Gama de Ecrãs Suportados

O Android oferece suporte para vários tamanhos de ecrã e densidades.

Pode-se usar os recursos do sistema para optimizar a interface do utilizador da aplicação para cada configuração de ecrã e garantir que a aplicação não só adequa correctamente como proporciona a melhor experiência possível em cada ecrã.

O Android divide a gama de tamanhos de ecrã real e densidades em:

- Um conjunto de quatro **tamanhos** generalizada: *small, normal, large* e *xlarge*
- Um conjunto de quatro **densidades** generalizada: ldpi (low dots per inch), mdpi (medium dots per inch), hdpi (high dots per inch), e xhdpi (extra large dots per inch)

Cada tamanho ou densidade generalizada abrange uma gama de tamanhos de ecrã real ou densidade.

A Figura 34 ilustra como diferentes tamanhos e densidades são classificados em diferentes dimensões e grupos de densidade.

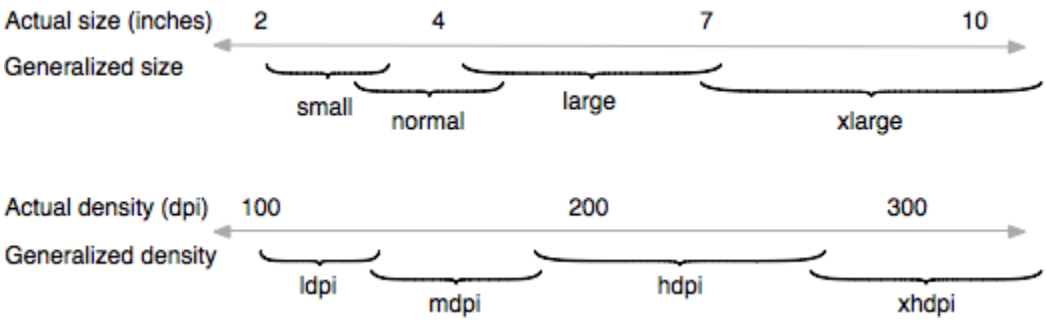


Figura 34. Diagrama de tamanhos e densidades de ecrãs [8]

Para optimizar a interface do utilizador da aplicação para os diferentes tamanhos de ecrã e densidades, pode-se fornecer recursos alternativos para qualquer dos tamanhos e densidades generalizada. Normalmente, deve-se fornecer *layouts* alternativos para alguns dos diferentes tamanhos de ecrã e imagens *bitmap* alternativa para densidades de ecrã diferente. Na execução, o sistema utiliza os recursos apropriados tamanho ou densidade para a aplicação, com base no tamanho generalizada ou densidade do ecrã do dispositivo actual.

Não é preciso fornecer recursos alternativos para cada combinação de tamanho de ecrã e densidade. O sistema oferece recursos de compatibilidade robusta que pode fazer a maioria do trabalho de prestar a aplicação em qualquer ecrã do dispositivo

4.11.4 Independência da Densidade

A aplicação consegue "independência da densidade" quando se preserva o tamanho físico (do ponto de vista do utilizador) dos elementos de interface do utilizador ao ser exibida em ecrãs com diferentes densidades.

Manter a independência de densidade é importante porque, sem ele, um elemento de interface do utilizador (como um botão) aparece fisicamente maior num ecrã de baixa densidade e menor num ecrã de alta densidade.

As figuras 35 e 36 mostram a diferença entre uma aplicação quando ela não fornece independência de densidade e quando isso acontece, respectivamente.

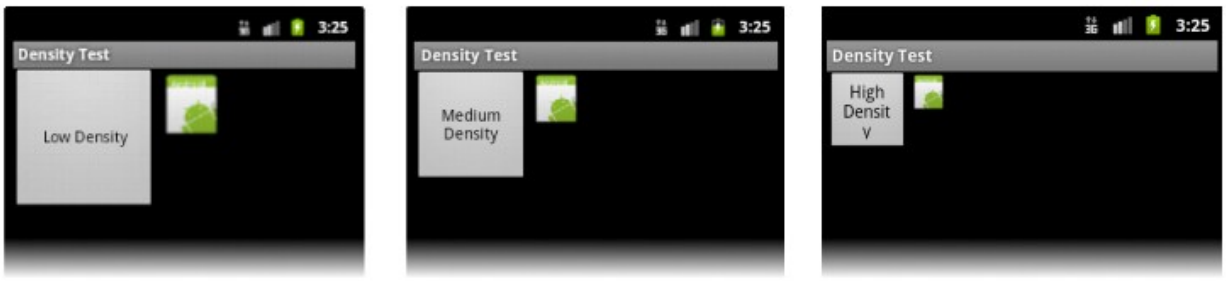


Figura 35. Imagem de aplicação em ecrãs de diferentes densidades sem independência da densidade [8]

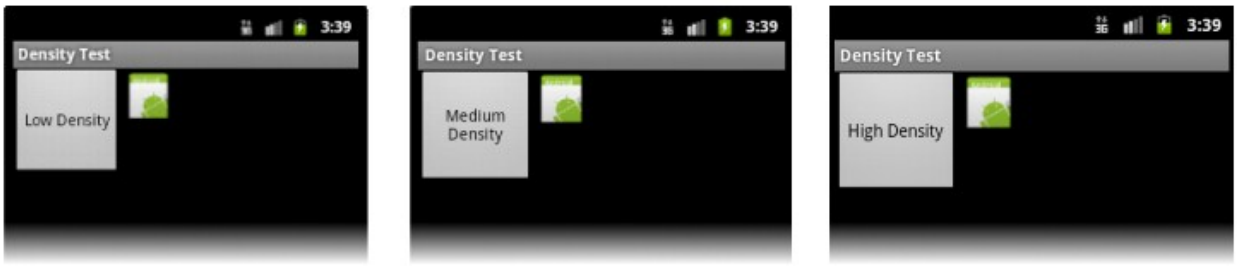


Figura 36. Imagem de aplicação em ecrãs de diferentes densidades com independência da densidade [8]

O sistema Android ajuda a aplicação a alcançar a independência da densidade de duas formas:

- O sistema de escalas de unidades dp conforme apropriado para a densidade do ecrã actual
- As escalas do sistema de recursos *drawable* para o tamanho adequado, com base na densidade do ecrã actual.

Na figura 35 a exibição de texto e bitmap *drawable* têm dimensões especificadas em pixels (`px` unidades) assim que os elementos são fisicamente maiores em um ecrã de baixa densidade e menor em um ecrã de alta densidade.

Na figura 36 as dimensões do *layout* são especificados na densidade independentes de pixels (unidades `dp`). Para os ecrãs de baixa densidade e alta densidade, as escalas a densidade independente do sistema de valores de pixel para baixo e para cima, respectivamente, para caber no ecrã.

Na maioria dos casos pode-se garantir a independência da densidade simplesmente especificando todos os valores de dimensão *layout* na densidade independentes de pixels (unidades `dp`) ou com `"wrap_content"` , conforme o caso. Usando o sistema *drawables* a fim de exibir no tamanho adequado, com base no factor de escala apropriada para a densidade do ecrã actual.

No entanto, o dimensionamento *bitmap* pode resultar em *bitmaps* desfocados ou pixelizados,. Para evitar esses artefactos, deve-se fornecer recursos de bitmap alternativa para diferentes densidades. Por exemplo, deve-se fornecer *bitmaps* de alta resolução para de ecrãs de alta densidade e o sistema utiliza-os em vez de redimensionar o bitmap projectado para ecrãs de média densidade.

4.11.5 Suportar para Múltiplos Ecrãs

O Android faz a maioria do trabalho para tornar a aplicação correcta em cada configuração de ecrã, *layouts* de escala para se ajustar ao tamanho da ecrã / densidade e *drawables bitmap* de escala para a densidade da ecrã, conforme o caso. Para utilizar bem com as configurações de ecrã diferente, no entanto, deve-se também:

- **Fornecer *layouts* diferentes para diferentes tamanhos de ecrã**

Por omissão, o Android redimensiona o *layout* da aplicação para caber no ecrã do dispositivo actual. Na maioria dos casos, isso funciona bem. Noutros casos, a interface do utilizador pode não parecer tão bom e pode precisar de ajustes para diferentes tamanhos de ecrã. Por exemplo, num ecrã maior, pode-se

querer ajustar a posição e o tamanho de alguns elementos para aproveitar o espaço na ecrã adicional, ou num ecrã menor, pode ser necessário ajustar os tamanhos para que tudo possa caber na ecrã.

- **Fornecer *drawables bitmap* diferente para diferentes densidades de ecrã**

Por omissão, o Android redimensiona os *drawables bitmap* (ficheiros `.png` , `.jpg` e `.gif`) e *Nine Patch-drawables* (ficheiros `.9.png`) para que eles transformem o tamanho físico apropriado em cada dispositivo.

Para garantir que os *bitmaps* se vejam melhor, deve-se incluir versões alternativas em diferentes resoluções de ecrã e diferentes densidades.

Os qualificadores de configuração que se pode usar para a densidade específicos são `ldpi` (baixo), `mdpi` (médio), `hdpi` (alta), e `xhdpi` (extra alta). Por exemplo, *bitmaps* para ecrãs de alta densidade devem estar em `drawable-hdpi/`.

4.11.6 Qualificadores para Diferentes Ecrãs

O Android suporta qualificadores para diversas configurações que permitem controlar a forma como o sistema selecciona os recursos alternativos com base nas características da ecrã do dispositivo actual. Um qualificador é uma *string* que se pode acrescentar a um directório de recursos no projecto Android e especifica a configuração para o qual os recursos são projectados.

Screen characteristic	Qualifier	Description
Size	`small`	Resources for *small* size screens.
	`normal`	Resources for *normal* size screens. (This is the baseline size.)
	`large`	Resources for *large* size screens.
	`xlarge`	Resources for *extra large* size screens.
Density	`ldpi`	Resources for low-density (*ldpi*) screens (~120 dpi).
	`mdpi`	Resources for medium-density (*mdpi*) screens (~160dpi). (This is the baseline density.)
	`hdpi`	Resources for high-density (*hdpi*) screens (~240dpi).
	`xhdpi`	Resources for extra high-density (*xhdpi*) screens (~320dpi).
	`nodpi`	Resources for all densities. These are density-independent resources. The system does not scale resources tagged with this qualifier, regardless of the current screen's density.
Orientation	`land`	Resources for screens in the landscape orientation (wide aspect ratio).
	`port`	Resources for screens in the portrait orientation (tall aspect ratio).
Aspect ratio	`long`	Resources for screens that have a significantly taller or wider aspect ratio (when in portrait or landscape orientation, respectively) than the baseline screen configuration.

	notlong	Resources for use screens that have an aspect ratio that is similar to the baseline screen configuration.

Figura 37. Tabela de qualificadores de ecrãs [8]

A seguir temos alguns exemplos de directórios de recursos de uma aplicação que fornece designs layout diferentes para diferentes tamanhos de ecrã e diferentes bitmaps para ecrãs de diferentes densidade.

```
res/layout/my_layout.xml              // layout for normal screen size ("default")
res/layout-small/my_layout.xml        // layout for small screen size
res/layout-large/my_layout.xml        // layout for large screen size
res/layout-xlarge/my_layout.xml       // layout for extra large screen size
res/layout-xlarge-land/my_layout.xml  // layout for extra large in landscape orientation

res/drawable-mdpi/my_icon.png         // bitmap for medium density
res/drawable-hdpi/my_icon.png         // bitmap for high density
res/drawable-xhdpi/my_icon.png        // bitmap for extra high density
```

4.11.7 Teste da Aplicação em Múltiplos Ecrãs

Antes de publicar a aplicação, deve-se testar em todos os tamanhos e densidades de ecrã.

Para configurar um ambiente para testar o suporte de aplicação no ecrã, deve-se criar uma série de AVD, usando configurações de ecrã que emulam os tamanhos de ecrã e densidades que a aplicação poderá correr. Para isso, pode-se usar o Android SDK e AVD Manager para criar o AVD.

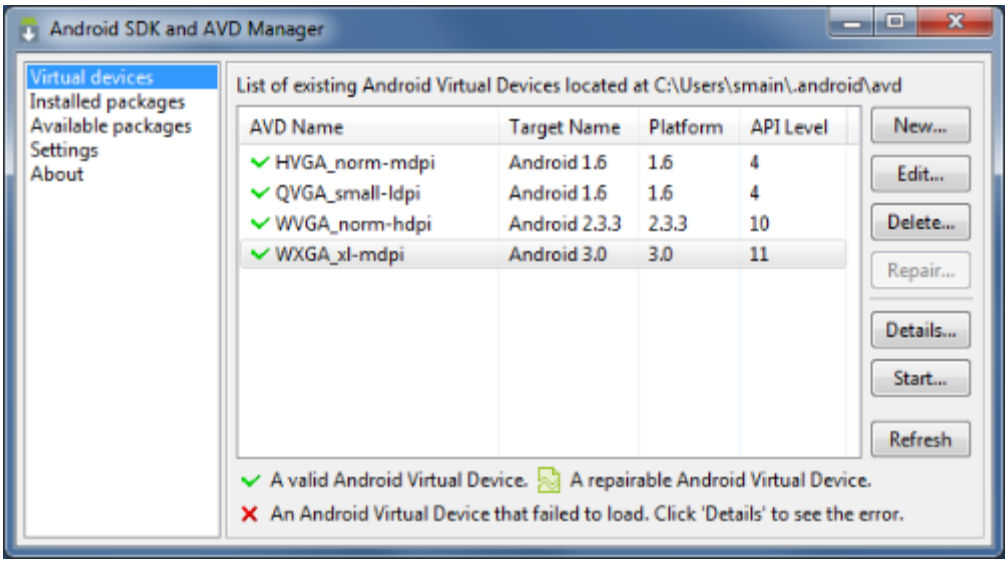


Figura 38. Janela em Eclipse com diferentes AVD para diferentes ecrãs

Na tabela seguinte podemos ver várias configurações de ecrã disponíveis.

Tabela 5. Tabela de comparação de ecrãs ao nível da densidade e tamanho [8]

	Low density (120), *ldpi*	Medium density (160), *mdpi*	High density (240), *hdpi*	Extra high density (320), *xhdpi*

***Small* screen**	**QVGA (240x320)**		480x640	
***Normal* screen**	**WQVGA400 (240x400)** **WQVGA432 (240x432)**	**HVGA (320x480)**	**WVGA800 (480x800)** **WVGA854 (480x854)** 600x1024	640x960
***Large* screen**	**WVGA800** (480x800)** **WVGA854** (480x854)**	**WVGA800* (480x800)** **WVGA854* (480x854)** 600x1024		
***Extra Large* screen**	1024x600	**WXGA (1280x800)**† 1024x768 1280x768	1536x1152 1920x1152 1920x1200	2048x1536 2560x1536 2560x1600

* To emulate this configuration, specify a custom density of 160 when creating an AVD that uses a WVGA800 or WVGA854 skin.
** To emulate this configuration, specify a custom density of 120 when creating an AVD that uses a WVGA800 or WVGA854 skin.
† This skin is available with the Android 3.0 platform

Também podemos testar a aplicação num emulador configurado para executar num tamanho físico que se aproxima um dispositivo real. Isto torna muito mais fácil de comparar os resultados em vários tamanhos e densidades. Para fazer isso é necessário saber a densidade aproximada, em dpi, do monitor do computador

Quando se inicia um AVD do Android SDK e AVD Manager, pode-se especificar o tamanho do ecrã para o emulador e o dpi do monitor nas opções de inicialização, como vemos na figura 39:

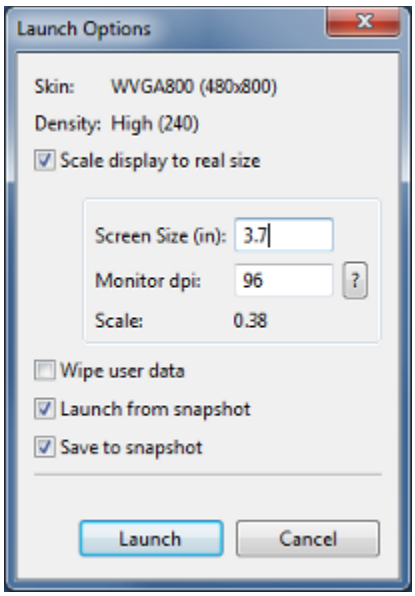


Figura 39. Janela em Eclipse de opções de densidade e tamanho para o AVD

Tamanho e densidade de opções que se pode configurar, quando se inicia uma AVD do Android SDK e AVD Manager.

4.11.8 Estado Actual de Tamanhos e Densidades de Ecrãs

Relativamente aos tamanhos e densidades de ecrã dos dispositivos activos actualmente podemos ver no gráfico seguinte os dispositivos Android que acederam ao Android Market num período de 7 dias até ao dia 1 de Junho de 2011.

Nota: Esta informação é baseada no número de dispositivos Android que acederam ao Android Market num período de 7 dias até ao dia 1 de Junho de 2011.

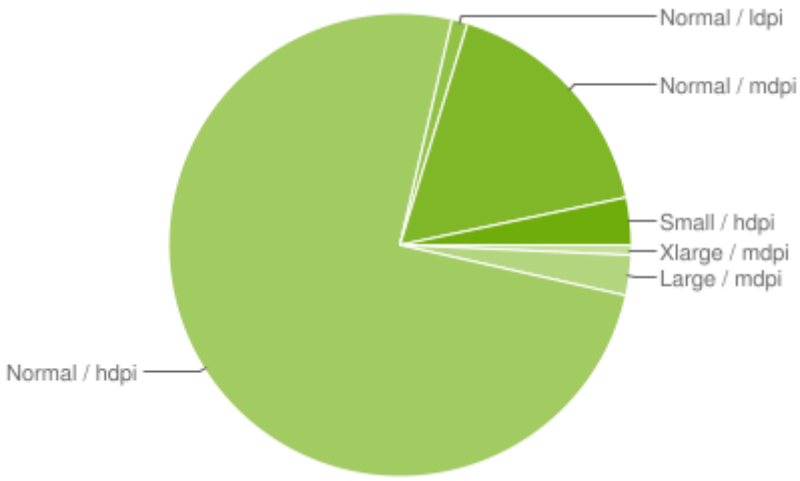


Figura 40. Gráfico de ecrãs dos dispositivos móveis no acesso ao Android Market [8]

De notar que hoje em dia, a grande maioria dos dispositivos estão a usar ecrãs de tamanho *Normal* e densidade *hdpi* e mais de 90% usam ecrã de tamanho *Normal* e densidades *mdpi ou hdpi.*

4.12 Áudio e Vídeo

A plataforma Android tem embutido um sistema de codificação / descodificação para uma variedade de tipos de multimédia comuns para se integrar vídeo, áudio e imagens nas aplicações.

O Android permite reproduzir áudio e vídeo de vários tipos de fontes de dados. Pode reproduzir áudio ou vídeo a partir de ficheiros armazenados na pasta "*raw*" da aplicação, a partir de ficheiros autónomos no sistema de ficheiros ou de um *stream* através de uma conexão de rede. Para reproduzir o áudio ou vídeo usa-se a Classe *MediaPlayer*

A plataforma também permite gravar áudio e vídeo, quando suportado pelo *hardware* do dispositivo móvel. Para gravar áudio ou vídeo usa-se a Classe MediaRecorder.

O emulador não tem *hardware* para captura de áudio ou vídeo.

4.12.1 Reprodução a Partir de um Recurso Local

Para a reprodução de áudio ou vídeo a partir de um ficheiro local deve:

1. Por o ficheiro na pasta `res/raw` do seu projecto, onde o plugin do Eclipse vai encontrá-lo e torná-lo em um recurso que pode ser referenciado a partir de sua classe R
2. Criar uma instância do `MediaPlayer` , fazendo referência a esse recurso usando `MediaPlayer.create` , e depois chamar `start()` na instância:

```
     MediaPlayer mp = MediaPlayer.create (contexto,
R.raw.sound_file_1);
     mp.start ();
```

Para parar a reprodução, use `stop()` . Se você quiser mais tarde fazer replay, deve fazer `reset()` e `prepare()` antes de chamar `start()` novamente.

Para parar momentaneamente a reprodução, use `pause()` . Para retomar a reprodução a partir de onde você parou use `start()` .

4.12.2 Reprodução a Partir de um Ficheiro ou Stream

Você pode reproduzir ficheiros de mídia do sistema de ficheiros ou uma URL web:

1. Criar uma instância do `MediaPlayer` usando `new`
2. Chame `setDataSource()` com uma String contendo o caminho (ficheiro local ou URL) para o ficheiro que você quer reproduzir
3. Primeiro `prepare()` , em seguida, `start()` na instância:

```
     MediaPlayer mp = MediaPlayer new ();
     mp.setDataSource (path_to_file);
     mp.prepare ();
     mp.start ();
```

`stop()` e `pause()` funcionam da mesma forma como referido acima.

4.12.3 Protocolos de Rede

Os seguintes protocolos de rede são suportados para reprodução de áudio e vídeo:

- Real Time Streaming Protocol (RTSP) (RTP, SDP)
- HTTP streaming
- HTTP live streaming (Android 3.0 e acima)

O protocolo HTTPS ainda não é suportado.

4.12.4 Formatos de Multimédia

A tabela da figura 41 descreve o suporte ao formato de multimédia incorporado na plataforma Android.

Type	Format / Codec	Encoder	Decoder	Details	Supported File Type(s) / Container Formats
Audio	AAC LC/LTP	•	•	Mono/Stereo content in any combination of standard bit rates up to 160 kbps and sampling rates from 8 to 48kHz	3GPP (.3gp), and MPEG-4 (.mp4, .m4a). ADTS raw AAC (.aac, decode only, ADIF not supported, Android 3.1+).
	HE-AACv1 (AAC+)		•		
	HE-AACv2 (enhanced AAC+)		•		
	AMR-NB	•	•	4.75 to 12.2 Kib/s amostrado a 8 kHz	3GPP (.3gp)
	AMR-WB	•	•	9 rates from 6.60 Kib/s to 23.85 Kib/s amostrado a 16 kHz	3GPP (.3gp)
	FLAC		• (Android 3.1+)	Mono/Stereo (no multichannel). Sample rates up to 48 kHz (but up to 44,1 kHz is recommended on devices with 44,1 kHz output, as the 48 to 44,1 kHz downsampler does not include a low-pass filter). 16-bit recommended; no dither applied for 24-bit.	FLAC (.flac) only
	MP3		•	Mono/Stereo 8-320 Kib/s constant (CBR) or variable bit-rate (VBR)	MP3 (.mp3)
	MIDI		•	MIDI Type 0 and 1. DLS Version 1 and 2. XMF and Mobile XMF. Support for ringtone formats RTTTL/RTX, OTA, and iMelody	Type 0 and 1 (.mid, .xmf, .mxmf). Also RTTTL/RTX (.rtttl, .rtx), OTA (.ota), and iMelody (.imy)
	Ogg Vorbis		•		Ogg (.ogg)
	PCM/WAVE		•	8- and 16-bit linear PCM (rates up to limit of hardware)	WAVE (.wav)
Image	JPEG	•	•	Base+progressive	JPEG (.jpg)

	GIF		•		GIF (.gif)
	PNG	•	•		PNG (.png)
	BMP		•		BMP (.bmp)
Video	H.263	•	•		3GPP (.3gp) and MPEG-4 (.mp4)
	H.264 AVC	• (Android 3.0+)	•	Baseline Profile (BP)	3GPP (.3gp) and MPEG-4 (.mp4). MPEG-TS (.ts, AAC audio only, not seekable, Android 3.0+)
	MPEG-4 SP		•		3GPP (.3gp)
	VP8		• (Android 2.3.3+)		WebM (.webm)

Figura 41. Tabela de formatos multimédia utilizados no Android [8]

4.12.5 Codificações de Vídeo

A tabela 6 lista exemplos de perfis de codificação de vídeo e parâmetros que o Android suporta para reprodução. Além destes parâmetros de codificação, os perfis de um dispositivo de gravação de vídeo disponíveis podem ser usados como uma *proxy* para as capacidades de reprodução de multimédia. Estes perfis podem ser inspeccionadas usando a classe `CamcorderProfile`, que está disponível desde o API nível 8.

Tabela 6. Tabela de codificações de vídeo utilizados no Android [8]

	Lower quality	Higher quality
Video codec	H.264 Baseline Profile	H.264 Baseline Profile
Video resolution	176 x 144 px	480 x 360 px
Video frame rate	12 f/s	30 f/s
Video bitrate	56 Kib/s	500 Kib/s
Audio codec	AAC-LC	AAC-LC
Audio channels	1 (mono)	2 (stereo)
Audio bitrate	24 Kib/s	128 Kib/s

4.13 Conclusões

O Android é um sistema *open source* da Google concebido para dispositivos móveis, quer sejam s*martphones* ou t*ablets*.

Esta área está actualmente com muita dinâmica de mercado, tornando-se numa área importante de desenvolvimento.

Como foi feita uma longa caminhada na aprendizagem nesta plataforma, criou-se esta documentação que descreve desde a instalação das ferramentas necessárias, passando pelos conceitos mais importante da plataforma e, por fim, referindo alguns recursos disponíveis mais vocacionados para o tema desta dissertação.

Este sistema de desenvolvimento tem uma muito boa documentação *online.* No entanto, parece que o emulador (AVD), embora extremamente útil e interessante, ainda terá que evoluir um pouco mais quer a nível de desempenho, quer a nível de suporte de alguns recursos.

Por fim espera-se que esta documentação seja útil para futuros estudos desta plataforma.

5. Arquitectura e Implementação

Neste capítulo são abordados os aspectos práticos do trabalho, ou seja, desde as decisões tomadas após um estudo inicial até aos resultados obtidos por recurso a várias experiências práticas.
No início do capítulo são descritos vários cenários possíveis que podem ocorrer e que seria interessante analisar e testar a eficiência do sistema.
De seguida é apresentada a informação de contexto que considero relevante o Agente gere.
Apresenta-se no anexo A a listagem de classes da API do Android que utilizadas na implementação.
Por fim é demonstrada a utilização das referidas classes com código de pequenas aplicações e capturas de ecrã das mesmas quando executadas no emulador.

5.1 Cenários

A estrutura inicialmente definida é uma estrutura simples em que o equipamento móvel em Android irá pedir um vídeo a um servidor HTTP e enviando a sua informação de contexto através de um agente.
Temos também um servidor HTTP para emissão de vídeo que possui várias versões do mesmo vídeo, mas com características distintas para adaptar o vídeo disponível ao perfil indicado pela informação de contexto recebida.
A situação é dinâmica, isto é, ao longo da visualização do vídeo, caso o agente de contexto encontre alguma alteração relevante, envia essa informação ao servidor que por sua vez irá novamente adaptar o vídeo à nova realidade.

De seguida, apresenta-se um conjunto de cenários possíveis que poderão servir para desenvolver e analisar o sistema.

Cenário 1

Temos um rapaz muito activo com um equipamento móvel em Android de última geração, i.e., com boas características de *hardware*. Ele encontra-se num escritório a trabalhar. Faz uma pausa e pretende ver um determinado vídeo no seu PDA. Como ele se encontra registado no sistema é disponibilizado a versão na qualidade máxima. Passados alguns instantes ele desloca-se dentro do edifício, sempre a observar o vídeo, e entra dentro do

elevador – local em que o nível de sinal WiFi diminui substancialmente. Nesse momento a resolução do vídeo baixa para manter a sua fluidez. Quando sai do elevador, a largura de faixa disponível sobe novamente e, então, a resolução do vídeo aumenta.

Cenário 2

Uma senhora que no perfil indicou que as suas línguas preferidas são apenas português e francês, encontra-se a utilizar o seu PDA e pediu para visualizar um vídeo. O vídeo requerido encontra-se na versão inglesa. Devido ao seu perfil, é enviada a versão legendada em português. Ao fim de algum tempo o nível de bateria baixa e o sistema adapta-se baixando a resolução espacial/temporal do vídeo para não comprometer o funcionamento do PDA.

Cenário 3

Um visitante entra no edifício e, enquanto aguarda pela pessoa que o vai receber, resolve ir ao bar que tem música ambiente e um elevado nível de ruído. Ele senta-se, utiliza o seu equipamento Android. Verifica que está disponível um vídeo na rede e faz a ligação. Como o nível de ruído é alto, o sistema opta por colocar a versão legendada para que o utilizador tenha a possibilidade de perceber totalmente o que está a visualizar. Passado algum tempo chega o amigo e o nosso utilizador fala-lhe do vídeo que visualizou. Este último pega no seu PDA, que já não é tão actual e tem um ecrã com uma resolução baixa, faz a visualização do vídeo, que lhe é apresentado na versão adequada para as características do seu equipamento.

Os cenários desenvolvidos permitiram identificar a informação de contexto que seria relevante considerar de forma a ser possível atingir os objectivos propostos.

5.2 Informação Contextual

A informação de contexto de interesse para o sistema é:

Contexto computacional

- CPU
- Tamanho do ecrã (em *pixels*)
- Tipo de ecrã (número de cores)
- Suporte de recursos (WAP e Java)
- Versão do *browser Web*
- Memória máxima para plataformas WAP e Java MIDlets
- Conectividade de rede
- Largura de faixa de comunicação

Contexto do utilizador (Perfil do utilizador)

- Nome

Sexo
Idade
Estado civil
Morada
Línguas preferidas
Actividade profissional
Áreas de interesse
Nível de experiência com equipamentos móveis
Dificuldades de audição ou de visão

Contexto físico

Temperatura
Iluminação (dentro/fora, bolso/mesa)
Níveis de ruído
Acelerómetro (gestos, movimento, orientação)

Contexto temporal

Data /Hora

Como já foi referido, o esquema a usar para representação destes metadados será o **UaProf**.

5.3 Implementação de Aplicações de Testes

Como implementação foram desenvolvidos alguns módulos na plataforma Android com intuito de serem utilizados no Agente de contexto.
Foi desenvolvido um módulo complexo que permite recolher variados tipos de informação de contexto relacionados com o *hardware* do dispositivo. Esse módulo utiliza as classes Build, Context, ViewDisplay e DisplayManager.
Para testes de diferentes resoluções e densidades de ecrã foi desenvolvido um módulo que é um *slideshow* utilizando a classe widget.ImageView.
Criou-se um módulo para recolher o estado das ligações GSM e WiFI utilizando as classes WifiInfo, WifiManager, NetworkInfo e WifiInfo.
Foi ainda testado um Mediaplayer utilizando a classe MediaPlayer para visualização de vídeo por streaming com vista a futuros testes da eficiência do Agente de contexto.
No anexo B apresenta-se a listagem dos módulos criados.
Seguidamente apresentam-se as capturas de ecrã dos testes em emulador (AVD) dos módulos referidos.

5.3.1 Contexto do Hardware

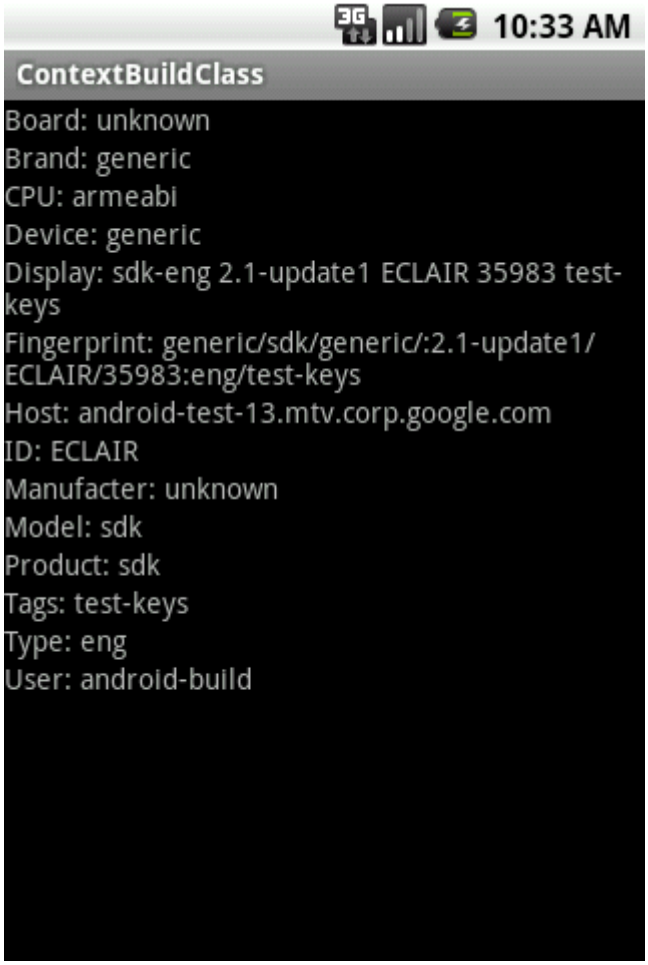


Figura 42. Captura de ecrã do AVD da aplicação ContextBuildClass

5.3.2 Contexto - diversos

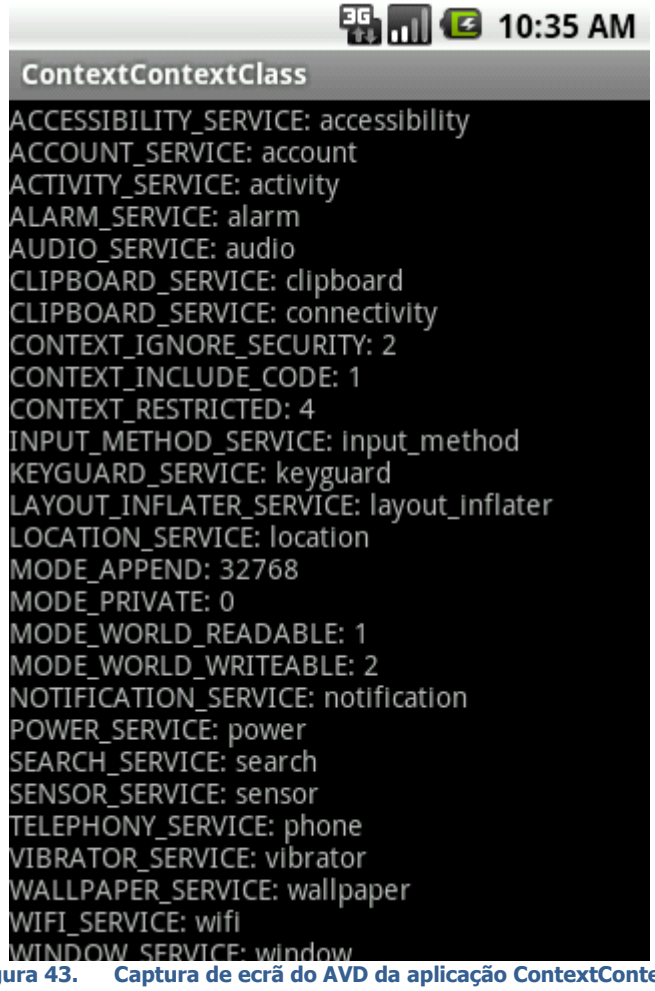


Figura 43. Captura de ecrã do AVD da aplicação ContextContextClass

5.3.3 Contexto de Resolução e Densidade do Ecrã

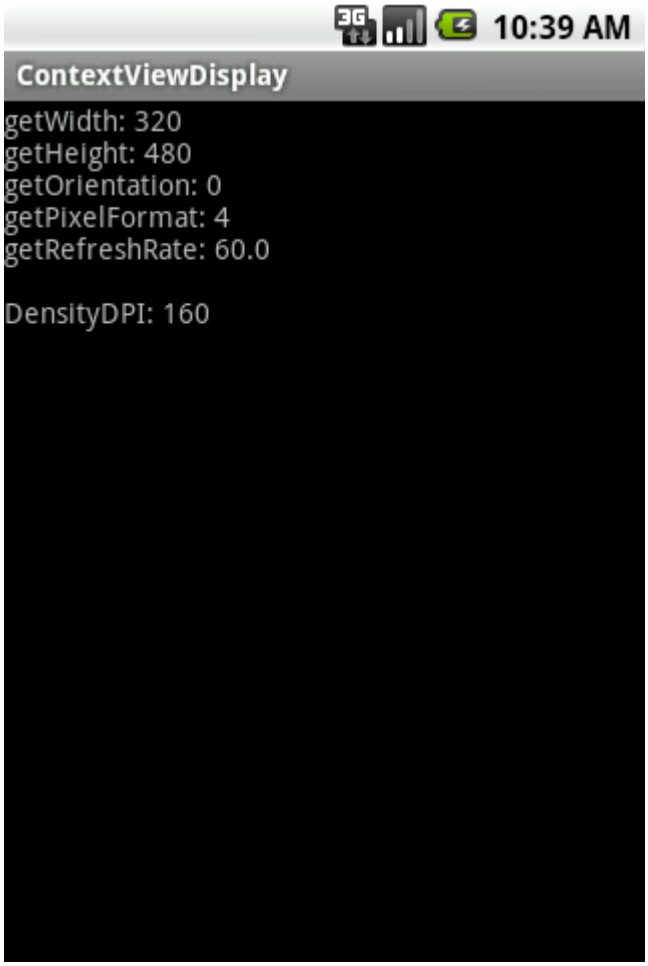


Figura 44. Captura de ecrã do AVD da aplicação ContextViewDisplay

5.3.4 Slideshow para Diferentes Resoluções de Densidades de Ecrã

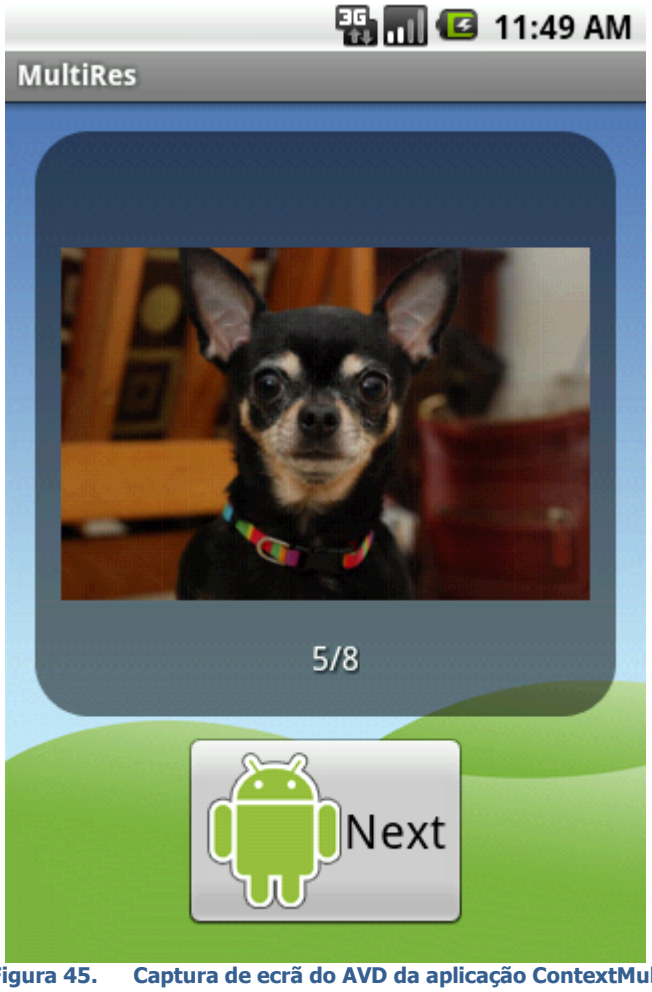


Figura 45. Captura de ecrã do AVD da aplicação ContextMultiRes

5.3.5 VideoPlayer

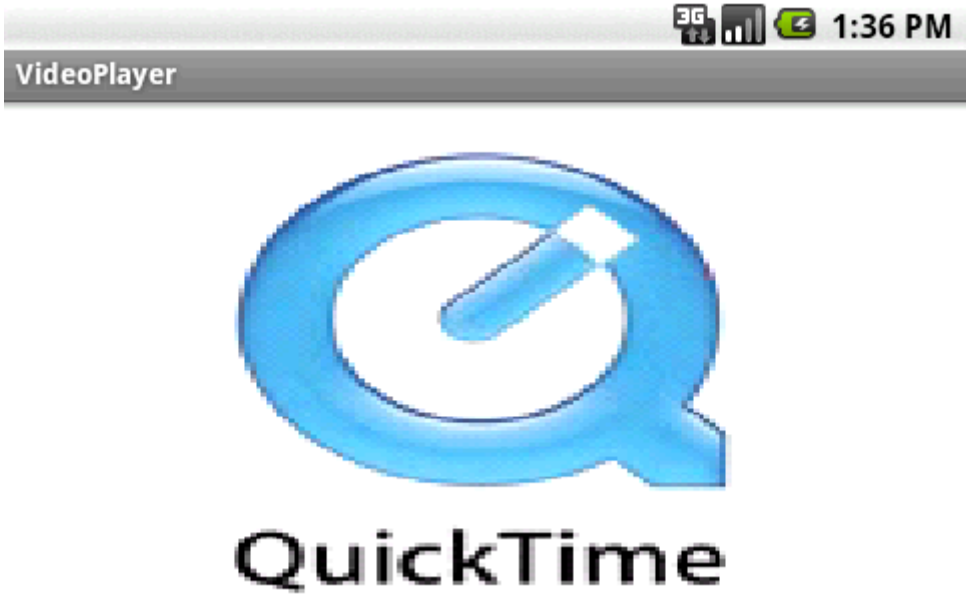


Figura 46. Captura de ecrã do AVD da aplicação ContextVideoPlayer

5.3.6 Estado das Ligações GSM e Wifi

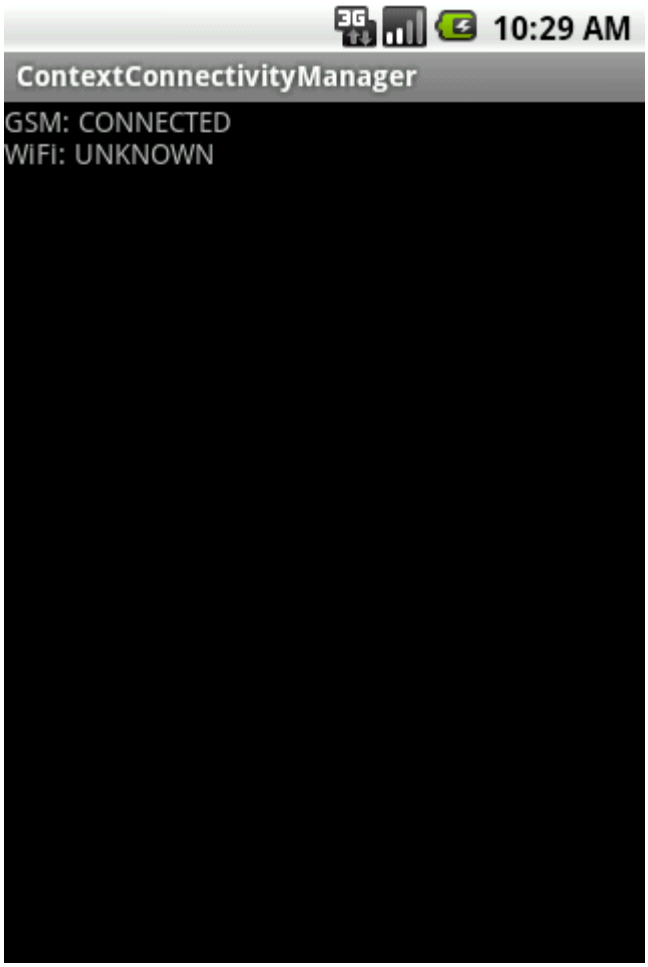


Figura 47. Captura de ecrã do AVD da aplicação ContextConnectivityManager

5.3.7 Contexto do GSM e Wifi

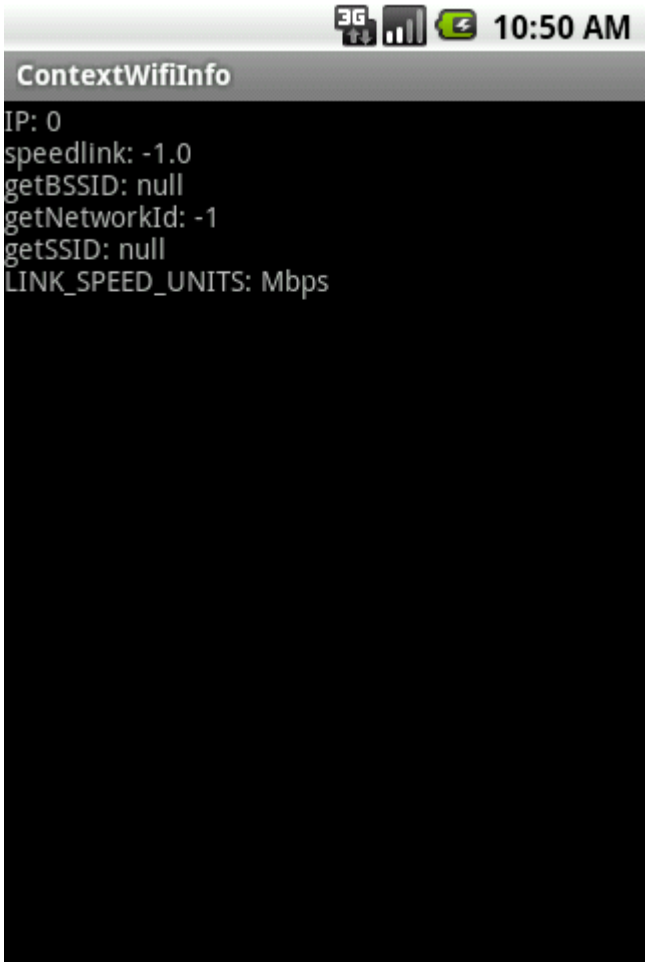


Figura 48. Captura de ecrã do AVD da aplicação ContexWifiInfo

5.3.8 ConnectivityDemo

Figura 49. Captura de ecrã do AVD da aplicação ContextConnectivityDemo

5.4 Conclusões

As rotinas listadas acima são de enorme importância visto ser a base da criação da informação para o Agente de Contexto.

Além das rotinas demonstradas, fez-se ainda estudos sobre a localização (ver classe: android.location) e sobre a câmara de vídeo. No entanto, não foi se conseguiu que o emulador reconhecesse a *webcam*. Houve também dificuldades na emulação do Global Navigation Satellite Systems (GNSS). No final consegui-se, utilizando a ferramenta DDMS/Emulator Control do Eclipse que simula uma localização, não sendo contudo possível incorporar aqui essa demonstração.

Foram ainda estudados os formatos de multimédia suportados (áudio, imagem e vídeo) e os protocolos de rede suportados (RTSP (RTP, SDP), HTTP *progressive streaming* e HTTP *live streaming draft protocol* (Android 3.0)), que são suficientes para uma implementação desta solução.

6. Conclusões e Trabalho Futuro

O prolongado tempo de estudo e ambientação à plataforma Android fez com que não se realizasse o módulo de Agente nem os testes de comunicação com servidor de *streaming* de vídeo, o que seria deveras interessante para demonstrar a utilidade do Agente de Contexto.

6.1 Conclusões

Os resultados não são pois conclusivos em termos de eficiência de um Agente de Contexto numa plataforma Android.
O tempo de desenvolvimento foi notoriamente insuficiente visto ser um trabalho individual e ter havido uma necessidade de ter de estudar uma plataforma (Android) que é muito elaborada e diversificada e está em permanente evolução.
Considero que gerei uma boa documentação de base para um trabalho futuro de continuidade do tema.
De referir ainda que, conforme previsto, criei um *site* de suporte ao desenvolvimento da dissertação em http://paginas.fe.up.pt/~ee86019/index.html onde se pode encontrar este documento bem como outras informações da evolução do trabalho.
Por fim e talvez o mais importante, foi muito gratificante o trabalho desenvolvido visto ter criado uma excelente visão das potencialidades da plataforma Android bem como da necessidade de um Agente de Contexto na eficiência do serviço de *streaming* para dispositivos móveis.

6.2 Trabalho Futuro

Penso que esta Dissertação embora não tenha sido muito conclusiva pode servir como uma boa referência para trabalho futuros neste tema.
Os passos seguintes que se podem dar são gerar o ficheiro XML em formato UAProf com a descrição de contexto definidas neste documento e implementar um servidor de *streaming* de vídeo para testar a eficiência, com recurso por exemplo ao FFmpeg.
Sugiro ainda a consulta de fóruns e *blogs* de excelente qualidade como o http://android-developers.blogspot.com, o http://marakana.com/forums/blog/ e o http://www.bogotobogo.com/android.html, que tão úteis me foram.

Bibliografia

- H. W. Gellersen, A. Schmidt and M. Beigl: "Multi-Sensor Context-Awareness in Mobile Devices and Smart Artifacts", Mobile Networks and Applications, Volume 7, Issue 5 (October 2002), pp: 341 – 351, 2002.
- A. K. Dey: "Understanding and Using Context", Personal and Ubiquitous Computing, Special Issue on Situated Interaction and Ubiquitous Computing, 2001.
- A. Perkis, P. Drege, O. I. Hillestad: "UMA enabled environment for mobile media using MPEG-21 client and server technology", 2005.
- Christian Timmerer, Johannes Jabornig, Hermann Hellwagner: "A Survey on Delivery Context Description Formats – A Comparison and Mapping Model", Department of Information Technology (ITEC), Klagenfurt University, Austria 2009.
- Hongjiang Zhang: "ADAPTIVE CONTENT DELIVERY: A NEW APPLICATION AREA FORMEDIA COMPUTING RESEARCH", Microsoft Research, Beijing 100084, China
- Tayeb Lemlouma, Nabil Layaıda: "Basic Ideas for User Constraints Specification in CC/PP", OPERA Project, Montbonnot, Saint Martin, France 2002.
- Christian Timmerer: "Standard's Support for UMA", Klagenfurt University, Klagenfurt, AUSTRIA 2008.
- Tayeb Lemlouma: "Universal Access to Multimedia Information on the Web", Microsoft Research Silicon Valley, USA 2004.
- Tayeb LEMLOUMA, Nabil LAYAÏDA: "An Application Example of Universal Profiling for Content Negotiation", Opera Project, INRIA RHONE ALPES, 2002.
- Tayeb LEMLOUMA, Nabil LAYAÏDA: "Universal Profiling Schema for Content Negotiation", Opera Project, INRIA RHONE ALPES, 2002.
- Keara Barrett, Ruaidhri Power: "State of the Art: Context Management", Trinity College Dublin, 2003.
- Abhishek Singh, Michael Conway: "Survey of Context aware Frameworks - Analysis and Criticism", UNC-Chapel Hill ITS, 2006.
- Panu Vartiainen: "Using Metadata and Context Information in Sharing Personal Content of Mobile Users", UNIVERSITY OF HELSINKI, Department of Computer Science 2003.
- Harry Chen, Tim Finin, and Anupam Joshi: "An Ontology for Context-Aware Pervasive Computing Environments", Department of Computer Science and Electrical Engineering, University of Maryland Baltimore County.
- Thomas Strang, Claudia Linnho-Popien, Korbinian Frank: "CoOL: A Context Ontology Language to enable Contextual Interoperability", German Aerospace Center (DLR), Oberpfaenhofen, Germany.

- Tarak Chaari, Frédérique Laforest, Augusto Celentano: "Service-Oriented Context-Aware Application Design", LIRIS, INSA Lyon, France.
- S. Pokraev,P. D. Costa, J. G. Pereira Filho, M. Zuidweg, J. W. Koolwaaij, M. van Setten: "Context-aware services - State-of-the-art", Telematica Instituut, Ericsson, CTIT, 2002.
- Julien Pauty, Davy Preuveneers, Peter Rigole, Yolande Berbers: "Research Challenges in Mobile and Context-Aware, Service Development", Department of Computer Science, K. U. Leuven, B-3001 Leuven, Belgium.
- Julien Pauty, Davy Preuveneers, Peter Rigole, Yolande Berbers: "Middleware support for component-based ubiquitous and mobile computing applications", Department of Computer Science, K. U. Leuven, B-3001 Leuven, Belgium.
- Tao Gu, Hung Keng Pung, Da Qing Zhang: "A service-oriented middleware for building context-aware services", Network Systems and Services Laboratory, Department of Computer Science, National University of Singapore, 2004.
- Matthias Baldauf, Schahram Dustdar, Florian Rosenberg: "A Survey on Context-Aware Systems", Distributed Systems Group, Information Systems Institute, Vienna University of Technology, Austria.
- Waltenegus Dargie: "A Distributed Architecture for Computing Context in Mobile Devices", Dresden University of Technology, 2006.
- Maria Teresa Andrade: "Apontamentos da unidade curricular Televisão Digital e Novos Serviços", FEUP, 2010.

Referências

[1] Maria Teresa Andrade: "Apontamentos da unidade curricular Televisão Digital e Novos Serviços", FEUP, 2010.

[2] Panu Vartiainen: "Using Metadata and Context Information in Sharing Personal Content of Mobile Users", UNIVERSITY OF HELSINKI, Department of Computer Science 2003.

[3] Christian Timmerer, Johannes Jabornig, Hermann Hellwagner: "A Survey on Delivery Context Description Formats – A Comparison and Mapping Model", Department of Information Technology (ITEC), Klagenfurt University, Austria 2009.

[4] Abhishek Singh, Michael Conway: "Survey of Context aware Frameworks - Analysis and Criticism", UNC-Chapel Hill ITS, 2006.

[5] BBC news technology, http://www.bbc.co.uk/news/technology-10839034, acesso em 22/06/2011.

[6] http://www.java.com/, Oracle, Java, acesso em 22/06/2011

[7] http://www.eclipse.org/, The Eclipse Foundation, Eclipse, acesso em 22/06/2011

[8] http://developer.android.com, Google, Android Developers, acesso em 22/06/2011

Anexo A – Classes de API Android Usadas

Perante a informação de contexto que se pretende representar, fez-se uma pesquisa exaustiva, visto ser de uma dimensão enorme, das classes da API do Android a usar para criar o Agente. Optou-se por fazer aqui uma listagem dessas Classes para futura consulta mais rápida.

public class
Build
↳android.os.Build

Informações sobre a compilação actual, extraído das propriedades do sistema.

Fields		
public static final String	BOARD	The name of the underlying board, like "goldfish".
public static final String	BOOTLOADER	The system bootloader version number.
public static final String	BRAND	The brand (e.g., carrier) the software is customized for, if any.
public static final String	CPU_ABI	The name of the instruction set (CPU type + ABI convention) of native code.
public static final String	CPU_ABI2	The name of the second instruction set (CPU type + ABI convention) of native code.
public static final String	DEVICE	The name of the industrial design.
public static final String	DISPLAY	A build ID string meant for displaying to the user
public static final String	FINGERPRINT	A string that uniquely identifies this build.
public static final String	HARDWARE	The name of the hardware (from the kernel command line or /proc).
public static final String	HOST	
public static final String	ID	Either a changelist number, or a label like "M4-rc20".
public static final String	MANUFACTURER	The manufacturer of the product/hardware.
public static final String	MODEL	The end-user-visible name for the end product.
public static final String	PRODUCT	The name of the overall product.
public static final String	RADIO	The radio firmware version number.
public static final	SERIAL	A hardware serial number, if available.

String		
public static final String	TAGS	Comma-separated tags describing the build, like "unsigned,debug".
public static final long	TIME	
public static final String	TYPE	The type of build, like "user" or "eng".
public static final String	USER	

Figura 50. Tabela de campos da Classe: android.os.Build [8]

public abstract class

Context

↳android.content.Context

Interface com informações globais sobre o contexto da aplicação.

Constants		
String	ACCESSIBILITY_SERVICE	Use with `getSystemService(String)` to retrieve a `AccessibilityManager` for giving the user feedback for UI events through the registered event listeners.
String	ACCOUNT_SERVICE	Use with `getSystemService(String)` to retrieve a `AccountManager` for receiving intents at a time of your choosing.
String	ACTIVITY_SERVICE	Use with `getSystemService(String)` to retrieve a `ActivityManager` for interacting with the global system state.
String	ALARM_SERVICE	Use with `getSystemService(String)` to retrieve a `AlarmManager` for receiving intents at a time of your choosing.
String	AUDIO_SERVICE	Use with `getSystemService(String)` to retrieve a `AudioManager` for handling management of volume, ringer modes and audio routing.
int	BIND_AUTO_CREATE	Flag for `bindService(Intent, ServiceConnection, int)`: automatically create the service as long as the binding exists.
int	BIND_DEBUG_UNBIND	Flag for `bindService(Intent, ServiceConnection, int)`: include debugging help for mismatched calls to unbind.
int	BIND_NOT_FOREGROUND	Flag for `bindService(Intent, ServiceConnection, int)`: don't allow this binding to raise the target service's process to the foreground scheduling priority.
String	CLIPBOARD_SERVICE	Use with `getSystemService(String)` to retrieve a `ClipboardManager` for accessing and modifying the contents of the global clipboard.
String	CONNECTIVITY_SERVICE	Use with `getSystemService(String)` to retrieve a `ConnectivityManager` for handling management of network connections.

int	CONTEXT_IGNORE_SECURITY	Flag for use with `createPackageContext(String, int)`: ignore any security restrictions on the Context being requested, allowing it to always be loaded.
int	CONTEXT_INCLUDE_CODE	Flag for use with `createPackageContext(String, int)`: include the application code with the context.
int	CONTEXT_RESTRICTED	Flag for use with `createPackageContext(String, int)`: a restricted context may disable specific features.
String	DEVICE_POLICY_SERVICE	Use with `getSystemService(String)` to retrieve a `DevicePolicyManager` for working with global device policy management.
String	DOWNLOAD_SERVICE	Use with `getSystemService(String)` to retrieve a `DownloadManager` for requesting HTTP downloads.
String	DROPBOX_SERVICE	Use with `getSystemService(String)` to retrieve a `DropBoxManager` instance for recording diagnostic logs.
String	INPUT_METHOD_SERVICE	Use with `getSystemService(String)` to retrieve a `InputMethodManager` for accessing input methods.
String	KEYGUARD_SERVICE	Use with `getSystemService(String)` to retrieve a `NotificationManager` for controlling keyguard.
String	LAYOUT_INFLATER_SERVICE	Use with `getSystemService(String)` to retrieve a `LayoutInflater` for inflating layout resources in this context.
String	LOCATION_SERVICE	Use with `getSystemService(String)` to retrieve a `LocationManager` for controlling location updates.
int	MODE_APPEND	File creation mode: for use with `openFileOutput(String, int)`, if the file already exists then write data to the end of the existing file instead of erasing it.
int	MODE_MULTI_PROCESS	SharedPreference loading flag: when set, the file on disk will be checked for modification even if the shared preferences instance is already loaded in this process.
int	MODE_PRIVATE	File creation mode: the default mode, where the created file can only be accessed by the calling application (or all applications sharing the same user ID).
int	MODE_WORLD_READABLE	File creation mode: allow all other applications to have read access to the created file.
int	MODE_WORLD_WRITEABLE	File creation mode: allow all other applications to have write access to the created file.
String	NFC_SERVICE	Use with `getSystemService(String)` to retrieve a `NfcManager` for using NFC.
String	NOTIFICATION_SERVICE	Use with `getSystemService(String)` to retrieve a `NotificationManager` for informing the user of background events.
String	POWER_SERVICE	Use with `getSystemService(String)` to retrieve a `PowerManager` for controlling power management, including "wake locks," which let you keep the device on while you're running long tasks.
String	SEARCH_SERVICE	Use with `getSystemService(String)` to retrieve a

		`SearchManager` for handling searches.
String	SENSOR_SERVICE	Use with `getSystemService(String)` to retrieve a `SensorManager` for accessing sensors.
String	STORAGE_SERVICE	Use with `getSystemService(String)` to retrieve a `StorageManager` for accessing system storage functions.
String	TELEPHONY_SERVICE	Use with `getSystemService(String)` to retrieve a `TelephonyManager` for handling management the telephony features of the device.
String	UI_MODE_SERVICE	Use with `getSystemService(String)` to retrieve a `UiModeManager` for controlling UI modes.
String	USB_SERVICE	Use with `getSystemService(String)` to retrieve a `UsbManager` for access to USB devices (as a USB host) and for controlling this device's behavior as a USB device.
String	VIBRATOR_SERVICE	Use with `getSystemService(String)` to retrieve a `Vibrator` for interacting with the vibration hardware.
String	WALLPAPER_SERVICE	Use with `getSystemService(String)` to retrieve a com.android.server.WallpaperService for accessing wallpapers.
String	WIFI_SERVICE	Use with `getSystemService(String)` to retrieve a `WifiManager` for handling management of Wi-Fi access.
String	WINDOW_SERVICE	Use with `getSystemService(String)` to retrieve a `WindowManager` for accessing the system's window manager.

Figura 51. Tabela de campos da Classe: android.content.Context [8]

public class
Display
↳android.view.Display

Constants		
int	DEFAULT_DISPLAY	Specify the default Display

Public Methods	
int	getDisplayId() Returns the index of this display.
int	getHeight() Returns the raw height of the display, in pixels.
void	getMetrics(DisplayMetrics outMetrics) Initialize a DisplayMetrics object from this display's data.
int	getOrientation() *This method is deprecated. use* `getRotation()`
int	getPixelFormat() Return the native pixel format of the display.
float	getRefreshRate() Return the refresh rate of this display in frames per second.
int	getRotation() Returns the rotation of the screen from its "natural" orientation.

int	getWidth() Returns the raw width of the display, in pixels.

Figura 52. Tabela de campos da Classe: android.view.Display [8]

public class
MediaPlayer
↳android.media.MediaPlayer

A classe MediaPlayer pode ser usada para controlar a reprodução de ficheiros áudio / vídeo e streams.

Há a necessidade de declarar a permissão WAKE_LOCK .

Constants		
int	MEDIA_ERROR_NOT_VALID_FOR_PROGRESSIVE_PLAYBACK	The video is streamed and its container is not valid for progressive playback i.e the video's index (e.g moov atom) is not at the start of the file.
int	MEDIA_ERROR_SERVER_DIED	Media server died.
int	MEDIA_ERROR_UNKNOWN	Unspecified media player error.
int	MEDIA_INFO_BAD_INTERLEAVING	Bad interleaving means that a media has been improperly interleaved or not interleaved at all, e.g has all the video samples first then all the audio ones.
int	MEDIA_INFO_BUFFERING_END	MediaPlayer is resuming playback after filling buffers.
int	MEDIA_INFO_BUFFERING_START	MediaPlayer is temporarily pausing playback internally in order to buffer more data.
int	MEDIA_INFO_METADATA_UPDATE	A new set of metadata is available.
int	MEDIA_INFO_NOT_SEEKABLE	The media cannot be seeked (e.g live stream)
int	MEDIA_INFO_UNKNOWN	Unspecified media player info.
int	MEDIA_INFO_VIDEO_TRACK_LAGGING	The video is too complex for the decoder: it can't decode frames fast enough.

Public Constructors	
	MediaPlayer() Default constructor.

Public Methods	
void	attachAuxEffect(int effectId) Attaches an auxiliary effect to the player.
static MediaPlayer	create(Context context, Uri uri, SurfaceHolder holder) Convenience method to create a MediaPlayer for a given Uri.
static MediaPlayer	create(Context context, int resid) Convenience method to create a MediaPlayer for a given resource id.
static MediaPlayer	create(Context context, Uri uri) Convenience method to create a MediaPlayer for a given Uri.
int	getAudioSessionId() Returns the audio session ID.

int	getCurrentPosition() Gets the current playback position.
int	getDuration() Gets the duration of the file.
int	getVideoHeight() Returns the height of the video.
int	getVideoWidth() Returns the width of the video.
boolean	isLooping() Checks whether the MediaPlayer is looping or non-looping.
boolean	isPlaying() Checks whether the MediaPlayer is playing.
void	pause() Pauses playback.
void	prepare() Prepares the player for playback, synchronously.
void	prepareAsync() Prepares the player for playback, asynchronously.
void	release() Releases resources associated with this MediaPlayer object.
void	reset() Resets the MediaPlayer to its uninitialized state.
void	seekTo(int msec) Seeks to specified time position.
void	setAudioSessionId(int sessionId) Sets the audio session ID.
void	setAudioStreamType(int streamtype) Sets the audio stream type for this MediaPlayer.
void	setAuxEffectSendLevel(float level) Sets the send level of the player to the attached auxiliary effect .
void	setDataSource(String path) Sets the data source (file-path or http/rtsp URL) to use.
void	setDataSource(FileDescriptor fd, long offset, long length) Sets the data source (FileDescriptor) to use.
void	setDataSource(FileDescriptor fd) Sets the data source (FileDescriptor) to use.
void	setDataSource(Context context, Uri uri) Sets the data source as a content Uri.
void	setDisplay(SurfaceHolder sh) Sets the `SurfaceHolder` to use for displaying the video portion of the media.
void	setLooping(boolean looping) Sets the player to be looping or non-looping.
void	setOnBufferingUpdateListener(MediaPlayer.OnBufferingUpdateListener listener) Register a callback to be invoked when the status of a network stream's buffer has changed.
void	setOnCompletionListener(MediaPlayer.OnCompletionListener listener) Register a callback to be invoked when the end of a media source has been reached during playback.

void	setOnErrorListener(MediaPlayer.OnErrorListener listener) Register a callback to be invoked when an error has happened during an asynchronous operation.
void	setOnInfoListener(MediaPlayer.OnInfoListener listener) Register a callback to be invoked when an info/warning is available.
void	setOnPreparedListener(MediaPlayer.OnPreparedListener listener) Register a callback to be invoked when the media source is ready for playback.
void	setOnSeekCompleteListener(MediaPlayer.OnSeekCompleteListener listener) Register a callback to be invoked when a seek operation has been completed.
void	setOnVideoSizeChangedListener(MediaPlayer.OnVideoSizeChangedListener listener) Register a callback to be invoked when the video size is known or updated.
void	setScreenOnWhilePlaying(boolean screenOn) Control whether we should use the attached SurfaceHolder to keep the screen on while video playback is occurring.
void	setVolume(float leftVolume, float rightVolume) Sets the volume on this player.
void	setWakeMode(Context context, int mode) Set the low-level power management behavior for this MediaPlayer.
void	start() Starts or resumes playback.
void	stop() Stops playback after playback has been stopped or paused.

Figura 53. Tabela de campos da Classe: android.media.MediaPlayer [8]

public class
VideoView
↳android.widget.VideoView

Exibe um ficheiro de vídeo. A classe VideoView pode carregar imagens de diversas fontes (tais como recursos ou como fornecedores de conteúdos) e oferece várias opções de ecrã como o dimensionamento.

Public Constructors	
	VideoView(Context context)
	VideoView(Context context, AttributeSet attrs)
	VideoView(Context context, AttributeSet attrs, int defStyle)
Public Methods	
boolean	canPause()
boolean	canSeekBackward()
boolean	canSeekForward()
int	getBufferPercentage()
int	getCurrentPosition()
int	getDuration()

boolean	isPlaying()
boolean	onKeyDown(int keyCode, KeyEvent event) Default implementation of `KeyEvent.Callback.onKeyDown()`: perform press of the view when `KEYCODE_DPAD_CENTER` or `KEYCODE_ENTER` is released, if the view is enabled and clickable.
boolean	onTouchEvent(MotionEvent ev) Implement this method to handle touch screen motion events.
boolean	onTrackballEvent(MotionEvent ev) Implement this method to handle trackball motion events.
void	pause()
int	resolveAdjustedSize(int desiredSize, int measureSpec)
void	resume()
void	seekTo(int msec)
void	setMediaController(MediaController controller)
void	setOnCompletionListener(MediaPlayer.OnCompletionListener l) Register a callback to be invoked when the end of a media file has been reached during playback.
void	setOnErrorListener(MediaPlayer.OnErrorListener l) Register a callback to be invoked when an error occurs during playback or setup.
void	setOnPreparedListener(MediaPlayer.OnPreparedListener l) Register a callback to be invoked when the media file is loaded and ready to go.
void	setVideoPath(String path)
void	setVideoURI(Uri uri)
void	start()
void	stopPlayback()
void	suspend()

Figura 54. Tabela de campos da Classe: android.widget.VideoView

public class

WifiManager

↳android.net.wifi.WifiManager

Essa classe serve para gerir todos os aspectos de conectividade Wi-Fi.

Constants		
String	ACTION_PICK_WIFI_NETWORK	Activity Action: Pick a Wi-Fi network to connect to.
int	ERROR_AUTHENTICATING	The error code if there was a problem authenticating.
String	EXTRA_BSSID	The lookup key for a String giving the BSSID of the access point to which we are connected.
String	EXTRA_NETWORK_INFO	The lookup key for a `NetworkInfo` object associated with the Wi-Fi network.
String	EXTRA_NEW_RSSI	The lookup key for an `int` giving the new RSSI in dBm.

String	EXTRA_NEW_STATE	The lookup key for a `SupplicantState` describing the new state Retrieve with `getParcelableExtra(String)`.
String	EXTRA_PREVIOUS_WIFI_STATE	The previous Wi-Fi state.
String	EXTRA_SUPPLICANT_CONNECTED	The lookup key for a boolean that indicates whether a connection to the supplicant daemon has been gained or lost.
String	EXTRA_SUPPLICANT_ERROR	The lookup key for a `SupplicantState` describing the supplicant error code if any Retrieve with `getIntExtra(String, int)`.
String	EXTRA_WIFI_STATE	The lookup key for an int that indicates whether Wi-Fi is enabled, disabled, enabling, disabling, or unknown.
String	NETWORK_IDS_CHANGED_ACTION	The network IDs of the configured networks could have changed.
String	NETWORK_STATE_CHANGED_ACTION	Broadcast intent action indicating that the state of Wi-Fi connectivity has changed.
String	RSSI_CHANGED_ACTION	The RSSI (signal strength) has changed.
String	SCAN_RESULTS_AVAILABLE_ACTION	An access point scan has completed, and results are available from the supplicant.
String	SUPPLICANT_CONNECTION_CHANGE_ACTION	Broadcast intent action indicating that a connection to the supplicant has been established (and it is now possible to perform Wi-Fi operations) or the connection to the supplicant has been lost.
String	SUPPLICANT_STATE_CHANGED_ACTION	Broadcast intent action indicating that the state of establishing a connection to an access point has changed.One extra provides the new `SupplicantState`.
int	WIFI_MODE_FULL	In this Wi-Fi lock mode, Wi-Fi will be kept active, and will behave normally, i.e., it will attempt to automatically establish a connection to a remembered access point that is within range, and will do periodic scans if there are remembered access points but none are in range.
int	WIFI_MODE_FULL_HIGH_PERF	In this Wi-Fi lock mode, Wi-Fi will be kept active as in mode `WIFI_MODE_FULL` but it operates at high performance with minimum packet loss and low packet latency even when the device screen is off.
int	WIFI_MODE_SCAN_ONLY	In this Wi-Fi lock mode, Wi-Fi will be kept active, but the only operation that will be supported is initiation of scans, and the subsequent reporting of scan results.
String	WIFI_STATE_CHANGED_ACTION	Broadcast intent action indicating that Wi-Fi has been enabled, disabled, enabling, disabling, or unknown.
int	WIFI_STATE_DISABLED	Wi-Fi is disabled.
int	WIFI_STATE_DISABLING	Wi-Fi is currently being disabled.
int	WIFI_STATE_ENABLED	Wi-Fi is enabled.
int	WIFI_STATE_ENABLING	Wi-Fi is currently being enabled.
int	WIFI_STATE_UNKNOWN	Wi-Fi is in an unknown state.
Public Methods		

int	addNetwork(WifiConfiguration config) Add a new network description to the set of configured networks.
static int	calculateSignalLevel(int rssi, int numLevels) Calculates the level of the signal.
static int	compareSignalLevel(int rssiA, int rssiB) Compares two signal strengths.
WifiManager.MulticastLock	createMulticastLock(String tag) Create a new MulticastLock
WifiManager.WifiLock	createWifiLock(int lockType, String tag) Creates a new WifiLock.
WifiManager.WifiLock	createWifiLock(String tag) Creates a new WifiLock.
boolean	disableNetwork(int netId) Disable a configured network.
boolean	disconnect() Disassociate from the currently active access point.
boolean	enableNetwork(int netId, boolean disableOthers) Allow a previously configured network to be associated with.
List<WifiConfiguration>	getConfiguredNetworks() Return a list of all the networks configured in the supplicant.
WifiInfo	getConnectionInfo() Return dynamic information about the current Wi-Fi connection, if any is active.
DhcpInfo	getDhcpInfo() Return the DHCP-assigned addresses from the last successful DHCP request, if any.
List<ScanResult>	getScanResults() Return the results of the latest access point scan.
int	getWifiState() Gets the Wi-Fi enabled state.
boolean	isWifiEnabled() Return whether Wi-Fi is enabled or disabled.
boolean	pingSupplicant() Check that the supplicant daemon is responding to requests.
boolean	reassociate() Reconnect to the currently active access point, even if we are already connected.
boolean	reconnect() Reconnect to the currently active access point, if we are currently disconnected.
boolean	removeNetwork(int netId) Remove the specified network from the list of configured networks.
boolean	saveConfiguration() Tell the supplicant to persist the current list of configured networks.
boolean	setWifiEnabled(boolean enabled) Enable or disable Wi-Fi.
boolean	startScan() Request a scan for access points.
int	updateNetwork(WifiConfiguration config)

	Update the network description of an existing configured network.

Figura 55. Tabela de campos da Classe: android.net.wifi.WifiManager [8]

public class
ConnectivityManager
↳android.net.ConnectivityManager

Classe que responde a consultas sobre o estado de conectividade de rede. Ele também notifica os aplicaçõesquando há mudanças de conectividade de rede. As principais responsabilidades desta classe são:

1. Monitor conexões de rede (Wi-Fi, GPRS, UMTS, etc)
2. Enviar intenções à operadora quando há mudanças na conectividade de rede
3. Tentativa de "fail over" para outra rede quando a conectividade a uma rede é perdida
4. Fornecer uma API que permite que as aplicações possam consultar o estado das redes disponíveis

Constants		
String	ACTION_BACKGROUND_DATA_SETTING_CHANGED	Broadcast Action: The setting for background data usage has changed values.
String	CONNECTIVITY_ACTION	A change in network connectivity has occurred.
int	DEFAULT_NETWORK_PREFERENCE	
String	EXTRA_EXTRA_INFO	The lookup key for a string that provides optionally supplied extra information about the network state.
String	EXTRA_IS_FAILOVER	The lookup key for a boolean that indicates whether a connect event is for a network to which the connectivity manager was failing over following a disconnect on another network.
String	EXTRA_NETWORK_INFO	The lookup key for a `NetworkInfo` object.
String	EXTRA_NO_CONNECTIVITY	The lookup key for a boolean that indicates whether there is a complete lack of connectivity, i.e., no network is available.
String	EXTRA_OTHER_NETWORK_INFO	The lookup key for a `NetworkInfo` object.
String	EXTRA_REASON	The lookup key for a string that indicates why an attempt to connect to a network failed.
int	TYPE_MOBILE	The Default Mobile data connection.
int	TYPE_MOBILE_DUN	A DUN-specific Mobile data connection.
int	TYPE_MOBILE_HIPRI	A High Priority Mobile data connection.
int	TYPE_MOBILE_MMS	An MMS-specific Mobile data connection.
int	TYPE_MOBILE_SUPL	A SUPL-specific Mobile data connection.

int	TYPE_WIFI	The Default WIFI data connection.
int	TYPE_WIMAX	The Default WiMAX data connection.

Public Methods	
NetworkInfo	getActiveNetworkInfo()
NetworkInfo[]	getAllNetworkInfo()
boolean	getBackgroundDataSetting() Returns the value of the setting for background data usage.
NetworkInfo	getNetworkInfo(int networkType)
int	getNetworkPreference()
static boolean	isNetworkTypeValid(int networkType)
boolean	requestRouteToHost(int networkType, int hostAddress) Ensure that a network route exists to deliver traffic to the specified host via the specified network interface.
void	setNetworkPreference(int preference)
int	startUsingNetworkFeature(int networkType, String feature) Tells the underlying networking system that the caller wants to begin using the named feature.
int	stopUsingNetworkFeature(int networkType, String feature) Tells the underlying networking system that the caller is finished using the named feature.

Figura 56. Tabela de campos da Classe: android.net.ConnectivityManager [8]

public class
NetworkInfo
↳android.net.NetworkInfo

Descreve o estado de uma interface de rede de um determinado tipo GSM ou Wifi.

Public Methods	
NetworkInfo.DetailedState	getDetailedState() Reports the current fine-grained state of the network.
String	getExtraInfo() Report the extra information about the network state, if any was provided by the lower networking layers., if one is available.
String	getReason() Report the reason an attempt to establish connectivity failed, if one is available.
NetworkInfo.State	getState() Reports the current coarse-grained state of the network.
int	getSubtype() Return a network-type-specific integer describing the subtype of the network.
String	getSubtypeName() Return a human-readable name describing the subtype of the network.
int	getType() Reports the type of network (currently mobile or Wi-Fi) to which the info in this

	object pertains.
String	getTypeName() Return a human-readable name describe the type of the network, for example "WIFI" or "MOBILE".
boolean	isAvailable() Indicates whether network connectivity is possible.
boolean	isConnected() Indicates whether network connectivity exists and it is possible to establish connections and pass data.
boolean	isConnectedOrConnecting() Indicates whether network connectivity exists or is in the process of being established.
boolean	isFailover() Indicates whether the current attempt to connect to the network resulted from the ConnectivityManager trying to fail over to this network following a disconnect from another network.
boolean	isRoaming() Indicates whether the device is currently roaming on this network.
String	toString() Returns a string containing a concise, human-readable description of this object.

Figura 57. Tabela de campos da Classe: android.net.NetworkInfo [8]

Anexo B – Listagens dos Módulos

Contexto do Hardware

```
package com.ContextBuildClass;

//Contexto do Hardware
//junho 2011, Pedro Garrido

import android.app.Activity;
import android.os.Build;
import android.os.Bundle;
import android.widget.TextView;

public class ContextBuildClass extends Activity {
      private TextView text;
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
      text =  (TextView) findViewById(R.id.textBoard);
      text.append(Build.BOARD);
            text =  (TextView) findViewById(R.id.textBrand);
            text.append(Build.BRAND);
            text =  (TextView) findViewById(R.id.textCpu);
            text.append(Build.CPU_ABI);
            text =  (TextView) findViewById(R.id.textDevice);
            text.append(Build.DEVICE);
            text =  (TextView) findViewById(R.id.textDisplay);
            text.append(Build.DISPLAY);
            text =  (TextView) findViewById(R.id.textFinger);
            text.append(Build.FINGERPRINT);
            text =  (TextView) findViewById(R.id.textHost);
            text.append(Build.HOST);
            text =  (TextView) findViewById(R.id.textId);
            text.append(Build.ID);
            text =  (TextView) findViewById(R.id.textManuf);
            text.append(Build.MANUFACTURER);
            text =  (TextView) findViewById(R.id.textModel);
            text.append(Build.MODEL);
            text =  (TextView) findViewById(R.id.textProduct);
            text.append(Build.PRODUCT);
            text =  (TextView) findViewById(R.id.textTags);
            text.append(Build.TAGS);
            text =  (TextView) findViewById(R.id.textType);
            text.append(Build.TYPE);
            text =  (TextView) findViewById(R.id.textUser);
            text.append(Build.USER);
      }
}
```

Contexto - diversos

```
package com.ContextContextClass;

//Contexto "Context"
//junho 2011, Pedro Garrido

import android.app.Activity;
import android.content.Context;
import android.os.Bundle;
import android.widget.TextView;

public class ContextContextClass extends Activity {
      private TextView text;
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        text =(TextView) findViewById(R.id.textView1);
      text.append("ACCESSIBILITY_SERVICE: "+Context.ACCESSIBILITY_SERVICE);
            text.append("\nACCOUNT_SERVICE: "+Context.ACCOUNT_SERVICE);
            text.append("\nACTIVITY_SERVICE: "+Context.ACTIVITY_SERVICE);
            text.append("\nALARM_SERVICE: "+Context.ALARM_SERVICE);
            text.append("\nAUDIO_SERVICE: "+Context.AUDIO_SERVICE);
            text.append("\nCLIPBOARD_SERVICE: "+Context.CLIPBOARD_SERVICE);
            text.append("\nCLIPBOARD_SERVICE: "+Context.CONNECTIVITY_SERVICE);
            text.append("\nCONTEXT_IGNORE_SECURITY:
"+Context.CONTEXT_IGNORE_SECURITY);
            text.append("\nCONTEXT_INCLUDE_CODE:
"+Context.CONTEXT_INCLUDE_CODE);
            text.append("\nCONTEXT_RESTRICTED: "+Context.CONTEXT_RESTRICTED);
            text.append("\nINPUT_METHOD_SERVICE:
"+Context.INPUT_METHOD_SERVICE);
            text.append("\nKEYGUARD_SERVICE: "+Context.KEYGUARD_SERVICE);
            text.append("\nLAYOUT_INFLATER_SERVICE:
"+Context.LAYOUT_INFLATER_SERVICE);
            text.append("\nLOCATION_SERVICE: "+Context.LOCATION_SERVICE);
            text.append("\nMODE_APPEND: "+Context.MODE_APPEND);
            text.append("\nMODE_PRIVATE: "+Context.MODE_PRIVATE);
            text.append("\nMODE_WORLD_READABLE:
"+Context.MODE_WORLD_READABLE);
            text.append("\nMODE_WORLD_WRITEABLE:
"+Context.MODE_WORLD_WRITEABLE);
            text.append("\nNOTIFICATION_SERVICE:
"+Context.NOTIFICATION_SERVICE);
            text.append("\nPOWER_SERVICE: "+Context.POWER_SERVICE);
            text.append("\nSEARCH_SERVICE: "+Context.SEARCH_SERVICE);
            text.append("\nSENSOR_SERVICE: "+Context.SENSOR_SERVICE);
            text.append("\nTELEPHONY_SERVICE: "+Context.TELEPHONY_SERVICE);
            text.append("\nVIBRATOR_SERVICE: "+Context.VIBRATOR_SERVICE);
            text.append("\nWALLPAPER_SERVICE: "+Context.WALLPAPER_SERVICE);
            text.append("\nWIFI_SERVICE: "+Context.WIFI_SERVICE);
            text.append("\nWINDOW_SERVICE: "+Context.WINDOW_SERVICE);
    }
}
```

Contexto de Resolução e Densidade do Ecrã

```
package com.ContextViewDisplay;

//Contexto resolução e densidade do ecrã
//junho 2011, Pedro Garrido

import android.app.Activity;
import android.os.Bundle;
import android.view.Display;
import android.view.WindowManager;
import android.widget.TextView;

public class ContextViewDisplay extends Activity {
      private TextView text;
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        Display display = ((WindowManager)
getSystemService(WINDOW_SERVICE)).getDefaultDisplay();
            text =(TextView) findViewById(R.id.textView1);
            text.append("getWidth: "+display.getWidth());
            text.append("\ngetHeight: "+display.getHeight());
            text.append("\ngetOrientation: "+display.getOrientation());
            text.append("\ngetPixelFormat: "+display.getPixelFormat());
            text.append("\ngetRefreshRate: "+display.getRefreshRate());
            text.append("\n\nDensityDPI:
"+getResources().getDisplayMetrics().densityDpi);
    }
}
```

Slideshow para Diferentes Resoluções de Densidades de Ecrã

```
package com.example.android.multires;

//slideshow para diferentes resoluções de densidades de ecrã
//Junho 2011, Pedro Garrido

import android.app.Activity;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.ImageView;
import android.widget.TextView;
import java.util.Random;

public final class MultiRes extends Activity {
      private int mCurrentPhotoIndex =0;
      //    Random rand = new Random();
      //private int mCurrentPhotoIndex =rand.nextInt(7);// arranca numa foto
aleatoria
    private int[] mPhotoIds = new int[] { R.drawable.sample_0,
            R.drawable.sample_1, R.drawable.sample_2, R.drawable.sample_3,
            R.drawable.sample_4, R.drawable.sample_5, R.drawable.sample_6,
            R.drawable.sample_7 };
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
```

```
        showPhoto(mCurrentPhotoIndex);
        // Handle clicks on the 'Next' button.
        Button nextButton = (Button) findViewById(R.id.next_button);
        nextButton.setOnClickListener(new View.OnClickListener() {
            public void onClick(View v) {
                mCurrentPhotoIndex = (mCurrentPhotoIndex + 1)
                        % mPhotoIds.length;
                showPhoto(mCurrentPhotoIndex);
            }
        });
    }
      @Override
    protected void onSaveInstanceState(Bundle outState) {
        outState.putInt("photo_index", mCurrentPhotoIndex);
        super.onSaveInstanceState(outState);
    }
    @Override
    protected void onRestoreInstanceState(Bundle savedInstanceState) {
      mCurrentPhotoIndex = savedInstanceState.getInt("photo_index");
        showPhoto(mCurrentPhotoIndex);
        super.onRestoreInstanceState(savedInstanceState);
    }
    private void showPhoto(int photoIndex) {
        ImageView imageView = (ImageView) findViewById(R.id.image_view);
        imageView.setImageResource(mPhotoIds[photoIndex]);
        TextView statusText = (TextView) findViewById(R.id.status_text);
        statusText.setText(String.format("%d/%d", photoIndex + 1,
                mPhotoIds.length));
    }
}
```

VideoPlayer

```
package com.audiovideo.MediaPlayer;

//Context VideoPlayer
//Player de audio e video para ficheiros locais e STREAM
//adaptação
//Junho 2011, Pedro Garrido
//creditos http://www.bogotobogo.com/android.html

import android.app.Activity;
import android.media.AudioManager;
import android.media.MediaPlayer;
import android.media.MediaPlayer.OnBufferingUpdateListener;
import android.media.MediaPlayer.OnCompletionListener;
import android.media.MediaPlayer.OnPreparedListener;
import android.media.MediaPlayer.OnVideoSizeChangedListener;
import android.os.Bundle;
import android.util.Log;
import android.view.SurfaceHolder;
import android.view.SurfaceView;

public class MyVideoPlayer extends Activity implements
        OnBufferingUpdateListener, OnCompletionListener,
        OnPreparedListener, OnVideoSizeChangedListener, SurfaceHolder.Callback
{

    private static final String TAG = "AudioVideo";
    private int mVideoWidth;
```

```
    private int mVideoHeight;
    private MediaPlayer mMediaPlayer;
    private SurfaceView mPreview;
    private SurfaceHolder holder;
    private String path;
    private Bundle extras;

    private static final String AUDIOVIDEO = "audiovideo";

    private static final int LOCAL_VIDEO = 4;
    private static final int STREAM_VIDEO = 5;
    private static final int RESOURCES_VIDEO = 6;

    private boolean mIsVideoSizeKnown = false;
    private boolean mIsVideoReadyToBePlayed = false;

    /**
     * Called when the activity is first created.
     */
    @Override
    public void onCreate(Bundle icicle) {
        super.onCreate(icicle);
        setContentView(R.layout.mediaplayer_2);
        mPreview = (SurfaceView) findViewById(R.id.surface);
        holder = mPreview.getHolder();
        holder.addCallback(this);
        holder.setType(SurfaceHolder.SURFACE_TYPE_PUSH_BUFFERS);
        extras = getIntent().getExtras();
    }

    private void playVideo(Integer AUDIOVIDEO) {
        doCleanUp();
        try {

            switch (AUDIOVIDEO) {
                case LOCAL_VIDEO:
                    /*
                     * TODO: Set the path variable to a local media file path.
                     */
                    path = "/sdcard/sample.mov";
                    // Create a new media player and set the listeners
                    mMediaPlayer = new MediaPlayer();
                    mMediaPlayer.setDataSource(path);
                    mMediaPlayer.setDisplay(holder);
                    mMediaPlayer.prepare();
                    mMediaPlayer.setOnBufferingUpdateListener(this);
                    mMediaPlayer.setOnCompletionListener(this);
                    mMediaPlayer.setOnPreparedListener(this);
                    mMediaPlayer.setOnVideoSizeChangedListener(this);

mMediaPlayer.setAudioStreamType(AudioManager.STREAM_MUSIC);
                    break;
                case RESOURCES_VIDEO:
                    /**
                     * TODO: Upload a video file to res/raw folder
                     */
                    mMediaPlayer = MediaPlayer.create(this, R.raw.samplemp4);
                    mMediaPlayer.start();
                case STREAM_VIDEO:
                    /*
                     * TODO: Set path variable to progressive streamable mp4
or
                     * 3gpp format URL. Http protocol should be used.
```

```
                     * Mediaplayer can only play "progressive streamable
                     * contents" which basically means: 1. the movie atom has
to
                     * precede all the media data atoms. 2. The clip has to be
                     * reasonably interleaved.
                     *
                     */
                    path = "http://www.bogotobogo.com/Video/sample.3gp";
                    // Create a new media player and set the listeners
                    mMediaPlayer = new MediaPlayer();
                    mMediaPlayer.setDataSource(path);
                    mMediaPlayer.setDisplay(holder);
                    mMediaPlayer.prepare();
                    mMediaPlayer.setOnBufferingUpdateListener(this);
                    mMediaPlayer.setOnCompletionListener(this);
                    mMediaPlayer.setOnPreparedListener(this);
                    mMediaPlayer.setOnVideoSizeChangedListener(this);

mMediaPlayer.setAudioStreamType(AudioManager.STREAM_MUSIC);
                    break;
                }
            } catch (Exception e) {
                Log.e(TAG, "error: " + e.getMessage(), e);
            }
        }

        public void onBufferingUpdate(MediaPlayer arg0, int percent) {
            Log.d(TAG, "onBufferingUpdate percent:" + percent);

        }

        public void onCompletion(MediaPlayer arg0) {
            Log.d(TAG, "onCompletion called");
        }

        public void onVideoSizeChanged(MediaPlayer mp, int width, int height) {
            Log.v(TAG, "onVideoSizeChanged called");
            if (width == 0 || height == 0) {
                Log.e(TAG, "invalid video width(" + width + ") or height(" +
height + ")");
                return;
            }
            mIsVideoSizeKnown = true;
            mVideoWidth = width;
            mVideoHeight = height;
            if (mIsVideoReadyToBePlayed && mIsVideoSizeKnown) {
                startVideoPlayback();
            }
        }

        public void onPrepared(MediaPlayer mediaplayer) {
            Log.d(TAG, "onPrepared called");
            mIsVideoReadyToBePlayed = true;
            if (mIsVideoReadyToBePlayed && mIsVideoSizeKnown) {
                startVideoPlayback();
            }
        }

        public void surfaceChanged(SurfaceHolder surfaceholder, int i, int j, int
k) {
            Log.d(TAG, "surfaceChanged called");

        }
```

```
    public void surfaceDestroyed(SurfaceHolder surfaceholder) {
        Log.d(TAG, "surfaceDestroyed called");
    }

    public void surfaceCreated(SurfaceHolder holder) {
        Log.d(TAG, "surfaceCreated called");
        playVideo(extras.getInt(AUDIOVIDEO));
    }

    @Override
    protected void onPause() {
        super.onPause();
        releaseMediaPlayer();
        doCleanUp();
    }

    @Override
    protected void onDestroy() {
        super.onDestroy();
        releaseMediaPlayer();
        doCleanUp();
    }

    private void releaseMediaPlayer() {
        if (mMediaPlayer != null) {
            mMediaPlayer.release();
            mMediaPlayer = null;
        }
    }

    private void doCleanUp() {
        mVideoWidth = 0;
        mVideoHeight = 0;
        mIsVideoReadyToBePlayed = false;
        mIsVideoSizeKnown = false;
    }

    private void startVideoPlayback() {
        Log.v(TAG, "startVideoPlayback");
        holder.setFixedSize(mVideoWidth, mVideoHeight);
        mMediaPlayer.start();
    }
}
```

Estado das Ligações GSM e Wifi

```
package com.ContextConnectivityManager;

//Estado das ligações GSM e Wifi
//junho 2011, Pedro Garrido

import android.app.Activity;
import android.content.Context;
import android.net.ConnectivityManager;
import android.os.Bundle;
import android.widget.TextView;

public class ContextConnectivityManager extends Activity {
```

```
      private TextView text;
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);

        ConnectivityManager conMan = (ConnectivityManager)
getSystemService(Context.CONNECTIVITY_SERVICE);

        text =(TextView) findViewById(R.id.textView1);
        text.append("GSM: " + conMan.getNetworkInfo(0).getState());//GSM
        text.append("\nWiFi: " + conMan.getNetworkInfo(1).getState());//Wifi
    }
}
```

Contexto do GSM e Wifi

```
package com.ContextWifiInfo;

//Contexto Wifi info
//junho 2011, Pedro Garrido

import android.app.Activity;
import android.net.wifi.WifiInfo;
import android.net.wifi.WifiManager;
import android.os.Bundle;
import android.widget.TextView;

public class ContextWifiInfo extends Activity {
      private TextView text;
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        WifiManager wifiManager = (WifiManager)
getSystemService(WIFI_SERVICE);
        WifiInfo wifiInfo = wifiManager.getConnectionInfo();
        int ipAddress = wifiInfo.getIpAddress();
        float speedlink = wifiInfo.getLinkSpeed();
        text =(TextView) findViewById(R.id.textView1);
        text.append("IP: "+ipAddress);
            text.append("\nspeedlink: "+speedlink);
            text.append("\ngetBSSID: "+wifiInfo.getBSSID());
            text.append("\ngetNetworkId: "+wifiInfo.getNetworkId());
            text.append("\ngetSSID: "+wifiInfo.getSSID());
            text.append("\nLINK_SPEED_UNITS: "+wifiInfo.LINK_SPEED_UNITS);
    }
}
```

ConnectivityDemo

```
package connectivityDemo;

//ConnectivityDemo
//junho 2011, Pedro Garrido

import android.app.Activity;
import android.content.Context;
import android.net.ConnectivityManager;
import android.net.NetworkInfo;
import android.net.wifi.WifiInfo;
import android.os.Bundle;
import android.widget.TextView;

public class ConnectivityDemo extends Activity {
  ConnectivityManager connectivity;
  NetworkInfo wifiInfo, mobileInfo,preferenceInfo;
  WifiInfo WifiVelocidade;
  TextView textStatus;
  @Override
  public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.main);
    // Get UI
    textStatus = (TextView) findViewById(R.id.textStatus);
    // Setup Connectivity
    connectivity = (ConnectivityManager)
getSystemService(Context.CONNECTIVITY_SERVICE);
    wifiInfo = connectivity.getNetworkInfo(ConnectivityManager.TYPE_WIFI);
    mobileInfo = connectivity.getNetworkInfo(ConnectivityManager.TYPE_MOBILE);
    // print info
    textStatus.append("\n\n" + wifiInfo.toString());
    textStatus.append("\n\n" + mobileInfo.toString());
  }
}
```